XY-pic Complete Sources with TeXnical Commentary

Kristoffer H. Rose  ⟨krisrose@tug.org⟩¹  Ross Moore  ⟨ross.moore@mq.edu.au⟩²

XY-pic version 3.8.9 ⟨2013/10/06⟩

¹IBM T.J.Watson Research Center, P.O.Box 704, Yorktown Heights, NY 10598, USA
²MPCE (Mathematics dept.), Macquarie University, North Ryde, Sydney, Australia NSW 2109.
Printed October 5, 2013, from xysource.man version 3.28.
This report is typeset from the XY-pic sources, version 3.8.9 released October 6, 2013. It includes all of the text in the XY-pic Reference Manual [16].
XY-pic related files can be retrieved from http://xy-pic.sourceforge.net.

XY-pic Complete Sources with TeXnical Commentary
The XY-pic package is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
The XY-pic package is distributed in the hope that it will be useful, but without any warranty; without even the implied warranty of merchantability or fitness for a particular purpose. See the GNU General Public License for more details.
You should have received a copy of the GNU General Public License along with the XY-pic package; if not, see http://www.gnu.org/licenses/.
Preface

In this report we present the TeX and METAFONT sources of the XY-pic package with TeXnical commentary. It assumes that you are familiar with the use of XY-pic as described in the User’s Guide [15] as well as with the fundamentals of TeX [6] and METAFONT [7].

Overview. Chapter 1 explains the TeX sources of the XY-picture kernel drawing language. Chapter 2 explains the sources of the standard extensions providing extended graphic capabilities, and chapter 3 the sources of standard features providing specialised notation for particular diagram types. Chapter 5 exposes the METAFONT sources of the standard fonts provided in the distribution.

The appendices contain additional information: Appendix A contains answers to all the exercises of the main text. Appendix B summarises the backwards compatibility with XY-pic version 2. Appendix C contains the GNU General Public License detailing the conditions of use of XY-pic, and appendix D various support files contained in the distribution. Finally a bibliography and the index.

The report includes most of the information in the reference manual [16].

License. Most of the XY-pic package is copyright by the individual authors. We have chosen to use the “GNU Copyleft” to make the package available without restriction to as many users as possible and ensure that it is attributed to its authors without risk of persecution. The disclaimer can be found on the inside cover page; the full text of the license is reproduced as appendix C.
Contents

1 Kernel: xy.doc  
1.1 The \( \text{XY-pic} \) implementation ................................................................. 1  
1.1.1 Loading \( \text{XY-pic} \) ................................................................. 1  
1.1.2 Logo, version, and messages ................................................................. 14  
1.1.3 Fonts ......................................................................................... 15  
1.1.4 Allocations ................................................................................. 16  
1.1.5 Utility macros ........................................................................... 18  
1.2 Picture basics ................................................................................. 20  
1.2.1 Positions ................................................................................ 21  
1.2.2 Objects ................................................................................. 21  
1.2.3 Connections ........................................................................... 21  
1.2.4 Decorations ........................................................................... 22  
1.2.5 The \( \text{XY-pic} \) state ................................................................. 22  
1.3 Positions ..................................................................................... 23  
1.4 Objects ......................................................................................... 46  
1.5 Decorations .................................................................................. 66  
1.6 Kernel object library ........................................................................ 71  
1.6.1 Directionals ........................................................................ 71  
1.6.2 Circle segments ..................................................................... 86  
1.6.3 Text ......................................................................................... 90  
1.7 \( \text{XY-pic} \) options ........................................................................... 91  
1.7.1 Loading ................................................................................ 91  
1.7.2 Option file format ................................................................. 92  
1.7.3 Driver options ....................................................................... 94  
1.8 Algorithms .................................................................................... 98  
1.8.1 Directions ........................................................................ 98  
1.8.2 Edges ................................................................................... 104  
1.8.3 Connections ........................................................................ 112  

2 Extensions ................................................................................. 123  
2.1 Curve and Spline extension .......................................................... 123  
2.1.1 Curved connections ............................................................... 124  
2.1.2 Circles and Ellipses ............................................................... 192  
2.1.3 Quadratic Splines ................................................................. 197  
2.2 Frame and Bracket extension ......................................................... 201  
2.2.1 Frames ............................................................................... 203  
2.2.2 Brackets ............................................................................. 211  
2.2.3 Filled regions ...................................................................... 214  
2.2.4 Framing as object modifier .................................................. 215
2.2.5 Using curves for frames .................................................. 217
2.3 More Tips extension ......................................................... 219
  2.3.1 End & log ................................................................. 222
2.4 Line styles extension ...................................................... 223
2.5 Rotate and Scale extension .............................................. 233
2.6 Colour extension ........................................................... 240
2.7 Pattern and Tile extension .............................................. 248
2.8 Import graphics extension .............................................. 254
2.9 Movie Storyboard extension ........................................... 259
2.10 PostScript backend ...................................................... 261
  2.10.1 Choosing the DVI-driver ........................................... 262
  2.10.2 Why use PostScript \(^1\) ........................................ 267
  2.10.3 Hooking into X\textup{-}pic .......................................... 269
  2.10.4 Kernel improvements .............................................. 270
2.11 TPIC backend ............................................................. 277
2.12 em-TeX backend .......................................................... 285
2.13 Necula's extensions ...................................................... 288
  2.13.1 Expansion ............................................................. 289
  2.13.2 Polygon shapes ...................................................... 290
2.14 LaTeX Picture extension .............................................. 299

3 Features .......................................................................... 303
  3.1 All features .................................................................. 303
  3.2 Dummy option .............................................................. 305
  3.3 Arrow and Path feature ................................................ 306
    3.3.1 Paths ................................................................. 307
    3.3.2 Arrows ............................................................... 319
  3.4 Two-cell feature .......................................................... 329
    3.4.1 Typesetting 2-cells in Diagrams ................................. 330
    3.4.2 Standard Options ................................................. 336
    3.4.3 Nudging .............................................................. 337
    3.4.4 Extra Options ...................................................... 338
    3.4.5 After Parsing........................................................ 343
    3.4.6 2-cells in general X\textup{-}pictures .............................. 344
  3.5 Matrix feature ............................................................. 355
    3.5.1 X\textup{-}matrices ...................................................... 355
    3.5.2 New coordinate formats ........................................ 365
    3.5.3 Spacing and rotation ............................................. 368
    3.5.4 Entries .............................................................. 370
  3.6 Graph feature ............................................................. 375
  3.7 Polygon feature ........................................................... 384
  3.8 Lattice and web feature .............................................. 404
  3.9 Circle, Ellipse, Arc feature ........................................... 410
    3.9.1 Full Circles ......................................................... 412
    3.9.2 Ellipses ............................................................. 415
    3.9.3 Drawing arcs ................................ ....................... 415
    3.9.4 Circular and Elliptical Arcs .................................... 420
  3.10 Knots and Links feature ............................................... 433

\(^1\text{PostScript} \text{ is a registered Trademark of Adobe, Inc. \cite{adobe}.}\)
3.11 Smart Path option ................................................................. 461

4 Drivers .......................................................... 477
  4.1 Support for Specific Drivers ............................................... 477
  4.2 dvidrv driver .............................................................. 477
  4.3 DVIPS driver ............................................................... 479
  4.4 DVIOPS driver .............................................................. 484
  4.5 OzTeX driver ............................................................... 490
  4.6 OzTeX v1.7 driver ......................................................... 494
  4.7 Textures driver ............................................................ 499
  4.8 Textures v1.6 driver ....................................................... 503
  4.9 XDVI driver ............................................................... 508
  4.10 PDF driver ............................................................... 513
  4.11 Extra features with POSTSCRIPT support ......................... 513
    4.11.1 xyps-ps.doc ....................................................... 513
    4.11.2 Installation ....................................................... 515
    4.11.3 Extensions ........................................................ 522
    4.11.4 xyps-c.doc ....................................................... 524
    4.11.5 Colour ............................................................... 525
    4.11.6 xyps-f.doc ....................................................... 527
    4.11.7 Frames ............................................................... 528
    4.11.8 xyps-l.doc ....................................................... 532
    4.11.9 Line-styles ........................................................ 532
    4.11.10 xyps-r.doc ..................................................... 535
    4.11.11 Rotations and scaling .......................................... 536
    4.11.12 xyps-t.doc ..................................................... 538
    4.11.13 Patterns and tiles .............................................. 539

5 Fonts .......................................................... 543
  5.1 Semidirectional font ..................................................... 543
    5.1.1 xyd2.mf ............................................................ 543
    5.1.2 xydash10 .......................................................... 544
  5.2 Directional font ........................................................ 546
    5.2.1 xyd.mf ............................................................. 546
    5.2.2 xyatip10 .......................................................... 547
    5.2.3 xybtip10 .......................................................... 549
    5.2.4 xybsql10 ............................................................ 550
  5.3 Special fonts ............................................................ 552
    5.3.1 xycirc10 ........................................................... 552
  5.4 Optional fonts .......................................................... 555
    5.4.1 xycmat10.mf ....................................................... 556
    5.4.2 xycmbt10.mf ....................................................... 556
    5.4.3 xycmat11 ........................................................... 557
    5.4.4 xycmbt11 ........................................................... 558
    5.4.5 xycmat12.mf ....................................................... 559
    5.4.6 xycmbt12.mf ....................................................... 560
    5.4.7 xyeuat10.mf ....................................................... 561
    5.4.8 xyeubt10.mf ....................................................... 561
    5.4.9 xyeuat11 ........................................................... 562
    5.4.10 xyeubt11 .......................................................... 563
CONTENTS

5.4.11 xyeuat12.mf .................................................. 564
5.4.12 xyebt12.mf .................................................. 565
5.4.13 xylu.mf ....................................................... 565
5.4.14 xyatri.mf .................................................... 566
5.4.15 xybtri.mf .................................................... 567
5.4.16 xluat10.mf ................................................... 568
5.4.17 xlb10.mf ...................................................... 568
5.4.18 xluat11 ....................................................... 569
5.4.19 xlb11 ......................................................... 570
5.4.20 xluat12.mf ................................................... 571
5.4.21 xlb12.mf ...................................................... 572

A Answers to all exercises ........................................ 575

B Backwards Compatibility ........................................ 583
  B.1 Version 2 Compatibility .................................... 583
    B.1.1 Unsupported incompatibilities .......................... 584
    B.1.2 Obsolete kernel features .............................. 584
    B.1.3 Obsolete extensions & features ....................... 589
    B.1.4 Obsolete loading ...................................... 593
    B.1.5 Compiling v2-diagrams ................................ 595
  B.2 Obsolete fonts ................................................ 597
    B.2.1 xyline10 ................................................ 597
    B.2.2 xyqc10 .................................................. 600
    B.2.3 xymisc10 ................................................ 602

C Licenses .......................................................... 607
  C.1 GNU General Public License ............................... 607
  C.2 Font General Public License Exceptions .................... 612

D Distribution support files ...................................... 615
  D.1 Hype .......................................................... 615
    D.1.1 CATALOG ............................................... 615
    D.1.2 README ................................................. 615
    D.1.3 TRAILER ............................................... 617
  D.2 Installation instructions .................................... 618
    D.2.1 INSTALL ................................................ 618
  D.3 Generation .................................................. 621
    D.3.1 MAKE ................................................... 621
    D.3.2 Makefile ................................................. 623
Bibliography ........................................................ 640

Index ............................................................... 643
<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>(pos)itions.</td>
<td>24</td>
</tr>
<tr>
<td>1.2</td>
<td>Computing angle vectors.</td>
<td>33</td>
</tr>
<tr>
<td>1.3</td>
<td>All directions.</td>
<td>36</td>
</tr>
<tr>
<td>1.4</td>
<td>Example (place)s.</td>
<td>38</td>
</tr>
<tr>
<td>1.5</td>
<td>(object)s.</td>
<td>47</td>
</tr>
<tr>
<td>1.6</td>
<td>(decor)ations.</td>
<td>66</td>
</tr>
<tr>
<td>1.7</td>
<td>Kernel library (dir)ectionals</td>
<td>73</td>
</tr>
<tr>
<td>1.8</td>
<td>(cir)cles.</td>
<td>86</td>
</tr>
<tr>
<td>2.1</td>
<td>Syntax for curves.</td>
<td>129</td>
</tr>
<tr>
<td>2.2</td>
<td>Plain (frame)s.</td>
<td>204</td>
</tr>
<tr>
<td>2.3</td>
<td>Bracket (frame)s.</td>
<td>204</td>
</tr>
<tr>
<td>2.4</td>
<td>Rotations, scalings, and flips</td>
<td>236</td>
</tr>
<tr>
<td>2.5</td>
<td>Colour names after \texttt{UseCrayolaColors}.</td>
<td>241</td>
</tr>
<tr>
<td>2.6</td>
<td>The 38 standard Macintosh patterns.</td>
<td>249</td>
</tr>
<tr>
<td>2.7</td>
<td>Importing a graphic for labelling.</td>
<td>256</td>
</tr>
<tr>
<td>3.1</td>
<td>(path)s.</td>
<td>308</td>
</tr>
<tr>
<td>3.2</td>
<td>(arrow)s.</td>
<td>472</td>
</tr>
<tr>
<td>3.3</td>
<td>Pasting diagram.</td>
<td>472</td>
</tr>
<tr>
<td>3.4</td>
<td>(twocell)s.</td>
<td>473</td>
</tr>
<tr>
<td>3.5</td>
<td>(graph)s.</td>
<td>474</td>
</tr>
<tr>
<td>3.6</td>
<td>Trigonometry tables for Polygon vertices.</td>
<td>474</td>
</tr>
<tr>
<td>3.7</td>
<td>(knot-piece) construction set.</td>
<td>475</td>
</tr>
<tr>
<td>3.8</td>
<td>Knot crossings with orientations and label positions.</td>
<td>476</td>
</tr>
<tr>
<td>3.9</td>
<td>Knot joins, with orientations, labels, and shifts.</td>
<td>476</td>
</tr>
<tr>
<td>5.1</td>
<td>Font table for \texttt{xydash10 scaled 2000}.</td>
<td>545</td>
</tr>
<tr>
<td>5.2</td>
<td>Font table for \texttt{xyatip10 scaled 2000}.</td>
<td>548</td>
</tr>
<tr>
<td>5.3</td>
<td>Font table for \texttt{xybtip10 scaled 2000}.</td>
<td>549</td>
</tr>
<tr>
<td>5.4</td>
<td>Font table for \texttt{xybsq10 scaled 2000}.</td>
<td>551</td>
</tr>
<tr>
<td>5.5</td>
<td>Font table for \texttt{xcirc10}.</td>
<td>553</td>
</tr>
<tr>
<td>5.6</td>
<td>Font table for \texttt{ycmat11 scaled 2000}.</td>
<td>558</td>
</tr>
<tr>
<td>5.7</td>
<td>Font table for \texttt{ycmbt11 scaled 2000}.</td>
<td>559</td>
</tr>
<tr>
<td>5.8</td>
<td>Font table for \texttt{xyeumat11 scaled 2000}.</td>
<td>562</td>
</tr>
<tr>
<td>5.9</td>
<td>Font table for \texttt{xyeubt11 scaled 2000}.</td>
<td>564</td>
</tr>
<tr>
<td>5.10</td>
<td>Font table for \texttt{xyluat11 scaled 2000}.</td>
<td>570</td>
</tr>
<tr>
<td>5.11</td>
<td>Font table for \texttt{xylubt11 scaled 2000}.</td>
<td>571</td>
</tr>
<tr>
<td>B.1</td>
<td>Font table for \texttt{xyline10 scaled 2000}.</td>
<td>598</td>
</tr>
</tbody>
</table>
B.2 Font table for xyqc10 scaled 2000. .................................................. 600
B.3 Font table for xymisc10 scaled 578. .................................................. 603
Chapter 1

Kernel: xy.doc

After giving an overview of the X\textgamma{}-pic environment in §1.1, this chapter document the basic concepts of X\textgamma{}-picture construction in §1.2, including the maintained ‘graphic state’. The following sections give the precise syntax rules of the main X\textgamma{}-pic constructions: the position language in §1.3, the object constructions in §1.4, and the picture ‘decorations’ in §1.5. §1.6 presents the kernel repertoire of objects for use in pictures; §1.7 documents the interface to X\textgamma{}-pic options like the standard ‘feature’ and ‘extension’ options.

Section §1.8 documents the more complicated algorithms used to compute directions, edges, and connections.

1.1 The X\textgamma{}-pic implementation

This section briefly discusses the various aspects of the present X\textgamma{}-pic kernel implementation of which the user should be aware.

1.1.1 Loading X\textgamma{}-pic

X\textgamma{}-pic is careful to set up its own environment in order to function with a large variety of formats. For most formats a single line with the command

\begin{verbatim}
\input xy
\end{verbatim}

in the preamble of a document file should load the kernel (see ‘integration with standard formats’ below for variations possible with certain formats, in particular \LaTeX{} [10]).

The rest of this section describes things you need to consider if you need to use X\textgamma{}-pic together with other macro packages, style options, or formats. The less your environment deviates from plain \TeX{} the easier it should be.

**File header:** Here is what actually happens in the header of xy.doc. It contains the copyright message, protection against loading the file more than once, and then bootstrap code to handle category codes and the \texttt{DOCMODE} format—we explain each separately below:

1  \%
2  $\Id: xy.doc,v 3.35 2013/10/06 01:14:17 krisrose Exp$
3  
4  \%
5  
6  \%
7  \% Basic Xy-pictures: Xy-pic bootstrap and kernel macros.
8  \% Copyright (c) 1991-2011 Kristoffer H. Rose <krisrose@tug.org>
9  
10  \% This file is part of the Xy-pic package for graphs and diagrams in \TeX{}.
%% Copyright (c) 1991-2011 Kristoffer H. Rose <krisrose@tug.org>
%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, see http://www.gnu.org/licenses/.
%%
\ifx\xyloaded\undefined\else\message{not reloaded}\endinput\fi
\let\xyloaded=\relax

% NOTE: Apart from the actual macros (as also found in xy.tex), this file
% contains both the Xy-pic kernel reference manual and TeXnical documentation.
% See xyrefer.man and xysource.man for how to typeset this information.
\message{Bootstrap'ing'}
{\catcode96 12\catcode`\#6\catcode`\.12\catcode`\:12\catcode`\'12\catcode`\@11
\ifx\xywarnifdefined\undefined\else \immediate\write16{\%}
\immediate\write16{Xy-pic Warning: \string\xywarnifdefined\space redefined.\%}
\immediate\write16{\%}
\ifx\xywarnifdefined\undefined\else \immediate\write16{\%}
\immediate\write16{Xy-pic Warning: \string\xywarnifdefined\space redefined.\%}
\immediate\write16{\%}
\gdef\xywarnifdefined#1{\ifx#1\undefined\else \immediate\write16{\%}
\immediate\write16{Xy-pic Warning: \string#1\space redefined.\%}
\immediate\write16{\%}
\gdef\xydef@\gdef\xydef@#1{\xywarnifdefined#1\gdef#1}
\xywarnifdefined\xylet@\gdef\xylet@#1{\xywarnifdefined#1\global\let#1}
\xywarnifdefined\xynew@\gdef\xynew@#1{#1\global\let\xywarnifdefined\xynew@}
\gdef\xywarnifdefined\xynew@#1\xynew@#1\xywarnifdefined\xynew@\xynew@#1[\xywarnifdefined\xynew@#1\csname new#1\endcsname\xynew@#1]}
\message{catcodes,string',}
\xywarnifdefined\xyuncatcodes
\xywarnifdefined\xyreuncatcodes \def\xyreuncatcodes{\edef\xyreuncatcodes{\%}
\catcode92 0 \catcode123 1 \catcode125 2 \catcode37 14
\catcode9 \the\catcode9 \catcode10 \the\catcode10 \catcode12 \the\catcode12
\catcode35 \the\catcode35 \catcode36 \the\catcode36 \catcode38 \the\catcode38
\catcode43 \the\catcode43 \catcode45 \the\catcode45 \catcode46 \the\catcode46
\catcode47 \the\catcode47
\catcode60 \the\catcode60 \catcode61 \the\catcode61 \catcode62 \the\catcode62
\catcode64 \the\catcode64 \catcode96 \the\catcode96
\newlinechar \the\newlinechar \newlinechar \the\newlinechar }
\xyreuncatcodes
\xywarnifdefined\xycatcodes \def\xycatcodes{\%}
\catcode9 10
\catcode35 6 \catcode36 3 \catcode38 4
Privacy: Xy-pic will warn about control sequences it redefines—thus you can be sure that there are no conflicts between Xy-pic-defined control sequences, those of your format, and other macros, provided you load Xy-pic last and get no warning messages like

Xy-pic Warning: ‘...’ redefined.

In general the Xy-pic kernel will check all control sequences it redefines except that (1) generic temporaries like \next are not checked, (2) predefined font identifiers (see §1.1.3) are assumed intentionally preloaded, and (3) some of the more exotic control sequence names used internally (like @{-}) are only checked to be different from \relax.

This is handled by \xywarnifdefined—at we have ensured that it is unique itself\footnote{This may seem paranoid but in fact many inconveniences in the \TeX world stem from the fact that somebody copied somebody else’s definition of, say, \xywarnifdefined, modified it, and then used it in something that somehow got distributed! The ‘flat name space’ problem remains \TeX largest problem as a programming language in this \TeX hackers opinion.}. \xydef @, \xylet @, and \xynew@⟨type⟩ are abbreviations used to this end throughout Xy-pic instead of \let, \def, and the \new⟨type⟩ commands.
Next some auxilliaries: \texttt{\textbackslash xydefcsname\texttt{@}} is similar to \texttt{\textbackslash xydef\texttt{@}} except that it builds the control sequence with \texttt{\textbackslash csname ... \textbackslash endcsname} which means that it is \texttt{\textbackslash relax} when undefined—there is thus no way to prevent redefinition of control sequences bound to \texttt{\textbackslash relax} ⌢.

\begin{verbatim}
\xydef@\xydefcsname@#1{\DN@{#1}\DNii@##1{%
\ifx ##1\relax\else \xywarning@{\string\textbackslash ##1\string' redefined}\fi
\def##1}%
\expandafter\nextii@\csname\codeof\next@\endcsname}
\end{verbatim}

\texttt{\textbackslash xyletcsnamecsname@\texttt{\textbackslash let}} one weird control sequence be the same as another using several \texttt{\expandafter}s:

\begin{verbatim}
\xydef@\xyletcsnamecsname@#1#2{\def\1{#1}\def\2{#2}\DN@##1##2{%
\ifx ##1\relax\else \xywarning@{\string\textbackslash ##1\string' redefined}\fi
\let##1=##2}%
\expandafter\expandafter\expandafter\next@\expandafter\csname\expandafter\codeof\expandafter\1\expandafter\endcsname\csname\codeof\2\endcsname}
\end{verbatim}

Finally \texttt{\textbackslash codeof}: a useful hack used to allow any characters in control sequences: \texttt{\codeof\texttt{\textbackslash}} expands to the characters of the control sequence \langle \texttt{cs} \rangle as a string of ‘other’ characters, \textit{i.e.}, all of category 12 and with a \texttt{\textbackslash }12 after every control sequence. The \langle \texttt{cs} \rangle must be a macro or it blows up.

\begin{verbatim}
\xywarnifdefined\codeof
\xywarnifdefined\codeof@
{\catcode`\:=12 % to ensure that all of :-} and <-: are other :-)
\gdef\codeof#1{\expandafter\codeof@\meaning#1<-:}
\gdef\codeof@#1:-{#2}
\end{verbatim}

\textbf{Category codes:} The situation is complicated by the flexibility of \texttt{T\textbackslash EX}'s input format. The culprit is the ‘category code’ concept of \texttt{T\textbackslash EX} (\textit{cf.} [6, p.37]): when loaded \texttt{XY-pic} requires the characters \texttt{\{}} (the first is a space) to have their standard meaning and all other printable characters to have the \texttt{same category as when XY-pic will be used}—in particular this means that (1) you should surround the loading of \texttt{XY-pic} with \texttt{\makeatother \ldots \makeatletter} when loading it from within a \texttt{\textbackslash AT\textbackslash EX} package, and that (2) \texttt{XY-pic} should be loaded after files that change category codes like the \texttt{german.sty} that makes " active. Some styles require that you reset the catcodes for every diagram, \textit{e.g.}, with \texttt{french.sty} you should use the command \texttt{\english} before every \texttt{\xymatrix}.

We define \texttt{\textbackslash xyuncatcodes} to restore the current catcodes, and \texttt{\xycatcodes} to install our own.

Here is an exact list of the category codes which \texttt{XY-pic} requires (all standard in plain \texttt{T\textbackslash EX}):

\begin{table}[h]
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline
character(s) & \{} & \} & \texttt{\textbackslash } & \texttt{\textbackslash CR} & \texttt{\textbackslash TAB} & \texttt{\textbackslash SP} & \texttt{A-Z} & \texttt{a-z} & 0-9 & \%
\hline
\hline
\texttt{\textbackslash codeof} & 0 & 1 & 2 & 5 & 10 & 11 & 12 & 14
\hline
\hline
\end{tabular}
\end{table}

Furthermore none of the remaining printable ASCII characters

\begin{verbatim}
!'"#$'()*+,-./:;<>@[]?^_`{|}~
\end{verbatim}

may be of category 0, 1, 2, 9, 14, or 15, because all should be tokens allowed in the replacement text of a \texttt{\textbackslash def}—this also means that they may not be active characters defined to be "\texttt{\textbackslash outer}!"

All other catcodes needed are established using \texttt{\xycatcodes} defined above—this is the reason the macros must be loaded at a time where the category codes are stable (otherwise it will make them stable!).
1.1. THE XY-PIC IMPLEMENTATION

Internally XY-pic enforces the following category codes:

<table>
<thead>
<tr>
<th>character</th>
<th>#</th>
<th>$</th>
<th>&amp;</th>
<th>'</th>
<th>+</th>
<th>-</th>
<th>.</th>
<th>&lt;</th>
<th>=</th>
<th>&gt;</th>
<th>@</th>
<th>`</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCII code</td>
<td>35</td>
<td>36</td>
<td>38</td>
<td>39</td>
<td>43</td>
<td>45</td>
<td>46</td>
<td>60</td>
<td>61</td>
<td>62</td>
<td>64</td>
<td>96</td>
</tr>
<tr>
<td>category code</td>
<td>6</td>
<td>3</td>
<td>4</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>abbreviation</td>
<td>HASH</td>
<td>DOLL</td>
<td>AND</td>
<td>RQ</td>
<td>PLUS</td>
<td>DASH</td>
<td>DOT</td>
<td>LT</td>
<td>EQ</td>
<td>GT</td>
<td>AT</td>
<td>LQ</td>
</tr>
</tbody>
</table>

with special control sequences named \add(abbreviation)@ that take an argument and expand to it followed by the original character token, i.e., many tests throughout the program look like \addDOT@\ifx \next ...

However, it is possible to 'repair' the problem in case any of the characters #$&’+-.<>=‘ change category code:

\xyresetcatcodes

will load the file xyrecat.tex (version 3.7) to do it.
The Xy-pic package is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by the
Free Software Foundation; either version 2 of the License, or (at your
option) any later version.

The Xy-pic package is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License
for more details.

You should have received a copy of the GNU General Public License along
with this package; if not, see http://www.gnu.org/licenses/.

Header: The command is defined in xy.doc: the purpose of this file is to be reloadable such that
the characters can be reread by TeX with fresh catcodes.

The code redefines the special \add...@ control sequences used for parsing of just those characters,
assuming \next is defined correctly before loading it:

\xyuncatcodes
\xyrecat @ 64 \catcode 64 11
\xymakeADD@\addHASH@ # 35
\xymakeADD@\addDOLL@ $ 36
\xymakeADD@\addAND@ & 38
\xymakeADD@\addRQ@ ' 39
\xymakeADD@\addPLUS@ + 43
\xymakeADD@\addDASH@ - 45
\xymakeADD@\addDOT@ . 46
\xymakeADD@\addLT@ < 60
\xymakeADD@\addEQ@ = 61
\xymakeADD@\addGT@ > 62
\xymakeADD@\addLQ@ ` 96
\endinput
The last block of the \texttt{Xy-pic} header bootstraps the “\texttt{DOCMODE format}” used in \texttt{.doc} variants of \texttt{Xy-pic} macro files in order to keep documentation and macros together in a literal programming style (this is redundant in the \texttt{xy.tex} macro file where all instances of \texttt{DOCMODE} have been eliminated (see chapter §D.3.2 for how this is accomplished) but it is included anyway since users may load options still in \texttt{DOCMODE} format). The details of \texttt{DOCMODE} are described in \texttt{xydoc.sty}, a special \LaTeX{} package used to typeset \texttt{Xy-pic} documentation; please read it if you intend to write \texttt{Xy-pic} options.

**Integration with standard formats**  This is handled by the \texttt{xyidioms.tex} file and the integration as a \LaTeX{} [10] package by \texttt{xy.sty}.

We input \texttt{xyidioms.tex} from the kernel:

\begin{verbatim}
\input xyidioms
\end{verbatim}

\texttt{xyidioms.doc}:  This included file provides some common idioms whose definition depends on the used format such that \texttt{Xy-pic} can use predefined dimension registers etc. and yet still be independent of the format under which it is used. The current version (3.7) handles plain \TeX{} (version 2 and 3 [6]), \texttt{AMS-\TeX{}} (version 2.0 and 2.1 [18]), \LaTeX{} (version 2.09 [9] and 2ε [10]), \texttt{AMS-\LaTeX{}} (version 1.0, 1.1 [2], and 1.2), and eplain (version 2.6 [3])\footnote{The ‘v2’ feature introduces some name conflicts, in order to maintain compatibility with earlier versions of \texttt{Xy-pic}.}.

\begin{verbatim}
%% $Id: xyidioms.doc,v 3.7 2011/03/14 20:14:00 krisrose Exp $
%%
%% Format-dependent idioms for Xy-pic.
%% Copyright (c) 1991-1997 Kristoffer H. Rose <krisrose@tug.org>
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-2011 Kristoffer H. Rose <krisrose@tug.org>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License
\end{verbatim}
CHAPTER 1. KERNEL: XY.DOC

You should have received a copy of the GNU General Public License along
with this package; if not, see http://www.gnu.org/licenses/.

\ifx\xyidiomsloaded\empty \message{not reloaded}\endinput \fi
\let\xyidiomsloaded=\empty

Header: This is a separate file so that we can abort the definitions easily using \endinput.

Scratch registers: All the formats currently treated allow the use of the following plain TeX scratch
register conventions (cf. [6, p.346])

- \count@ is available as a local scratch counter register.
- \dimen@, \dimen@i, \dimen@ii, \dimen@iii, \dimen@iv to \dimen@ix, and \dimen@x to \dimen@x
  are available as scratch dimension registers.
- \skip@, \skip@ to \skip@, and \skip@ to \skip@ are available as scratch skip registers.
- \toks@ and \toks@ to \toks@ are available as scratch token lists.
- \box@ and \box@ to \box@ are available as scratch box registers.

with the constraint that \global should never (always) be used when assigning to number 0, 2, 4, 6, 8, and 255 (1, 3, 5, 7, and 9).

\textbf{XY-specific scratch registers:} We first give new names to dimension registers 4, 6, and 8, since
none of the formats do that: \A@, \B@, and \R@ are specific to XY-pic and will be referred to as \textit{A}, \textit{B},
and \textit{R} in comments.

\begin{verbatim}
\xywarnifdefined\A@ \dimendef\A@=4
\xywarnifdefined\B@ \dimendef\B@=6
\xywarnifdefined\R@ \dimendef\R@=8
\end{verbatim}

Some shared scratch registers are defined last.

Idioms: First check that \undefined is indeed undefined...or rather: the same as an extremely
unlikely control sequence that we are making up:

\begin{verbatim}
\ifx\undefined\AveryUNLIKELYc@ntr@lSEQUENCE@@\else
  \errmessage{Xy-pic Error: \string\undefined\space defined.}\fi
\end{verbatim}

These idioms are so common they are just conditionally defined:

\begin{verbatim}
\ifx\undefined\literal@ \def\literal@#1{#1}\fi
\ifx\undefined\eat@ \def\eat@#1{}\fi
\ifx\undefined\xyFN@ \def\xyFN@{\futurelet\next} % Now private because of AMS-LaTeX change.
\ifx\undefined\DN@ \def\DN@{\def\next@}\fi
\ifx\undefined\DNii@ \def\DNii@{\def\nextii@}\fi
\ifx\undefined\setboxz@h \def\setboxz@h{\setbox\z@\hbox}\fi
\ifx\undefined\wdz@ \def\wdz@{\wd\z@}\fi
\ifx\undefined\boxz@ \def\boxz@{\box\z@}\fi
\ifx\undefined\W@ \def\W@{\immediate\write16 }\fi
\end{verbatim}
Delayed setup: This is used when it is advantageous to delay loading of something until after the preamble, i.e., after all options, etc., have been loaded. This is particularly true for nested \nxywithoption uses that will otherwise be executed repeatedly. Currently properly supported with \LaTeX\ 2\epsilon and the amsppt style of AMS-\TeX.

Similarly, the following provides a place to insert ‘trailing messages’ if the format supports it; otherwise it just throws away the argument (useful for repeating the most important warnings).

Shared scratch registers: Everything else in this file is also done by AMS-\TeX so we exit here when using that format after renaming their \toks@@ to \toks@ii...
\xynew@{count}\count@@
\xynew@{count}\count@@@
\else
\xylet\count@@=\@tempcnta
\xylet\count@@@=\@tempcntb
\fi

Finally the second scratch token register not available with \LaTeX.

\ifx\undefined\toks@ii \toksdef\toks@ii=2 \fi

\endinput

\% $Log: xyidioms.doc,v $ 
\% Revision 3.7  2011/03/14  20:14:00  krisrose
\% Preparing for release 3.8.6.
\% 
\% Revision 3.6  2010/06/10  18:45:50  krisrose
\% Reference to GPL by URL.
\% 
\% Revision 3.5  2010/04/16  06:06:52  krisrose
\% Preparing for a new release...
\% 
\% Revision 3.4  1997/05/18  01:14:25  krisrose
\% Essential bugfixes.
\% 
\% Revision 3.3  1996/12/19  03:31:56  krisrose
\% Maintenance release
\% 
\% Revision 3.1  1995/09/05  20:31:32  kris
\% Releasing!
\% 
\% Revision 3.0  1995/07/07  20:14:21  kris
\% Major release w/new User’s Guide!
\% 
\% Revision 2.13  1995/07/04  15:11:17  kris
\% Ready to release v3?
\% 
\% Revision 2.12  1994/10/25  11:34:25  kris
\% Interim release just before v3 [works with AMS-LaTeX 1.2]...
\% 
\% Revision 2.11  1994/07/05  10:37:32  kris
\% Third 3beta release [bug fixes].
\% Experimental graph feature included (for ECCT-94 presentation).
\% 
\% Revision 2.9  1994/06/09  15:02:49  kris
\% Release 3beta.
\% 
\% Revision 2.8  1994/04/08  04:30:00  kris

\textbf{End & log:} That’s all.
1.1. THE XY-PIC IMPLEMENTATION

%% $Id: xy.sty,v 3.10 2011/03/14 20:14:00 krisrose Exp $
%%
%% "Xy-pic as LaTeX 2.09 style option and LaTeX 2e package".
%% Copyright (c) 1993-1996 Kristoffer H. Rose <krisrose@tug.org>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.

\usepackage[⟨option⟩,...]{xy}

where the ⟨option⟩s will be interpreted as if passed to \xyoption (cf. §1.7).

The only exceptions to this are the options having the same names as those driver package options of chapter 4, which appear in cf. [4, table 11.2, p.317] or the \LaTeX\2e \texttt{graphics} bundle. These will automatically invoke any backend extension required to best emulate the \LaTeX\2e \texttt{behaviour. (This means that, e.g., \texttt{dvips} and \texttt{textures} can be used as options to the \texttt{documentclass} command, with the normal effect.)}

The file also works as a \LaTeX\ 2.09 [9] ‘style option’ although you will then have to load options with the \xyoption mechanism described in §1.7.

Here is the raw source of \texttt{xy.sty}.

\texttt{xy.sty:} If you use \LaTeX then this file makes it possible to load \textsc{Xy-pic} as a ‘package’ using the \texttt{\LaTeX\2e \texttt{usepackage}} command:
\% Copyright (c) 1991-2011 Kristoffer H. Rose <krisrose@tug.org>
\%
\% The Xy-pic package is free software; you can redistribute it and/or modify
\% it under the terms of the GNU General Public License as published by the
\% Free Software Foundation; either version 2 of the License, or (at your
\% option) any later version.
\%
\% The Xy-pic package is distributed in the hope that it will be useful, but
\% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
\% or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License
\% for more details.
\%
\% You should have received a copy of the GNU General Public License along
\% with this package; if not, see http://www.gnu.org/licenses/.
\%
\% Load Xy-pic with catcode of @ set as it will be in user text:
\%\count255=\the\catcode`@ {\catcode`@=11 \xdef\xystycatcode{\the\count255}}
\makeatletter
% REPAIR bug of the 2e version of AMS-LaTeX where \makeatother is broken...
\ifx\@ifpackageloaded\undefined \makeatother \else \if\@ifpackageloaded{amstex}{\catcode`@=\xystycatcode} \fi \fi
\input xy
% That is all unless this is called as a LaTeX2e 'native mode' package. Is
% there an official way to check this? Anyway, this seems to work:
\makeatletter
\if\@compatibility\undefined \catcode`@=\xystycatcode \endinput \fi
\input xy
% Ensure that we are *really* running LaTeX2e...
\NeedsTeXFormat{LaTeX2e}
% This file provides the current version of Xy-pic!
\edef\next{%
\noexpand\ProvidesPackage{xy}[\xydate\space Xy-pic version \xyversion]\next
% Recognise some standard LaTeX2e class options (cf. old 'LaTeX Companion' p.317):
\DeclareOption{cmactex}{\xyoption{dvips}} \% Thomas Kiffe's CMacTeX uses dvips
\DeclareOption{dvips}{\xyoption{dvips}\xyoption{ps}}
\DeclareOption{dvitops}{\xyoption{dvitops}\xyoption{ps}}
\DeclareOption{emtex}{\xyoption{emtex}}
\DeclareOption{ln}{\xywarning@{`ln' DVI driver not yet fully supported}}
\DeclareOption{oztex}{\xyoption{oztex}}
\DeclareOption{textures}{\xyoption{textures}}
\DeclareOption{xdvi}{\xyoption{xdvi}}
% Recognise standard aliases for the pdf option.
\DeclareOption{pdftex}{\xyoption{pdf}}
\DeclareOption{dvipdfm}{\xyoption{pdf}}
\DeclareOption{dvipdfmx}{\xyoption{pdf}}
% This is special.
\DeclareOption{colour}{\xyoption{color}} % :-)
% This (obsolete) cmtip style option should be activated after loading:
\DeclareOption{cmtip}{\xyoption{cmtip}\UseComputerModernTips}
% Pass the desired tip size if defined
\DeclareOption{10pt}{\xywithoption{tips}{\def\tipsize@@{10}}}
\DeclareOption{11pt}{\xywithoption{tips}{\def\tipsize@@{11}}}
\DeclareOption{12pt}{\xywithoption{tips}{\def\tipsize@@{12}}}
% Other options are just loaded.
\DeclareOption*{\edef\next{\noexpand\xyoption{\CurrentOption}}\next}
% Reinstall catcode of @...
\catcode`@=\xystycatcode
% ...and process the options.
\ProcessOptions\relax
% Now, if no backend is loaded but we can generate PDF, do.
\ifpdf\makeatletter
  \ifx\xydriversloaded@@\xydriversloaded@none
    \makeatother\xyoption{pdf}\fi\fi
% That's all.
\endinput
% $Log: xy.sty,v $
% Revision 3.10  2011/03/14 20:14:00  krisrose
% Preparing for release 3.8.6.
%
% Revision 3.9  2010/06/10 18:45:49  krisrose
% Reference to GPL by URL.
%
% Revision 3.8  2010/04/28 07:14:30  krisrose
% New Xy-pic home page installed.
%
% Revision 3.7  2010/04/26 01:45:23  krisrose
% First proper integration of xypdf into Xy-pic "make dist".
%
% Revision 3.6  2010/04/25 21:48:06  krisrose
% First proper integration of xypdf into Xy-pic "make dist".
%
% Revision 3.5  2010/04/20 20:36:43  krisrose
% Documentation updates.
%
% Revision 3.4  2010/04/16 06:06:52  krisrose
% Preparing for a new release...
%
% Revision 3.3  1996/12/19 03:31:56  krisrose
% Maintenance release
%
% Revision 3.1  1995/09/05 20:31:32  kris
% Releasing!
1.1.2 Logo, version, and messages

Loading \texttt{XY-pic} prints a banner containing the version and author of the kernel; small progress messages are printed when each major division of the kernel has been loaded. Any options loaded will announce themself in a similar fashion.

Of these, \texttt{stripRCS} is a very useful hack for extracting the first component of an RCS $\ldots$ keyword value.

\begin{verbatim}
\xydef\stripRCS$#1${\stripRCS@#1: @@ @@@}
\xydef\stripRCS@#1: #2@ #3@@@{% 
  \ifx @#2\string?\else\ifx :#2\else\stripRCS@@#2\fi\fi}
\xydef\stripRCS@@#1 #2: @{#1}
\xydef\xyversion{3.8.9} % Must be in sync with *VERSION in Makefile.
\edef\next{\stripRCS\Date: 2013/10/06 01:14:17 $}
\xylet\xydate=\next
\xdef\Xygreet@{%
  \W@{}%
  \W@{ Xy-pic version \xyversion\space<\xydate>}%
  \W@{ Copyright (c) 1991-2013 by Kristoffer H. Rose <krisrose@tug.org> and others}%
  \W@{ Xy-pic is free software: see the User's Guide for details.}%
  \W@{}}
\Xygreet@
\expandafter\everyjob\expandafter{\the\everyjob\Xygreet@}
\message{Loading kernel:}
\end{verbatim}

If you refer to \texttt{XY-pic} in your written text (please do ☺) then you can use the command \texttt{\textbackslash Xy-pic} to typeset the “\texttt{XY-pic}” logo. The version of the kernel is typeset by \texttt{\textbackslash xyversion} and the release date by \texttt{\textbackslash xydate} (as found in the banner). By the way, the \texttt{XY-pic} name\footnote{No description of a \TeX program is complete without an explanation of its name.} originates from the fact that the first
version was little more than support for \((x, y)\) coordinates in a configurabe coordinate system where the main idea was that all operations could be specified in a manner independent of the orientation of the coordinates. This property has been maintained except that now the package allows explicit absolute orientation as well.

\[
\text{\texttt{\leavevmode}}\quad \text{\texttt{\hbox{\kern-.1em X\kern-.3em\lower.4ex\hbox{\kern-.15em}Y}}}\]

Messages that start with \textit{"Xy-pic Warning"} are indications that something needs your attention; an \textit{"Xy-pic Error"} will stop \TeX{} because Xy-pic does not know how to proceed.

We use the input line number if available and rudimentary help in the form of a reference to the manual if no specific help string is given. \texttt{\newlinechar} is set locally to \texttt{^^J} while writing such that messages of several lines can be written.

\[
\text{\texttt{\message{messages;}}}\]

\texttt{xywarnifdefined\edef\thelineno@{\string？}}

\[
\text{\texttt{\xydef@{\string[]jobname:\thelineno@}}}
\]

\[
\text{\texttt{\xydef@{\string[\jobname:\thelineno@\string]}}}\]

\[
\text{\texttt{\xydef@{\xywarning@#1{{\newlinechar=10 %}}}}}
\]

\[
\text{\texttt{\xydef@{\xytracelineno@{\string[\jobname:\thelineno@\string]}}}\xytracelineno@{}\xytracelineno@{}}\]

\[
\text{\texttt{\xycircfont}}\quad 1/8\quad \text{circle segments} \quad \xycirc10
\]

The first four contain variations of characters in a large number of directions, the last contains 1/8 circle segments.
**Note:** The default fonts are not part of the \textit{XY-pic} kernel \textit{specification}: they just set a standard for what drawing capabilities should at least be required by an \textit{XY-pic} implementation. Implementations exploiting capabilities of particular output devices are in use. Hence the fonts are only loaded by \textit{XY-pic} if the control sequence names are undefined—this is used to preload them at different sizes or prevent them from being loaded at all.

To be more precise, \textit{XY-pic} requires \texttt{\textbackslash xydashfont} to be a \textit{semidirectional} font as \textit{METAFONT} will generate with the driver file \texttt{xyd2.mf}—this is very important because the \textit{italic corrections} of the characters in this particular font are used to approximate trigonometric computations, so if you replace \texttt{\textbackslash xydashfont} be sure to replace it with another semidirectional font! Similarly, the three fonts \texttt{\textbackslash yatipfont}, \texttt{\textbackslash btipfont}, and \texttt{\textbackslash bsqllfont} should be \textit{directional} as \textit{METAFONT} will generate with the driver file \texttt{xyd.mf}.

Finally, \texttt{\textbackslash xycircfont} should contain 1/8 circle segments of various radii as described in \texttt{xycirc10.mf}.

The following code loads the fonts \textit{unless it was already loaded} and defines some associated dimensions for \texttt{\textbackslash xydashfont} and \texttt{\textbackslash xybsqllfont}: for each of these \( f \) we define \( f_L \) as the length of a unit in the current direction (used when juxtaposing for connections), \( f_h \) as the height of the unit (used for several parallel connections), and \( f_w \) as the ‘line width’ of the unit (to know how to interface to rules).

\begin{verbatim}
557 \message{fonts;} \\
559 \xydef\xyfont@#1{\ifx#1\undefined \DN@{\global\font#1}\expandafter\next@ \\
560 \else \xywarning@{Using previously loaded \string#1\space font}\fi} \\
562 \xyfont@\xydashfont=xydash10 \\
563 \xydef\xydashl@{\fontdimen6\xydashfont} \\
564 \xydef\xydashh@{\fontdimen5\xydashfont} \\
565 \xydef\xydashw@{\fontdimen8\xydashfont} \\
567 \xyfont@\yatipfont=yatip10 \\
568 \xyfont@\btipfont=btip10 \\
570 \xyfont@\xybsqllfont=xybsqll0 \\
571 \xydef\xybsqllh@{\fontdimen6\xybsqllfont} \\
572 \xydef\xybsqllh@{\fontdimen5\xybsqllfont} \\
574 \xydef\xybsqlw@{\fontdimen8\xybsqlwfont} \\
575 \xyfont@\xycircfont=xycirc10
\end{verbatim}

### 1.1.4 Allocations

One final thing that you must be aware of is that \textit{XY-pic} allocates a significant number of dimension registers and some counters, token registers, and box registers, in order to represent the state and do computations. The current kernel allocates 4 counters, 28 dimensions, 2 box registers, 4 token registers, 1 read channel, and 1 write channel (when running under \texttt{\textbackslash LaTeX}; some other formats use slightly more because standard generic temporaries are used). Options may allocate further registers (currently loading \textit{everything} loads 6 dimen-, 3 toks-, 1 box-, and 9 count-registers in addition to the kernel ones).

\begin{verbatim}
598 \message{allocations:} \\
\end{verbatim}

See §1.1.1 for scratch register allocations.

**Picture state:** These realise the picture state as described in §1.2.5: \( c, p \), the \textit{base}, and the picture size:

\begin{verbatim}
609 \message{state,}
\end{verbatim}
1.1. THE XY-PIC IMPLEMENTATION

\xynew@{dimen}\X@c
\xynew@{dimen}\Y@c
\xynew@{dimen}\U@c
\xynew@{dimen}\D@c
\xynew@{dimen}\L@c
\xynew@{dimen}\R@c
\xynew@{toks}\Edge@c
\xynew@{dimen}\X@p
\xynew@{dimen}\Y@p
\xynew@{dimen}\U@p
\xynew@{dimen}\D@p
\xynew@{dimen}\L@p
\xynew@{dimen}\R@p
\xynew@{toks}\Edge@p
\xynew@{dimen}\X@origin \X@origin=\z@
\xynew@{dimen}\Y@origin \X@origin=\z@
\xynew@{dimen}\X@xbase \X@xbase=1mm
\xynew@{dimen}\Y@xbase \Y@xbase=\z@
\xynew@{dimen}\X@ybase \X@ybase=\z@
\xynew@{dimen}\Y@ybase \Y@ybase=1mm
\xynew@{dimen}\X@min
\xynew@{dimen}\Y@min
\xynew@{dimen}\X@max
\xynew@{dimen}\Y@max

Drop and connect: \lastobjectbox@ stores the most recently dropped object.

\xynew@{box}\lastobjectbox@
\zerodotbox@ is of zero size with a ‘dot’ in the form of a rule the width and height as the line width of the line font; \zz @ is ‘almost-zero-check’. To Do: overshoot the size a bit as the dots appear very small.

\xynew@{box}\zerodotbox@
\setbox\zerodotbox@=\hbox{\dimen@=.5\xydashw@
\kern-\dimen@ \vrule width\xydashw@ height\dimen@ depth\dimen@}
\wd\zerodotbox@=\z@ \ht\zerodotbox@=\z@ \dp\zerodotbox@=\z@
\xynew@{dimen}\almostz@ \almostz@=50sp
\xydef\zz@#1{\ifdim#1<\z@-\fi#1<\almostz@ \relax}
\xynew@{if}\iftmp@

Direction state: The direction state is rather complicated and described in detail in §1.8.1.

\message{direction,}
\xynew@{dimen}\d@X
\xynew@{dimen}\d@Y
\xydef@s@X{}
\xydef@s@Y{}
\xynew@{count}\K@ \K@=1024
CHAPTER 1. KERNEL: XY.DOC

Miscellaneous: Finally some generic allocations used in the following:

\xynew{read}\xyread@ \% for `safe input'
\xynew{write}\xywrite@ \% for `saving' to .xyc file
\xynew{count}\csp@ \% for `control stack pointer'
\xynew{dimen}\quotPTK@ \% for `fractions'

The required temporaries are defined by xyidioms.tex.

1.1.5 Utility macros

Finally we define some utility macros.

\message{utility macros;}

Simple queue: Just appending to the \toks @ list.
\xydef@\addtotoks@ #1\{\toks@=\expandafter{\the\toks@#1}\}

Safe input: Check that file is available before input. Tries the alternate extension .doc in case the .tex file is not there. The second is the lowest level function used when the desired files cannot be preloaded and do not exist in .doc form.
\xydef@\xyinputorelse@ #1#2\{% 
\expandafter\let\expandafter\next@\csname#1loaded\endcsname 
\ifx\next@\empty \DN@{\xyinputorelse@@{#1}{\xyinputorelse@@{#1.doc}{#2}}}\% 
\fi \next@\}
\xydef@\xyinputorelse@@ #1#2\{\openin\xyread@=#1 % 
\ifeof\xyread@ \DN@{#2}\else \DN@{\closein\xyread@\input#1 }\fi \next@\}

Continuation stack: This is used to `enter’ a new context and `leave’ to the previous context. It works as a stack defining a control sequence for each level, thus using a counter as the stack pointer. Defines the following

\csp@ \% `Continuation Stack Pointer'
\enter@{(code)} \% Enter new block with ⟨code⟩ expanded as continuation
\enter@{(code)} \% Enter new block with ⟨code⟩ as continuation
\dontleave@ \% Execute continuation without leaving block
\unenter@ \% Leave block without executing its continuation
1.1. THE \textsc{xy-Pic} Implementation

\leave@ Leave block (execute its continuation)

So \texttt{\enter@\{}\leave@ is a noop and \leave@ is the same as \texttt{\dontleave@\unenter@}.

\global\csp@=\z@

\texttt{xdef@\enter@\#1{\global\advance\csp@\@ne}
\expandafter\xdef\csname cs@\number\csp@\endcsname{#1}\ignorespaces}

\texttt{xdef@\nter@\#1{\global\advance\csp@\@ne}
\expandafter\gdef\csname cs@\number\csp@\endcsname{#1}\ignorespaces}

\texttt{xdef@\dontleave@{\csname cs@\number\csp@\endcsname}
\xydef@\unenter@{\global\advance\csp@\@ne}
\xydef@\leave@{\expandafter\unenter@\csname cs@\number\csp@\endcsname}

\textbf{Fractions:} Below we often use a factor on the form of a quotient $A/B$. Here is a hack to get it; it is not very precise but suffices for our needs.

\quotient@ (cs) \{A\} \{B\} Defines (cs) to expand (immediately) to the factor corresponding to $A/B$; $A$, $B$ must be dimensions where $|A| < \text{\maxdimen}/\KK$ and $|B| > \KK$

\quotient@@ (cs) \{A\} \{B\} Same, but uses $8\KK$ for $\KK$.

\textbf{Notes:} (1) If $c$ is a count register, then \{1\}$c$ is a legal dimension. (2) Really computes

\[
((A \times |KK|)/(B/|KK|)) \times (ipt/|K|)
\]

and then defines (cs) to expand to the resulting pt value. This means that results are only reasonable for $|A| << \text{\maxdimen}/\KK$ and $|B| \gg \KK$.

\texttt{\quotPTK@=\p@ \divide\quotPTK@\K@
\xylet@\quotsign@@=\empty
\xywarnifdefined\removePT@
\{\catcode`p=12 \catcode`t=12 \gdef\removePT@#1pt{#1}\}
\xydef@\quotient@#1#2#3{\A@=#2\relax \B@=#3\relax
\ifdim\A@<\z@\def\quotsign@@{-}\else\def\quotsign@@{+}\fi
\ifdim\quotsign@@\A@<15pt \multiply\A@51pt \multiply\A@\KK@
\else\ifdim\quotsign@@\A@<51pt \multiply\A@\KK@
\advance\B@\ifdim\B@<\z@-\fi 16\text{sp} \divide\B@\KK@
\else
\advance\B@\ifdim\B@<\z@-\fi 512\text{sp} \divide\B@\KK@
\fi
\fi
\ifdim\quotient@#1#2#3\{\A@=#2\relax \B@=#3\relax \multiply\A@\quotPTK@ \edef#1{\expandafter\removePT@\the\A@}}
\fi
\ifdim\quotient@#1#2#3\{\A@=#2\relax \B@=#3\relax \multiply\A@\quotPTK@ \edef#1{\expandafter\removePT@\the\A@}}
\fi
\ifdim\quotient@#1#2#3\{\A@=#2\relax \B@=#3\relax \multiply\A@\quotPTK@ \edef#1{\expandafter\removePT@\the\A@}}
\fi
Loops: XY-pic uses its own \loop@ to avoid interference with plain \loop.

```
\xydef@\loop@\#1\repeat@{(\def\body@\#1\iterate@)\xylset@\repeat@=}\fi
\xydef@\iterate@{(\body@\expandafter\iterate@\else\fi}
```

Execution: All execution of XY-commands should be ‘indirect’, i.e., execute

```
\xy@{(source)\{\{internal\ commands\}\}}
```

where the \{internal commands\} directly do the desired operation(s). This is used for tracing and can be used to separate parsing and execution by changing \xy@; \oxy@ is kept stable such that \let\xy@=\oxy@ will reestablish a sane state; it should only be changed with \change@oxy@ to make this easy to verify.

```
\xydef@\xyinitial@\#1\#2{\DN@\#1%
\xyerror@{command used out of context: \codeof\next@}{}%}
\xylet@\xy@=\xyinitial@
\xylet@\oxy@=\xy@
\xydef@\change@oxy@\#1{\let\oxy@=\#1\relax
% \W@{*OXY@:=\meaning\oxy@}%
}
```

This is also used to check whether an XY-picture is already active; use as \if\inxy@...\else...\fi:

```
\xydef@\inxy@{T\ifx\xy@\xyinitial@ F\else T\fi}
```

The final execution command is a trick used to put bits of the user’s input inside the \next@ scratch macro with the user’s catcodes intact: \xy@\ix@\{...\} is the same as \xy@\{\global\toks9=\{...\}\} except for the category codes used for the ....

```
\xydef@\xyxy@@ix@\{\begingroup\xyuncatcodes\afterassignment\endgroup\global\toks9=\}
```

This to save some tokens – maybe not worth it:

```
\xydef@\xy@@\{\xy@{}\}
```

Finally this to establish a sane state – only use within a group!

```
\xydef@\plainxy@{\let\xy@=\xyxy@ \change@oxy@\xy@ \let\xy@@ix@=\xyxy@@ix@}
```

To Do: Clean up all uses of these such that \{source\} is always that and only that. Define a method for ‘inner’ aka ‘implied’ \{source\} that doesn’t really count in that it is a consequence of some other source...

### 1.2 Picture basics

The basic concepts involved when constructing XY-pictures are positions and objects, and how they combine to form the state used by the graphic engine.

The general structure of an XY-picture is as follows:

\[
\xy\{\text{pos}\}\{\text{decor}\}\endxy
\]

builds a box with an XY-picture (\LaTeX users may substitute \begin{xy} ... \end{xy} if they prefer). \{pos\} and \{decor\} are components of the special ‘graphic language’ which XY-pictures are specified in. We explain the language components in general terms in this § and in more depth in the following §§.

The code for the \xy...\endxy command is presented last in this section.
1.2. PICTURE BASICS

1.2.1 Positions

All positions may be written \(<X,Y>\) where \(X\) is the \(\TeX\) dimension distance right and \(Y\) the distance up from the zero position 0 of the \(\XY\)-picture (0 has coordinates \(<0\text{mm},0\text{mm}>\), of course). The zero position of the \(\XY\)-picture determines the box produced by the \(\xy\ldots\endxy\) command together with the four parameters \(X_{\text{min}}, X_{\text{max}}, Y_{\text{min}},\) and \(Y_{\text{max}}\) set such that all the objects in the picture are ‘contained’ in the following rectangle:

```
      Y_{\text{max}}
     /      /
    /  \    /
  X_{\text{min}}  0  X_{\text{max}}
   \    \
   \    \
Y_{\text{min}}

```

where the distances follow the “up and right > 0” principle, e.g., the indicated \(\TeX\) reference point has coordinates \(<X_{\text{min}},0\text{pt}>\) within the \(\XY\)-picture. The zero position does not have to be contained in the picture, but \(X_{\text{min}} \leq X_{\text{max}} \wedge Y_{\text{min}} \leq Y_{\text{max}}\) always holds. The possible positions are described in detail in §1.3.

When an \(\XY\)-picture is entered in math mode then the reference point becomes the “vcenter” instead, i.e., we use the point \(<X_{\text{min}},-\text{the fontdimen22}>\) as reference point.

1.2.2 Objects

The simplest form of putting things into the picture is to ‘drop’ an object at a position. An object is like a \(\TeX\) box except that it has a general Edge around its reference point—in particular this has the extents (i.e., it is always contained within) the dimensions \(L, R, U,\) and \(D\) away from the reference point in each of the four directions left, right, up, and down. Objects are encoded in \(\TeX\) boxes using the convention that the \(\TeX\) reference point of an object is at its left edge, thus shifted \(<-L,0\text{pt}>\) from the center—so a \(\TeX\) box may be said to be a rectangular object with \(L = 0\text{pt}\). Here is an example:

```
      0
     /      /
    /  \    /
  L 0  R
   \    \
   \    \
 D

```

The object shown has a rectangle edge but others are available even though the kernel only supports rectangle and circle edges. It is also possible to use entire \(\XY\)-pictures as objects with a rectangle edge, 0 as the reference point, \(L = -X_{\text{min}}, R = X_{\text{max}}, D = -Y_{\text{min}},\) and \(U = Y_{\text{max}}\). The commands for objects are described in §1.4.

1.2.3 Connections

Besides having the ability to be dropped at a position in a picture, all objects may be used to connect the two current objects of the state, i.e., \(p\) and \(c\). For most objects this is done by ‘filling’ the straight line between the centers with as many copies as will fit between the objects:

```
   c
```

The ways the various objects connect are described along with the objects.
1.2.4 Decorations

When the \xy command reaches something that can not be interpreted as a continuation of the position being read, then it is expected to be a decoration, i.e., in a restricted set of \TeX commands which add to pictures. Most such commands are provided by the various user options (cf. §1.7)—only a few are provided within the kernel to facilitate programming of such options (and user macros) as described in §1.5.

1.2.5 The \textsc{Xy}-pic state

Finally we summarise the user-accessible parts of the \textsc{Xy}-picture state of two positions together with the last object associated with each: the previous, \textit{p}, is the position \langle X_p, Y_p \rangle with the object \textit{L}_p, \textit{R}_p, \textit{D}_p, \textit{U}_p, \textit{Edge}_p, and the current, \textit{c}, is the position \langle X_c, Y_c \rangle with the object \textit{L}_c, \textit{R}_c, \textit{D}_c, \textit{U}_c, \textit{Edge}_c.

Furthermore, \textsc{Xy}-pic has a configurable cartesian coordinate system described by an origin position \langle X_{\text{origin}}, Y_{\text{origin}} \rangle and two base vectors \langle X_{\text{base}}, Y_{\text{base}} \rangle and \langle X_{\text{ybase}}, Y_{\text{ybase}} \rangle accessed by the usual notation using parentheses:

\[
(x, y) = \langle X_{\text{origin}} + x \times X_{\text{base}} + y \times Y_{\text{base}}, Y_{\text{origin}} + x \times Y_{\text{base}} + y \times Y_{\text{ybase}} \rangle
\]

This is explained in full when we show how to set the base in note 1.3d of §1.3.

Finally typesetting a connection will setup a “placement state” for referring to positions on the connection that is accessed through a special ? position construction; this is also discussed in detail in §1.3.

The \textsc{Xy}-pic state consists of all these parameters together. They are initialised to zero except for \textit{X}_{\text{base}} = \textit{Y}_{\text{ybase}} = 1\text{mm}.

The edges are are available to the programmer as token lists; see §1.8.2 for details.

**Procedure:** \texttt{\textbackslash xy . . . endxy} builds an object from an \textsc{Xy}-pic \langle pos \rangle \langle decor \rangle sequence as follows:

\begin{itemize}
  \item \texttt{\textbackslash xy} starts the \texttt{\hbox} to contain the \textsc{Xy}-picture,
  \item \texttt{\textbackslash endxy} starts an inner box to be resized appropriately later, sets \texttt{\xy@} to just execute immediately, and makes a fresh scope for global internal names, and
  \item \texttt{\textbackslash \textbackslash} initialises the \textsc{Xy}-pic state (setting the size to a ridiculously large negative value), and finally passes control to the \langle pos \rangle parser.
\end{itemize}

Some care is taken to `lift' the diagram a bit to `vcenter' it when in math mode, and \texttt{\mathsurround} is set to zero to make changes into math mode safe.

\begin{verbatim}
\message{pictures: \string\xy,}
\xydef@\xy{\ifmmode\expandafter\xymath@\else\expandafter\xynomath@\fi}
\xydef@\xymath@{\hbox\bgroup \dimen@=\the\fontdimen22\textfont\tw@ \xyinside@}
\xydef@\xynomath@{\hbox\bgroup \dimen@=\z@ \xyinside@}
\xydef@\xyinside@{%
  \saveXyStyle@ \aftergroup\xycheck@end
  \setboxz@h\bgroup
  \plainxy@
  \X@c=\z@ \Y@c=\z@ \czeroEdge@
  \X@c=\z@ \Y@c=\z@ \D@c=\l@ \L@c=\z@ \R@c=\z@ \Edge@c={\zeroEdge}%%
  \X@cmin=\hsize \X@cmax=-\hsize \Y@cmin=\hsize \Y@cmax=-\hsize
  \mathsurround=\z@ \expandafter\POS\everyxy@@}
\xdef@\czeroEdge@{%\U@c=\z@ \D@c=\U@c \L@c=\U@c \R@c=\U@c \Edge@c={\zeroEdge}}
\end{verbatim}
1.3. Positions

A \langle\text{pos}\rangle\text{ition} is a way of specifying locations as well as dropping objects at them and decorating them—in fact any aspect of the \text{X\text-Y-pic} state can be changed by a \langle\text{pos}\rangle but most will just change the coordinates and/or shape of \(c\).

All possible positions are shown in figure 1.1 with explanatory notes below.
<table>
<thead>
<tr>
<th>Syntax</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>⟨pos⟩ →</td>
<td>⟨coord⟩</td>
</tr>
<tr>
<td>⟨pos⟩ + ⟨coord⟩</td>
<td>c ← ⟨pos⟩ + ⟨coord⟩</td>
</tr>
<tr>
<td>⟨pos⟩ − ⟨coord⟩</td>
<td>c ← ⟨pos⟩ − ⟨coord⟩</td>
</tr>
<tr>
<td>⟨pos⟩ ! ⟨coord⟩</td>
<td>c ← ⟨pos⟩ then skew.c by ⟨coord⟩</td>
</tr>
<tr>
<td>⟨pos⟩ . ⟨coord⟩</td>
<td>c ← ⟨pos⟩ but also covering ⟨coord⟩</td>
</tr>
<tr>
<td>⟨pos⟩ , ⟨coord⟩</td>
<td>c ← ⟨pos⟩ then c ← ⟨coord⟩</td>
</tr>
<tr>
<td>⟨pos⟩ ; ⟨coord⟩</td>
<td>c ← ⟨pos⟩, swap p and c, c ← ⟨coord⟩</td>
</tr>
<tr>
<td>⟨pos⟩ :: ⟨coord⟩</td>
<td>c ← ⟨pos⟩, set base, c ← ⟨coord⟩</td>
</tr>
<tr>
<td>⟨pos⟩ * ⟨object⟩</td>
<td>c ← ⟨pos⟩, drop ⟨object⟩</td>
</tr>
<tr>
<td>⟨pos⟩ ** ⟨object⟩</td>
<td>c ← ⟨pos⟩, connect using ⟨object⟩</td>
</tr>
<tr>
<td>⟨pos⟩ ? ⟨place⟩</td>
<td>c ← ⟨pos⟩, c ← ⟨place⟩</td>
</tr>
<tr>
<td>⟨pos⟩ @ ⟨stacking⟩</td>
<td>c ← ⟨pos⟩, do ⟨stacking⟩</td>
</tr>
<tr>
<td>⟨pos⟩ = ⟨saving⟩</td>
<td>c ← ⟨pos⟩, do ⟨saving⟩</td>
</tr>
<tr>
<td>⟨coord⟩ →</td>
<td>⟨vector⟩</td>
</tr>
<tr>
<td>⟨empty⟩</td>
<td>c reuses last c (do nothing)</td>
</tr>
<tr>
<td>p</td>
<td>p</td>
</tr>
<tr>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>s ⟨digit⟩</td>
<td>s {number}</td>
</tr>
<tr>
<td>&quot;(id)&quot;</td>
<td>restore what was saved as ⟨id⟩ earlier</td>
</tr>
<tr>
<td>{ ⟨pos⟩ ⟨decor⟩ }</td>
<td>the c resulting from interpreting the group</td>
</tr>
<tr>
<td>⟨vector⟩ →</td>
<td>0</td>
</tr>
<tr>
<td>⟨dimen⟩ , ⟨dimen⟩</td>
<td>absolute with equal dimensions</td>
</tr>
<tr>
<td>⟨dimen⟩ &gt;</td>
<td>angle in current base</td>
</tr>
<tr>
<td>⟨factor⟩ , ⟨factor⟩</td>
<td>The ⟨factor⟩ multiplied with ⟨factor⟩</td>
</tr>
<tr>
<td>/ ⟨direction⟩ ⟨dimen⟩ /</td>
<td>vector ⟨dimen⟩ in ⟨direction⟩</td>
</tr>
<tr>
<td>⟨corner⟩ →</td>
<td>L</td>
</tr>
<tr>
<td>CL</td>
<td>CR</td>
</tr>
<tr>
<td>LD</td>
<td>RD</td>
</tr>
<tr>
<td>E</td>
<td>P</td>
</tr>
<tr>
<td>A</td>
<td>vertical offset to math axis</td>
</tr>
<tr>
<td>⟨place⟩ →</td>
<td>⟨place⟩</td>
</tr>
<tr>
<td>⟨factor⟩</td>
<td>⟨place⟩</td>
</tr>
<tr>
<td>⟨slide⟩</td>
<td>f ← ⟨factor⟩</td>
</tr>
<tr>
<td>! ⟨pos⟩ ⟨slide⟩</td>
<td>pick place and apply ⟨slide⟩</td>
</tr>
<tr>
<td>⟨slide⟩ →</td>
<td>/ ⟨dimen⟩ /</td>
</tr>
<tr>
<td>⟨empty⟩</td>
<td>slide ⟨dimen⟩ further along connection</td>
</tr>
<tr>
<td></td>
<td>no slide</td>
</tr>
</tbody>
</table>

Figure 1.1: ⟨pos⟩itions.
Exercise 1.1: Which of the positions 0, \(0,0\), \(<0pt,0pt>\), \(0\), and \(/0pt/\) is different from the others? (p. 575)

Parsing: First the \POS and \afterPOS (decor)ations, and similar \afterCOORD and \afterVECTORorEMPTY ones. They handle parsing of \pos, \coord, and \vector; parsing of \corner and \place is presented along with note 1.3n and 1.3h explaining them.

The \afterVECTORorEMPTY command is special in that it takes two arguments: the ‘continuation’ if a \vector was found and the continuation if \empty was found (this is not applicable to the other two since \empty is a legal \coord and thus also a legal \pos).

Next we proceed with the actual parsing primitives: \COORD, \POS, and \VECTOR. These are bound to \xyCOORD, \xyPOS, and \xyVECTOR in order to be extendable, e.g., the matrix option extends \coord to support the \[row, column\] format by redefining \COORD to first test for this new format and then call \xyCOORD.

The parsing commands above are set up such that they all first call the \VECTOR command. \coord and \pos parsing then proceeds with calling the \COORD if there was no \vector. \pos parsing then calls \POS to continue the \pos (in both cases).

First \vector:

All letters used for \vector are uppercase \corner except for a used for angles (where the main code is in note 1.3e); this is also where we introduce the auxiliary \notrelaxorelse that takes two control sequences and expands the first unless it is relax in which case it expands the second.
The ⟨corner⟩ trick is to do nothing when there is nothing and initialise both X and Y in all other cases.

The remaining ⟨vector⟩ forms just set X and Y.

The ⟨factor⟩ also; this does no harm as it was never called if the first character was a ⟨.⟩.
1.3. POSITIONS

Hack: escape out in case of the special (* introducer...picked up by the ⟨coord⟩ parser, then.

xydef\VECTOR@other@open{%
\ifx *
ext \DN@{\VECTORempty@true \xyFN@afterVECTOR@}%
\else
\DN@{##1}{\xy@{##1}\vfromcartesian@{##1}\VECTORempty@false\afterVECTOR@}%
\fi \next@}

Next ⟨coord⟩inates that are not ⟨vector⟩s:

xydef\xyCOORD@{%
\ifx \space@\next \expandafter\DN@{\xyFN@COORD@}%
gobble spaces
\else \ifcat A\noexpand\next \let\next@=\xyCOORD@letter
\else \let\next@=\xyCOORD@other \fi\fi \next@}
\xylet@COORD@=\xyCOORD@
\xydef\xyCOORD@letter{%
\ifx c\next
\DN@ c{\xy@{c}{}\afterCOORD@}%
\else\ifx p\next
\DN@ p{\xy@{p}\cfromp@ \afterCOORD@}%
\else\ifx x\next
\DN@ x{\xy@{x}{\R@c=\X@xbase \U@c=\Y@xbase \intersect@}\afterCOORD@}%
\else\ifx y\next
\DN@ y{\xy@{y}{\R@c=\X@ybase \U@c=\Y@ybase \intersect@}\afterCOORD@}%
\else\ifx s\next
\DN@ s##1{\xy@{s{##1}}{\cfroms@{##1}}\afterCOORD@}%
\else \let\next@=\afterCOORD@ \fi\fi\fi\fi\fi \next@}
\xydef\xyCOORD@other{%
\ifx "\next %"
\DN@"##1"{\xy@{"##1"}{\cfromid@{##1}\afterCOORD@}%
\else\ifx b\group\next
\DN@{##1}{\xy@{##1}\enter@{\pfromthep@\basefromthebase@}}%
\silencexy@ \POS##1\relax \unsilencexy@ \xy@leave@ \afterCOORD@}%
\else\ifx (\next %)
\DN@{\xyFN@xyCOORD@other@open}%
\else \let\next@=\afterCOORD@ \fi\fi \next@}
xynew@if\ifsilentxy@
\xydef\silencexy%@%
\ifsilentxy@ \nter@{}%
\else \nter@{\silentxy@false \let\xy@=\unsilent@\xy@}\
\silencexy@true \let\unsilent@\xy@=\xy@ \def\xy@##1##2{\unsilent@\xy@{##1##2}}%
\fi
\xydef\unsilencexy@{\leave@}
\xydef\xyCOORD@other@open{%
\ifx *\next
\DN@"##1"{\xy@{*}{\enter@{\pfromthep@\basefromthebase@}}%
Finally (pos) parsing after (coord) (possibly (vector)) is interpreted:

```
\xydef\xyPOS@% 
\ifx \space@ \next \expandafter\DN@\space{\xyFN@\POS@} % gobble spaces
\else \addPLUS@ \ifx \next \addPLUS@ \DN@{\xy@+{\enter@\cplusthec@} \afterCOORD{\xy@{\leave@ \xyFN@\POS@}}} 
\else \addDASH@ \ifx \next \addDASH@ \DN@{\xy@-{\enter@\cplusthec@} \afterCOORD{\xy@{\X@c=-\X@c \Y@c=-\Y@c \leave@ \xyFN@\POS@}}} 
\else \ifx ! \next \DN@ !{\xy@!{\enter@\cskewthec@} \afterCOORD{\xy@{\leave@ \xyFN@\POS@}}} 
\else \addDOT@ \ifx \next \addDOT@ \DN@{\xy@.{\enter@\cmergethec@} \afterCOORD{\xy@{\leave@ \xyFN@\POS@}}} 
\else \ifx , \next \DN@ ,{\xy@,{\comma@@} \afterCOORD{\xyFN@\POS@}} 
\else \ifx ; \next \DN@ ;{\xy@;{\swap@} \afterCOORD{\xyFN@\POS@}} 
\else \ifx : \next \DN@ :{\xyFN@\POS@colon} 
\else \addEQ@ \ifx \next \addEQ@ \DN@{\xyFN@\saveid@} 
\else \ifx * \next \DN@ *(\xyFN@\POS@star) 
\else \ifx ? \next \DN@ ?{\xy@?{} \afterPLACE{\xyFN@\POS@}} 
\else \addAT@ \ifx \next \addAT@ \DN@{\xyFN@\STACK@} 
\else \let \next@=\afterPOS@ 
\fi 
\fi 
\fi 
\fi 
\fi 
\fi 
\fi 
\fi 
\fi 
\fi 
\fi 
\fi 
\fi 
\fi 
\fi 
```

```
\comma@@ is a hook used to change the operation of ,, e.g., when reading a stack setup where it means ‘push’. The final functions serve only to distinguish between the single character :/* and dual character ::/** operators:
```

}\xydef@\POS@colon{\DNii@{\afterCOORD{\xyFN@\POS@}}% 
\ifx : \next \xy@::{}{\setbase@@\X@c\Y@c\DN@:\nextii@}% 
\else \xy@::{\setbase@\X@p\Y@p\X@c\Y@c\let\next@=\nextii@ \fi 
\next@}% 
\xydef@\POS@star% 
\ifx * \next 
\DN@*#1##{\nextii@{#1}}% 
\DNii@#1##2{\xy@{#1}{#2}}%
```
1.3. POSITIONS

Simple actions: Next follow the simplest actions; the complicated ones are explained along with their notes below.

Next the parsing of coordinate pairs in <>:

The next group of commands are used to store on the control stack with the \enter@ command, so they expand to something useful:

Notes

1.3a. When doing arithmetic with \(+\) and \(-\) then the resulting current object inherits the size of the \(\langle\text{coord}\rangle\), \text{i.e.}, the right argument—this will be zero if the \(\langle\text{coord}\rangle\) is a \(\langle\text{vector}\rangle\).

**Exercise 1.2:** How do you set \(c\) to an object the same size as the saved object "ob" but moved \(<X,Y>\)? \(\text{(p.575)}\)

1.3b. *Skewing* using \(!\) just means that the reference point of \(c\) is moved with as little change to the shape of the object as possible, \text{i.e.}, the edge of \(c\) will remain in the same location except that it will grow larger to avoid moving the reference point outside \(c\).

**Exercise 1.3:** What does the \(\langle\text{pos}\rangle\ldots!\text{R-L}\) do? \(\text{(p.575)}\) **Bug:** The result of \(!\) is always a rectangle currently.

**Procedure:** \(!\) moves the center of \(c\) by a temporarily read \(c'\) and then readjusts the extents:

\[
\begin{align*}
D_c &= Y' + Y_c - \min(Y' - D', Y' + Y_c) = \max(Y_c + D', 0) \\
U_c &= \max(Y' + U', Y' + Y_c) - (Y' + Y_c) = \max(U' - Y_c, 0) \\
Y_c &= Y' + Y_c \\
L_c &= X' + X_c - \min(X' - L', X' + X_c) = \max(X_c + L', 0) \\
R_c &= \max(X' + R', X' + X_c) - (X' + X_c) = \max(R' - X_c, 0) \\
X_c &= X' + X_c
\end{align*}
\]

1.3c. A \(\langle\text{pos}\rangle\) *covers* another if it is a rectangle with size sufficiently large that the other is "underneath". The \(.\) operation “extends” a \(\langle\text{pos}\rangle\) to cover an additional one—the reference point of \(c\) is not moved but the shape is changed to a rectangle such that the entire \(p\) object is covered.

**Bug:** non-rectangular objects are first “translated” into a rectangle by using a diagonal through the object as the diagonal of the rectangle.

**Procedure:** \(.\) takes a temporary object \(c'\) and adjusts the extents of \(c\) such that it is covered.

\[
\begin{align*}
L_c &= X' - \min(X' - L_c, X - L) = \max(L_c, A + L) \\
R_c &= \max(X' + R_c, X + R) - X' = \max(R_c, -A + R) \\
D_c &= Y' - \min(Y' - D_c, Y - D) = \max(D_c, B + D) \\
U_c &= \max(Y' + U_c, Y + U) - Y' = \max(U_c, -B + U)
\end{align*}
\]
with $<A, B> = <X'-X, Y'-Y>$. First method 2 of the object is used to convert it into a rectangle.

1.3d. The operations : and :: set the base used for \(\langle coord\rangle\)inates having the form \((x, y)\). The : operation will set \(\langle X\text{\_origin}, Y\text{\_origin}\rangle\) to \(p\), \(\langle X\text{\_base}, Y\text{\_base}\rangle\) to \(c - \text{\_origin}\), and \(\langle X\text{\_ybase}, Y\text{\_ybase}\rangle\) to \((-Y\text{\_base}, X\text{\_base})\) (this ensures that it is a usual square coordinate system). The :: operation may then be used afterwards to make nonsquare bases by just setting \(y\text{\_base}\) to \(c - \text{\_origin}\). Here are two examples: firstly \(0; <1\text{\,cm}, 0\text{\,cm}>\): sets the coordinate system

![Diagram](attachment:image)

where in each case the \(\circ\) is at 0, the base vectors have been drawn and the \(\times\) is at (1,1).

When working with cartesian coordinates these three special \(\langle factor\rangle\)s are particularly useful:

\[
\begin{align*}
\text{\halfroottwo} &= 0.70710678 \approx \frac{1}{2} \sqrt{2} \\
\text{\partroottwo} &= 0.29289322 \approx 1 - \frac{1}{2} \sqrt{2} \\
\text{\halfrootthree} &= 0.86602540 \approx \frac{1}{2} \sqrt{3}
\end{align*}
\]

More can be defined using \def (or \newcommand in \LaTeX).

Procedure: The code chosen by the parsing is very simple; the only tricky bit is to ensure that \\text{\\@\textbf{\texttt{substr}} the current base.}
1.3e. An angle $\alpha$ in XY-pic is the same as the coordinate pair $(\cos \alpha, \sin \alpha)$ where $\alpha$ must be an integer interpreted as a number of degrees. Thus the vector $a(0)$ is the same as $(1,0)$ and $a(90)$ as $(0,1)$, etc.

The translation involves several steps: (l.1962) Normalise the argument to be within $[0^\circ : 360^\circ]$. (l.1965) Flip angle around $x$-axis and then $y$-axis to ensure it is in the first quadrant, i.e., within $[0^\circ : 90^\circ]$. (l.1970) Flip around diagonal to ensure angle within $[0^\circ : 45^\circ]$. (l.1973) Find values $\phi < \psi$ from the table in figure 1.2 (using recursive table lookup – at most 3 tests needed). (l.2030) Build vector $(x,y)$ interpolated between the sin/cos values for $\phi$ and $\psi$ using the formula

$$(\cos \phi + k(\cos \psi - \cos \phi), \sin \phi + k(\sin \psi - \sin \phi)),$$

where $k = \frac{\alpha - \phi}{\psi - \phi}$

(l.2041) Build the chosen vector.
Vectors for angles in $[0^\circ : 45^\circ]$ contains all angles required to typeset fractions up to $\frac{n}{12} \times 2\pi$, $\frac{1}{16} \times 2\pi$, and $\frac{1}{24} \times 2\pi$ exactly, and two extra low ones to ensure that all gaps are less than $5^\circ$ and the precision of all sine/cosines better than $\frac{1}{1000}$.

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$(\cos \alpha, \sin \alpha)$</th>
<th>fractions of $2\pi$</th>
<th>flipped fractions of $2\pi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$(1, 0)$</td>
<td>$\frac{3}{8}$</td>
<td>$\frac{1}{2}, \frac{1}{4}, \frac{1}{10}, \frac{1}{16}, \frac{1}{20}, \frac{1}{12}$</td>
</tr>
<tr>
<td>4.090909</td>
<td>(.99677570, .08023846)</td>
<td>$-$</td>
<td>$-$</td>
</tr>
<tr>
<td>6</td>
<td>(.99452190, .10452846)</td>
<td>$\frac{3}{11}$</td>
<td>$\frac{8}{11}$</td>
</tr>
<tr>
<td>8.181818</td>
<td>(.98982144, .14231484)</td>
<td>$\frac{7}{9}$</td>
<td>$\frac{7}{9}$</td>
</tr>
<tr>
<td>10</td>
<td>(.98480775, .17364818)</td>
<td>$\frac{2}{7}$</td>
<td>$\frac{5}{7}$</td>
</tr>
<tr>
<td>12.857143</td>
<td>(.97492791, .22252093)</td>
<td>$\frac{1}{12}$</td>
<td>$\frac{5}{12}$</td>
</tr>
<tr>
<td>15</td>
<td>(.96592583, .25881905)</td>
<td>$\frac{1}{12}$</td>
<td>$\frac{5}{12}$</td>
</tr>
<tr>
<td>16.363636</td>
<td>(.95949297, .28173256)</td>
<td>$\frac{6}{11}$</td>
<td>$\frac{5}{11}$</td>
</tr>
<tr>
<td>18</td>
<td>(.95105652, .30901699)</td>
<td>$\frac{3}{10}, \frac{4}{10}, \frac{8}{10}$</td>
<td>$\frac{1}{5}, \frac{2}{5}, \frac{7}{10}$</td>
</tr>
<tr>
<td>20</td>
<td>(.93969262, .34202014)</td>
<td>$\frac{5}{9}$</td>
<td>$\frac{4}{9}$</td>
</tr>
<tr>
<td>22.5</td>
<td>(.92387953, .38268343)</td>
<td>$\frac{1}{10}$</td>
<td>$\frac{2}{10}$</td>
</tr>
<tr>
<td>24.545455</td>
<td>(.90963200, .41541501)</td>
<td>$\frac{7}{11}$</td>
<td>$\frac{7}{11}$</td>
</tr>
<tr>
<td>25.714286</td>
<td>(.90096887, .43388374)</td>
<td>$\frac{4}{7}$</td>
<td>$\frac{3}{7}$</td>
</tr>
<tr>
<td>30</td>
<td>(.86602540, .5)</td>
<td>$\frac{1}{3}, \frac{1}{12}, \frac{2}{5}, \frac{3}{10}, \frac{5}{10}, \frac{7}{12}, \frac{10}{12}$</td>
<td>$\frac{1}{5}, \frac{3}{10}, \frac{2}{5}, \frac{7}{10}, \frac{5}{10}, \frac{3}{10}, \frac{1}{12}$</td>
</tr>
<tr>
<td>32.727273</td>
<td>(.84125353, .54064082)</td>
<td>$\frac{1}{10}$</td>
<td>$\frac{1}{10}$</td>
</tr>
<tr>
<td>36</td>
<td>(.80901699, .58778525)</td>
<td>$\frac{1}{10}, \frac{3}{10}, \frac{6}{10}$</td>
<td>$\frac{2}{5}, \frac{4}{10}, \frac{9}{10}$</td>
</tr>
<tr>
<td>38.571429</td>
<td>(.78183148, .62348980)</td>
<td>$\frac{5}{7}$</td>
<td>$\frac{5}{7}$</td>
</tr>
<tr>
<td>40</td>
<td>(.76604444, .64278761)</td>
<td>$\frac{1}{9}$</td>
<td>$\frac{8}{9}$</td>
</tr>
<tr>
<td>40.909091</td>
<td>(.75574957, .65486073)</td>
<td>$\frac{1}{11}$</td>
<td>$\frac{7}{11}$</td>
</tr>
<tr>
<td>45</td>
<td>(.70710678, .70710678)</td>
<td>$\frac{1}{8}, \frac{3}{8}, \frac{5}{8}$</td>
<td>$-$</td>
</tr>
</tbody>
</table>

Figure 1.2: Computing angle vectors
\begin{verbatim}
184 \chooseangleinterval@
185 {}\%
186 \{8.181818\{.98982144,.14231484\}\%
187 \{\}
188 \{10\}\{.98480775,.17364818\}%
189 \chooseangleinterval@
190 {}\%
191 \{12.857143\}\.97492791,.22252093\}%
192 \{\}
193 \{15\}\{.96592583,.25881905\}%
194 \chooseangleinterval@
195 \chooseangleinterval@
196 {}\%
197 \{16.363636\}\.95949297,.28173256\}%
198 \{\}
199 \{18\}\{.95105652,.30901699\}%
200 \chooseangleinterval@
201 {}\%
202 \{20\}\{.93969262,.34202014\}%
203 \{\}
204 \{22.5\}\{.92387953,.38268343\}%
205 \chooseangleinterval@
206 \chooseangleinterval@
207 \chooseangleinterval@
208 {}\%
209 \{24.545455\}\.90963200,.41541501\}%
210 \{\}
211 \{25.714286\}\.90096887,.43388374\}%
212 \{\}
213 \{30\}\.86602540,.50\}%
214 \chooseangleinterval@
215 \chooseangleinterval@
216 {}\%
217 \{32.727273\}\.84125353,.54064082\}%
218 \{\}
219 \{36\}\.80901699,.58778525\}%
220 \chooseangleinterval@
221 \chooseangleinterval@
222 {}\%
223 \{38.571429\}\.78183148,.62348980\}%
224 \{\}
225 \{40.909091\}\.75574957,.65486073\}%
226 \chooseangleinterval@
227 {}\%
228 \{40\}\.76604444,.64278761\}%
229 \{\}
230 \A@=\R@ \advance\A@-\dimen@
231 \ifdim\ifdim\A@<\z@-\fi\A@<.01\p@ \edef\next@{\expandafter\nextiii@\next@}%
232 \else \B@=\dimen@ii \advance\B@-\R@
\end{verbatim}
1.3. POSITIONS

1.3f. To drop an ⟨object⟩ at \( c \) with * means to actually physically typeset it in the picture with reference position at \( c \)—how this is done depends on the ⟨object⟩ in question and is described in detail in §1.4. The intuition with a drop is that it typesets something at \( <X_c, Y_c> \) and sets the edge of \( c \) accordingly.

Procedure: \( (l.2110) \) sets up the direction to allow for directionals and builds the requested ⟨object⟩ in the (global) \( \lastobjectbox@ \) box, \( (l.2112) \) adjust the picture size unless it is a hidden object, setting \( \dimen@ = X_c - L_c \), and \( (l.2118) \) drop the object in the picture at the right point by setting \( \text{box0} \) and using the \( \Drop@@ \) method.
Figure 1.3: All directions.

**Note:** All typesetting into a picture should use or emulate \drop@!

1.3g. The *connect* operation **will** first compute a number of internal parameters describing the direction from *p* to *c* and then typesets a connection filled with copies of the (object) as illustrated in §1.2.3. The exact details of the connection depend on the actual (object) and are described in general in §1.4. The intuition with a connection is that it typesets something connecting *p* and *c* and sets the ? (pos) operator up accordingly.

**Procedure:** Set up the direction to allow for directional objects, then save *c*, build the (object) in \lastobjectbox@, restore *c*, and perform the \Connect@@ method to connect using \lastobjectbox@.

```latex
\xydef@\connect@#1#2{\setupDirection@ \enter@{\cfromthec@}\
\global\setbox\lastobjectbox@=\object#1{#2}\leave@
\Connect@@}
\xydef@\preconnect@#1#2{\setupDirection@ \enter@{\cfromthec@}\
\global\setbox\lastobjectbox@=\object#1{#2}\leave@ \connectStore@
\ifInvisible@ \Connect@@ \else \Invisible@true\Connect@@\Invisible@false \fi}
```

The \preconnect command is for internal use by arrow ... it makes use of the following:

```latex
\xynew@{box}\connectobjectbox@@
\xylet@\connectDrop@@=\empty
\xylet@\connectpreXY@style@=\empty
\xylet@\connectpostXY@style@=\empty
```
1.3. POSITIONS

See note ‘define ⟨shape⟩’ for the use of \preXY@style@ and \postXY@style@ to apply special ⟨style⟩s, specified via ⟨object-modifier⟩s; being saved here by \connectStore@ as \connectpreXY@style@ and \connectpostXY@style@ to be reset by \connectRestore@.

1.3h. Using ? will “pick a place” along the most recent connection typeset with **. What exactly this means is determined by the object that was used for the connection and by the modifiers described in general terms here.

The “shave” modifiers in a ⟨place⟩, < and >, change the default ⟨factor⟩, f, and how it is used, by ‘moving’ the positions that correspond to (0) and (1) (respectively): These are initially set equal to p and c, but shaving will move them to the point on the edge of p and c where the connection “leaves/enters” them, and change the default f as indicated. When one end has already been shaved thus then subsequent shaves will correspond to sliding the appropriate position(s) a \text{T}E\text{X} \jot (usually equal to 3pt) further towards the other end of the connection (and past it). Finally the pick action will pick the position located the fraction f of the way from (0) to (1) where f = 0.5 if it was not set (by <, >, or explicitly).

All this is probably best illustrated with some examples: each ⊗ in figure 1.4 is typeset by a sequence of the form p; c **@{.} ?⟨place⟩ *⟨oplus⟩ where we indicate the ⟨place⟩ in each case. (We also give examples of ⟨slide⟩s.)

Procedure: The code for parsing ⟨place⟩ is the following. To get first <> to move to edge and the remaining to move a \text{jot} we have both initial and continuing versions for each, the idea being that the second and following go to the edge of a small temporary object with radius \text{jot}.

Note: This parser tests the new parsing principle that \text{xy}@ should always be called as \text{xy}@{ source } \text{target }.
Figure 1.4: Example ⟨place⟩s
1.3i. A \langle slide \rangle will move the position a dimension further along the connection at the picked position.
For straight connections (the only ones kernel \texttt{Xy-pic} provides) this is the same as adding a vector in the tangent direction, \textit{i.e.}, \( \ldots/A \) is the same as \( \ldots + A/ \).

1.3j. This special \langle place \rangle finds the point where the last connection intercepts with the line from \( p \) to \( c \) as setup by the \langle pos \rangle, thus usually this will have the form \( \{ \langle \text{coord} \rangle ; \langle \text{coord} \rangle \} \), for example, \textbf{Bug:} Only works for straight arrows at present.

\texttt{\textbackslash xy <1cm,0cm>:}
\begin{verbatim}
(0,0)*=0{+}="+
(2,1)*=0{\times}="**@{.},
(1,0)++{A} ; (2,2)++{B} ++@{-}
?!{"+";"*"} *{\bullet}
endxy
\end{verbatim}

will typeset

\begin{center}
\begin{tikzpicture}
\coordinate (A) at (0,0);
\coordinate (B) at (2,1);
\coordinate (C) at (1,0);
\coordinate (D) at (2,2);
\draw (A) -- (B);
\draw (C) -- (D);
\end{tikzpicture}
\end{center}

1.3k. The positions denoted by the \textit{axis intersection} \langle coord \rangle indicates \( x \) and \( y \) are the points where the line through \( p \) and \( c \) intersects with each axis. The following figure illustrates this:

\begin{center}
\begin{tikzpicture}
\coordinate (A) at (0,0);
\coordinate (B) at (2,1);
\coordinate (C) at (1,0);
\coordinate (D) at (2,2);
\draw (A) -- (B);
\draw (C) -- (D);
\end{tikzpicture}
\end{center}

\begin{enumerate}
\item \textbf{Exercise 1.4:} Given predefined points \( A, B, C, \) and \( D \) (stored as objects \texttt{"A"}, \texttt{"B"}, \texttt{"C"}, and \texttt{"D"}), write a \langle coord \rangle specification that will return the point where the lines \( AB \) and \( CD \) cross.
\end{enumerate}

\footnote{The braces can be replaced by \langle \ldots \rangle \textit{once}, \textit{i.e.}, there can be no other braces nested inside it.}
CHAPTER 1. KERNEL: XY.DOC

(the point marked with a large circle here):

\[ \text{Procedure:} \quad \text{We solve the following equation in } a, b: \]
\[ \text{origin} + a \times <R_c, U_c> = c - b \times (c - p) \]

and then set
\[ <X_c, Y_c> := <X_c, Y_c> - b \times (c - p) \quad \text{with zero size} \quad D_c, U_c, L_c, R_c := 0, 0, 0, 0. \]

The code uses \( c = (X_c, Y_c, D_c, U_c, L_c, R_c) \) and \( A, B \) as temporaries and computes:
\[ <dX, dY> := <X_c, Y_c> - <X_p, Y_p> \]
\[ <A, B> := <X_c, Y_c> - <X_{\text{origin}}, Y_{\text{origin}}> \]
\[ <D_c, L_c> := \begin{bmatrix} R & dX \\ U & dY \end{bmatrix} \begin{bmatrix} R & A \\ U & B \end{bmatrix} \]
\[ <X_c, Y_c> := <X_c, Y_c> - (L_c/D_c) \times <dX, dY> \]

where we really do \( D := (R/pt)dY - (U/pt)dX \) and similarly for \( L \).

When there is no intersection point a wrong answer is returned, accompanied by a warning message. This behaviour can be altered by assigning a different value to the hook: \texttt{\textbackslash zeroDivide@}. This macro must store a number in \texttt{\textbackslash next@}.
By specifying \( \text{zeroDivideLimit}(\text{num}) \) the user can locally establish that \( \frac{x}{0} = \text{sgn}(x)(\text{num}) \), whenever such a division by zero would otherwise occur in an intersection calculation.

1.3l. A \( \langle \text{pos} \rangle \) (decor) \textit{grouped} in \{\}\textit{-}braces\(^5\) is interpreted in a local scope in the sense that any \( p \) and \( \text{base} \) built within it are forgotten afterwards, leaving only the \( c \) as the result of the \( \langle \text{coord} \rangle \). \textbf{Note:} Only \( p \) and \( \text{base} \) are restored – it is not a \TeX\ group.

\textbf{Exercise 1.5:} What effect is achieved by using the \( \langle \text{coord} \rangle \text{inate} \{;\} \)? \((\text{p.575})\)

The code is inside \POS@.

1.3m. The vector /\( Z /\), where \( Z \) is a \( \langle \text{dimen} \rangle \)sion, is the same as the vector \( <Z \cos \alpha, Z \sin \alpha> \) where \( \alpha \) is the angle of the last direction set by a connection (\textit{i.e.}, with **) or subsequent placement (\?) position.

\texttt{2520 xydef@vfromslide@#1{\enter@\DirectionfromtheDirection@ \begingroup}
2521 \plainxy@afterDIRECTIONorEMPTY\vfromslide@i\vfromslide@i#1@}\endgroup
2523 \edef\next{\endgroup
2524 \ifx\next@empty \dimen@=.5pc \else \dimen@=#1\relax \fi
2526 \X@c=\cosDirection\dimen@ \Y@c=\sinDirection\dimen@}\next
2527 \leave@}

It is possible to give a \( \langle \text{direction} \rangle \) as described in the next section (figure 1.5, note 1.4l in particular) that will then be used to set the value of \( \alpha \). It is also possible to omit the \( \langle \text{dimen} \rangle \) in which case it is set to a default value of .5pc.

1.3n. A \( \langle \text{corner} \rangle \) is an offset from the current \( <X_c, Y_c> \) position to a specific position on the edge of the \( c \) object (the two-letter ones may be given in any combination):

![Diagram of corners and edges]

The ‘edge point’ \( E \) lies on the edge along the line from \( p \) to the centre of the object, in contrast to the ‘proportional’ point \( P \) which is also a point on the edge but computed in such a way that the object looks as much ‘away from \( p \)’ as possible. The \( A \) point vector is special: it is equal to \(<0pt, \text{fontdimen}22, \text{textfont2}> \) and useful for recentering entries.

Finally, a following \( \langle \text{f} \rangle \) suffix will multiply the offset vector by the \( \langle \text{factor} \rangle \) \( f \).

\textbf{Exercise 1.6:} What is the difference between the \( \langle \text{pos} \rangle \text{itions} \ c? < \text{ and } c+E? \) \((\text{p.576})\)

\textbf{Exercise 1.7:} What does this typeset?

\texttt{\xy *=<3cm,1cm>\txt{Box}*\frm{-}\endxy}

\texttt{!U!(.5) *\frm{..}*{\bullet} \endxy}

\(^5\)One can use \(*\ldots\ast\) instead also here.
Hint: \texttt{\textbackslash frm} is defined by the frame extension and just typesets a frame of the kind indicated by the argument. (p.576)

**Bug:** Currently only the single-letter corners (L, R, D, U, C, E, and P) will work for any shape—the others silently assume that the shape is rectangular.

### 1.3o. The stack is a special construction useful for storing a sequence of \langle pos\rangle itions that are accessible using the special \langle coord\rangle inates $s_n$, where $n$ is either a single digit or a positive integer in $\{s\}$: $s_0$ is always the ‘top’ element of the stack and if the stack has depth $d$ then the ‘bottom’ element of the stack has number $s\{d - 1\}$. The stack is said to be ‘empty’ when the depth is 0 and then it is an error to access any of the $s_n$ or ‘pop’ which means remove the top element, shifting what is in $s_1$ to $s_0$, $s_2$ to $s_1$, etc. Similarly, ‘push $c$’ means to shift $s_0$ to $s_1$, etc., and then insert the $c$ as the new $s_0$.

The stack is manipulated as follows:

<table>
<thead>
<tr>
<th>@ (stacking)</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>@+ (coord)</td>
<td>push (coord)</td>
</tr>
<tr>
<td>@- (coord)</td>
<td>$c \leftarrow $ (coord) then pop</td>
</tr>
<tr>
<td>@= (coord)</td>
<td>load stack with (coord)</td>
</tr>
<tr>
<td>@@ (coord)</td>
<td>do (coord) for $c \leftarrow $ stack</td>
</tr>
<tr>
<td>@i</td>
<td>initialise</td>
</tr>
<tr>
<td>@(</td>
<td>enter new frame</td>
</tr>
<tr>
<td>@)</td>
<td>leave current frame</td>
</tr>
</tbody>
</table>

To ‘load stack’, means to load the entire stack with the positions set by \langle coord\rangle within which $c$, means ‘push $c$’.

To ‘do \langle coord\rangle for all stack elements’ means to set $c$ to each element of the stack in turn, from the bottom and up, and for each interpret the \langle coord\rangle. Thus the first interpretation has $c$ set to the bottom element of the stack and the last has $c$ set to $s_0$. If the stack is empty, the \langle coord\rangle is not interpreted at all.

These two operations can be combined to repeat a particular \langle coord\rangle for several points, like this:

\begin{verbatim}
\xy
@={(0,-10),(10,3),(20,-5)} @@{*{P}}
@endxy
\end{verbatim}

will typeset

$$\begin{array}{c}
\hspace{10ex} P \\
\hspace{10ex} P \\
\hspace{10ex} P
\end{array}$$

Finally, the stack can be forcibly cleared using @i, however, this is rarely needed because of @(), which saves the stack as it is, and then clears it, such when it has been used (and is empty), and @) is issued, then it is restored as it was at the time of the @().

**Exercise 1.8:** How would you change the example above to connect the points as shown below?

![Diagram of a triangle]
First the stack top and bottom, both initially $-1$:

Next the function to set $c \leftarrow s_n$, i.e., test that $bot < n + bot < top$ and then run the associated stack element.

Finally the actual code to do the stack operations: it depends on the `code' passed after @; spaces are not allowed:

Finally the actual code to do the stack operations: it depends on the `code' passed after @; spaces are not allowed:
1.3p. It is possible to define new \langle coord \rangle inates on the form "\langle id \rangle" by \textit{saving} the current \texttt{c} using the ..."\langle id \rangle" \langle pos \rangle\rangle form. Subsequent uses of "\langle id \rangle" will then reestablish the \texttt{c} at the time of the saving.

Using a "\langle id \rangle" that was never defined is an error, however, saving into a name that was previously defined just replaces the definition without warning, \textit{i.e.}, "\langle id \rangle" always refers to the last thing saved with that \langle id \rangle.

However, many other things can be ‘saved’: in general =<saving> has either of the forms

\texttt{=:"\langle id \rangle" \langle id \rangle" restores current base
=\langle coord \rangle"\langle id \rangle" \langle id \rangle" reinterprets \langle coord \rangle}
The first form defines "⟨id⟩" to be a macro that restores the current base.

The second does not depend on the state at the time of definition at all; it is a macro definition. You can pass parameters to such a macro by letting it use coordinates named "1", "2", etc., and then use="1", ="2", etc., just before every use of it to set the actual values of these. **Note:** it is not possible to use a (coord) of the form "⟨id⟩" directly: write it as {"⟨id⟩"}

**Exercise 1.9:** Write a macro "dbl" to double the size of the current c object, e.g., changing it from the dotted to the dashed outline in this figure:

```
+   |
```

(p.576)

The final form defines a special kind of macro that should only be used after the @= stack operation: the entire current stack is saved such that the stack operation @="⟨id⟩" will reload it.

**Note:** There is no distinction between the ‘name spaces’ of ⟨id⟩s used for saved coordinates and other things.

This parser distinguishes between the cases:

```latex
\texttt{\begin{verbatim}
xydef\saveid@{
  \ifx \space@ \next \expandafter\DN@ \space@ \endcsname{\xyFN@\saveid@}%
obble spaces
  \else \ifx "\next\DN@":\next@"\next@"{\xy@{=}\next@}{\idfromc@{\next@}}\xyFN@\POS@%
    \else \ifx :
ext\DN@:\next@"\next@"{\xy@{=}\next@}{\idfrombase@{\next@}}\xyFN@\POS@%
    \else \addAT@ \ifx \next@\next@}
    \addAT@ \DN@\saveid@COORD@={\saveid@COORDii}\xyFN@\POS@%
  \else \iffalse \next@}
    \DN@\saveid@COORD@={\xyFN@\saveid@COORD@}%
  \else \let\saveid@COORD@@=\saveid@COORDi \let
    \next@=\saveid@COORD@={\saveid@COORDi \next@=\saveid@COORD@=%relax
    \fi\fi\fi\fi\fi\fi \next@}
  \addAT@ \DN@\saveid@COORD@={\xyFN@\saveid@COORD@}%
  \else \let\saveid@COORD@@=\saveid@COORDi \let
    \next@=\saveid@COORD@={\saveid@COORDii \next@=\saveid@COORD@%
    \fi\fi\fi\fi\fi\fi \next@}
  \addAT@ \DN@\saveid@COORD@={\xyFN@\saveid@COORD@}%
\end{verbatim}}
```

Here is the code for saving/restoring a position and a base.

```latex
\texttt{\begin{verbatim}
xydef\idfromc@#1{\DN@{#1}}%
  \expandafter\edef\csname Q@\endcodeof\next@\endcsname{\cfromthec@}}
\xydef\idfrombase@#1{\DN@{#1}}%
  \expandafter\edef\csname Q@\endcodeof\next@\endcsname{\basefromthebase@}}
\xydef\idfromstack@#1{%
  \toks@={\if\sempty@\else
    \expandafter\DN@{\xyFN@\next@\endsname{\sinit@\fi}}%}
  \count@=\s@bot \advance\count@\@ne
  \ifnum\count@>\s@top\else
    \xywarning@{loading on top of non-empty stack}\sinit@ \fi}%
  \count@=\s@bot \advance\count@\@ne
  \ifnum\count@>\s@top\else
    \fi\fi\fi\fi\fi\fi \next@}
\end{verbatim}}
```
1.4 Objects

Objects are the entities that are manipulated with the * and ** ⟨pos⟩ operations above to actually get some output in XYPictures. As for ⟨pos⟩itions the operations are interpreted strictly from left to right, however, the actual object is built before all the ⟨modifier⟩s take effect. The syntax of objects is given in figure 1.5 with references to the notes below.

Remark: It is never allowed to include braces {} inside ⟨modifier⟩s! In case you wish to do something that requires {...} then check in this manual whether you can use (*...*) instead. If not then you will have to use a different construction.

We first discuss the parser and then summarise the required methods. The entry point to use of objects is \object described in note 1.4c. This should always be used because it initialises the token list and redefines \xy to be \addtotoks such that we can use ⟨pos⟩ parser routines within the ⟨object⟩!

Parsing: The ⟨object⟩ parser \OBJECT will first parse the ⟨modifier⟩s, storing the action of each in sequence on the \toks token list. When there are no more modifiers we insert \objectbox if we have reached the { otherwise we just assume that the rest of the ⟨object⟩ is some kind of box construction.

Note: The ⟨modifier⟩ actions doing shifts are implemented by having an independent vector for the shift: \(<R_p, U_p>\) always contains the vector from the current to the original \TeX reference point; furthermore the initial \(L_c\) is saved as \(L_p\) such that we can retrieve the original (XYPic) reference point again. Modifying the \(p\) values is safe because all actual changes are done in a local scope after the entire ⟨object⟩ is parsed and we have built the object box (in \OBJECT).
### 1.4. OBJECTS

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>⟨object⟩</td>
<td>→ ⟨modifier⟩ ⟨object⟩</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>⟨objectbox⟩</td>
<td>→ { ⟨text⟩ }</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>⟨modifier⟩</td>
<td>→ ! ⟨vector⟩</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>⟨add op⟩</td>
<td>→ +</td>
</tr>
<tr>
<td>⟨size⟩</td>
<td>→ ⟨empty⟩</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>⟨direction⟩</td>
<td>→ ⟨diag⟩</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>⟨diag⟩</td>
<td>→ ⟨empty⟩</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>⟨composite⟩</td>
<td>→ ⟨object⟩</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 1.5: ⟨object⟩s.
used when building the \langle object \rangle is right, and restored in the right sequence while evaluating the \langle modifier \rangle s afterwards.

\xydef@\OBJECT@{\
\ifx \space@ \next \expandafter\DN@ \space\{\xyFN@\OBJECT@\}\% gobble spaces\
\else \ifcat A \noexpand \next \let \next@=\OBJECT@letter \\
\else \let \next@=\OBJECT@other \fi \fi \next@}

\xydef@\OBJECT@letter{% 
\ifx i \next \DN@ i{\addtotoks@\Invisible@true \xyFN@\OBJECT@\}% 
\else \ifx h \next \DN@ h{\addtotoks@\Hidden@true \xyFN@\OBJECT@\}% 
\else \ifx o \next \DN@ o{\xywarning@{Obsolete \texttt{o} \text{ modifier used}} \OBJECT@shape{o} \}% 
\else \ifx x \next \DN@ x{\xywarning@{Obsolete \texttt{x} \text{ modifier used}} \OBJECT@shape{} \}% 
\else \ifx @ \next \DN@ @##1##1{\
\xywarning@{Impossible \texttt{@} \text{(letter)} should not be here!!} \\
\else \
\fi}

\xydef@\OBJECT@other{% 
\ifx ! \next \DN@!{\OBJECT@shift} \% 
\else \addPLUS@ \ifx \next \DN@{\OBJECT@change} \%
\else \addDASH@ \ifx \next \DN@{\OBJECT@change} \%
\else \addEQ@ \ifx \next \DN@{\OBJECT@set} \%
\else \ifx \![ \next \%
\DN@{[#1]}{\xy@{[#1]}{\OBJECT@shape{##1}}} \%
\else \ifx ^ \next \let \next@=\OBJECT@direction \\
\else \ifx _ \next \let \next@=\OBJECT@direction \\
\else \ifx : \next \let \next@=\OBJECT@direction \\
\else \ifx ? \next \%
\DN@ nullable{\xywarning@{\texttt{?} \text{ modifier used}} \xyFN@\OBJECT@direction} \%
\else \ifx \next \%
\let \next@=\OBJECT@direction \\
\else \addAT@ \ifx \next \addAT@ \DN@{\OBJECT@{#1}} \%
\else \DN@{#1}{\OBJECT@{#1}} \%
\fi}

\OBJECT@ is where we actually build the box by first 1.3137 saving the previous edge in case it is needed, then 1.3138 setting the defaults (including temporarily resetting \xy@ to execute in case any \xy@s are used internally), 1.3143 building the object box (which might change them) using \objectbox if no other command specified and setting the $D, U, L, R$ dimensions as required using the \Leftness@ and \Upness@ methods, and finally 1.3147 setting up the $R_p, U_p$ dimensions (as discussed above) and applying the \langle modifier \rangle s stored in \toks@ and dumping the modified box.

\xydef@\prevEdge@@\{\zeroEdge\}

\xydef@\OBJECT@#1#2{% 
\expandafter\def\expandafter\prevEdge@@\expandafter{\the\Edge@c} \\
\expandafter\Edge@c \expandafter{\objectEdge} \\
\Invisible@false \Hidden@false \\
\def\Leftness@{.5}\def\Upness@{.5}\%
As an optimisation \OBJECT@@ sets the edge type of all zero-sized objects to \zeroEdge.

\OBJECT@x cleans up the object by ensuring that it defines all the required methods: Essentially it terminates the box with the sequence \hbox{...} where each \hbox{...} is set to the method defined within the object creation environment (started with \hbox{ in \OBJECT@@ or possibly elsewhere). We use rather heavy expansion hacking with \toks@ to create the sequence so please look the other way... 😊

Methods: In addition to the “current object properties” for c (cf. 1.2.5) the following methods are set up by all objects:
\texttt{\textbackslash In\textbackslash visible\texttt{\textbackslash true} or \texttt{\textbackslash In\textbackslash visible\texttt{\textbackslash false}} whether object is invisible\textsuperscript{1.4i}
\texttt{\textbackslash Hidden\texttt{\textbackslash true} or \texttt{\textbackslash Hidden\texttt{\textbackslash false}} whether object is hidden\textsuperscript{1.4h}
\texttt{\textbackslash def\texttt{\textbackslash Left\textbackslash ness\tt\{}\langle\text{factor}\rangle\texttt{\textbackslash\}} the desired $L/(L+R)$
\texttt{\textbackslash def\texttt{\textbackslash Up\textbackslash ness\tt\{}\langle\text{factor}\rangle\texttt{\textbackslash\}} the desired $U/(D+U)$
\texttt{\textbackslash def\texttt{\textbackslash Drop\tt\{}\langle\text{factor}\rangle\texttt{\textbackslash\}} code that outputs the object, assuming $\texttt{\textbackslash Box\tt\{}\langle\text{factor}\rangle\texttt{\textbackslash\}}$ is of zero size and has the object displaced $<X,Y>$ inside—usually just \texttt{\textbackslash def’d to \texttt{\textbackslash styledbox\tt\{}\langle\text{factor}\rangle\texttt{\textbackslash\}}
\texttt{\textbackslash def\texttt{\textbackslash Connect\tt\{}\langle\text{factor}\rangle\texttt{\textbackslash\}} code that builds a connection from $p$ to $c$, assuming $\texttt{\textbackslash lastobject\tt\{}\langle\text{factor}\rangle\texttt{\textbackslash\}}$ contains the object.

It is important to \texttt{\textbackslash def} and not \texttt{\textbackslash let} the last four methods since the \TeX	extsuperscript{nique} used in \texttt{\textbackslash OBJECT\tt\{}x (and elsewhere) of passing them to surrounding scopes depends on it. The \texttt{\textbackslash Connect\tt\{} method should in turn setup several submethods as described in detail in 1.8.3.

Suitable defaults are set up by \texttt{\textbackslash OBJECT\tt\{} above which is why any box generating command can be used to construct objects as explained in note 1.4a below.

Here are the declarations:

\texttt{\textbackslash xynew\tt\{}\langle\text{factor}\rangle\texttt{\textbackslash if\textbackslash In\textbackslash visible\texttt{\textbackslash\}}
\texttt{\textbackslash xynew\tt\{}\langle\text{factor}\rangle\texttt{\textbackslash if\textbackslash Hidden\texttt{\textbackslash\}}
\texttt{\textbackslash xydef\texttt{\{}\langle\text{factor}\rangle\texttt{\textbackslash\}} code that outputs the object, assuming $\texttt{\textbackslash Box\tt\{}\langle\text{factor}\rangle\texttt{\textbackslash\}}$ is of zero size and has the object displaced $<X,Y>$ inside—usually just \texttt{\textbackslash def’d to \texttt{\textbackslash styledbox\tt\{}\langle\text{factor}\rangle\texttt{\textbackslash\}}
\texttt{\textbackslash xydef\texttt{\{}\langle\text{factor}\rangle\texttt{\textbackslash\}} code that builds a connection from $p$ to $c$, assuming $\texttt{\textbackslash lastobject\tt\{}\langle\text{factor}\rangle\texttt{\textbackslash\}}$ contains the object.

Notes

1.4a. An \langle object \rangle is built using \texttt{\textbackslash object\tt\{}\langle text\rangle\texttt{\}}. \texttt{\textbackslash object\tt\{} is initially defined as

\texttt{\textbackslash def\texttt{\{}\langle text\rangle\texttt{\textbackslash\}}
\texttt{\textbackslash hbox\tt\{}\langle\text{factor}\rangle\texttt{\textbackslash\}}
\texttt{\textbackslash let\tt\{}\langle\text{factor}\rangle\texttt{\textbackslash\}}

but may be redefined by options or the user. The \langle text \rangle should thus be in the mode required by the \texttt{\textbackslash object\tt\{} command—with the default \texttt{\textbackslash object\tt\{} shown above it should be in math mode.

Actually it is

\texttt{\textbackslash xydef\texttt{\{}objectbox\tt\{}\langle text\rangle\texttt{\textbackslash\}}
\texttt{\textbackslash hbox\tt\{}\langle\text{factor}\rangle\texttt{\textbackslash\}}
\texttt{\textbackslash let\tt\{}\langle\text{factor}\rangle\texttt{\textbackslash\}}

1.4b. An \langle object \rangle built from a \TeX box with dimensions $w \times (h+d)$ will have $L_c = R_c = w/2$, $U_c = D_c = (h+d)/2$, thus initially be equipped with the adjustment $!C$ (see note 1.4f). In particular: in order to get the reference point on the (center of) the base line of the original \langle \TeX box \rangle then you should use the \langle modifier \rangle $!C$; to get the reference point identical to the \TeX reference point use the modifier $!!L$.

\TeXnical remark: Any macro that expands to something that starts with a \langle box \rangle may be used as a \langle \TeX box \rangle here.

This is done by the parsing above.
1.4. OBJECTS

1.4c. Takes an object and constructs it, building a box; it is then processed according to the preceding modifiers. This form makes it possible to use any ⟨object⟩ as a TeX box (even outside of XY-pictures) because a finished object is always also a box.

This macro is the main entry point to the ⟨object⟩ parser. It furthermore initialises the modifier list and the previous object edge.

\begin{verbatim}
\xydef@object{\hbox\bgroup\resetStyle@\object@}
\xydef@object@{%
\edef\next@{={\DirectionfromtheDirection@}}\expandafter\toks@\next@
\plainxy@ \xyFN@\OBJECT@}
\end{verbatim}

The initial value of \toks@ (modifier) list is explained in note 1.4l below.

1.4d. Several ⟨object⟩s can be combined into a single object using the special command \composite with a list of the desired objects separated with *s as the argument. The resulting box (and object) is the least rectangle enclosing all the included objects.

First we collect all the objects smash on top of each other in box0 while we maintain the maximal extents in (DULR)p. Then we reset box0 to contain the same but with the right spacing around.

\begin{verbatim}
\xydef@\composite{\hbox\bgroup\composite@{#1}}
\xydef@\composite@#1#2{%
  D\NO@{#1}\ifx next@{}\empty\else\xwarning@{no variants of }
  \string\composite space allowed}\fi
  \global\setbox9=\hbox\bgroup
  \D@p=-\maxdimen \U@p=-\maxdimen \L@p=-\maxdimen \R@p=-\maxdimen
  \xyFN@\composite@i#2@}
\xydef@\composite@i{%
  \ifx \space@\next \expandafter\DN@\space@{%}
gobble spaces
  \else\ifx *\next \DN@ *{%}
gobble spaces
  \else\ifx @\next \DN@ @{%}
\composite@x \fi
  \else\DN@{%}
\composite@ii\next@\fi\fi
\xydef@\composite@ii#1#{\composite@iii{#1}}
\xydef@\composite@iii#1#2{%
  \setbox\z@=\object#1{#2}\
  \ifInvisible@ \setbox\z@h\z@=\z@ \dp\z@=\z@ \wd\z@=\z@ {\Drop@@}\fi
  \ifHidden@\else
  \ifdim\U@p<\U@c \U@p=\U@c \fi
  \ifdim\D@p<\D@c \D@p=\D@c \fi
  \ifdim\R@p<\R@c \R@p=\R@c \fi
  \ifdim\L@p<\L@c \L@p=\L@c \fi
  \fi
  \xyFN@\composite@iv}
\xydef@\composite@iv{%
  \ifx \space@\next \expandafter\DN@\space@{%}
gobble spaces
  \else \ifx @\next \DN@ @{%}
\composite@x \fi
  \else\let\next@=\composite@i \fi\fi
\xydef@\composite@x{%
\edef\tmp@{\egroup
\end{verbatim}
1.4. Take an entire \texttt{X Y}-picture and wrap it up as a box as described in \S 1.2.1. Makes nesting of \texttt{XY}-pictures possible: the inner picture will have its own zero point which will be its reference point in the outer picture when it is placed there.

This is simple exploiting the fact that \texttt{\endxy} actually sets up the extents of the ‘object’:

\begin{verbatim}
\def\xybox#1{\xy#1\endxy \Edge@c={\rectangleEdge}\computeLeftUpness@}
\end{verbatim}

1.4f. An object is \textit{shifted} a \langle vector \rangle by moving the point inside it which will be used as the reference point. This effectively pushes the object the same amount in the opposite direction.

Shifting uses the special value of $R_p$ and $U_p$ used while evalutaing the \langle modifier \rangle s. The fact that shifts like !C refer to the initial object’s size means that we should parse the \langle vector \rangle such that its \langle add op \rangle erations happen at modification time... hence \texttt{\xytotoks@} is used to delay execution.

\begin{verbatim}
\def\OBJECT@shift{% 
\let\xy@=\xytotoks@ \afterVECTORorEMPTY 
{\OBJECT@shift@}% 
{\addtotoks@{\X@c=-\L@c \advance\X@c\R@p \advance\X@c\L@p \Y@c=\U@p}% 
\OBJECT@shift@}}
\end{verbatim}

\textbf{Exercise 1.10:} What is the difference between the \langle position \rangle $0*{a}!DR$ and $0*!DR{a}$? \hspace{1cm} (p.576)

1.4g. A \langle size \rangle is a pair $<W,H>$ of the width and height of a rectangle. When given as a \langle vector \rangle these are just the vector coordinates, \textit{i.e.}, the \langle vector \rangle starts in the lower left corner and ends in the upper right corner. The possible \langle add op \rangle erations that can be performed are described in the
1.4. **OBJECTS**

In each case the \textlangle add op\textrangle may be omitted which invokes the “default size” for the particular \textlangle add op\textrangle:

<table>
<thead>
<tr>
<th>\textlangle add op\textrangle</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>grow</td>
</tr>
<tr>
<td>-</td>
<td>shrink</td>
</tr>
<tr>
<td>=</td>
<td>set to</td>
</tr>
<tr>
<td>+</td>
<td>grow to at least</td>
</tr>
<tr>
<td>-</td>
<td>shrink to at most</td>
</tr>
</tbody>
</table>

The defaults for the first three are set with the commands

\begin{verbatim}
\objectmargin \langle add op\rangle \{\dimen\}\n\objectwidth \langle add op\rangle \{\dimen\}\n\objectheight \langle add op\rangle \{\dimen\}\n\end{verbatim}

where \textlangle add op\textrangle is interpreted in the same way as above.

\begin{verbatim}
\xylet\objectmargin@=\jot
\xylet\objectwidth@=\z@\n\xylet\objectheight@=\z@
\xydef\objectmargin{\afterADDOP{\Addop@\objectmargin@}}
\xydef\objectwidth{\afterADDOP{\Addop@\objectwidth@}}
\xydef\objectheight{\afterADDOP{\Addop@\objectheight@}}
\end{verbatim}

The defaults for +=/−= are such that the resulting object will be the smallest containing/largest contained square.

**Exercise 1.11:** How are the objects typeset by the \textlangle pos\rangle tions “*+UR{\sum}” and “*+DL{\sum}” enlarged? (p.576)

**Bug:** Currently changing the size of a circular object is buggy—it is changed as if it is a rectangle and then the change to the \textit{R} parameter affects the circle. This should be fixed probably by a generalisation of the \textit{o} shape to be ovals or ellipses with horizontal/vertical axes.

The three cases distinguished by the parsing above are handled similarly: they insert the parsed/default vector into \textit{X,Y} in the modifications and then perform the operation at that time using the \texttt{xytotoks@} trick described in note 1.4f:

\begin{verbatim}
\xydef\OBJECT@change{%
\afterADDOP{%
\addEQ@\ifx \next
\addtotoks@{\X@c=c \D@c \advance\X@c\U@c \Y@c=c \L@c \advance\Y@c\R@c}%
\else
\addtotoks@{\X@c=\objectmargin@ \advance\X@c\X@c \Y@c=\X@c}%
\fi
\end{verbatim}
The real work is done by the following command: a hack using expansion tricks to make use of the \Addop@@ known now on the values in $X, Y$ at modification time.

It is clearly crucial that \Addop@@ expands to its action immediately! Also note that enlarging changes the initial box offset in the horizontal direction only, i.e., $R_p$.

Finally the code to interpret an ⟨add op⟩ used above: This simply parses it and creates a macro \Addop@@ that takes a control sequence and a parameter ⟨dimen⟩ as arguments, and expands directly to commands that perform the ⟨add op⟩ of the ⟨dimen⟩ on the control sequence:

<table>
<thead>
<tr>
<th>⟨add op⟩</th>
<th>effect of \Addop@@ $cs \ D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$+$</td>
<td>$cs \leftarrow cs + D$</td>
</tr>
<tr>
<td>$-$</td>
<td>$cs \leftarrow cs - D$</td>
</tr>
<tr>
<td>$=$</td>
<td>$cs \leftarrow D$</td>
</tr>
<tr>
<td>$+$</td>
<td>$cs \leftarrow \begin{cases} D &amp; \text{if } D \leq cs \ cs &amp; \text{if } D &gt; cs \end{cases}$</td>
</tr>
<tr>
<td>$-$</td>
<td>$cs \leftarrow \begin{cases} D &amp; \text{if } D \geq cs \ cs &amp; \text{if } D &lt; cs \end{cases}$</td>
</tr>
</tbody>
</table>

Furthermore \afterADDOP leaves the \next token set to = in the last three cases only (this is used to determine the right default value in the size changes above).

The \afterADDOP macro is relatively simple because ⟨add op⟩s don’t nest:
1.4. \textit{OBJECTS}

The work is done by the general \Addop@ \{ f \} \{ ± \} \{ ▷◁ \} \{ cs \} \{ D \} that defines

\[
\text{cs} \equiv \begin{cases} 
  f \times cs \pm D & \text{if } -(f \times cs \pm D) \bowtie D \\
  cs & \text{otherwise}
\end{cases}
\]

and also leaves the dimension in \dimen@.

1.4h. A \textit{hidden} object will be typeset but hidden from \textsc{Xy-pic} in that it won’t affect the size of the entire picture as discussed in §1.2.1.

This is handled by the \ifHidden@ conditional allocated with the methods.

1.4i. An \textit{invisible} object will be treated completely normal except that it won’t be typeset, \textit{i.e.}, \textsc{Xy-pic} will behave as if it was.

This is handled by the \ifInvisible@ conditional allocated with the methods.

1.4j. Setting the \textit{shape} of an object forces the shape of its edge to be as indicated. The kernel provides three shapes that change the edge, namely [·], [], and [o], corresponding to the outlines

\[
\begin{align*}
\times, & \quad \begin{array}{c}
\begin{array}{c}
\text{L} \\
\text{B}
\end{array}
\end{array}
\end{align*}
\]

and

where the \times denotes the point of the reference position in the object (the first is a point). Extensions can provide more shapes, however, all shapes set the extent dimensions \textit{L}, \textit{R}, \textit{D}, and \textit{U}.

The default shape for objects is [] and for plain coordinates it is [·].

Furthermore the \langle \textit{shape}\rangle s \{ r \}, \{ l \}, \{ u \}, and \{ d \}, are defined for convenience to adjust the object to the indicated side by setting the reference point such that the reference point is the same distance from the opposite of the indicated edge and the two neighbour edges but never closer to the indicated side than the opposite edge, \textit{e.g.}, the object \{ r \} \hbox{Wide text} has reference
Note: Extensions can add new \langle shape \rangle object \langle modifier \rangle s which are then called \langle style \rangle s. These will always be either of the form \[ \langle keyword \rangle \] or \[ \langle character \rangle \langle argument \rangle \]. Some of these \langle style \rangle s do other things than set the edge of the object.

A “simple shape” is just a control sequence \shape \[ \langle shape \rangle \] setting the appropriate edge. When such a \[ \langle shape \rangle \] modifier is encountered then we expand this control sequence onto the modifier queue, unless a control sequence \style \[ \langle shape \rangle \] exists: then that is expected to do it (and whatever else is required).

Add more simple shapes by defining more commands like these and proceed with coding the \Edge command as described in §1.8.2.

Alternatively it is a “complex shape” of which none are defined in the kernel but some options like
more variation... It is characterised by its first token and the rest of the contents of the []s is the argument (remember: no {} characters!); it is searched for as control sequence \[(shape)\ldots\] overridden in the same way by \*stylechar@(shape)@.

\def\OBJECT@shapei[#1#2]{\DN@{shape [#1\ldots]}}%  
\expandafter\let\expandafter\next\csname\codeof\next@\endcsname
\ifx\next\relax\DN@{*stylechar@#1@}%  
\expandafter\let\expandafter\next\csname\codeof\next@\endcsname
\ifx\next\relax\DNii@{shape [#1#2]}%  
\xywarning@{illegal [shape] ignored: \codeof\nextii@ space not defined}%%  
\DN@{\xyFN@\OBJECT@}%%  
\else%% Delete this: \expandafter\addtotoks@\expandafter{\next{#2}}%  
\DN@{\next{#2}\xyFN@\OBJECT@}%%  
\fi
\else
\expandafter\addtotoks@\expandafter{\next{#2}\xyFN@\OBJECT@}%%  
\fi \next@}

Bug: The above is messy!

1.4k. While typesetting an object, some of the properties are considered part of the ‘current object style’. Initially this means nothing but some of the ⟨style⟩s defined by extensions have this status, e.g., colours [red], [blue] say, using the xycolor extension, or varying the width of lines using xyline. Such styles are processed left-to-right: for example,

*[red][green]=[NEW][blue]{A}  

will typeset a blue A and define [NEW] to set the colour to green (all provided that xycolor has been loaded, of course).

The method of requesting a special effect is via the ⟨shape⟩ modifiers. However the code required for implementation is quite different. In particular it must allow for the different ways in which the ⟨driver⟩ may implement the \specials required to activate, and turn off, the effect. Another difficulty is that the ⟨style⟩ applies also to nested sub-parts of any ⟨object⟩. This means that the code cannot be wholly delayed until after the ⟨object⟩ has been constructed.

Usually some value has to be stored, so as to be accessible to sub-parts, though the \special commands cannot be fully constructed until the ⟨object⟩ is completely known and ready to be \Drop@@ or \Connect@@'ed.

Finally, the overhead in using \special commands can be quite high with some dvi-drivers. Thus it is not best that every ⟨style⟩ request generate a \special command; indeed two \special commands, since each style-change also requires a command to revert to the previous style, after the change is no longer applicable.

Thus, as with ⟨shape⟩s, all the style-change requests are stored until the ⟨object⟩ is ready to be \Drop@@'ed. This also helps with getting the sequencing correct when later requests compound on, or override, earlier requests.

When the ⟨style⟩ requests are processed, after the ⟨object⟩ has been constructed, information of the form ⟨control-word⟩{(data)} is added to a global list called \preXY@style@. Similar information is added to another global list \postXY@style@, for the purpose of reverting the style parameters, within the ⟨driver⟩-file, to their previous values prior to the current ⟨object⟩.
Both lists are initially \langle empty \rangle at the start of an \langle object \rangle. The contents of these lists are processed when the macros \texttt{\xypre@Style@@} and \texttt{\xypost@Style@@} are encountered in the expansion of \texttt{\styledboxz@}.

The \langle data \rangle contains the code needed to implement the requested style-change, with the particular \langle driver \rangle being used.

The \langle control-word \rangle is a macro whose expansion could be fixed, or could depend upon what other style-change requests are made for this \langle object \rangle. Its purpose is to decide whether or not the \langle data \rangle is actually required. If so, it is added to the token-list \texttt{\styletoks@} for later use.

Whether the \langle control-word\{\langle data\} \rangle tokens are prepended or appended to the global lists depends upon the nature of the style-change requested. For example, with \texttt{*[red][green]} \langle object \rangle the information \texttt{\{set-color\{\langle green\}}} could be appended to \texttt{\xypre@Style@@}, with \texttt{\{revert-to-color\{\langle red\}}} prepended to \texttt{\xypost@Style@@}. Then the dvi-output would set the color to \langle red \rangle then \langle green \rangle and restore it first to \langle red \rangle then \langle black \rangle.

\textbf{To Do:} A more sophisticated implementation could instead prepend \texttt{\{set-color\{\langle green\}}} to \texttt{\xypre@Style@@} and append \texttt{\{revert-to-color\{\langle red\}}} to \texttt{\xypost@Style@@}. Now the first \texttt{\{set-color\}} encountered adds its \langle data \rangle to \texttt{\styletoks@} then rebinds \texttt{\{set-color\}} to \texttt{\eat@}, to ignore subsequent occurrences. Similarly \texttt{\{revert-to-color\}} is rebound so that only the is restoration to \langle black \rangle is retained.

This latter strategy is preferred when the overhead for a \texttt{\special} command is high, within the \langle driver \rangle’s output. Indeed the \texttt{POSTSCRIPT} back-end is even more efficient. Since \texttt{POSTSCRIPT} is itself a programming language, the \langle data \rangle can be just the \texttt{POSTSCRIPT} code. A single \texttt{\special} command is sufficient to accommodate \textit{all} the style-changes for an individual \langle object \rangle. This is achieved by rebinding \texttt{\xydoprestyles@@} and \texttt{\xydopoststyles@@} to include the \langle data \rangle within an appropriately prepared \texttt{\special} command.

The \langle data \rangle to be used is collected in the token-list \texttt{\styletoks@}. If this is \langle empty \rangle for an \langle object \rangle then the style processing is skipped completely. For safety, in case multiple occurrences of \texttt{\xypre@Style@@} have somehow slipped into the token stream, these and \texttt{\xypost@Style@@} are bound to \texttt{\relax} until the \langle object \rangle is finished. Indeed \texttt{\xypost@Style@@} is always bound to \texttt{\relax} unless \texttt{\styletoks@} is non-empty.
Some extensions use the following macros to add style ⟨data⟩ to the global macros. Here code is added sequentially to \preXY@style@ so that it is acted upon in the order of occurrence (FIFO) of the ⟨shape⟩ modifiers. Code is added to \postXY@style@ in reverse order (LIFO), so that each addition to \preXY@style@ can be closed off, if necessary, in correctly nested sequence.

The \preStyle@ and \postStyle@ temporarily hold style ⟨data⟩. The macro \preXYStyle@ is used to check whether there is any ⟨data⟩ to process. Normally this is just an alias for \preStyle@, but some back-ends may use the two control-names differently.
Initially styles are activated but can be suppressed using \Unloadstyle@.

Global macros are used, so that the same styles can be reused by successive objects without having to re-interpret ⟨shape⟩ modifiers, as described next.

**Saving styles:** Once specified for an ⟨object⟩, the collection of ⟨style⟩s can be assigned a name, using [≡⟨word⟩]. Then ⟨⟨word⟩⟩ becomes a new ⟨style⟩, suitable for use with the same or other ⟨objects⟩s. Use a single ⟨word⟩ built from ordinary letters. If ⟨⟨word⟩⟩ already had meaning the new definition will still be imposed, but the following type of warning will be issued:

Xy-pic Warning: Redefining style [⟨word⟩]

The latter warning will appear if the definition occurs within an \xymatrix. This is perfectly normal, being a consequence of the way that the matrix code is handled. Similarly the message may appear several times if the style definition is made within an \ar.

The following illustrates how to avoid these messages by defining the style without typesetting anything.

\setbox0=\hbox{\xy\drop[OrangeRed][≡A]{}}\endxy

Note 1: The current colour is regarded as part of the style for this purpose.

Note 2: Such namings are global in scope. They are intended to allow a consistent style to be easily maintained between various pictures and diagrams within the same document.

If the same ⟨style⟩ is intended for several ⟨object⟩s occurring in succession, the [|*] ⟨modifier⟩ can be used on the later ⟨object⟩s. This only works when [|*] precedes any other ⟨style⟩ modifiers; it is local in scope, recovering the last ⟨style⟩s used at the same level of TeX grouping.
Defining new effects  Allow new effects to be declared and a default action provided. The name is constructed from #1 and the action to be taken is passed as #2. If #3 is non-empty then overwrite any existing definition for a control sequence of the same name.

Support for a new ⟨style⟩ takes the following form, in which \thestyle@ expands to the current value of the ⟨style⟩ information to be maintained. If this is kept in a ⟨count⟩ or ⟨dimen⟩ register, named \xystyle@ say, then replace \thestyle@ by \the\style@. (However due to the large number of ⟨dimen⟩s already allocated, it is preferable to keep a text form of the value in \thestyle@.)

In the above the macros \modifystyle@@ and \stylespecial@ are adapted to the particular ⟨style⟩ information, see below. Typically ⟨how⟩ is actual T\TeX code and \modifystyle@@ is simply \literal@ to execute this code. Typesetting is done by \stylespecial@#1 which is typically of the form \special{⟨key⟩ #1}.

The \transformstyle@ is required in case a dvi-(driver) needs information in a form that is dif-
different to how \texttt{XY-pic} maintains it in \texttt{\thestyle@}. A \langle\texttt{driver}\rangle file should define \langle\texttt{driver}\rangle@\texttt{style@@@} to perform the transformation and place its result in \texttt{\next@}. Install this macro using \texttt{\let\transformstyle@=}\langle\texttt{driver}\rangle@\texttt{style@@@} within the \langle\texttt{driver}\rangle installation macro.

Next we discuss \texttt{\applystyle@}; which uses \texttt{\prestyle@}, \texttt{\prestyle@@@}, \texttt{\poststyle@} and \texttt{\poststyle@@@}. There are two strategies here, affecting how much information is placed into the dvi-file, indicated by the alternative expansions given above. The appropriateness for a given style must be decided by the author of the style-option.

Suppose several modifications are made to a particular style parameter. We must build a list of commands \texttt{\prestyle@{(value)}} to be executed later, which will recover the required value. Also we build a list of commands \texttt{\poststyle@{(value)}} to reset to the previous value.

In most cases it is only the result of the modifications that need be placed into the dvi-file. Since we do not know how many, if any, more modifications follow we could store each new result after the previous, so that its effect will override the previous value. This is building a FIFO list for the \texttt{\prestyle@} commands, with a FILO list for the \texttt{\poststyle@} commands:

```latex
\xydef\applyFIFOstyle@#1#2#3#4{\bgroup
\styletoks@={\egroup\gdef\preXY@style@}\
\expandafter\toks@\expandafter{\expandafter#1\expandafter{#2}}\expandafter\expandafter{\expandafter#3\expandafter{#4}}
\gdef\postXY@style@)
\the\styletoks@}
```

Use this via: \texttt{\applyFIFOstyle@\prestyle@{(\thestyle@)}\poststyle@{(\prevstyle@)}.}

With this method each modification places two entries into the dvi-file; there is no need for separate \texttt{\prestyle@} and \texttt{\poststyle@@@} macros.

Perhaps a more efficient strategy is to build a LIFO list for the \texttt{\prestyle@} commands. This way the required value is encountered first. This first instance should then cancel all subsequent instances of \texttt{\prestyle@}. This is where the need comes for a \texttt{\prestyle@@@}. Accompanying this is a FIFO list for the \texttt{\poststyle@} commands, with a corresponding \texttt{\poststyle@@@} to kill subsequent instances:

```latex
\xydef\applyLIFOstyle@#1#2#3#4{\bgroup
\styletoks@={\egroup\gdef\preXY@style@}\
\expandafter\toks@\expandafter{\expandafter#1\expandafter{#2}}\expandafter\expandafter{\expandafter#3\expandafter{#4}}
\gdef\postXY@style@)
\the\styletoks@}
```

Use this via: \texttt{\applyLIFOstyle@\prestyle@{(\thestyle@)}\poststyle@{(\prevstyle@)}.}

The advantage here is that only two pieces of data need be output to the dvi-file, no matter how many modifications are requested.
1.4. Setting the current direction is simply pretending for the typesetting of the object (and the following (modifier)s) that some connection set it – the (empty) case just inherits the previous direction.

The code just calls the general ⟨direction⟩ parser below:

1.4l. Diagonal directions. It is particularly easy to set ⟨diag⟩onal directions:

\begin{center}
\begin{tikzpicture}
\node {\textbullet} coordinate (c);
\draw (c) -- node[above]{$u$} (c + 1,0) coordinate (u);
\draw (c) -- node[above]{$r$} (c + 0,1) coordinate (r);
\draw (c) -- node[below]{$d$} (c + 0,-1) coordinate (d);
\draw (c) -- node[left]{$l$} (c - 1,0) coordinate (l);
\draw (c) -- node[left]{$u$} (c + 0,1) coordinate (ul);
\draw (c) -- node[right]{$r$} (c + 1,0) coordinate (ur);
\draw (c) -- node[below]{$d$} (c + 0,-1) coordinate (dl);
\draw (c) -- node[below]{$l$} (c - 1,0) coordinate (dr);
\end{tikzpicture}
\end{center}
More to the point, ⟨diag⟩onals are stored internally as

\[ \begin{array}{c c c}
1 & 2 & 3 \\
5 & 0 & 4 \\
6 & 7 & 0
\end{array} \]

Expanding \texttt{\afterDIAG\{stuff\}\langle diag\rangle} will result in \texttt{\count@} being set to the ⟨diag⟩ code (not changed in case the ⟨diag⟩ is ⟨empty⟩) before expanding ⟨stuff⟩.

\begin{verbatim}
\def\afterDIAG#1{\def\afterDIAG@{#1}\xyFN@\DIAG@}
\xydef@\DIAG@{\ifx d\next \DN@ d{\count@=1 \xyFN@\DIAG@}%
\else\ifx r\next \DN@ r{\count@=3 \xyFN@\DIAG@}%
\else\ifx u\next \DN@ u{\count@=5 \xyFN@\DIAG@}%
\else\ifx l\next \DN@ l{\count@=7 \xyFN@\DIAG@}%
\else \let\next@=\afterDIAG@ 
\fi\fi\fi\fi \next@}
\xydef@\DIAG@@{\ifcase\count@ 
\or \xy@@{dlDirection@\dimen@}%
\or \xy@@{drDirection@\dimen@}%
\or \xy@@{urDirection@\dimen@}%
\or \xy@@{ulDirection@\dimen@}%;
\or \xy@@{rDirection@\dimen@}%
\or \xy@@{uDirection@\dimen@}%
\or \xy@@{ldDirection@\dimen@}%;
\or \xy@@{ldDirection@\dimen@}%;
\or \xy@@{ldDirection@\dimen@}%;
\or \xy@@{ldDirection@\dimen@}%;
\or % 8 is legal and means change nothing
\else\xybug@{impossible <diag> number}\fi \next@}
\xydef@\DIAG@@@#1#2#3#4{\ifx #1\next \count@=#2\DN@#1{\afterDIAG@}%
\else \ifx #3\next \count@=#4\DN@#3{\afterDIAG@}%
\else \let\next@=\afterDIAG@ \fi\fi}
\def\Directionfromdiag@{\ifcase\count@
\xy@@{dlDirection@\dimen@}%;
\or \xy@@{dlDirection@\dimen@}%;
\or \xy@@{drDirection@\dimen@}%;
\or \xy@@{drDirection@\dimen@}%;
\or \xy@@{urDirection@\dimen@}%;
\or \xy@@{urDirection@\dimen@}%;
\or \xy@@{ulDirection@\dimen@}%;
\or \xy@@{ulDirection@\dimen@}%;
\or \xy@@{ldDirection@\dimen@}%;
\or % 8 is legal and means change nothing
\else\xybug@{impossible <diag>}\fi \DIRECTIONempty@false \xyFN@\DIRECTION@i}
\end{verbatim}

The action in case of a ⟨diag⟩ is simply to pick the right direction setup routine according to the encoding, getting the ⟨diag⟩ from \texttt{\count@} and the length of the \texttt{d} vector from \texttt{\dimen@}:

\begin{verbatim}
\def\Directionfromdiag@{\ifcase\count@
\xy@@{dlDirection@\dimen@}%;
\or \xy@@{dlDirection@\dimen@}%;
\or \xy@@{drDirection@\dimen@}%;
\or \xy@@{drDirection@\dimen@}%;
\or \xy@@{urDirection@\dimen@}%;
\or \xy@@{urDirection@\dimen@}%;
\or \xy@@{ulDirection@\dimen@}%;
\or \xy@@{ulDirection@\dimen@}%;
\or \xy@@{ldDirection@\dimen@}%;
\or % 8 is legal and means change nothing
\else\xybug@{impossible <diag>}\fi \DIRECTIONempty@false \xyFN@\DIRECTION@i}
\end{verbatim}

**Vector directions.** Alternatively \texttt{v\langle vector\rangle} sets the direction as if the connection from 0 to the ⟨vector⟩ had been typeset except that the \texttt{origin} is assumed zero such that directions \texttt{v\langle x,y\rangle} mean the natural thing, \textit{i.e.}, is the direction of the connection from \texttt{(0,0)} to \texttt{(x,y)}.
The action for a \( v \) reads a \( \langle \text{vector} \rangle \) and sets the direction accordingly using some expansion hackery to propagate it out. The \emph{origin} is cleared locally to make \( v(x,y) \) behave as it should.

\begin{verbatim}
\xydef@DIRECTION@v{\enter@{\cfromthec@ \X@origin=\the\X@origin \Y@origin=\the\Y@origin
 \X@p=\the\X@p \Y@p=\the\Y@p}\
 \X@origin=\z@ \Y@origin=\z@}\
\afterVECTORorEMPTY
\{\xy@{\X@p=\z@ \Y@p=\z@ \setupDirection@ \leave@}\%
\DIRECTIONempty@false \xyFN@DIRECTION@i}\%
\{\xy@\leave@ \xyerror@{<vector> expected after v}\%
\DIRECTIONempty@false \xyFN@DIRECTION@i}\%
\end{verbatim}

\textbf{Complex direction.} In case the direction is not as simple, you can construct \( \{ \langle \text{pos} \rangle \langle \text{decor} \rangle \} \) that sets up \( p \) and \( c \) such that \( pc \) has the desired direction. \textbf{Note:} that you must use the \((\ast\ldots\ast)\) form if this is to appear in an object \langle \text{modifier} \rangle!

The code for this is rather like the one for \langle \text{vector} \rangle s.

\begin{verbatim}
\xydef@DIRECTION@group#1{\xy@@{\begin{group}\xy@@ix@{#1}\xy@@{\plainxy@\expandafter\POS\the\toks9\relax
 \setupDirection@\edef
\next@{\endgroup \DirectionfromtheDirection@}\next@}\%
\DIRECTIONempty@false \xyFN@DIRECTION@i}\%
\{\xy@\leave@ \xyerror@{<vector> expected after \text{group}}\%
\DIRECTIONempty@false \xyFN@DIRECTION@i}\%
\end{verbatim}

\textbf{Exercise 1.12:} What effect is achieved by using \langle \text{modifier} \rangle s \( v/1\text{pc/} \) and \( v/-1\text{pc/} \)? (p.576)

1.4. \textbf{OBJECTS} Once the initial direction is established as either the last one or an absolute one then the remainder of the \langle \text{direction} \rangle is interpreted.

Adding a single \( ^\_ \) or \( _\_ \) denotes the result of rotating the default direction a right angle in the positive and negative direction, \textit{i.e.}, anti-/clockwise, respectively. \textbf{Note:} Do \textit{not} use \( ^^ \) but only \( _\_ \) to reverse the direction!

A trailing \( :\langle \text{vector} \rangle \) is like \( v\langle \text{vector} \rangle \) but uses the \langle \text{direction} \rangle to set up a standard square base such that \( :a(0,1) \) and \( :a(0,-1) \) mean the same as \( :a(90) \) and \( :a(-90) \) and as \( ^\_ \) and \( _\_ \), respectively.

\begin{verbatim}
\xydef@DIRECTION@i{\ifx ^\next \DN@ ^{\xy@^{\aboveDirection@\xydashl@}\
\DIRECTIONempty@false \xyFN@DIRECTION@i}\%
\else\ifx _\next \DN@ _{\xy@_{\belowDirection@\xydashl@}\
\DIRECTIONempty@false \xyFN@DIRECTION@i}\%
\else\ifx :\next \DN@ :{\xy@{:}{\enter@{\cfromthec@ \basefromthebase@ \X@p=\the\X@p \Y@p=\the\Y@p}\
 \X@origin=\z@ \Y@origin=\z@}
 \X@xbase=\cosDirection\xydashl@ \Y@xbase=\sinDirection\xydashl@
 \X@ybase=-\Y@xbase \Y@ybase=\X@xbase}\
\afterVECTORorEMPTY
\{\xy@{\X@p=\z@ \Y@p=\z@ \setupDirection@ \leave@}\%
\DIRECTIONempty@false \xyFN@DIRECTION@i}\%
\{\xy@\leave@ \xyerror@{<vector> expected after :}{}\%
\DIRECTIONempty@false \xyFN@DIRECTION@i}\%
\else
\end{verbatim}
### 1.5 Decorations

Decorations are actual \TeX{} macros that decorate the current picture in manners that depend on the state. They are allowed after the \emph{position} either of the outer \texttt{\begin{xy}} \dots \texttt{\end{xy}} or inside \texttt{\{\ldots\}}. The possibilities are given in figure 1.6 with notes below.

Most options add to the available \emph{decor}, in particular the v2 option loads many more since \texttt{XY-pic} versions prior to 2.7 provided most features as \emph{decor}.

#### Simple decorations: \texttt{\POS} and \texttt{\afterPOS} have already been defined; the following are just simple applications of previously defined commands:

\begin{verbatim}
\let\next@=\afterDIRECTION@ \fi \fi \fi \next@}
\end{verbatim}

#### Exercise 1.13: What effect is achieved by using \emph{modifier}s \texttt{v/1pc/:(1,0)} and \texttt{v/-1pc/__}? 

(p.576)
Notes

1.5a. Saving and restoring allows `excursions' where lots of things are added to the picture without affecting the resulting \text{XY-pic} state, \emph{i.e.}, \text{c, p, and base}, and without requiring matching \{\}s. The independence of \{\} is particularly useful in conjunction with the \texttt{\afterPOS} command, for example, the definition

\begin{verbatim}
def\ToPOS{\save\afterPOS{\POS**{}?>*@2{>}*@{-}\restore};p,}
\end{verbatim}

will cause the code \texttt{\ToPOS(pos)} to construct a double-shafted arrow from the current object to the \langle pos \rangle (computed relative to it) such that \texttt{\xy *{A} \ToPOS +<10mm,2mm>\endxy} will typeset the picture $\xrightarrow{A}$.

\textbf{Note:} Saving this way in fact uses the same state as the \{\} ‘grouping’, so the code \texttt{p\{save\}, \ldots \{restore\}} will have $c = p_1$ both at the \ldots and at the end!

1.5b. One very tempting kind of \LaTeX\ commands to perform as \langle decor \rangle is arithmetic operations on the \text{XY-pic} state. This will work in simple \text{XY-pic} pictures as described here but be warned: \emph{it is not portable} because all \text{XY-pic} execution is indirect, and this is used by several options in nontrivial ways. Check the \LaTeX\-nical documentation \[17\] for details about this!

Macros that expand to \langle decor \rangle will always do the same, though.
\LaTeX\ hackers like the author may enjoy changing the \text{XY-pic} state directly using \langle decor \rangle of the form \texttt{\xy@\{\langle id \rangle\}\{\langle code \rangle\}...}

1.5c. \texttt{\xyecho} will turn on echoing of all interpreted \text{XY-pic} \langle pos \rangle characters. \textbf{Bug:} Not completely implemented yet. \texttt{\xyverbose} will switch on a tracing of all \text{XY-pic} commands executed, with line numbers. \texttt{\xytracing} traces even more: the entire \text{XY-pic} state is printed after each modification. \texttt{\xyquiet} restores default quiet operation.

The trick is to replace the \texttt{\xy@} command such that it calls the ‘normal’ one between writing out a trace message and the state.
1.5d. Ignoring means that the (pos) (decor) is still parsed the usual way but nothing is typeset and the Xy-pic state is not changed.

We ignore in a group to ensure that nothing done inside ‘leaks’ to the outside.

1.5e. It is possible to save an intermediate form of commands that generate parts of an Xy-pic to a file such that subsequent typesetting of those parts is significantly faster: this is called compiling. The produced file contains code to check that the compiled code still corresponds to the same (pos)(decor) as well as efficient Xy-code to redo it; if the (pos)(decor) has changed then the compilation is redone.

There are two ways to use this. The direct is to invent a (name) for each diagram and then embrace it in \xycompileto{⟨name⟩}{{...}} – this dumps the compiled code into the file ⟨name⟩.xyc. When many diagrams are compiled then it is easier to add \xycompile{{...}} around the (pos)(decor) to be compiled. This will assign file names numbered consecutively with a (prefix) which is initially the expansion of \jobname-- but may be set with

\CompilePrefix{(prefix)}

This has the disadvantage, however, that if additional compiled Xy-pictures are inserted then all subsequent pictures will have to be recompiled. One particular situation is provided, though: when used within constructions that typeset their contents more than once (such as most \AMS-LaTEX equation constructs) then the declaration

\CompileFixPoint{(id)}
can be used inside the environment to fix the counter to have the same value at every passage.

It is done by just writing all \xy@-commands to the file. The file establishes the correct input mode through use of the appropriate commands itself.

Recompilation is done by just writing all \xy@-commands to the file. The file establishes the correct input mode and terminates itself; after it has been finished it is simply reread to actually get the drawing done in the document.
Hack1: The \ifxysaving@ can never be locally switched off! Anyway it is used to allow a gross hack avoiding building a queue in the matrix option that will generate too long lines!!

Hack2: \xysave@@toksix@ is not doing the catcode jive because it can never be invoked while loading a file (knock, knock ☹).

The initial command in all .xyc files check that this is the right file and that neither the version of XY-pic nor the user’s code has changed:

```
\xydef@\xycompiled#1#2#3#4{\DN@{#1}\edef\next@{\codeof\next@}\ifx\next@\compilename@@\else
\xywarning@{This file does not contain the result of
string\xycompileto{\compilename@@}{...}^^J
but of string\xycompileto{\next@}\fi
\edef\next{Xy-pic \xyversion}\DN@{#3}\ifx\next\nextii@ \xycatcodes \outlines@@\relax\readoutline@\else \def\xyrecompile@@{recompiling to}\endinput \fi
\else \def\xyrecompile@@{Xy-pic version change - recompiling}\endinput \fi}
```

Finally the fix-point thing.

```
\xydef@\CompileFixPoint#1{%
\expandafter\ifx\csname#1FIX@@\endcsname\relax
\xdef\xycompileno@@{%
\else
\xdef\xycompileno@@{\csname#1FIX@@\endcsname}\fi}
```

Finally, when many ‘administrative typesetting runs’ are needed, e.g., readjusting L\LaTeX\ cross references and such, then it may be an advantage to not typeset any XY-pictures at all during the intermediate runs. This is supported by the following declarations which for each compilation creates a special file with the extension .xyd containing just the size of the picture:

```
\MakeOutlines
\OnlyOutlines
\ShowOutlines
\NoOutlines
```

The first does no more. The second uses the file to typesets a dotted frame of the appropriate size instead of the picture (unless the picture has changed and is recompiled, then it is typeset as always and the .xyd file is recreated for subsequent runs). The third shows the outlines as dotted rectangles. The last switches outline processing completely off.

The implementation is all handled by \outlines@@ which is called at strategic places in the compilation macros above. It just expands to nothing when the state is ‘No’, the first argument when it is ‘Make’, and the second when it is ‘Only’.

```
\xydef@\outlines@no#1#2{}
```
1.6. Kernel object library

In this section we present the library objects provided with the kernel language—several options add more library objects. They fall into three types: Most of the kernel objects (including all those usually used with ** to build connections) are directionals, described in §1.6.1. The remaining kernel library objects are circles of §1.6.2 and text of §1.6.3.

1.6.1 Directionals

In this section we present the library objects provided with the kernel language—several options add more library objects. They fall into three types: Most of the kernel objects (including all those usually used with ** to build connections) are directionals, described in §1.6.1. The remaining kernel library objects are circles of §1.6.2 and text of §1.6.3.
The kernel provides a selection of *directionals*: objects that depend on the current direction. They all take the form

$$\textbackslash \text{dir}(\text{dir})$$

to typeset a particular ⟨dir⟩ectional object. All have the structure

$$⟨\text{dir}⟩ \rightarrow ⟨\text{variant}⟩\{⟨\text{main}⟩\}$$

with ⟨variant⟩ being ⟨empty⟩ or one of the characters ^_23 and ⟨main⟩ some mnemonic code.

We will classify the directionals primarily intended for building connections as *connectors* and those primarily intended for placement at connection ends or as markers as *tips*.

Figure 1.7 shows all the ⟨dir⟩ectionals deﬁned by the kernel with notes below; each ⟨main⟩ type has a line showing the available ⟨variant⟩s. Notice that only some variants exist for each ⟨dir⟩—when a nonexistent variant of a ⟨dir⟩ is requested then the ⟨empty⟩ variant is used silently. Each is shown in either of the two forms available in each direction as applicable: connecting a ⊙ to a □ (typeset by **\textbackslash dir(\text{dir})**) and as a tip at the end of a dotted connection of the same variant (i.e., typeset by the ⟨pos⟩ **\textbackslash dir(\text{variant}⟩\{.⟩ \Rightarrow \ast\textbackslash dir(\text{dir})⟩**).

As a special case an entire ⟨object⟩ is allowed as a ⟨dir⟩ by starting it with a *: \textbackslash dir* is equivalent to \textbackslash object.

**Setup:** \textbackslash dir starts an ⟨object⟩ and passes control to a ‘ﬁnisher’ named \textbackslash dir(⟨variant⟩\{⟨main⟩\}) otherwise to the one corresponding to an ⟨empty⟩ ⟨variant⟩. The kernel ones described here have in common that they make use of the generic \textbackslash straight@ defined in §1.8.3.

```latex
\xydef\dir{\hbox{bgroup}\xyFN\dir@i}
\xydef\dir@i{\ifx \ast\next \DN@*{\resetStyle\object@}{\fi \let\next@=\dir@ii \fi \next@}
\xydef\dir@ii#1#{\dir@{#1}}
\xydef\dir@#1#2{\DN@{dir#1{#2}}{\expandafter\let\expandafter\next\csname\codeof\next@\endcsname}
\if\next@\relax\DN@{dir{#2}}{\expandafter\let\expandafter\next\csname\codeof\next@\endcsname}
\if\next@\relax\DN@{dir#1{#2}}{\xyerror{illegal \langle \text{dir} \rangle: \langle \text{code of next} \rangle \space not defined}}{}
\let\next=\no@ \fi \fi}\next}
```

**Notes**

1.6a. You may use \textbackslash dir{} for a “dummy” directional object (in fact this is used automatically by **{}). This is useful for a uniform treatment of connections, e.g., making the ? ⟨pos⟩ able to find a point on the straight line from p to c without actually typesetting anything.

Uses an empty dropping, the \textbackslash no@ connection. All the variants are defined for optimisation reasons and it is also named \textbackslash dir{ } to allow spurious spaces:

```latex
\xydefcsname@{dir{}}{\no@}
\xyletcsnamecsname@{dir0{}}{dir{}}
\xyletcsnamecsname@{dir1{}}{dir{}}
\xyletcsnamecsname@{dir^{}{}}{dir{}}
\xyletcsnamecsname@{dir_{}}{dir{}}
```
1.6. KERNEL OBJECT LIBRARY

Plain connectors

Plain tips

Constructed tips

Figure 1.7: Kernel library (dir)ectionals
1.6b. The \textit{plain connectors} group contains basic directionals that lend themselves to simple connections.

The bulk of the code is in fact in the description of these. First each of the three types—lines, dots, and squiggles—then the code for doubling and tripling.

\textbf{Lines:} A single \texttt{\dir{-}} object is a dash in the current direction: build box with character \texttt{C} of the semidirectional \texttt{\xydashfont}; use the characters natural width \texttt{w} and construct a height/depth from \(d = |\sin(\text{Direction})| \text{em}\) (where \texttt{1em} is the dash length, \textit{cf. xydash10.mf}) as follows:

\begin{align*}
C & \quad L & R & D & U & \text{flip if} \\
0\ldots30 & 0 & w & 0 & d & dY < 0 \\
31\ldots63 & 0 & w & d & 0 & dY > 0 \\
64\ldots95 & 0 & w & d & 0 & dX < 0 \\
96\ldots127 & 0 & w & 0 & d & dX < 0
\end{align*}

where \textit{flip} means shift the box opposite vertically and horizontally, \textit{i.e.}, \((L, R, D, U) := (R, L, U, D)\), and then lower the box \(D – U\).

\textit{Procedure:} (1.5142) Compute \texttt{d}, (1.5143) set \(D, U\), and flip condition, (1.5147) build box to get \texttt{w, L, R}, (1.5149) dump box that is flipped if condition holds, and (1.5158) finally setup the required parameters properly.


\edef\tmp@{\egroup \U@c=\the\U@c \D@c=\the\D@c \L@c=\the\L@c \R@c=\the\R@c}%
\tmp@
\Edge@c={\rectangleEdge}%%%%%
\ifdim\z@<\U@c \def\Upness@{1}\else \def\Upness@{0}\fi
\ifdim\z@<\L@c \def\Leftness@{1}\else \def\Leftness@{0}\fi
\def\Drop@@{\styledboxz@}\def\Connect@@{\solid@}
\xydef@line@@{{\xydashfont\SemiDirectionChar/}}

\textbf{Bug:} \line@ should allow the size of the object to be changed after typesetting—this should make \Connect@@ do dashing. Hm.

As mentioned above a dash will ‘Connect’ to make lines by using rules when strictly horizontal or vertical. This is controlled by enabling or disabling the test \ifjusthvtest@ discussed below.

Finally, we give the algorithm for ‘spreading’ the dashes along a solid line: just add an extra dash so they always overlap (see §1.8.3 for a proper definition of the requirements to spreading).

By default XY-pic will typeset horizontal and vertical \dir{-} connections using \TeX{} rules. Unfortunately rules is the feature of the DVI format most commonly handled wrong by DVI drivers. Therefore XY-pic provides the \texttt{\langle decor\rangle} \NoRules \UseRules that will switch the use of such off and on.

They simply redefine the conditional used to select typesetting with rules in \solid@ above:

\xylet@\ifjusthvtest@=\ifdim
\xydef@\NoRules{\let\ifjusthvtest@=\iffalse}
\xydef@\UseRules{\let\ifjusthvtest@=\ifdim}

The actual typesetting essentially means calling \drop@ to box with a rule of the appropriate length and with line width set to that of \xydashfont (as stored in \xydashw@).
\def\Cbreak@@{%
% \W@{VRULE Cbreak}\
% 
\connectRestore@ \swap@\lastbreak@@\swap@
\solidvrule@typeset \edef\lastbreak@@{\cfromthec@}\Creset@@%
\def\Clast@@{%
% \W@{VRULE Clast}%
% \let\Clast@@=\undefined
% 
\connectRestore@ \Creset@@\swap@\lastbreak@@\swap@
\solidvrule@typeset \Creset@@ \edef\lastbreak@@{\cfromthec@}}%
}
\xydef@\solidvrule@typeset{%
% \W@{VRULE TYPESET}\xystatus@cp:%
% 
\ifInvisible@ \let\next@=\relax \else \let\next@=\solidvrule@typeset@ \fi
\checkoverlap@@ \next@}
\xydef@\solidvrule@typeset@{{%
% 
\advance\X@c\X@p \X@c=.5\X@c \X@p=.5\X@c
\advance\X@c-.5\xydashw@ \advance\X@p.5\xydashw@
% 
\ifdim\Y@c<\Y@p \advance\Y@c\U@c \advance\Y@p-\D@p
\else \swapdimen@\Y@c\Y@p \advance\Y@c\U@p \advance\Y@p-\D@c \fi
% 
\Droprule@}}
\xydef@\solidhrule@{%\no@@ \solidhrule@typeset \ifHidden@\else\noinsert@\fi
\solidhrule@pre}
\xydef@\solidhrule@pre{%
% \W@{HRULE SETUP}%
% 
\def\Cbreak@@{%
% \W@{HRULE Cbreak}%
% 
\connectRestore@ \swap@\lastbreak@@\swap@
\solidhrule@typeset \edef\lastbreak@@{\cfromthec@}\Creset@@%
\def\Clast@@{%
% \W@{HRULE Clast}%
% \let\Clast@@=\undefined
% 
\connectRestore@ \Creset@@\swap@\lastbreak@@\swap@
1.6. KERNEL OBJECT LIBRARY

\solidhrule\typeset \Creset@ \edef\lastbreak@{\cfromthep@}{% \
oexpand\solidhrule\typeset{% 
\% \W@{HRULE TYPESET}\xystatus@cp:% 
\% 
\ifInvisible@ \let\next@=\relax \else \let\next@=\solidhrule\typeset\fi
\checkoverlap@@ \next@}
\xydef\solidhrule\typeset@{{
\ifdim\X@c<\X@p \advance\X@c\R@c \advance\X@p-\L@c
\else \swapdimen@\X@c\X@p \advance\X@c\R@p \advance\X@p-\L@c \fi
\advance\Y@c\Y@p \Y@c=.5\Y@c \Y@p=\Y@c
\advance\Y@c-.5\xydashw@ \advance\Y@p.5\xydashw@
\Droprule@}}

\Droprule@ puts the actual ink on the page: with lower left corner at $c$ and upper right corner at $p$.

\xydef\Droprule@{\advance\X@p-\X@c
\% \W@{: kern\the\X@c\space width\the\X@p\space depth-\the\Y@c\space height\the\Y@p}\
\setboxz@h{\kern\X@c \vrule width\X@p depth-\Y@c height\Y@p}\
\ht\z@=\z@ \wd\z@=\z@ \dp\z@=\z@ \Drop@@}

Dots: \dir{.} creates a very boring dot when used as an (object), but interesting dotted lines when used to connect. \zerodot should expand to a zero-sized box with a dot (initialised to use \zerodotbox@); the (object) is built using \pointlike@ (text) (spread-dimen) that we will use again later.

\xydef\zerodot{\copy\zerodotbox@}
\xydef\pointlike@\dir{.}{{\point@}
\xydef\double@{\xydashh@}
\xydef\triple@{\xydashh@}
\xydef\dir0@{0\dir{.}}\dir0@
\xydef\dir1@{\dir{.}}\dir1@
\xydef\dir2@{\dir{.}}\dir2@

This is reflected by the rather complicated spreading routine: ‘Dotting’ is the art of putting zero-sized objects together with equal distance independent of the chosen direction. So we must recompute the number of segments $N$ (likely to be very big or $-1$) with trigonometry; using $radius$
for the individual dots this becomes

\[\begin{align*}
A &= \cos \text{Direction} \times \text{radius} \\
B &= \sin \text{Direction} \times \text{radius}
\end{align*}\]

if \(A + B > |d\@X| + |d\@Y|\) then \(N = 0\) and exit

\[\langle A, B \rangle := 2 \times \langle A, B \rangle\]

as realised below:

\begin{verbatim}
\xydef@dottedSpread@#1{%
\setupDirection@
\dimen@=#1\relax \DN@{
\A@=2\A@ \B@=2\B@ \dottedSpread@i}%
\A@=\sd@X\cos\text{Direction}\dimen@ \B@=\sd@Y\sin\text{Direction}\dimen@
\dimen@=\A@ \advance\dimen@\B@
\dimen@ii=\sd@X\d@X \advance\dimen@ii\sd@Y\d@Y
\ifdim\dimen@>\dimen@ii \DN@{\count@@=\z@}\fi
\next@
}
\xydef@dottedSpread@i{%
\setupDirection@
\kern.5\A@ box \lastobjectbox@\kern.5\A@
\kern.5\B@ box \lastobjectbox@\kern.5\B@}
\dimen@ii=\sd@X\cos\text{Direction}\dimen@ \advance\dimen@.1\p@
\dimen@ii=\sd@Y\sin\text{Direction}\dimen@
\squiggle@@%
\edef\tmp@{\egroup \U@c=the\dimen@ii \L@c=the\dimen@}
\wdz@=2\L@c \R@c=\L@c \ht\z@=\U@c \D@c=\U@c \dp\z@=\U@c
\rectangleEdge{\Edge@c={rectangleEdge}}%
\Invisible@false \Hidden@false
\Leftness@{.5} \Upness@{.5} 
\end{verbatim}

A particular aspect of this is that we have to enlarge and recenter the actual box used for the typesetting (\lastobjectbox@).

**Squiggles:** These are just a lot of box maneuvering using the directional characters of \xybsqlfont (see xybsql10.mf for details):

\begin{verbatim}
\xydefcsname@{dir1{~}}{\squiggle@}
\xydefcsname@{dir2{~}}{\squiggle@ \double@\xybsqlh@}
\xydefcsname@{dir3{~}}{\squiggle@ \triple@\xybsqlh@}
\xyletcsnamecsname@{dir0{~}}{dir{}}
\xyletcsnamecsname@{dir{~}}{dir1{~}}
\xydef@{\squiggle@}{\xybsqlfont}
\dimen@=\sd@X\cos\text{Direction}\xybsqlll@ \advance\dimen@\p@
\dimen@ii=\sd@Y\sin\text{Direction}\xybsqlll@
\kern\dimen@\squiggle@@
\edef\tmp@{\egroup \the\dimen@ii \the\dimen@}
\wdz@=2\L@c \R@c=\L@c \ht\z@=\U@c \D@c=\U@c \dp\z@=\U@c
\rectangleEdge{\Edge@c={rectangleEdge}}%
\Invisible@false \Hidden@false
\Leftness@{.5} \Upness@{.5} 
\end{verbatim}
1.6. KERNEL OBJECT LIBRARY

The interesting bit is that they spread by not spreading, i.e., by centering between the endpoints—this means

\[ X := X - d/2, dX := dX - d \quad \text{where} \quad d = \|dX\| - N \times A + .1pt \]
\[ Y := Y - d/2, dY := dY - d \quad \text{where} \quad d = \|dY\| - N \times B + .1pt \]

**Double and triple directionalss:** As can be seen by the last two columns, these (and most of the other connectors) also exist in double and triple versions with a 2 or a 3 prepended to the name. For convenience \(\texttt{\dir{=}\text{and}}\ \texttt{\dir{:}}\) are synonyms for \(\texttt{\dir2{-}}\) and \(\texttt{\dir2{.}}\), respectively; similarly \(\texttt{\dir{==}}\) is a synonym for \(\texttt{\dir2{-}}\).

This is very simple, really: \(\texttt{\double@}\) and \(\texttt{\triple@}\) do the work by redefining the \(\texttt{\Drop@@}\) method to do its job twice and thrice. **To Do:** should probably extend any already defined \(\texttt{\Drop@@}\) method?
**Dashing directionals:** First traditional dashing:

\begin{verbatim}
\xydefcsname@{dir1{--}}{\dash@}
\xydefcsname@{dir2{--}}{\dash@ \double@\xydashh@}
\xydefcsname@{dir3{--}}{\dash@ \triple@\xydashh@}
\xyletcsnamecsname@{dir0{--}}{dir{}}
\xyletcsnamecsname@{dir{--}}{dir1{--}}
\xyletcsnamecsname@{dir{==}}{dir2{--}}
\xydef@\dash@{\line@ \wdz@=2\wdz@ \ht\z@=2\ht\z@ \dp\z@=2\dp\z@}
\multiply\D@c\tw@ \multiply\U@c\tw@ \multiply\L@c\tw@ \multiply\R@c\tw@
\def\Connect@@{\straight@\dashedSpread@}
\end{verbatim}

Since the dashes should reach the endpoints we do this:

\begin{verbatim}
if \text{if } N > 0 \text{ then } N := N + 1 \\
dX := dX + d/2 \text{ where } d = \sd@X A \\
dY := dY + d/2 \text{ where } d = \sd@Y B \\
if dX > 0 \text{ then } X := X + A/2 \\
Y := Y + \sd@Y A/2
\end{verbatim}

\begin{verbatim}
\xydef@\dashedSpread@{\ifnum\z@<\count@@ \advance\count@@\@ne \fi \\
\advance\d@X\sd@X.5\A@ \advance\d@Y\sd@Y.5\B@ \\
\ifdim\z@<\d@X \advance\X@c.5\A@ \fi \advance\Y@c\sd@Y.5\B@}
\end{verbatim}

Dashed dashing of squiggled lines are simpler since squiggles are symmetric:

\begin{verbatim}
\xydefcsname@{dir1{~~}}{\dashsquiggle@}
\xydefcsname@{dir2{~~}}{\dashsquiggle@ \double@\xybsqlh@}
\xydefcsname@{dir3{~~}}{\dashsquiggle@ \triple@\xybsqlh@}
\xyletcsnamecsname@{dir0{~~}}{dir{}}
\xyletcsnamecsname@{dir{~~}}{dir1{~~}}
\xydef@\dashsquiggle@{\squiggle@ \\
\multiply\D@c\tw@ \multiply\U@c\tw@ \multiply\L@c\tw@ \multiply\R@c\tw@
\dimen@=\L@c \advance\dimen@\R@c \wdz@=\dimen@ \ht\z@=\U@c \dp\z@=\D@c \\
\def\Connect@@{\straight@\dashsquiggledSpread@}}
\end{verbatim}

The spreading of squiggles is similarly simpler: we just shave 1/4 squiggle size of each end of the connection in order to eliminate the blank space at both ends:

\begin{verbatim}
\xydef@\dashsquiggledSpread@{\ifnum\z@<\count@@ \advance\count@@\@ne \fi \\
\advance\X@c.5\A@ \advance\d@Y\sd@Y.5\B@ \advance\Y@c\sd@Y.5\B@}
\end{verbatim}

Finally “dashed dotting” synonyms:

\begin{verbatim}
\xyletcsnamecsname@{dir1{..}}{dir{.}}
\xyletcsnamecsname@{dir2{..}}{dir2{.}}
\xyletcsnamecsname@{dir3{..}}{dir3{.}}
\xyletcsnamecsname@{dir{..}}{dir1{.}}
\xyletcsnamecsname@{dir{.}}{dir2{.}}
\end{verbatim}

1.6c. The group of *plain tips* contains basic objects that are useful as markers and arrowheads making connections, so each is shown at the end of a dotted connection of the appropriate kind.
They may also be used as connectors and will build dotted connections. \textit{e.g.}, **@{>** typesets $ightarrow$

\textbf{Exercise 1.14:} Typeset the following two $+$s and a tilted square:

$$+
\uparrow$$

\textit{Hint:} the dash created by $\texttt{dir{-}}$ has the length $5\text{pt}$ (here). (p.576)

\textbf{To Do:} Change tips to have a tiny size of $2\text{sp}$ which may be taken as an indication that it is a tip (this can be used by some features, \textit{e.g.}, ‘arrow’).

\textbf{Arrow heads:} The ones intended for single connections are just characters from \texttt{\xyatipfont} and \texttt{\xybtipfont}.

	\texttt{\xydefcsname\{dir{>}\}\{\tip@\}}
\texttt{\xydefcsname\{dir^{>}\}\{\atip@\}}
\texttt{\xydefcsname\{dir_{>}\}\{\btip@\}}
\texttt{\xydefcsname\{dir0{>}\}\{\dir{}\}}
\texttt{\xydefcsname\{dir{>}\}\{\dir{}\}}
\texttt{\xydefcsname\{dir1{>}\}\{\dir1{}\}}
\texttt{\xydefcsname\{dir2{>}\}\{\Tip@\}}
\texttt{\xydefcsname\{dir2{<}\}\{\reverseDirection@\Tip@\}}
\texttt{\xydefcsname\{dir_{<}\}\{\reverseDirection@\atip@\}}
\texttt{\xydefcsname\{dir0{<}\}\{\dir{}\}}
\texttt{\xydefcsname\{dir{<}\}\{\dir{}\}}
\texttt{\xydefcsname\{dir1{<}\}\{\dir1{}\}}
\texttt{\xydefcsname\{dir2{<}\}\{\Tip@\}}
\texttt{\xydefcsname\{dir2{<}\}\{\reverseDirection@\Tip@\}}
\texttt{\xydefcsname\{dir_{<}\}\{\reverseDirection@\atip@\}}
\texttt{\xydefcsname\{dir0{<}\}\{\dir{}\}}
\texttt{\xydefcsname\{dir{<}\}\{\dir{}\}}
\texttt{\xydefcsname\{dir1{<}\}\{\dir1{}\}}
\texttt{\xydefcsname\{dir2{<}\}\{\Tip@\}}
\texttt{\xydefcsname\{dir2{<}\}\{\reverseDirection@\Tip@\}}
\texttt{\xydefcsname\{dir_{<}\}\{\reverseDirection@\atip@\}}
\texttt{\xydefcsname\{dir0{<}\}\{\dir{}\}}
\texttt{\xydefcsname\{dir{<}\}\{\dir{}\}}
\texttt{\xydefcsname\{dir1{<}\}\{\dir1{}\}}
\texttt{\xydefcsname\{dir2{<}\}\{\Tip@\}}
\texttt{\xydefcsname\{dir2{<}\}\{\reverseDirection@\Tip@\}}
\texttt{\xydefcsname\{dir_{<}\}\{\reverseDirection@\atip@\}}
\texttt{\xydefcsname\{dir0{<}\}\{\dir{}\}}
\texttt{\xydefcsname\{dir{<}\}\{\dir{}\}}
\texttt{\xydefcsname\{dir1{<}\}\{\dir1{}\}}
\texttt{\xydefcsname\{dir2{<}\}\{\Tip@\}}
\texttt{\xydefcsname\{dir2{<}\}\{\reverseDirection@\Tip@\}}
\texttt{\xydefcsname\{dir_{<}\}\{\reverseDirection@\atip@\}}
\texttt{\xydefcsname\{dir0{<}\}\{\dir{}\}}
\texttt{\xydefcsname\{dir{<}\}\{\dir{}\}}
\texttt{\xydefcsname\{dir1{<}\}\{\dir1{}\}}
\texttt{\xydefcsname\{dir2{<}\}\{\Tip@\}}
\texttt{\xydefcsname\{dir2{<}\}\{\reverseDirection@\Tip@\}}
\texttt{\xydefcsname\{dir_{<}\}\{\reverseDirection@\atip@\}}
\texttt{\xydefcsname\{dir0{<}\}\{\dir{}\}}
\texttt{\xydefcsname\{dir{<}\}\{\dir{}\}}
\texttt{\xydefcsname\{dir1{<}\}\{\dir1{}\}}

Double and triple tips are realised by taking the two halves and ‘wringing them apart’; as the naming indicates they are meant to be put at the end of 2- and 3-connections. This is currently done the slightly hacky (but efficient) way of adding directly to \texttt{\DirectionChar}; maybe this should be using \texttt{\Direction@}?
Stopper: \[\text{dir}{}\] makes a ‘stopper’ using just the appropriate \texttt{\xydashfont} character rotated 90° and centered; the ^ and _ variants are just shifted appropriately and two are used to make the 2 and 3 variants longer.

Hooks: These are halfcircles opening towards or opposite \texttt{\direction} and fastened by their center or either endpoint. Build by lots of box manipulation with the \texttt{\xybsqlfont} quarter
circles...

Quarter turns: These are quarter circles fastened by their start or end point in \texttt{Direction}. Build by box manipulation of the \texttt{xybsqlfont} quarter circles. The intention is that the ' directionals are half the corresponding () directional.
1.6d. These tips are combinations of the plain tips provided for convenience (and optimised for efficiency). New ones can be constructed using \texttt{\composite} and by declarations of the form
\begin{verbatim}
newdir \langle dir \rangle \{ \langle composite \rangle \}
\end{verbatim}

which defines \texttt{\dir\langle dir \rangle} as the \langle composite \rangle (see note 1.4d for the details).

\newdir is simple:
\begin{verbatim}
\xydef@newdir#1#{\newdir@{#1}}
\xydef@
ewdir@#1#2#3{\xydefcsname@{dir#1{#2}}{\composite@{}{#3}}}
\end{verbatim}

Then the somewhat more efficient \texttt{\shiftdir@} used internally for moving a tip in the current direction—it does so by making a local hbox within which the argument tip is constructed and subsequently shifted and made of zero size. Use as
\begin{verbatim}
\shiftdir@\langle tip@\rangle\langle dimen\rangle\langle tip@\rangle
\end{verbatim}

where \langle tip@\rangle means a tip command without the leading \texttt{\hbox{}}.
\begin{verbatim}
\xydef@\shiftdir@#1#2{\setbox\z@=\hbox{\relax
\setboxz@h{\dimen@ii=#2\relax
\dimen@=-\cos\Direction\dimen@ii \lower\sin\Direction\dimen@ii}
\kern\dimen@ \boxz@}
\wdz@=\z@ \ht\z@=\z@ \dp\z@=\z@ \boxz@}
\end{verbatim}

Then the tips, with the \texttt{\tipjot@} hook allowing changing the spacing of tips used for single lines.
\begin{verbatim}
\xylet@\tipjot@=\jot
\end{verbatim}
\begin{verbatim}
\xydefcsname@{dir1{>>}}{\shiftdir@\tip@\tipjot@ \tip@}
\xydefcsname@{dir^{>>}}{\shiftdir@\atip@\tipjot@ \atip@}
\xydefcsname@{dir_{>>}}{\shiftdir@\btip@\tipjot@ \btip@}
\xydefcsname@{dir2{>>}}{\composite@{h!/\tipjot@/\dir2{>}}\dir2{>}}
\xydefcsname@{dir3{>>}}{\composite@{h!/\tipjot@/\dir3{>}}\dir3{>}}
\xydefcsname@{dir0{>>}}{\dir{}}
\xydefcsname@{dir{>>}}{\dir{}}
\xydefcsname@{dir1{<<}}{\reverse\Direction\shiftdir@\tip@\tipjot@ \tip@}
\xydefcsname@{dir^{<<}}{\reverse\Direction\shiftdir@\atip@\tipjot@ \atip@}
\xydefcsname@{dir_{<<}}{\reverse\Direction\shiftdir@\btip@\tipjot@ \btip@}
\xydefcsname@{dir2{<<}}{\composite@{h!/-\tipjot@/\dir2{<}}\dir2{<}}
\xydefcsname@{dir3{<<}}{\composite@{h!/-\tipjot@/\dir3{<}}\dir3{<}}
\xydefcsname@{dir0{<<}}{\dir{}}
\xydefcsname@{dir{<<}}{\dir{}}
\xydefcsname@{dir1{||}}{\shiftdir@\stopper@\xydashh@ \shiftdir@\stopper@\z@}
\xydefcsname@{dir^{||}}{\shiftdir@{\above\Direction\xydashl@\line@}\xydashh@}
\end{verbatim}
1.6.2 Circle segments

Circle (object)s are round and typeset a segment of the circle centered at the reference point. The syntax of circles is described in figure 1.8 with explanations below.

The \texttt{circ} command is the hub: it parses the optional \texttt{<radius>} (to \texttt{\R@}, default from \texttt{R_c}) and \texttt{{\cir}}, bailing out with a \texttt{\zerodot} if the radius is to small:

\begin{verbatim}
\xydef@\cir#1#{\hbox{\bgroup
\afterVECTORorEMPTY{\xy@@{\R@=\X@c}\cir@}{\xy@@{\R@=\R@c}\cir@}#1}\
\endgroup}
\end{verbatim}

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{\cir (radius) { \cir }}</td>
<td>\texttt{\cir}cle segment with \texttt{(radius)}</td>
</tr>
<tr>
<td>\texttt{\cir (radius)} \rightarrow \texttt{\cir (empty)}</td>
<td>use \texttt{R_c} as the radius</td>
</tr>
<tr>
<td>\texttt{</td>
<td>\cir (vector)}</td>
</tr>
<tr>
<td>\texttt{\cir} \rightarrow \texttt{\cir (empty)}</td>
<td>full circle of \texttt{(radius)}</td>
</tr>
<tr>
<td>\texttt{</td>
<td>\cir (diag) \cir (orient) \cir (diag)}</td>
</tr>
<tr>
<td>\texttt{\cir (orient)} \rightarrow \texttt{\cir (orient) \cir (orient)}</td>
<td>anticlockwise</td>
</tr>
<tr>
<td>\texttt{</td>
<td>\cir (orient) \cir (orient)}</td>
</tr>
</tbody>
</table>

Figure 1.8: \texttt{(circ)les}.

Syntax Action

\begin{verbatim}
\cir (radius) \{ \cir \}
\end{verbatim}

\begin{verbatim}
\cir (radius) \rightarrow \cir (empty)
\cir (vector)
\cir (empty)
\cir (diag) \cir (orient) \cir (diag)
\cir (orient) \rightarrow \cir (orient) \cir (orient)
\end{verbatim}
The code to actually typeset the \(\langle \text{cir} \rangle\) just parsed starts by checking that the \(\langle \text{cir} \rangle\) was immediately followed by the @ we put there in \texttt{\textbackslash cir}: 

\[
\xydef@\cir@cir{% \\
\ifx \space@ \next \expandafter\DN@ \space@ {\xyFN@ \cir@cir}% \gobble spaces \\
\else \ifx @ \next \DN@ @{% \cir@i} \\
\else \xyerror@{illegal \langle \text{cir} \rangle: must have form \langle \text{diag} \rangle <\text{orient} > <\text{diag} > \text{or} \langle \text{empty} \rangle \text{ or}}{} \\
\fi \fi \next@} 
\]

Similarly when an \(\langle \text{empty} \rangle\) one was given—the parser will recognise this as a \(\langle \text{diag} \rangle\) but we hack that here: 

\[
\xydef@\cir@diag{% \\
\DN@ {\xyerror@{illegal \langle \text{cir} \rangle: must have form \langle \text{diag} \rangle <\text{orient} > <\text{diag} > \text{or} \langle \text{empty} \rangle \text{ or}}{}} \\
\ifx @ \next \ifnum\count@ = 8 \\
\DN@ @{\def\CIRin@@{0}\def\CIRorient@@{\CIRfull@}\def\CIRout@@{7}\cir@i} \\
\fi \fi \next@} 
\]

...and then use the constructed methods to build it: 

\[
\xydef@\cir@i{% \\
\ifnum\CIRin@@ = 8 \xyerror@{incomplete \langle \text{cir} \rangle \text{ specification}}{% \\
The \langle \text{cir} \rangle \text{ you specified as \langle \text{diag} \rangle <\text{orient} > <\text{diag} > \text{ is not sufficiently specific.}}{% \\
\def\CIRin@@{0}\fi \\
\ifdim\R@ < .5\p@ \R@ = \z@ \zerodot \\
\else \CIRorient@@ \cirbuild@ \fi \\
\L@c = \R@ \R@c = \R@ \U@c = \R@ \D@c = \R@ \xyD{Leftness\{.5\}} \xyD{Upness\{.5\}} \\
\xyD{Drop\{\texttt{\styledboxz@}\}} \xyD{Connect\{\texttt{\straight@\relax}\}} \\
\xyD{\texttt{\circleEdge}} \xyD{\texttt{\OBJECT@x}} 
\]

Parsing: The \texttt{\afterCIRorDIAG} parser handles the parsing: it either 

- parses the \(\langle \text{cir} \rangle\) and sets \texttt{\textit{in}}, \texttt{\textit{orient}}, and \texttt{\textit{out}}, and passes control to the first argument continuation, 
  or 

- parses the single \(\langle \text{diag} \rangle\) specified, store it in \texttt{\count@} (as \texttt{8} if an \(\langle \text{empty} \rangle\) one given), and pass control to the second continuation argument, 

where the \(\langle \text{diag} \rangle\) internal representation number of note 1.4l of is used. An \(\langle \text{empty} \rangle\) circle is treated as an \(\langle \text{empty} \rangle\) diagonal; specifying an \(\langle \text{empty} \rangle\) first \(\langle \text{diag} \rangle\) of a \(\langle \text{cir} \rangle\) is equivalent to using the value of the \texttt{\textit{in}} method at call time.

The parser is very simple, setting methods stored in the usual \texttt{@@}-terminated control sequences (\textbf{To Do}: Rename all non-method control sequences that end in \texttt{@@}... to use \texttt{@\romannumeral} suffixes...):

\[
\xydef@\CIRin@@{3} \\
\xydef@\CIRout@@{3} \\
\xylet@\CIRorient@@ = \empty \\
\xydef@afterCIRorDIAG#1#2{% \def\afterCIR@{#1}\def\afterCIRDIA@{#2}\xyFN@ \CIR@} \\
\xylet@\afterCIR@ = \empty \\
\xylet@\afterCIRDIA@ = \empty \\
\xydef@\CIR@{\count@ = 8 \afterDIAG{\edef\CIRin@@{\the\count@}\xyFN@ \CIR@@}}
\]
The default is to generate a *full circle* with the specified radius, e.g.,

\xy*
\cir<4pt>{}
\endxy

\xy*
\M\cir{}
\endxy

All the other circle segments are subsets of this and have the shape that the full circle outlines.

Finally we present the *orient* methods. They use these ‘internal methods’ to actually draw the circles

\xylet\CIRtest@@=\relax
\xydef\CIRlo@@{0}
\xydef\CIRhi@@{0}
\xydef\CIRfull@{
  \def\CIRtest@@##1##2{##2}}
\xydef\cirbuild@{
  \cirrestrict@@ \multiply\count@8 %
  \circhar@0 \circhar@7 \kern\dimen@ 
  \circhar@1 \circhar@6 \kern\dimen@ 
  \circhar@2 \circhar@5 \kern\dimen@ 
  \circhar@3 \circhar@4 \kern\dimen@}
\xydef\circhar@#1{
  \setboxz@h{\circhar@@{#1}} \dimen@=\wdz@ \wdz@=\z@ \ht\z@=\R@ \dp\z@=\R@
  \CI\CIRtest@@1{\styledboxz@} \setbox\z@=\copy\voidb@x}
\xydef\circhar@@#1{\xycircfont \advance\count@1 \relax \char\count@}

\cirrestrict@@ computes the group \( g \) of circle segments to use from the radius \( r \) using the formula (the reverse of the one in \xycirc10.mf)

\[
g = \begin{cases} 
  \frac{r}{\text{pt}} - 1 & \text{if } 1\text{pt} \leq r < 8\text{pt} \\
  \frac{r}{4\text{pt}} + 3 & \text{if } 8\text{pt} \leq r < 16\text{pt} \\
  \frac{r}{10\text{pt}} + 7 & \text{if } 16\text{pt} \leq r < 32\text{pt} \\
  15 & \text{if } 32\text{pt} \leq r
\end{cases}
\]
(where we know from \( \text{\texttt{\textcirc}} \)) that \( r \geq \frac{1}{2} \text{pt} \), and then adjusts the radius to be exactly the one chosen through the use of group \( g \) using the formula in \texttt{xcirc10.mf}...this is necessary because of the restriction on \texttt{tfm} files that they can only have 15 different nonzero heights and depths. Subsequent calls to \texttt{\textcirc restrict@@} should compute the same values.

\[
\text{\texttt{\textcirc restrict@@}} \{ \begin{align*}
\texttt{\begingroup} \texttt{\dimen@=} \R @ \texttt{\endgroup}
\texttt{\setboxz@h{\xycircfont}\char \z@ \char \@ne} \A@=\wdz @
\texttt{\ifdim} \R @<8 \A @ \texttt{\count@=} \dimen@ \texttt{\advance} \texttt{\count@=} \A @
\texttt{\else} \texttt{\ifdim} \R @<16 \A @ \texttt{\count@=} \dimen@ \texttt{\divide} \texttt{\count@=} \texttt{\dimen@} \texttt{\advance} \texttt{\count@=}3 \%
\texttt{\else} \texttt{\ifdim} \R @<32 \A @ \texttt{\count@=} \dimen@ \texttt{\divide} \texttt{\count@=} \texttt{\dimen@} \texttt{\advance} \texttt{\count@=}7 \%
\texttt{\else} \texttt{\count@=}15 \texttt{\fi}
\R @=\A @
\texttt{\ifnum} \texttt{\count@}<8 \texttt{\multiply} \R @ \texttt{\count@} \texttt{\advance} \R @=\A @
\texttt{\else} \texttt{\ifnum} \texttt{\count@}<12 \texttt{\multiply} \R @ \texttt{\count@} \texttt{\multiply} \R @=2 \A @ \texttt{\divide} \texttt{\count@=} \texttt{\dimen@} \texttt{\advance} \texttt{\count@=}3 \%
\texttt{\else} \texttt{\ifnum} \texttt{\count@}<16 \texttt{\multiply} \R @ \texttt{\count@} \texttt{\multiply} \R @=4 \A @ \texttt{\divide} \texttt{\count@=} \texttt{\dimen@} \texttt{\advance} \texttt{\count@=}7 \%
\texttt{\else} \texttt{\multiply} \R @=32 \texttt{\fi}
\texttt{\edef} \texttt{\@tmp=} \texttt{\endgroup} \texttt{\R @}=\texttt{\the} \texttt{\R @} \texttt{\count@=}\texttt{\the} \texttt{\count@}\texttt{\@tmp}
\end{align*} \}
\]

Partial circle segments with \langle\text{orient}\rangle ation are the part of the full circle that starts with a tangent vector in the direction of the first \langle\text{diag}\rangleonal (see note 1.4l) and ends with a tangent vector in the direction of the other \langle\text{diag}\rangleonal after a clockwise (for \texttt{\_}) or anticlockwise (for \texttt{^}) turn, e.g.,

\[
\xy*\texttt{\textcirc<4pt}>{l^r}\endxy \text{ typesets } \texttt{"\textcirc"}
\]

\[
\xy*\texttt{\textcirc<4pt}>{l_r}\endxy \text{ — } \texttt{"\textcirc"}
\]

\[
\xy*\texttt{\textcirc<4pt}>{d_l^u}\endxy \text{ — } \texttt{"\textcirc"}
\]

\[
\xy*\texttt{\textcirc<4pt}>{d_l_u}\endxy \text{ — } \texttt{"\textcirc"}
\]

\[
\xy*+{M}^{\texttt{\textcirc<4pt>}{dr Ur}\endxy} \text{ — } \texttt{"M"}
\]

If the same \langle\text{diag}\rangle is given twice then nothing is typeset, e.g.,

\[
\xy*\texttt{\textcirc<4pt}>{u^u}\endxy \text{ typesets } \texttt{""}
\]

Special care is taken to setup the \langle\text{diag}\rangleonal defaults:

- After \texttt{^} the default is the diagonal 90° anticlockwise from the one before the \texttt{^}.
- After \texttt{_} the default is the diagonal 90° clockwise from the one before the \texttt{_}.

The \langle\text{diag}\rangle before \texttt{^} or \texttt{_} is required for \texttt{\textcirc} (objects).

**Exercise 1.15:** Typeset the following shaded circle with radius 5pt:

\[
\circ
\]

These two macros implement the defaults and setup of \texttt{lo} and \texttt{hi} for anticlockwise and clockwise segments. Here is what they set:

<table>
<thead>
<tr>
<th>\langle\textcirc\rangle</th>
<th>\texttt{lo}</th>
<th>\texttt{hi}</th>
<th>\texttt{test(s)}</th>
</tr>
</thead>
<tbody>
<tr>
<td>( d_1^&lt;d_2, d_1 \leq d_2 )</td>
<td>( d_1-8 )</td>
<td>( d_2-8 )</td>
<td>( lo \leq s \land s &lt; hi )</td>
</tr>
<tr>
<td>( d_1^&lt;d_2, d_1 &gt; d_2 )</td>
<td>( d_2-8 )</td>
<td>( d_1-8 )</td>
<td>( s &lt; lo \lor hi \leq s )</td>
</tr>
<tr>
<td>( d_1^&lt;d_2, d_1 &lt; d_2 )</td>
<td>( d_2+8 )</td>
<td>( d_2+8 )</td>
<td>( s &lt; lo \lor hi \leq s )</td>
</tr>
<tr>
<td>( d_1^&lt;d_2, d_1 \geq d_2 )</td>
<td>( d_2+8 )</td>
<td>( d_1+8 )</td>
<td>( lo \leq s \land s &lt; hi )</td>
</tr>
</tbody>
</table>
where \( +8 \) and \( -8 \) are \( + \) and \( - \) modulo \( 8 \); \( d_1 \) and \( d_2 \) are in \( \texttt{count@0} \) and \( \texttt{count@} \), respectively.

1.6.3 Text

Text in pictures is supported through the \langle \text \rangle construction that builds an object containing \langle \text \rangle typeset to \langle \text \rangle using \langle \text \rangle; \langle \text \rangle can be used as an explicit line break; all lines will be centered. \langle \text \rangle should either be a font command or some other stuff to do for each line of the \langle \text \rangle and \langle \text \rangle should be either \langle \dimen \rangle or \langle \empty \rangle.

The code just parses the \langle \text \rangle defaulting it to \langle \maxdimen \rangle which is recognised as ‘free form’.
1.7 Xy-pic options

Note: LaTeX 2ε users should also consult the paragraph on “xy.sty” in §1.1.1.

1.7.1 Loading

Xy-pic is provided with a growing number of options supporting specialised drawing tasks as well as exotic output devices with special graphic features. These should all be loaded using this uniform interface in order to ensure that the Xy-pic environment is properly set up while reading the option.

\xyoption \xyrequire

\xyoption will cause the loading of an Xy-pic option file which can have one of several names. These are tried in sequence: xy(option).tex, xy(option).doc, xy(short).tex, and xy(short).doc, where (short) is (option) truncated to 6 (six) characters to conform with the TWG-TDS [19].
\xyrequire is the same except it is ignored if an option with the same name is already present (thus does not check the version etc.).
properly installed before continuing.}}}

Sometimes some declarations of an option or header file or whatever only makes sense after some particular other option is loaded. In that case the code should be wrapped in the special command

\xywithoption {⟨option⟩} {⟨code⟩}

which indicates that if the ⟨option⟩ is already loaded then ⟨code⟩ should be executed now, otherwise it should be saved and if ⟨option⟩ ever gets loaded then ⟨code⟩ should be executed afterwards. **Note:** The ⟨code⟩ should allow more than one execution; it is saved with the catcodes at the time of the \xywithoption command.

Finally, it is possible to declare ⟨code⟩ as some commands to be executed before every actual execution of \xywithoption{⟨option⟩} {...}, and similarly ⟨code⟩ to be executed before every \xyoption{⟨option⟩} and \xyrequire{⟨option⟩} (collectively called ‘requests’):

\xyeverywithoption {⟨option⟩} {⟨code⟩}
\xyeveryrequest {⟨option⟩} {⟨code⟩}

This is most often used by an option to activate some hook every time it is requested itself.

These last two use the usual expansion trickery to define or append to the control sequences named \xyeveryrequest@⟨option⟩@ and \xyeverywithoption@⟨option⟩@ used in the definitions above.

### 1.7.2 Option file format

Option files must have the following structure:
1.7. XY-PIC OPTIONS

\%\% (identification)
\%\% (copyright, etc.)
\ifx\xyloaded\undefined \input xy \fi
\xyprovide{(option)}{(name)}{(version)}{%
  (author){(email)}{(address)}
}\texttt{(body of the option)}
\xyendinput

The 6 arguments to \texttt{xyprevide} should contain the following:

\texttt{(option)} Option load name as used in the \texttt{xyoption} command. This should be safe and distinguishable for any operating system and is thus limited to characters chosen among the lowercase letters (a–z), digits (0–9), and dash (–), and all options should be uniquely identifiable by the first 6 (six) characters only.

\texttt{(name)} Descriptive name for the option.

\texttt{(version)} Identification of the version of the option.

\texttt{(author)} The name(s) of the author(s).

\texttt{(email)} The electronic mail address(es) of the author(s) or the affiliation if no email is available.

\texttt{(address)} The postal address(es) of the author(s).

This information is used not only to print a nice banner but also to (1) silently skip loading if the same version was preloaded and (2) print an error message if a different version was preloaded.

The \texttt{xyprevide} command checks that the option is not already loaded and that the loaded version is the same as the preloaded one by checking the existence and contents of the macro \texttt{\xy\texttt{option}\texttt{loaded}}. Finally it calls \texttt{xycatcodes} such that the option internals are loaded in ‘\TeX\ programming mode’.

\texttt{\xyendinput} undoes this.

\def\xyprevide#1#2#3#4#5#6{%
  \def\xyoption@@{#1}\edef\xyoption@@{\codeof\xyoption@@}\edef\next@{#3}%
  \message{ Xy-pic option: #2 v.\next@}%
  \expandafter\let\expandafter\nextii@\csname xy\xyoption@@ loaded\endcsname
  \ifx \next@\nextii@ \message{not reloaded}\endinput
  \else
    \ifx \nextii@\relax\else \xyerror@{Option ‘\xyoption@@’ version mismatch}\{%
      You previously loaded, or the format has preloaded, a different version\^\^\^\\% of this option. Just continue to try to load this version instead (and\^\^\^\\% be prepared for a lot of warnings about redefinitions).\}%
    \fi
    \expandafter\let\expandafter\nextii@\csname xy\xyoption@@ version\endcsname\next@
    \expandafter\let\expandafter\xyenddocmode@\csname DOCMODE\endcsname
    \expandafter\let\csname DOCMODE\endcsname\xyprevide@docmode@
    \xycatcodes
    \fi \ignorespaces}
\def\xyendinput{\expandafter\let\csname DOCMODE\endcsname=\xyenddocmode@
  \message{loaded}\xyuncatcodes\endinput}
The above is further complicated because an option may be loaded in its DOCMODE form (ending in .doc). So we make sure that the mode that skips documentation, defined in the xy.tex file header, is active now.

The `dummy' option described in §3.2 is a minimal option using the above features. It uses the special DOCMODE format to include its own documentation for this document (like all official Xy-pic options) but this is not a requirement.

1.7.3 Driver options

The ⟨driver⟩ options described in chapter 4 require special attention because each driver can support several extension options, and it is sometimes desirable to change ⟨driver⟩ or even mix the support provided by several.6

A ⟨driver⟩ option is loaded as other options with \xyoption{⟨driver⟩} (or through \LaTeX\ 2ε class or package options as described in §1.1.1). The special thing about a ⟨driver⟩ is that loading it simply declares the name of it, establishes what extensions it will support, and selects it temporarily. Thus the special capabilities of the driver will only be exploited in the produced DVI file if some of these extensions are also loaded and if the driver is still selected when output is produced. Generally, the order in which the options are loaded is immaterial. (Known exceptions affect only internal processing and are not visible to the user in terms of language and expected output.) In particular one driver can be preloaded in a format and a different one used for a particular document.

The following declarations control this:

\UseSingleDriver forces one driver only
\MultipleDrivers allows multiple drivers
\xyReloadDrivers resets driver information

The first command restores the default behaviour: that only one ⟨driver⟩ is allowed, i.e., each loading of a ⟨driver⟩ option cancels the previous. The second allows consecutive loading of drivers such that when loading a ⟨driver⟩ only the extensions actually supported are selected, leaving other extensions supported by previously selected drivers untouched. Beware that this can be used to create DVI files that cannot be processed by any actual DVI driver program!

The last command is sometimes required to reset the Xy-pic ⟨driver⟩ information to a sane state, for example, after having applied one of the other two in the middle of a document, or when using simple formats like plain \TeX that do not have a clearly distinguished preamble.

Driver database. The main structure, the list of ‘loaded’ and ‘selected’ ⟨driver⟩s (ordered by age), is implemented as traditional \TeX ‘do-lists’ (applications of \do{⟨driver⟩} to each element).

This is used to provide entry points into a secondary structure that lists for each ⟨driver⟩ the extensions supported by that driver. As should be apparent from the above, only the \unload ⟨driver⟩ is present initially: it represents the implementation technique used for the extension when no ⟨driver⟩ is selected. Every extension must add an entry which reestablishes the default implementation of the extension:

6The kernel support described here is based on the (now defunct) xydriver include file by Ross Moore.
All (driver)s will have a similar structure named \texttt{\textbackslash xydriver@\langle driver\textrangle@support@@} which contains the extensions supported by that particular (driver): the ‘signature’ of each driver as \texttt{-list with two arguments (hence the \texttt{ii \textbackslash \textasciitilde\textbackslash \textasciitilde}): the extension and the command to install the (driver)-extension combination.

The two selection principles differ significantly. When only a single (driver) is selectable then all extensions need to be unloaded before the selection. When multiple (driver)s are allowed then only the added support needs to be executed.

Finally we define the following to get complete independence of load sequence: at every loading of an extension option known to be supported, we run through the list of presently selected drivers to activate any support for it, using the following:

As the above suggests it sometimes makes sense to load (driver)s in the actual textual part of a document, however, it is recommended that only drivers also loaded in the preamble are reloaded later, and that \texttt{\textbackslash xyReloadDrivers} is used when there is doubt about the state of affairs. In case of confusion the special command \texttt{\textbackslash xyShowDrivers} will list all the presently supported and selected driver-extension pairs to the \TeX log.
**Driver internals.** A ⟨driver⟩ option files must have the following structure:

```latex
\% (identification)
\% (copyright, etc.)
\ifx\xyloaded\undefined \input xy \fi
\xyprovide{(option)}{(name)}{(version)}%
   {(author)}{(email)}{(address)}
\newdriver{%
\xyaddsupport{⟨extension⟩}{cs}
:
}⟨body of the ⟨driver⟩ option⟩
\xyendinput
```

There should be an \xyaddsupport command for each supported ⟨extension⟩ which specifies the ⟨cs⟩ (control sequence) with which the indicated support is *activated*.

The \newdriver command first establishes the ‘support’ list for the ⟨driver⟩ (running the argument), then adds the ⟨driver⟩ to the ‘loaded’ list, and finally selects it now and for every subsequent request of this ⟨driver⟩ option.

It is important that the ⟨cs⟩ that *activates* each driver/extension combination only does *rebinding of hooks*. In order to see what hooks are available for an extension, look for declarations of the form

\begin{verbatim}
\xyaddunsupport {⟨extension⟩} ⟨cs⟩
\end{verbatim}

and see what rebindings that ⟨cs⟩ does: those are the hooks. *because* they are the ones that will be safely reestablished! This should be adhered to because both activation and deactivation ⟨cs⟩s may be executed many times.

The procedure is as follows: (1.6820) add the driver and clear its support control sequence unless it is already defined, (1.6826) execute the support commands, and (1.6827) setup to select it after loading.

```
\xydef\newdriver#1{%
\def\nextiii@##1{
\expandafter\def\expandafter\xydriversloaded@@
\expandafter{\xydriversloaded@@\do{##1}}%
\expandafter\let\csname xydriver@#1@support@@\endcsname=\empty}
\def\do##1{\DNii@{##1}\ifx\xyoption@@\nextii@ \let\nextiii@=\eat@ \fi}%
\xydriversloaded@@ \expandafter\nextiii@\expandafter{\xyoption@@}#1\relax
\DN@@\xywithoption{##1}{\selectdriver@{##1}\xyeveryrequest{##1}{\selectdriver@{##1}}}%
\expandafter\next@\expandafter{\xyoption@@}\ignorespaces
}
```

The support commands merely add to the support \do-lists which in turn (1.6856) adds \doii{(extension)}{...} to the \xydriver@(driver)@support@@ control sequence and adds a check for when to apply it every time that ⟨extension⟩ is requested!
Driver related messages. Most extensions will print a warning when a capability is used which is not supported by the presently loaded \(\langle\text{driver}\rangle\). Such messages are only printed once, however, (for some formats they are repeated at the end). Similarly, when the support of an extension that exploits a particular \(\langle\text{driver}\rangle\) is used a warning message will be issued that the DVI file is not portable.

This is implemented by defining a control sequence for each such warning and repeat it at the end. Warnings should be issued using one of

\begin{verbatim}
\xyununsupportwarning@ \langle\text{extension}\rangle \langle\text{message}\rangle
\xyunsupportwarning@ \langle\text{driver}\rangle \langle\text{extension}\rangle
\end{verbatim}

1.8 Algorithms

This section presents the more complicated algorithms used in Xy-pic.

1.8.1 Directions

The direction state is described by the following parameters:

<table>
<thead>
<tr>
<th>Direction</th>
<th>‘angle’ of the direction on ([-4K \ldots 4K]) unit square</th>
</tr>
</thead>
<tbody>
<tr>
<td>(dX, dY)</td>
<td>the vector (c - p)</td>
</tr>
</tbody>
</table>
\[ \text{sign of } dX \text{ and } dY \]
\[ \text{quotients } K \frac{dX}{dY} \text{ and } K \frac{dY}{dX} \text{ (as dimensions in sp)} \]
\[ \text{DirectionChar, SemiDirectionChar} \]
\[ \text{chardefs for directional and semidirectional fonts} \]
\[ \text{cosDirection, sinDirection} \]
\[ \text{factors in the range } [1 \ldots 1] \text{ corresponding to cos and sin of Direction} \]

where the “\([-4K \ldots 4K]\)” unit square” has the following angles:

![Unit Square Diagram]

where the intermediate \(K\) angle in each interval correspond to equidistant points on the unit square. Thus only for \(n \in \{-3, 2, 1, 0, 1, 2, 3, 4\}\) the angle of direction \(n \times K\) is exactly \(n \times 45^\circ - 135^\circ\) (\(0^\circ\) being the direction straight right).

As usual \(\text{DirectionfromtheDirection@}\) expands to code setting the current direction.

\[ \text{DirectionfromtheDirection@}\{
\text{noexpand}\text{Direction}=\text{the}\text{Direction}
\]

\[ \text{Noexpand}\text{dOX}=\text{the}\text{dOX} \text{ Noexpand}\text{dOY}=\text{the}\text{dOY}
\]

\[ \text{Noexpand}\text{K@dX}@Y=\text{the}\text{K@dX}@Y \text{ Noexpand}\text{K@dY}@X=\text{the}\text{K@dY}@X
\]

\[ \text{Chardef}\text{Noexpand}\text{DirectionChar}=\text{the}\text{DirectionChar}
\]

\[ \text{Chardef}\text{Noexpand}\text{SemiDirectionChar}=\text{the}\text{SemiDirectionChar}
\]

\[ \text{Def}\text{Noexpand}\text{cosDirection}\{	ext{cosDirection}\}
\]

\[ \text{Def}\text{Noexpand}\text{sinDirection}\{	ext{sinDirection}\}
\]

\[ \text{Noexpand}\text{resetupDirection@}\]

The actual direction computation is done using \(\text{setupDirection@}\).

**Procedure:** Is really not so complicated. \([1.7025]\) \(dX\) and \(dY\) are computed from \(c - p\) and we skip if the current setting is based on these (this is stored in the internal \(\text{Directiontest@@}\) method); \([1.7028]\) if the direction is one of the principal ones then proceed with an optimised special case for those; otherwise proceed with the generic code.
The procedures for the special (diag)onal cases are summarised in this table:

<table>
<thead>
<tr>
<th>(diag)onal</th>
<th>Direction</th>
<th>$\cos(Direction)$</th>
<th>$\sin(Direction)$</th>
<th>sign($dX, dY$)</th>
<th>Char</th>
<th>Semi</th>
</tr>
</thead>
<tbody>
<tr>
<td>dl</td>
<td>0</td>
<td>$\sqrt{\frac{1}{2}}$, $-\sqrt{\frac{1}{2}}$</td>
<td>$-\sqrt{\frac{1}{2}}$, $\sqrt{\frac{1}{2}}$</td>
<td>$-, -$</td>
<td>127</td>
<td>127</td>
</tr>
<tr>
<td>d</td>
<td>$K$</td>
<td>$0$, $-1$</td>
<td>$0$, $-1$</td>
<td>$+, -$</td>
<td>15</td>
<td>31</td>
</tr>
<tr>
<td>dr</td>
<td>$2K$</td>
<td>$\sqrt{\frac{1}{2}}$, $-\sqrt{\frac{1}{2}}$</td>
<td>$\sqrt{\frac{1}{2}}$, $-\sqrt{\frac{1}{2}}$</td>
<td>$+, +$</td>
<td>47</td>
<td>95</td>
</tr>
<tr>
<td>r</td>
<td>$3K$</td>
<td>$1$, $0$</td>
<td>$1$, $0$</td>
<td>$+, +$</td>
<td>63</td>
<td>127</td>
</tr>
<tr>
<td>ur</td>
<td>$4K$</td>
<td>$\sqrt{\frac{1}{2}}$, $\sqrt{\frac{1}{2}}$</td>
<td>$\sqrt{\frac{1}{2}}$, $\sqrt{\frac{1}{2}}$</td>
<td>$+, +$</td>
<td>79</td>
<td>31</td>
</tr>
<tr>
<td>u</td>
<td>$-3K$</td>
<td>$0$, $1$</td>
<td>$0$, $1$</td>
<td>$+, +$</td>
<td>95</td>
<td>63</td>
</tr>
<tr>
<td>ul</td>
<td>$-2K$</td>
<td>$-\sqrt{\frac{1}{2}}$, $\sqrt{\frac{1}{2}}$</td>
<td>$-\sqrt{\frac{1}{2}}$, $\sqrt{\frac{1}{2}}$</td>
<td>$-, +$</td>
<td>111</td>
<td>95</td>
</tr>
<tr>
<td>l</td>
<td>$-K$</td>
<td>$1$, $-1$</td>
<td>$1$, $-1$</td>
<td>$-, -$</td>
<td>111</td>
<td>95</td>
</tr>
</tbody>
</table>

In each case the argument is used as the unit circle, i.e., non-zero of $dX$ and $dY$, and $<K\frac{dX}{dy}, K\frac{dY}{dx}> := KK \times <dX, dY>\ldots
Here is the procedure for the generic code.

If \( K < KdYdX < K \) then direction is mostly left or right: if \( dX < 0 \) [left, 1.7206]: Direction := \( KdYdX - K \); else [right, 1.7208]: Direction := \( KdYdX + 3K \).

1.723 Compute character codes for direction and semidirection fonts. [1.7234]: DirectionChar := \((8K + Direction + K/32) \div (K/16) - 1\); while DirectionChar > 127 : DirectionChar := 128. [1.7237]: SemiDirectionChar := \((8K + Direction + K/64) \div (K/32) - 1\); while SemiDirectionChar > 127 : SemiDirectionChar := 128. In both cases the 8K are added to ensure that \( \TeX \) will round down. Hack: The 16, \( \KK@ \), and 64 in these lines are really \( \K@ \), \( \K@ \), and \( \K@ \).

1.724 Build cosDirection and sinDirection from appropriate characters in the \texttt{\xydashfont}. [1.7240]: cosDirection := \texttt{wd(\xydashfontSemiDirectionChar)}. [1.7242] C := SemiDirectionChar - 64, if \( C < 0 \): C := C + 128, sinDirection := \texttt{wd(\xydashfontC)}.

1.7218 Register this \( dX, dY \) for next time.

\begin{verbatim}
\xydef\setupDirection@ii{%
 \ifdim>dX<z0 \def>sd@X{-}\else \def>sd@X{+}\fi
 \ifdim>dY<z0 \def>sd@Y{-}\else \def>sd@Y{+}\fi
 \K@dXdY=sd@X\K@dYdX=sd@Y \ifdim>K@dXdY<500pt \multiply\K@dXdY\KK0 \fi
 \dimen@=sd@Y\d@Y
 \divide\dimen@\divide\dimen@\divide\dimen@
 \end\setupDirection@ii
\end{verbatim}
\documentclass{article}
\usepackage{xy}
\xymatrix{
  A & B \\
  C & D
}
\end{document}
Finally some special cases used by the ⟨direction⟩s and directional library objects. All manipulate the Direction dependent parameters and then call \resetupDirection@: \reverseDirection reverses it; \aboveDirection and \belowDirection are for ^ and _, and \vDirection@(x,y){L} is for :((x,y), i.e., computes a new direction as the vector

\[<X - x \times \cos \alpha \times L - y \times (-\sin \alpha) \times L, Y - x \times \sin \alpha \times L - y \times \cos \alpha \times L>\]

where \(\alpha\) is the previous direction angle.
CHAPTER 1. KERNEL: XY.DOC

The above all make use of the following; use them also when the direction state is known to be correct: \resetDirection@ should be called when \p and/or \c are moved along the line \vec{pc}, \resetupDirection when the entire direction state is changed in a consistent manner.

Finally the initial direction: up!

1.8.2 Edges

An Edge is a token list describing the edge of an object. It must have the form \{⟨expandable token⟩ ⟨unexpandable tokens⟩\}. To find the edge of an object then first make it the current object and then do

\the\Edge@c⟨code⟩

where ⟨code⟩ determines what should be done:

0 \c is changed to be equal to the point on the edge intersecting with the line segment from \p (for \E ⟨corner⟩, also this was the v2.6 behaviour).

Note: This should not change any of \A, \B, or any component of the state except \Xc and \Yc!

1 Test whether the center of \p, \ie, \langle X_p, Y_p \rangle, is ‘inside’ the \c object (or on the edge). Sets the test \ifInside@ accordingly.
2 Set \texttt{dimen@} to the distance from the center to the edge towards \( p \) (as set with code 0).

\textbf{Note:} This is only positive in the direction towards \( p \) (thus negative-sized circles and rectangles make it negative).

3 \( c \) is changed to be equal to the point on the edge furthest in the direction towards \( p \) (for \( \langle \text{corner} \rangle \)).

\textbf{Note:} This should not change any of \( A, B, \) or any component of the state except \( X_c \) and \( Y_c \)!

4 Replace \( c \) with rectangle with corners where the line from \( p \) intersects with the edge of \( c \) (thus this is the inner rectangle with corners as the current direction dictates).

5 Replace \( c \) with smallest rectangle that encloses the current object completely.

(if this reminds the reader of a ‘dictionary’ as used by class-oriented programming languages then they probably share this author’s regret that \TeX{} does not have type classes \( \odot \)).

\texttt{message\{edges,\}}
\texttt{\xynew@{if}\ifInside@}

\textbf{Points:} The simplest shape is none at all – a point.

\texttt{\xydef@\zeroEdge#1\{\ifcase#1\relax \or \Inside@false \or \dimen@=\z@ \or \else \Edge@c={\rectangleEdge}\fi}

\textbf{Circles and Ellipses:} Next we define round things, with several special cases for the individual variants described below: centered circular, centered elliptical, and general elliptical. The only ‘intelligent’ choice is for the ‘Under’ method where the object is forced to be a centered variant before applying either the circular or elliptical version.

\texttt{\xydef@\circleEdge#1\{\ifcase#1\expandafter\circleEdge@ \or \expandafter\circleUnder@ \or \dimen@=\R@c \or \expandafter\circleProp@ \or \expandafter\circleInner@ \else \expandafter\circleOuter@ \fi\}

\texttt{\xydef@\circleEdge@\{\DN@{%\ellipseEdge@% \ifdim\R@c=\L@c\relax \ifdim\U@c=\D@c\relax \ifdim\R@c=\U@c\DN@{%\circlecentredEdge@% \else \DN@{%\reverseDirection@\ellipsecentredEdge@% \fi\fi\fi\next@}

\texttt{\xydef@\circleProp@\{\DN@{%\reverseDirection@\ellipseEdge@% \ifdim\R@c=\L@c\relax \ifdim\U@c=\D@c\relax \ifdim\R@c=\U@c\DN@{%\circlecentredEdge@% \else \DN@{%\reverseDirection@\ellipsecentredEdge@% \fi\fi\fi\next@}

\texttt{\xydef@\circleUnder@\{\Inside@false \ifdim\X@c=\X@c \relax \ifdim\Y@c=\Y@c \Inside@true \fi \fi\ifInside@ \else \expandafter \circleCentre@ \fi}

\texttt{\xydef@\circleCentre@\{}

\texttt{\dimen@=\R@c\advance\dimen@-\L@c \divide\dimen@tw@}
CHAPTER 1. KERNEL: XY.DOC

True circles, centered: Code 0 moves $<X_c, Y_c>$ to the point $<X_c - R_c \times \cos \alpha, Y_c - R_c \times \sin \alpha>$ where $\alpha$ is the current direction angle, code 1 tests whether the $p$ center is located between those two points, code 2 just returns the radius, code 3 is as code 0 and code 4 is the only nontrivial one, replacing with the inner symmetric rectangle with corner at the point of code 0.

Ellipsis, centered: When $c$ is at the centre of an elliptical object, first rescale the axes to make the object circular. Rescale $dX$ and $dY$ appropriately, reset the direction and move to the edge as previously. Then scale back to the original shape, adjusting $X_c$ and $Y_c$ appropriately.
For elliptical objects the Outer-rectangle retains all the extents, so just change the edge-type.

Non-centered variants: The code for the more general \texttt{ellipseEdge@} uses a Newton iteration to solve a quadratic equation.

First locate the actual centre of the ellipse and the lengths of the major axis $a$ and minor axis $b$. Let $(x,y)$ denote the displacement of the required edge point from the centre of the ellipse, so that $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$. Now let $(dx,dy)$ be the displacement from the centre to the reference point; then $x = dx + s \cos \theta$ and $y = dy + s \sin \theta$ where $s$ is the distance to $(x,y)$ in the required direction $\theta$. We must find the value of $s$, presumed to satisfy $s \geq 0$.

If $a < b$ we search for a root of

\[
    f\left(\frac{s}{a}\right) = \left(dx + \frac{s}{a} \cos \theta\right)^2 + \left(dy + \frac{s}{a} \sin \theta\right)^2
\]

\[
    f'\left(\frac{s}{a}\right) = 2\left(dx + \frac{s}{a} \cos \theta\right) \cos \theta + 2\left(dy + \frac{s}{a} \sin \theta\right) \frac{a}{b} \sin \theta
\]

where $\tilde{dx} = dx/a$ and $\tilde{dy} = dy/b$. By choosing a starting value $s_0 = 2a + 2b$ the Newton-iteration converges rapidly to the correct root, provided $b > 0$. If $a > b$ then we solve a similar quadratic in $\frac{s}{b}$. 

To Do: The case of \(a = 0\) or \(b = 0\), so that the ellipse degenerates to a line segment, should be trapped earlier. Such an object would be more easily handled as a degenerate case of a rectangle.

To Do: In the case when the reference point lies outside the ellipse the Newton iteration is not guaranteed to converge. A simple test — \(s\) growing rather than decreasing in size — detects possible converge problems and exits the iteration. The following message is issued . . .

Xy-pic warning: poor convergence.

Often, particularly when the reference point is on or near the edge of the ellipse, an acceptable approximation to the edge has been found anyway.
The `Under' procedure is identical to the centered version.

Rectangles:
Rectangles intersection is slightly more complicated and handled separately for the horizontal and vertical case.
Bug: no 'Inner' for rectangles!

Sets $X,Y$ to the intersection of a line from $<X-dX,Y-dY>$ to $<X,Y>$ and the rectangle from $<X-L,Y-D>$ to $<X+R,Y+U>$:

- **%1a** $dY<0, dX<0$: $X := X + \min\{R, U*|dX/dY|\}$,
- **%1b** $dY<0, dX=0$: $Y := Y + U$,
- **%1c** $dY<0, dX>0$: $X := X - \min\{L, U*|dX/dY|\}$,
- **%2a** $dY=0, dX<0$: $X := X + R$,
- **%2b** $dY=0, dX=0$: $;$
- **%2c** $dY=0, dX>0$: $X := X - L$,

%3a $dY>0, dX<0$: $X := X + \min\{R, D*|dX/dY|\}$,
- **%3b** $dY>0, dX=0$: $Y := Y - D$,
- **%3c** $dY>0, dX>0$: $X := X - \min\{L, D*|dX/dY|\}$,
- **%4** \resetupDirection@ to register that even though $dX,dY$ changed all Direction parameters are still valid!

NOTE: $d=0$ really means $|d| < .05pt.$
Checking that \( p \) is under is simpler: Normally it is not but if \( c = p \) or \((L < X < p < R)\) and \((-D < Y < U < c)\) then it is.

Calculating the width is like computing the edge point, just simpler: (1.778) Set \( A \) to the horizontal and \( B \) to the vertical distance applicable; if either is zero (or close) use the other unmodified. (1.780) If both nonzero use \( \min(A=\cos \alpha, B=\sin \alpha) \) where \( \alpha \) is the current angle.

Setting \( c \) to the ‘proportional edge point’ is straight out of v2.6’s \texttt{setlabel@} macro...
1.8.3 Connections

Connections describe how a particular \langle object \rangle may be used to connect \langle p \rangle to \langle c \rangle. The following parameters and methods are defined by any connection; they should be used in the indicated sequence:

\\texttt{\textbackslash Invisible@, \textbackslash Hidden@, and \textbackslash Drop@:} as for \langle object \rangle.

\\texttt{\textbackslash Creset@:} (Re)set the connection parameters to allow use of the following to move to a point on the connection (this is what the interpretation of \langle pos \rangle \textbackslash? does first). All effects obtained by the following methods are undone: \\texttt{\textbackslash Creset@} reestablishes the state to what it was just after the \\texttt{\textbackslash Connect@} every time.

\\texttt{\textbackslash Cshavep@ or \textbackslash Cshavec@:} Change \langle p \rangle or \langle c \rangle to a zero-sized object at the start or finish of the connection (the first < and > \langle place \rangle components).

\\texttt{\textbackslash Calong@{(factor)}:} Move \langle c \rangle to point the \langle factor \rangle along the connection and set the direction parameters as a tangent to it in this point (the \langle (factor) \rangle \langle place \rangle component).

\\texttt{\textbackslash Cslidep@{(dimen)} or \textbackslash Cslidec@{(dimen)}:} Move \langle p \rangle or \langle c \rangle the \langle dimen \rangle sion further in the current direction. Can be used both before and after the \\texttt{\textbackslash Calong@} method (these handle subsequent < and > as well as the /\langle slide \rangle/ \langle place \rangle component).

\\texttt{\textbackslash Cintercept@:} Set \langle c \rangle to a zerosized object at the point where the connection crosses the straight line from \langle p \rangle to \langle c \rangle (the \langle \{\ldots\} \rangle \langle place \rangle component).

\\texttt{\textbackslash Cbreak@ and \textbackslash Clast@:} These support typesetting of the connection in several ‘subsegments’ as used by the arrow option. The idea is the following, using the internal \texttt{lastbreak} and \texttt{nextbreak} positions:
1. \Connect@@ sets \textit{lastbreak} := \textit{p} and \textit{nextbreak} := \textit{c}

2. The commands \Cslidec@@, \Cslidep@@, and \Cintercept@@, set \textit{nextbreak} logically to the position that is obtained by the manipulation.

3. \Cbreak@@ retypesets the connection except only the piece from \textit{lastbreak} to the current \textit{c} object, which \textit{must} be at on top of \textit{nextbreak}, is typeset. Afterwords it sets \textit{lastbreak} := \textit{nextbreak} and \textit{nextbreak} := \textit{c} (where \textit{c} is the original \textit{c} of the connection).

4. The command \Clast@@ typesets from \textit{lastbreak} and the remainder of the connection, and resets \textit{lastbreak} := \textit{p} and \textit{nextbreak} := \textit{c}.

Notice that it is not necessary for \textit{nextbreak} to actually exist, in fact in the kernel it is just an alias for \textit{c} at the time \Cbreak@@ is called.

These are the defaults for basic $X\text{Y}$-pictures that only has straight connections hence the macros all starting with \no... are rather simple – in particular they just merge with zerosized objects at the locations cutoff locations (this was inherited from the broken \arrow code $\bigcirc$).
And finally the dummy ‘break’ method which of course does nothing except the required shifting.

The kernel only defines one kind of connection which is used by default: ‘filling’ many copies of the connect object on the straight line from \( p \) to \( c \).

First the ‘dummy’ connection type \texttt{no} used for \langle<object\rangle\rangles where it doesn’t really make sense to ‘connect’, and used to provide defaults for easy operations:

Next ‘straight’ connections that use the following special parameters to customises how to repeat a ‘filler’ object as many times as needed to draw the connection: \texttt{\lastobjectbox@}: box to fill with; assumed to be ‘trimmed’ to have only the size needed for the filler; size is \( w \times (h+d) \).

The method is called as \texttt{\straight@{(spread)}} where \langle\textit{spread}\rangle is a macro to expand \textit{after} the number of fillers \( N \) (in \texttt{\count@@}) is known and with \( A = w \) and \( B = d + h \) but \textit{before} any filling is done. May change \( N \) as well as \( dX \); \( dY \); \( X \); \( Y \); \( w \); \( d \); \( h \) in order to affect the filling.

**Procedure:** \[ l.8057 \] setup direction parameters, clear the break state, and define the \texttt{\Creset@@} macro to reestablish this initial state; \[ l.8060 \] build macro to discover if the edges of the objects overlap (it edges were removed); \[ l.8068 \] move both \( p; c \) to the edges of the objects... and define the \texttt{\Cshave*@@} macros accordingly; \[ l.8069 \] choose either to ignore the connection if requested or there is overlap between the objects of choose either vbox or hbox version (see below)...; and \[ l.8074 \] build \texttt{\Calong@@}, \texttt{\Cbreak@@}, and \texttt{\Clast@@} macros, and reset \( pc \).

**Note:** Assumes that the direction is not tampered with between \texttt{\Creset} is defined and used...
The methods \texttt{\straightCbreak@} and \texttt{\straightClast@} implement the break method parametrised on the `typeset' primitive which is usually \texttt{\straight@typeset} which is a simplified version of \texttt{\straight@} that does not initialise. This relies on the connection being straight: the \texttt{\Cbrea@\}@ method executes $p, c := p, \texttt{lastbreak}, c, \texttt{nextbreak}$ and \texttt{\Clast@} executes $p, c := p, \texttt{lastbreak}, c$, where in both cases the \texttt{[Inner]} modifier is applied just before the merging to make sure the edge is respected by the merged object.
Now for the typesetting — to summarise: these are parametrised on direction parameters and use $X, Y$: endpoint of connection (`Direction end'). $dX, dY$: connection distance (`Direction vector'). \texttt{\Leftness\ of (object). \texttt{\lastobjectbox} to fill with; assumed to be `trimmed'. \texttt{\Spread} to expand to modify the default spreading used. \texttt{\Drop} to expand to actually typeset the finished connection when it is in box! \textbf{Note:} Must make box0 void to avoid `box leaks'.

Two variants exist: one for mostly horizontal and one for mostly vertical filling. We detail the `mostly horizontal' one:

\[ \texttt{\xydef@\straighth@\{\setbox\z@=\hbox{}} \]
\[ \texttt{\setbox8=\copy\lastobjectbox\relax \A@=\wd8\relax \B@=\dp8\relax \advance\B@\ht8\relax} \]
\[ \texttt{\ifdim \A@=\z@ \count@@=\m@ne \else \dimen@=\sd@X\d@X \divide\dimen@\A@ \count@@=\dimen@ \fi} \]
\[ \texttt{\Spread@@} \]
\[ \texttt{\ifdim\d@X>\z@ \advance\X@c-\wd8\relax\fi} \]
\[ \texttt{\dimen@=-\sd@X\wd8\relax} \]
\[ \texttt{\multiply\dimen@\K@dYdX \divide\dimen@\K@} \]
\[ \texttt{\ifdim\d@X>\z@ \advance\Y@c\dimen@ \divide\Y@c-\Leftness\dimen@} \]
\[ \texttt{\else \advance\Y@c\Leftness\dimen@ \fi} \]
\[ \texttt{\dimen@=\wd8\relax} \]
\[ \texttt{\advance\A@=\sd@X\d@X \divide\A@-\dimen@} \]
\[ \texttt{\B@=\sd@X\dimen@ \multiply\B@\K@dYdX \divide\B@\K@} \]
\[ \texttt{\ifdim\d@Y<\z@ \count@=\z@ \else \count@@=\z@ \fi} \]
\[ \texttt{\ifnum\z@<\count@ \divide\A@\count@ \divide\B@\count@ \fi} \]
\[ \texttt{\ifnum\z@<\count@ \divide\A@\count@ \divide\B@\count@ \fi} \]
\[ \texttt{\ifnum\z@<\count@ \divide\A@\count@ \divide\B@\count@ \fi} \]
\[ \texttt{\ifnum\z@<\count@ \divide\A@\count@ \divide\B@\count@ \fi} \]
\[ \texttt{\ifnum\z@<\count@ \divide\A@\count@ \divide\B@\count@ \fi} \]
\[ \texttt{\ifnum\z@<\count@ \divide\A@\count@ \divide\B@\count@ \fi} \]
\[ \texttt{\ifnum\z@<\count@ \divide\A@\count@ \divide\B@\count@ \fi} \]
\[ \texttt{\ifnum\z@<\count@ \divide\A@\count@ \divide\B@\count@ \fi} \]
\[ \texttt{\ifnum\z@<\count@ \divide\A@\count@ \divide\B@\count@ \fi} \]
\[ \texttt{\ifnum\z@<\count@ \divide\A@\count@ \divide\B@\count@ \fi} \]
\[ \texttt{\advance\Y@c\copy8\relax \divide\Y@c\B@ \repeat\}} \]
\[ \ht\z@=\z@ \wd\z@=\z@ \dp\z@=\z@ \quad \{\texttt{\Drop\} \}} \]

The mostly \texttt{\vertical} is analogous. \textbf{Bug:} \texttt{\dimen@} holds the unadjusted $h + d$ throughout — somehow it works anyway!

\[ \texttt{\xydef@\straightv@\{\setbox\z@=\vtop{}} \]
\[ \texttt{\setbox8=\copy\lastobjectbox\relax \A@=\wd8\relax \B@=\dp8\relax \advance\B@\ht8\relax} \]
\[ \texttt{\ifdim \B@=\z@ \count@@=\m@ne \else \dimen@=\sd@Y\d@Y \divide\dimen@\B@ \count@@=\dimen@ \fi} \]
\Spread@  
\dimen@=\dp8\relax \advance\dimen@-\ht8\relax  
\B@=\sd@Y\d@Y \divide\A@X \divide\A@K@ \divide\A@-\d@X  
\A@=\sd@X\A@ \count@=\count@@ \divide\count@\m@ne  
\ifnum\z@<\count@ \divide\B@\count@ \divide\A@\count@ \fi  
\B@=\sd@Y\B@ \A@=\sd@X\A@ \ht8=\B@ \dp8=\z@  
\ifdim\d@Y<\z@  
\advance\Y@c\dimen@ \advance\Y@c\Upness@\B@  
\else  
\advance\dimen@\Upness@\B@ \advance\Y@c\dimen@\B@  
\fi  
\advance\Y@c\B@  
\ifdim\d@X<\z@ \else \advance\X@c\wd8\relax \fi  
\null \kern-\Y@c \count@=\z@  
\loop\ifnum\count@\count@@ \divide\count@\one \ifnum\count@\count@@ \divide\count@\one \fi  
\moveright\X@c\copy8\relax \divide\A@A@X  
\repeat@}  
\ht\z@=\z@ \wd\z@=\z@ \dp\z@=\z@ \{\Drop@\}  

\End & log

\message{ Xy-pic loaded}\xyuncatcodes \endinput  

Xy-pic is maintained using the RCS “Revision Control System” by Walther F. Tichy. The following is the revision history since the first release to Usenet.

% $Log: xy.doc,v $  
% Revision 3.35 2013/10/06 01:14:17 krisrose  
% Backpatch for 3.8.9...  
%  
% Revision 3.34 2013/08/26 03:56:11 krisrose  
% Typo in documentation of saving fixed.  
%  
% Revision 3.33 2012/05/24 00:30:38 krisrose  
% Release 3.8.8 with xyframes fix by Norbert Preining.  
%  
% Revision 3.32 2011/08/28 22:19:06 krisrose  
% Font fix (stroke to outline) by Daniel.  
%  
% Revision 3.31 2011/05/27 04:51:17 krisrose  
% Ready to release.  
%  
% Revision 3.30 2011/04/24 02:56:34 krisrose  
% Latest from Daniel.  
%  
% Revision 3.29 2011/03/31 06:10:57 krisrose  
% !B changed to !V (and !H added).  
%  
%
% Link fixes in progress...

% Revision 3.11  2010/04/20 20:36:43  krisrose
% Documentation updates.

% Revision 3.10  2010/04/16 06:58:06  krisrose
% Version fixed by hand.

% Revision 3.9  2010/04/16 06:06:51  krisrose
% Preparing for a new release...

% Revision 3.8  2010/04/13 08:44:32  krisrose
% Old xydiff patches applied.

% Revision 3.7  1999/02/16 15:12:50  krisrose
% Interim release (Y&Y fonts now free).

% Revision 3.6  1998/03/06 01:28:05  krisrose
% Releasing (with Y&Y fonts).

% Revision 3.5  1997/05/28 13:05:01  krisrose
% Fixed missing breaks bug.

% Revision 3.4  1997/05/18 01:14:25  krisrose
% Essential bugfixes.

% Revision 3.3  1996/12/19 03:31:56  krisrose
% Maintenance release

% Revision 3.2  1995/09/19 18:22:27  kris
% Bug fix release.

% Revision 3.1  1995/09/05 20:31:32  kris
% Releasing!

% Revision 3.0  1995/07/07 20:14:21  kris
% Major release w/new User’s Guide!

% Revision 2.14  1995/07/05 22:20:09  kris
% Buglets...

% Revision 2.13  1995/07/04 15:11:17  kris
% Ready to release v3?

% Revision 2.12  1994/10/25 11:55:12  kris
% Interim release just before v3 [works with AMS-LaTeX 1.2]...

% Revision 2.11  1994/07/05 10:37:32  kris
% Third 3beta release [bug fixes].
Experimental graph feature included (for ECCT-94 presentation).

Revision 2.10 1994/06/15 12:55:07 kris
Second 3beta release: bug fixes.

Revision 2.9 1994/06/09 14:59:19 kris
Release 3beta.

Revision 2.8 1994/04/11 09:31:09 kris
Second (bug fix) 3alpha release [corrected].

Revision 2.7 1994/03/08 02:06:01 kris
Release 3alpha.

Revision 2.6.9.1 1994/03/07 04:22:46 kris
Last internal 3alpha and pre-2.7 release.

MAJOR REWRITE and REORGANISATION:
File xypic.doc split into separate files: xy.doc for 'kernel' and other
files with the 'extensions' and 'features'.
Documented in special DOCMODE LaTeX-based format supported by xydoc.sty.

Revision 2.6.1.1 1992/07/01 07:08:24 kris
Send to EuroTeX '92...

Revision 2.6 1992/06/24 01:23:34 kris
Added hooks using font xyqc10.
Added new POSitions: * and !.
Added triple lines \Ssolid and \Ddashed.

Revision 2.5 1992/02/24 03:30:54 kris
Fixed bugs in \Direction calculation logic...
Added (FACTOR) to \rotate to allow arbitrary rotation.
Intermediate points now accept an optional /RADIUS argument.
Added \Tip with wide tip.
[See ChangeLog for further details].

Revision 2.4 1992/01/22 02:15:10 kris
Fixed bugs [with thanks]:
No spurious arrow heads with LaTeX: \pit now undefined [Werner Struckmann]
\Solid works: sets \Density [Dave Bowen]
Short diagonal lines work...major rewrite of \connectv@ [Eric Domenjoud]

Revision 2.3 1992/01/13 23:28:12 kris
Swapped definitions of \ddtoX and \uutoX [found by Nico Verwer].
Diagonal lines were wrong [Eric Domenjoud].

Revision 2.2 1992/01/09 04:05:40 kris
Still problems with rules in frames and horizontal/vertical \solids. Grrr.

Revision 2.1 1992/01/05 03:40:14 kris
Fixed bugs and added \connectv@. Increase \Density to 3.14.

Revision 2.0 1991/12/25 22:45:42 kris
Release 2.0.

Revision 1.5 1991/12/20 13:20:37 kris
Fixed bugs in \rotate (\rotatebox).

Revision 1.4 1991/12/19 11:40:24 kris
Fixed bugs: \rotatebox had an extra space, \plot and \arrow do not have enough options.

Revision 1.3 1991/12/18 10:37:47 kris
Released as 1.3.

Revision 1.2 1991/12/17 05:35:32 kris
Fixed bugs in \arrow box.

Revision 1.1 1991/12/15 00:59:12 kris
Released as 1.1.

Revision 1.0 1991/12/13 11:11:18 kris
Release 1.0.

Revision 0.9 1991/12/08 04:55:38 kris
Initial version.
1.8. ALGORITHMS

Revision 2.1 1992/01/02 14:54:07 kris
Release version.

Revision 1.40 1991/12/17 04:53:23 kris
Version distributed as ‘final draft’ on Usenet.
Chapter 2

Extensions

This chapter describes the options that provide extensions to the kernel, i.e., support facilities that are not obtainably using the kernel. Most such features are only approximated, relying on a driver for the implementation.

2.1 Curve and Spline extension

Vers. 3.12 by Ross Moore (ross.moore@mq.edu.au)

This option provides Xy-pic with the ability to typeset spline curves by constructing curved connections using arbitrary directional objects and by encircling objects similarly. Warning: Using curves can be quite a strain on \TeX{}’s memory; you should therefore limit the length and number of curves used on a single page. Memory use is less when combined with a backend capable of producing its own curves; e.g., the \texttt{POSTSCRIPT} backend).

Header:

```latex
\%% $Id: xycurve.doc,v 3.12 2011/03/14 20:14:00 krisrose Exp$
\%%
\%% Xy-pic ‘‘Curves and Splines’’ extension.
\%% Copyright (c) 1993-1997 Ross Moore <ross.moore@mq.edu.au>
\%%
\%% This file is part of the Xy-pic package for graphs and diagrams in \TeX{}.
\%% See the companion README and INSTALL files for further information.
\%% Copyright (c) 1991-2011 Kristoffer H. Rose <krisrose@tug.org>
\%%
\%% The Xy-pic package is free software; you can redistribute it and/or modify
\%% it under the terms of the GNU General Public License as published by the
\%% Free Software Foundation; either version 2 of the License, or (at your
\%% option) any later version.
\%%
\%% The Xy-pic package is distributed in the hope that it will be useful, but
\%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
\%% or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License
\%% for more details.
\%%
\%% You should have received a copy of the GNU General Public License along
\%% with this package; if not, see http://www.gnu.org/licenses/.
```
Chapter 2. Extensions

2.1.1 Curved connections

Simple ways to specify curves in Xy-pic are as follows:

\begin{verbatim}
**\crv{⟨poslist⟩} curved connection
**\crvs{⟨dir⟩}     get ⟨poslist⟩ from the stack
\curve{⟨poslist⟩} as a ⟨decor⟩ation
\end{verbatim}

in which ⟨poslist⟩ is a list of valid ⟨pos⟩itions. The decoration form \curve is just an abbreviation for \connect\crv. As usual, the current p and c are used as the start and finish of the connection, respectively. Within ⟨poslist⟩ the ⟨pos⟩itions are separated by &. A full description of the syntax for \crv is given in figure 2.1.

We need a counter to track the number of control points, (i.e. the number of ⟨pos⟩s in ⟨poslist⟩) and provide a macro to read it. Also a token list which will store the vital information for later use, to allow curved connections to work properly.

\begin{verbatim}
\xynew{count}\crv@cnt@
\xydef@\xynumctrlpts@{\the\crv@cnt@}
\xynew{toks}\crvpts@
\end{verbatim}

If ⟨poslist⟩ is empty a straight connection is computed. When the length of ⟨poslist⟩ is one or two then the curve is uniquely determined as a single-segment Bézier quadratic or cubic spline. The tangents at p and c are along the lines connecting with the adjacent control point. With three or more ⟨pos⟩itions a cubic B-spline construction is used. Bézier cubic segments are calculated from the given control points.

\begin{verbatim}
\xydef@\setcurve{%
\edef\xysplineparams@{%
  \expandafter\noexpand\csname params@endcsname\endcsname}%
\edef\xysplineedges@{%
  \expandafter\noexpand\csname edges@endcsname\endcsname}%
\end{verbatim}


2.1. CURVE AND SPLINE EXTENSION

The previous picture was typeset using:
\begin{verbatim}
\xy (0,20)*+{A};(60,0)*+{B}
**\crv{}
**\crv{(30,30)}
**\crv{(20,40)&(40,40)}
**\crv{(10,20)&(30,20)&(50,-20)&(60,-10)}
\endxy
\end{verbatim}
except for the labels, which denote the number of entries in the \langle poslist \rangle. (Extending this code to include the labels is set below as an exercise).

The \texttt{?}-operator of \S 1.3 (note 1.3h) is used to find arbitrary \langle place \rangle s along a curve in the usual way.

\textbf{Exercise 2.1:} Extend the code given for the curves in the previous picture so as to add the labels giving the number of control points. (p.577)

Using \texttt{?} will set the current direction to be tangential at that \langle place \rangle, and one can \langle slide \rangle specified distances along the curve from a found \langle place \rangle using the \texttt{?.../dimen}/ notation:

\begin{center}
\begin{tikzpicture}
\draw[->] (0,0) to [out=90,in=180] (3,3) to [out=0,in=90] (6,0);
\node at (1,1) {Q};
\node at (2.5,0) {P};
\node at (0,0) {A};
\node at (6,0) {B};
\end{tikzpicture}
\end{center}

\textbf{Exercise 2.2:} Suggest code to produce something like the above picture; the spline curve is the same as in the previous picture. \textit{Hints:} The line is 140pt long and touches 0.28 of the way from \(A\) to \(B\) and the \(x\) is 0.65 of the way from \(A\) to \(B\). (p.577)

The positions in \langle poslist \rangle specify \textit{control points} which determine the initial and final directions of the curve—leaving \(p\) and arriving at \(c\)—and how the curve behaves in between, using standard spline constructions. In general, control points need not lie upon the actual curve.

A natural spline parameter varies in the interval \([0,1]\) monotonically along the curve from \(p\) to \(c\). This is used to specify \langle place \rangle along the curve, however there is no easy relation to arc-length. Generally the parameter varies more rapidly where the curvature is greatest. The following diagram illustrates this effect for a cubic spline of two segments (3 control points).
Exercise 2.3: Write code to produce a picture such as the one above. (Hint: Save the locations of places along the curve for later use with straight connections.) (p.577)

To have the same \langle pos \rangle occurring as a multiple control point simply use a delimiter, which leaves the \langle pos \rangle unchanged. Thus \texttt{\textbackslash curve\{\langle pos \rangle\&\}} uses a cubic spline, whereas \texttt{\textbackslash curve\{\langle pos \rangle\}} is quadratic.

Repeating the same control point three times in succession results in straight segments to that control point. Using the default styles this is an expensive way to get straight lines, but it allows for extra effects with other styles.

Curve Objects: At present the syntax is supported for using a \texttt{\textbackslash curve} object only as a decoration, constructed from a \texttt{\textbackslash crv} object used as a connection.

\texttt{% xydef@\curve\{\connect\crv%}

The more general \texttt{\textbackslash crv} object currently works as a connection. In future this object-type will be extended to allow closed curves as the boundaries of objects.

\texttt{% xydef@\crv#1#{\hbox \bgroup \crvresetbreaks@ \initshape@}
\def\pure@crv{\crv@saveStyles@}\crv@{#1}}

\texttt{% xydef@\crv@#1#2{%}
\DN@{#1}\ifx\empty\next@
\DN@{\def\afterCURVEaction@{\}\parsecurve@\xycurve@}@%
\else\DN@{\parsecurve@@\textbackslash curveSTYLE@\fi
\next@ #2\@endcurve\endcrv@ }
\texttt{% xydef@\curveSTYLE@{\xycurve@@}% default style}
\texttt{% xydef@\afterCURVEaction@{\% default after-curve action}
\texttt{% xydef@\endcurve@{\%}
\texttt{% xydef@\endcrv@@{\endcrv@\POS}%
\texttt{% xydef@\pure@crv{\relax}

The \@endcurve inserted here ensures that subsequent parsing with \texttt{\checkendcurve@} (see below) will terminate cleanly.

The \texttt{\textbackslash crvs} object has restricted functionality with choice of styles and is always used for curves only. It has a single braced argument which can be used to set curve styles. Its main use is for re-typesetting different portions of the same curve, for example the subsegments of a curved arrow/path segment.

This is done by first examining \texttt{\textbackslash bstartPLACE@}. If \texttt{\relax} or undefined, then the curve is set using \@crv@ to decide the style. If \texttt{\bstartPLACE@} is \texttt{\empty} then the curve is fully processed as a connection but nothing is actually typeset; this is used by curved arrows where the curve is first set as “invisible” before breaks and labels are processed. When \texttt{\bstartPLACE@} contains a number, normally within the range 0 to 1, this is interpreted as a \langle place \rangle along a curved connection that has already been established. No typesetting may occur before this \langle place \rangle.
In this case the control point information is discarded since the curve can be recovered using \texttt{\splitreset}. The current $p$ and $c$ are no longer the endpoints of the curve but are positions along the curve between which the typesetting should occur. The value of \texttt{\bstartPLACE} is typically a (place) along the curve which is within the (pos) at $p$. It is used to help locate the edge of this (pos) where typesetting should commence. Similarly \texttt{\bendPLACE} is typically a (place) along the curve which is within the (pos) at $c$.

\begin{verbatim}
\xywarnifdefined\crv@normaltemplate
\xywarnifdefined\crv@othertemplate
\xywarnifdefined\crv@specialtemplate@@
\{\xyuncatcodes \catcode`\@=11 \catcode`\#=6
\gdef\crv@normaltemplate#1{{}{~**#1\xy@@crvaddstack@}}
\gdef\crv@othertemplate#1{{}{~*=<2\jot>{}~**#1\xy@@crvaddstack@}}
\gdef\crv@specialtemplate@@#1{{}{~**\dir{#1}\xy@@crvaddstack@}}
\}
\xydef@\crvs#1#{\hbox \bgroup \def\pure@crv{\relax}\crvs@{#1}}%
\xydef@\crvi#1#{\hbox \bgroup \def\pure@crv{\relax}\invisbreaks@ \crvs@{#1}}%
\xydef@\crvs@#1{\DN@{\initshape@ \@crv@{#1}}%
\def\crv@defaultshape{-}%
\ifx\bstartPLACE@\relax \else
\ifx\bstartPLACE@\empty \Invisible@true \else
\splinetrace@{\bstartPLACE@=\bstartPLACE@, \bendPLACE@=\bendPLACE@}%
\DN@{\let\xy@crvstack@=\xy@samecurve@}
\let\saveshape@=\savesame@
\let\savectrlptsnum@=\relax
\let\startxycurve@=\recovercurve@
\crv@cnt@=\xycrvptsnum@\relax \@crv@{#1}}%
\fi \fi \next@ }
\xydef@\xy@@samecurve@{\xyFN@\checkendcurve@}
\xydef@\xy@samecurve@{\xyFN@\checkendcurve@}
\xydef@\xy@samecurve@{\xyFN@\checkendcurve@}
\xydef@\crv@cnt@=\xycrvptsnum@ \splinereset@ \recovercv@end
\ifx\bendPLACE@\relax \def\bendPLACE@{1}\fi
\ifdim\zz@\R@c \ifdim\zz@\L@c \ifdim\zz@\D@c \ifdim\zz@\U@c
\Edge@c={\zeroEdge}\fi\fi\fi
\edef\cv@end{\cfromthec@}\edef\cv@start{\cfromthep@}
\edef\cv@end{\cfromthec@}\edef\cv@start{\cfromthep@}
\xydef@\recovercv@end{\count@=\ptsnum@ \relax \advance\count@@one
\edef\tmp@{\the\count@}\csname cv@\tmp@\endcsname}%
\xydef@\crvSTYLE@@{}
\expandafter\xydef@\expandafter\crv@defaultshape\expandafter{\addDASH@{}}
\xydef@\crvSTYLE@@{}
\expandafter\xydef@\expandafter\crv@defaultshape\expandafter{\addDASH@{}}
\xydef@\crvSTYLE@@{}
\expandafter\xydef@\expandafter\crv@defaultshape\expandafter{\addDASH@{}}
\xydef@\crvSTYLE@@{\Invisible@true}
\explanafter{\xydef@\crv@defaultshape\expandafter{\addDASH@{}}}
\xydef@\crvSTYLE@@{\Invisible@true}
\explanafter{\xydef@\crv@defaultshape\expandafter{\addDASH@{}}}
\xydef@\crvSTYLE@@{\Invisible@true}
\explanafter{\xydef@\crv@defaultshape\expandafter{\addDASH@{}}}
\xydef@\crvSTYLE@@{\Invisible@true}
\explanafter{\xydef@\crv@defaultshape\expandafter{\addDASH@{}}}
\end{verbatim}
CHAPTER 2. EXTENSIONS

Bug: this should use a \Step@@ method to get the spacing for dotting; this will eliminate the need for templates...

Parsing: Two separate parsers are required: one for \curve..., the other for the contents of {...}.

Set \curveSTYLE@ to be the default of \xycurve@@ then examine the following tokens to see if this must be changed.

Set \curveSTYLE@ to be the default of \xycurve@@ then examine the following tokens to see if this must be changed.

Procedure: If the first token of #1 (from \crv@) is ~ then a letter should follow, determining how to set the style. If instead it is an active control sequence then issue a warning message, but let it do its thing anyway. However \parsecurve@ is a normal thing to encounter, so no warning is required. Currently a single letter without the ~ will be recognised, but two-letter combinations definitely need the ~.

\xydef\parseCURVE@{\{\def\curveSTYLE@{\xycurve@@}\def\afterCURVEaction@{}}% 
\xyFN@parseCURVE@@} 
\xydef\parseCURVE@@{% 
\ifx\space@\next\expandafter\DN@\space{\xyFN@parseCURVE@@}% gobble spaces 
\else\ifx ~\next \DN@ ~{\xy@~{}}\xyFN@setcurveSTYLE@@% 
\else\ifx\parsecurve@\next\DN@@%
### 2.1. CURVE AND SPLINE EXTENSION

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{curve}(\text{modifier}){\texttt{curve-object}{\texttt{poslist}}}</td>
<td>construct curved connection</td>
</tr>
</tbody>
</table>

- **(modifier)** → \texttt{(empty)}
  - | \texttt{~}(\text{curve-option}) (\text{modifier}) | zero or more modifiers possible; default is \texttt{C} |

- **(curve-option)** → \texttt{p} | \texttt{P} | \texttt{1} | \texttt{L} | \texttt{c} | \texttt{C} | \texttt{pc} | \texttt{pC} | \texttt{Pc} | \texttt{PC} | \texttt{lc} | \texttt{1C} | \texttt{1C} | \texttt{Lc} | \texttt{LC} | \texttt{cC} | \texttt{cC} |
  - show only control points (p=points), joined by lines (l=lines), or curve only (c=curve) |
  - show control points and curve \texttt{2.1f} and curve \texttt{2.1e} |
  - show lines joining control points and curve \texttt{2.1g} |
  - plot curve twice, with and without specified formatting \texttt{2.1e} |

- **(curve-object)** → \texttt{(empty)}
  - use the appropriate default style |
  - \texttt{~}(\text{object}) (\text{curve-object}) | \texttt{~}(\text{object}) (\text{curve-object}) |
  - specify the “drop” object \texttt{2.1a} and maybe more \texttt{2.1c} |
  - specify “connect” object \texttt{2.1b} and maybe more \texttt{2.1c} |

- **(poslist)** → \texttt{(empty)} | \texttt{~}(\text{pos}) (\texttt{delim}) (\texttt{poslist})
  - list of positions for control points |
  - \texttt{~}(\text{delim}) (\texttt{poslist}) add the current stack to the control points |

- **(delim)** → & allowable delimiter

---

**Figure 2.1:** Syntax for curves.

Options are normally specified using \texttt{~}. Sometimes this can be safely omitted; if such a case is encountered then a warning message is issued advising to use \texttt{~}. This nicety may be removed in future versions.

---

Examine the next token to see if it determines a curve style. Currently only \texttt{p}, \texttt{1} and \texttt{c} are recognised, along with the uppercase variants \texttt{P}, \texttt{L} and \texttt{C}. When one of these letters is encountered, set \texttt{curveSTYLE}\texttt{2} and keep examining tokens with \texttt{setafterCURVEaction}.
\DN@ c\{xyFN@\setafterCURVEaction@\%
\else\ifx\next C\def\curveSTYLE@\{\xycurve@@\%
\DN@ C\{xyFN@\setafterCURVEaction@\%
\else \DN@#1\{xywarning@\{unknown curve style ##1\}xyFN@\parseCURVE@@\%
\fi\fi\fi\fi\fi\fi \next@ 
\}

\setafterCURVEaction@ examines tokens to see if a 2-letter combination is being specified; e.g. pc, lC, etc. Alternatively another ~ can set a new specification.

\xydef@\setafterCURVEaction@{%
\ifx\next~\DN@~\{xyFN@\setcurveSTYLE@@\%
\else\ifcat a\noexpand\next\DN@\{\setafterCURVEaction@@\%
\else\DN@\{xyFN@\parseCURVE@@\%
\fi\fi \next@}

Currently the only 2-letter specifications have either c or C as the 2nd letter.

\xydef@\setafterCURVEaction@@{%
\ifx\next c\DN@c{%
\def\afterCURVEaction@\{\noexpand\endcurve\noexpand\xy@curve@@\%
\xyFN@\parseCURVE@@ \}%
\else\ifx\next C\DN@ C{%
\def\afterCURVEaction@\{\noexpand\endcurve\noexpand\xy@curve@@\%
\xyFN@\parseCURVE@@ \}%
\else \DN@#1\{xywarning@\{unknown after-curve action ##1\}xyFN@\parseCURVE@@ \%
\fi\fi \next@ \}

When \parsecurve@ is encountered, this signifies the end of this part of the parsing. The only active control sequence that should be encountered legitimately here is \@endcurve, which signifies that the (poslist) is empty and default styles are required. Otherwise assume we the subsequent tokens are from \#2 of \crv@. Control passes to \xycurve@@ to prepare for reading the (crv-object)s and (poslist).

\xydef@\parsecurve@\{\xyFN@\parsecurve@@\}
\xydef@\parsecurve@@{%
\ifx\space@\next\expandafter\DN@\space\{\xyFN@\parsecurve@@\%
\else\ifx\next@\endcurve\DN@\@endcurve\{\checkafterCURVE@\%
\else\ifcat\active\noexpand\next\DN@\{\relax\%
\else\DN@\{\xycurve@@\fi\fi\fi \next@\%

\xydef@\xycurve@\begingroup\afterCURVE\{\setcurve@\endgroup\crvobjects@
\startxycurve@\xycrvmods@}
\xydef@\xycurve@@\{\afterCURVE\{\setcurve@\crvobjects@\startxycurve@\xycrvmods@\}
\xydef@\xy@curve@\{\xy@curve@@\{\splinedefaulttol\crvobjects@\}
\xydef@\xy@curve@@\{\xy@curve@@\{\resetcrvobjects@\}
\xydef@\xy@curve@@@#1\{\ifx\cv@start\relax
\DN@\xywarning@\{There is no curve to plot\}.
\else\DN@\begingroup\afterCURVE\{\setcurve@\endgroup\#1\cv@end\%
\fi \next@ \}
\xydef@\crvobjects@\{\def\xycrvdrop@\{\def\xycrvconn@\{\}
\xydef@\resetcrvobjects@\{\crvobjects@\}
These were originally provided for sophisticated-user access. They are otherwise undocumented and may be removed.

\texttt{xylet@\savecurve=xycurve@@}
\texttt{xylet@\samecurve=xy@curve@}
\texttt{xy@crvmods@ handles reading the (curve-object)s, \textit{i.e.} the “drop” object and the “connect” object.}

**Procedure:**
1. **Get next token.**
2. **Is it a “?”**
3. **If not, exit.**
4. **If so, is it followed by a “*”?**
5. **If not, issue a message and go back to 1.**
6. **Otherwise,**
7. **is there a second “*”?**
8. **If not then read and store the “drop” object,**
9. **else, read and store the “connect” object.**
10. **When finished, in either case,**
11. **look for further specifications.**
12. **An empty “drop” object is not allowed, so this actually gives the default of \zerodot.**

Parsing of the \langle poslist \rangle is quite simple, recognising few special tokens:

- \texttt{\endcurve} and \texttt{\endxy} terminate reading of control points.
- \\
- \& and \texttt{\relax} delimit (pos)itions
- \\
- anything else is treated as a (pos), being handled by the \texttt{\POS@} parser.
There could be a problem in that an invalid token would cause an infinite loop, passing back and forth between `\checkendcurve` and `\POSt`. This is avoided by setting a flag `\setsearchflag` when the `\POSt` parser is called, not following a valid delimiter. Encountering a valid delimiter clears the flag. If `\checkendcurve` is called with the flag set, a warning message is issued and the token is skipped; the flag remains set. Processing continues, but it is possible that the wrong number of positions will be read; presumably there is some kind of error that needs to be fixed anyway.

It is possible for the final `\endcurve` to be omitted, but only when the next token is `\endxy`. A warning message is written to the log file, protesting against this sloppy T\(\TeX\)-ing.

Creation of the `\crv` is completed when `\endcrv` is encountered. At this point the usual methods `\Drop@@` and `\Connect@@` are defined. Extra information is retained, using `\saveshape` and `\savecrvparams` for the benefit of methods which treat the curve as a connection.
The control names \crvXY@pre and \crvXY@post are used to transfer style information passed with \preconnect to subsegments.

The values of \X@min, \X@max, etc. are used to compute the extents \L@c, \R@c, etc. of the object. This information must be maintained after the group is closed, unless the “hidden” attribute is required or we are inside a matrix construction.

In calculating the size of the box containing the curve \X@min, \X@max, etc. are initialised to describe the minimum rectangle enclosing p and c. At the same time we save the current scope.

To Do: This can be improved. For example, trace along the spline until an appropriate point is
found. There may be more than one such point, so extra criteria may be required.

Notes

2.1a. The “drop” object is set once, then “dropped” many times at appropriately spaced places along the curve. If directional, the direction from \( p \) to \( c \) is used. Default behaviour is to have tiny dots spaced sufficiently closely as to give the appearance of a smooth curve. Specifying a larger size for the “drop” object is a way of getting a dotted curve (see the example in the next note).

2.1b. The “connect” object is also dropped at each place along the curve. However, if non-empty, this object uses the tangent direction at each place. This allows a directional object to be specified, whose orientation will always match the tangent. To adjust the spacing of such objects, use an empty “drop” object of non-zero size as shown here:

\[
\xy (0,0)**{A}; (50,-10)**{B}
**\crv{z*=<4pt>{.} (10,10)&(20,0)&(40,15)}
**\crv{z*=<8pt>{.}**!/-5pt/\dir{>}(10,-20)
&(40,-15)} \endxy
\]

When there is no “connect” object then the tangent calculations are not carried out, resulting in a saving of time and memory; this is the default behaviour.

2.1c. The “drop” and “connect” objects can be specified as many times as desired. Only the last specification of each type will actually have any effect. (This makes it easy to experiment with different styles.)

2.1d. Complicated diagrams having several spline curves can take quite a long time to process and may use a lot of \TeX’s memory. A convenient device, especially while developing a picture, is to show only the location of the control points or to join the control points with lines, as a stylized approximation to the spline curve. The \langle curve-option\rangle s \texttt{\~p} and \texttt{\~l} are provided for this purpose. Uppercase versions \texttt{\~P} and \texttt{\~L} do the same thing but use any \langle curve-object\rangle s that may be specified, whereas the lowercase versions use plain defaults: small cross for \texttt{\~p}, straight line for \texttt{\~l}. Similarly \texttt{\~C} and \texttt{\~c} set the spline curve using any specified \langle curve-option\rangle s or as a (default) plain curve.

2.1e. Use of \texttt{\~p}, \texttt{\~l}, etc. is extended to enable both the curve and the control points to be easily shown in the same picture. Mixing upper- and lower-case specifies whether the \langle curve-option\rangle s are to be applied to the spline curve or the (lines joining) control points. See the examples accompanying the next two notes.

2.1f. By default the control points are marked with a small cross, specified by \texttt{\*\dir{x}}. The “connect” object is ignored completely.
2.1. CURVE AND SPLINE EXTENSION

2.1g. With lines connecting control points the default “drop” object is empty, while the “connect” object is \texttt{\dir{-}} for simple straight lines. If non-empty, the “drop” object is placed at each control point. The “connect” object may be used to specify a fancy line style.

2.1h. When a stack of \texttt{⟨pos⟩i}tions has been established using the \texttt{@i} and \texttt{@+} commands, these positions can be used and are appended to the \texttt{⟨poslist⟩}.

Spline Methods Each \texttt{\curve} has to set the sliding abilities. This is done by a call to \texttt{\crvconnect@@} which gives values to the methods \texttt{\Creset@@}, \texttt{\Cshavep@@}, etc. These values depend on the style of the curve itself.
When straight lines are required ...

These are the actual methods used for a sequence of straight segments.

These are the actual methods used for a sequence of straight segments.
2.1. CURVE AND SPLINE EXTENSION

\begin{verbatim}
1028 \advance\dimen0\segmentnum0\p@
1029 \advance\count0\segmentnum0\relax
1030 \def\segmentnum0{0}%
1031 \ifnum\count0<\count%
1032 \xywarning{parameter value #1 too large, using \the\count0}%
1033 \count0=\count0 \advance\count0\m@ne \edef\PLACEf@{{1}}%
1034 \else
1035 \advance\dimen0-\count0\p@
1036 \ifdim\zz0\dimen0\ifnum\count0>\z@\advance\count0\m@ne \edef\PLACEf@{{\expandafter\removePT@\the\dimen0}}%
1037 \else
1038 \edef\PLACEf@{{\expandafter\removePT@\the\dimen0}}%
1039 \fi
1040 \ifnum\count0=\z@\else \bgroup
1041 \csname cv@number\count@\endcsname
1042 \edef\tmp@{\egroup \X@c=\the\X@c \Y@c=\the\Y@c}\tmp@
1043 \fi
1044 \ifnum\count0<\count%
1045 \edef\PLACEf@{{\expandafter\removePT@\the\dimen0}}%
1046 \csname cv@number\count@\endcsname
1047 \edef\PLACEf@{{\expandafter\removePT@\the\dimen0}}%
1048 \csname cv@number\count@\endcsname
1049 \fi
1050 \d@X=\X@c \advance\d@X-\X@p \d@Y=\Y@c \advance\d@Y-\Y@p
1051 \expandafter\noCalong@@ \PLACEf@ \czeroEdge@ }
1055 \xydef@\straitintercept@{\noCintercept@@}
1056 \xydef@\straitbreak@{\straightCbreak@@}
1057 \xydef@\straitlast@{\straightClast@@}

When curved segments are required . . .

\begin{verbatim}
1063 \xydef@splineconnect@{%
1064 \let\crvreset@=\splinereset@
1065 \let\crvshavep@=\splinehavep@
1066 \let\crvshavec@=\splinehavec@
1067 \let\crvslidep@=\splineslidep@
1068 \let\crvslidesc@=\splineslidesc@
1069 \let\crvalong@=\splinealong@
1070 \let\crvintercept@=\splineintercept@
1071 \let\crvbreak@=\splinebreak@
1072 \let\crvlast@=\splinelast@}
1073 }
\end{verbatim}

These are the actual methods.

Reset the spline parameters:

\verbatim
\xycrvptsnum@: expands to a control sequence, whose name incorporates the correct scoping level, giving the number of control points.
\verbatim
\the\crvpts@: reads the control point locations and other information from a token list.
\verbatim
\xysplineparams@: set to a control sequence name, incorporating the correct scoping level, which will allow access to the Bezier control points for each spline segment, when required.
\verbatim
\end{verbatim}
\texttt{xysplineedges@}: set to a control sequence name, incorporating the correct scoping level, which gives the spline parameters of the edges of objects at \( p \) and \( c \);

\texttt{segmentnum@}: reset to 0.

\texttt{splinecorrect@p} and \texttt{splinecorrect@c}: reset to 0 until changed by a \texttt{\textbackslash Cshavep@} or \texttt{\textbackslash Cshavec@}.

\texttt{splinelength@}: reset to 0 until changed by a \texttt{\textbackslash Cshavep@}, \texttt{\textbackslash Cshavec@} or \texttt{\textbackslash Calong@}; this is needed by \texttt{\textbackslash Cshavep@} and \texttt{\textbackslash Cshavec@} to quickly locate where to slide from, especially when it is necessary to slide across more than a single spline segment.

\texttt{splineplace@}: reset to \( .5 \).

\texttt{xdef@splinereset@}{\the\crvpts@
edef\xycrvptsnum@{\csname ptsnum@\endcsname}\edef\xysplineparams@{\expandafter\noexpand\csname params@\endcsname}\edef\xysplineedges@{\expandafter\noexpand\csname edges@\endcsname}\def\segmentnum@{1}\def\splinecorrect@p{0}\def\splinecorrect@c{0}\splinelength@=\z@ \def\splineplace@f{.5}\}

\textit{Shaving} to the appropriate edge. This is simply a matter of reading the stored edge information and storing the appropriate value in \texttt{splinecorrect@p} or \texttt{splinecorrect@c}. The difficult part is to construct the correct control sequence name and then to extract the correct part of the text in its expansion.

\texttt{xdef@splineshavep@}{\begingroup\edef\xysplineedges@{\expandafter\noexpand\csname edges@\endcsname}\expandafter\expandafter\expandafter\getsplineedges@safe\xysplineedges@ @@\edef\tmp@{\noexpand\removePT@\the\dimen5}\edef\tmp@{\endgroup\noexpand\def\noexpand\splinecorrect@p{\tmp@}}\tmp@\setupDirection@ii }\texttt{xdef@splineshavec@}{\begingroup\edef\xysplineedges@{\expandafter\noexpand\csname edges@\endcsname}\expandafter\expandafter\expandafter\getsplineedges@safe\xysplineedges@ @@\edef\tmp@{\noexpand\removePT@\the\dimen5}\edef\tmp@{\endgroup\noexpand\def\noexpand\splinecorrect@c{\tmp@}}\tmp@\setupDirection@ii }\texttt{xdef@splineslidep@#1}{\enter@{\pfromthep@}% \begingroup \splinealong@@{0}{#1}+\splineslidep@@@ \setupDirection@ii \leave@ }\texttt{xdef@splineslidep@@@}{\def\tmp@{\postfind@}\tmp@ \dimen@=\splineval@ \d@X=\dimen5 \d@Y=\dimen7 \edef\tmp@{\postfind@}\tmp@ \dimen@=\splineval@}
2.1. CURVE AND SPLINE EXTENSION

\begin{verbatim}
\count@=\xycrvptsnum\relax
\ifnum\count@>\tw@ \advance\count@\m@ne
\advance\dimen@=\segmentnum\p@ \advance\dimen@-\p@
\fi
\edef\tmp@{\endgroup \endgroup
\noexpand\def\noexpand\xysplineval@{\the\splineval@}%
\noexpand\def\noexpand\splinecorrect@p{\expandafter\removePT@\the\dimen@}%
splinetrace{slide = \the\splineval}\%
\X@c=\dimen@i \Y@c=\dimen3 \% something may be wrong here!!
\d@X=\the\d@X \d@Y=\the\d@Y \%
\noexpand\def\noexpand\segmentnum@{\segmentnum\}} \tmp@ }
\xydef@\splineslidec@#1{\enter@{\pfromthep}@% 
\begingroup
\expandafter\splinealong@@\PLACEf@{#1}+\splineslidec@@@%
\setupDirection@ii \leave@ }
\xydef@\splineslidec@@@{\d@X=\dimen5 \d@Y=\dimen7 \edef\tmp@{\postfind@}\tmp@ \dimen@=\splineval@
\expandafter\count@=\xycrvptsnum\relax
\ifnum\count@>\tw@ \advance\count@\m@ne \advance\dimen@-\p@
\dimen@=-\dimen@ \advance\dimen@=\count@\p@
\else \dimen@=-\dimen@ \advance\dimen@=\p@
\fi %\W@{**slide**<\the\dimen@>***}%
\%% splinecorrect@c is probably wrong here!!
\edef\tmp@{\endgroup \endgroup
\noexpand\def\noexpand\xysplineval@{\the\splineval@}%
\noexpand\def\noexpand\splinecorrect@c{\expandafter\removePT@\the\dimen@}%
splinetrace{slide = \the\splineval\; correct@c=\noexpand\splinecorrect@c}%
\X@c=\dimen@i \Y@c=\dimen3 \%
\d@X=\the\d@X \d@Y=\the\d@Y \%
\noexpand\def\noexpand\segmentnum@{\segmentnum\}} \tmp@ }
\xydef@\checkslidemore@#1{% 
\d@X=-\dimen5 \d@Y=\dimen7 \edef\tmp@{\postfind@} \tmp@ \dimen@=\splineval@
\expandafter\count@=\xycrvptsnum\relax
\ifnum#1<\z@ \advance\count@\m@ne \dimen@=\p@
\else \advance\count@\@ne \dimen@=\z@
\fi
\ifnum\count@<\@ne \expandafter\splinesegment@\segmentnum\relax\fi
\dn@={\xywarning@{cannot slide beyond start of curve} \spline@find{\z@}{\z@}{} \splineslidec@@ \leave@ }%%%% ?
\else \splinetrace{not found yet, \the\dimen@i\space still to go}%
\dimen@i=#1\relax\ifdim\dimen@i<\z@\multiply\dimen@i\m@ne\fi
\dimen@=-\splineval\; \advance\dimen@=\dimen@i\relax
\ifdim\dimen@i<\z@ \dimen@i=-\dimen@i\else\dimen@i=\dimen@i\fi
\ifdim\dimen@i<.2\p@\dn@{\splineslidec@ \leave@}%%% ?
\else \splinetrace{not found yet, \the\dimen@i\space still to go}%
\dimen@i=#1\relax\ifdim\dimen@i<\z@\multiply\dimen@i\m@ne\fi
\dn@={\expandafter\splineslidemore@\expandafter{\the\dimen@i}}\fi
\next@}%
\xydef@\splineslidemore@#1{% 
\expandafter\count@=\segmentnum\relax
\ifdim#1<\z@ \advance\count@\m@ne \dimen@=\p@
\else \advance\count@\@ne \dimen@=\z@
\fi
\ifnum\count@<\@ne \expandafter\splinesegment@{\segmentnum\relax\fi
\dn@={\xywarning@{cannot slide beyond start of curve} \spline@find{\z@}{\z@}{} \splineslidec@ \leave@ }%%% ?
\else \expandafter\count@@=\xycrvptsnum\relax
\advance\count@@\m@ne
\expandafter\splinesegment@\segmentnum\relax
\expandafter\splineslidemore@\expandafter{\the\dimen@}\fi
\endgroup
\end{verbatim}
\end{document}
The method \texttt{splinealong@#1} finds places along the curve, when #1 is given as a factor, normally between 0 and 1. The location of the edges of the objects at \texttt{p} and \texttt{c} are stored in \texttt{splinecorrect@p} and \texttt{splinecorrect@c}. These are the places found if the factor is either 0 or 1. If an edge is non-zero then it is actually possible to move to points inside that edge by specifying a factor less than 0 or greater than 1.

The parameter #2 in \texttt{splinealong@@#1#2#3} allows for a subsequent slide through a given ⟨dimen⟩ along the curve, from the place specified by the factor in \texttt{#1}. The information supplied in \texttt{#1} must be converted to the segment number and parameter value of the specified point. These will be calculated and stored temporarily in \texttt{count@} and \texttt{dimen@} before the values are passed respectively to \texttt{splinesegment@} which chooses the segment, and \texttt{spline@find} to locate the point on this segment.

To do the calculation, first we correct for the parameter values of the edges of the objects at the end-points, which are stored in \texttt{splinecorrect@p} and \texttt{splinecorrect@c}. If the resulting number \(x\) is not an integer then \texttt{count@} is set to \(1 + \lfloor x \rfloor\) while \texttt{dimen@} becomes \((x - \lfloor x \rfloor)\p@\). If \(x\) is an integer: \(x = 0\) gives \texttt{count@=1} and \texttt{dimen@=\z@}, otherwise \texttt{count@} = \(x\) and \texttt{dimen@=\p@}.
2.1. CURVE AND SPLINE EXTENSION

\edef\tmp@{#1}\relax \dimen@=-\tmp@\dimen@ \relax
\expandafter\advance\expandafter\dimen@\splinecorrect@p\p@
\getcrvsegsnum@ \multiply \dimen@ by\count@\relax
\splinetrace@{actual:\the\dimen@; f:\the\dimen@ii; p:\splinecorrect@p, c:\splinecorrect@c}\
\count@=\dimen@ \divide\count@\p@ \DN@{\relax}\
\expandafter\ifnum\xycrvptsnum@<\count@
\xywarning@{parameter value #1 too large}\
\DNii@{\relax}\DN@{\begingroup}
\else
\edef\xysplineval@{\the\dimen@}\
\advance\dimen@-\count@\p@
\ifdim\dimen@=\z@ \splinetrace@{find knot point \the\count@}\
\ifnum\count@=\z@ \count@=\one \else \dimen@=\p@ \fi
\else \advance\count@\one
\splinetrace@{find point \the\dimen@ on spline segment \the\count@}\
\fi
\edef\segmentnum@{\the\count@}
\edef\next@{\begingroup\noexpand\splinesegment@{\the\count@}}
\edef\nextii@{\noexpand\spline@find{\the\dimen@}{#2}{}\noexpand\checksplinefind@{#2}}
\ifx>#3\relax
\edef\nextii@{\noexpand\spline@find{\the\dimen@}{#2}{#3}\noexpand\checksplinefind@{#2}}
\else\ifx<#3\relax \ifdim\dimen@<\z@ \relax \ifdim\dimen@<\p@
\dimen@=-\dimen@ \advance\dimen@\p@
\edef\nextii@{\noexpand\splinereverse@}
\noexpand\splinetrace@{find \the\dimen@ on spline segment \the\count@}\
\fi\fi
\fi\fi
\fi\fi
This macro checks whether the required sliding has been carried out, or whether it is necessary to
continue sliding along the next segment.
\xydef@\checksplinefind@#1{\
\dimen@ii=#1\relax \dimen@=\splinelength@\
\ifdim\dimen@ii=\z@\DN@{}\
\else\ifdim\dimen@ii<\z@ \relax \ifdim\dimen@<\p@
\dimen@=-\dimen@ \advance\dimen@\p@
\edef\nextii@{\noexpand\splinereverse@}
\noexpand\splinetrace@{find \the\dimen@ on spline segment \the\count@}\
\fi\fi
\fi\fi
\fi\fi\expandafter\next@\nextii@}
CHAPTER 2. EXTENSIONS

Typesetting breaks in curves is quite involved. The \texttt{\bstartPLACE@} and \texttt{\bendPLACE@} are used to hold parameter values for the positions of the beginning and end of a segment, respectively, when known. The default values of \texttt{\relax} indicate that these values need to be found, usually from the start- and end-edges of the curve. The value of \texttt{\empty} for \texttt{\bstartPLACE@} is used initially with a new curved connection, to set up all its parameters — no typesetting takes place. This is used by curved arrows to set up the connection before places can be found for any breaks.

\begin{verbatim}
\xydef@resetbreaks@{%
  \let\bstartPLACE@=\relax \let\bthisPLACE@=\empty
  \let\bendPLACE@=\relax \let\bprevPLACE@=\relax
\xydef@invisbreaks@{\resetbreaks@ \let\bstartPLACE@=\empty}
\xydef@crvresetbreaks@{\resetbreaks@ \let\bstartPLACE@=\relax }
\xydef@initbreaks@{%\texttt{\bstartPLACE@=\relax} \let\bthisPLACE@=\relax
  \def\bstartPLACE@{0}\def\bendPLACE@{1}\else
  \if\bstartPLACE@=\empty\def\bstartPLACE@{0}\def\bendPLACE@{1}\fi
  \fi}
\xydef@lastbreaks@{%\texttt{\bprevPLACE@=\relax} \def\bstartPLACE@{0}\else
  \expandafter \bprevPLACE@ \fi
  \let\bprevPLACE@=\relax \let\bendPLACE@=\relax
\xydef@\bstartPLACE@=\relax
\xydef@\bendPLACE@=\relax
\xydef@\bprevPLACE@=\relax
\xydef@\bthisPLACE@={1}
\end{verbatim}

These are called in the actual setting of curved sub-segments.

\begin{verbatim}
\xydef@\splinebreak@{\splinetrace@{***new break*** this:\bthisPLACE@ }% \ifs\bthisPLACE@=\empty\def\bendPLACE@={1}\fi
\end{verbatim}
Intersection with a curved connection  Just as the intersection of two lines (1.3j) can be found, so can the intersection of a straight line with a curved connection, or the intersection of a curve with a straight connection.

When the line separates the end-points of a curve an intersection can always be found. If there is more than one then that occurring earliest along the curve is the one found.

Presume the end-points of the spline segment are at \((X_p, Y_p)\) and \((X_c, Y_c)\) with \((L_p, U_p)\) and \((R_p, D_p)\) as two points on the line. Then the macro \texttt{\xysidesofinterval}\@ returns 3 pieces of information.
\next expands to 2 characters, each from \{+, -, 0\} indicating which side of, or on, the line lies either end-point;

\this expands to either + or - indicating which end is closer to the line, returning + if equi-distant;

\howclose contains a non-negative (num) indicating the distance to the nearest end-point.

\begin{verbatim}
\xydef\splineNointercept@{\enter@{\pfromthep@}}%
\begingroup \splinereset@ \csname cv@0\endcsname \swap@
\count@=\ptsnum@ \advance\count@\@ne \csname cv@\the\count@\endcsname
\edef\tmp@{\endgroup \X@origin=\the\X@p \Y@origin=\the\Y@p \R@c=\the\d@X \U@c=\the\d@Y}\tmp@
\loop@
\dimen@=}\ifdim\R@c<\z@-\fi\R@c
\advance\dimen@ \ifdim\U@c<\z@-\fi\U@c
\ifdim\dimen@>10\p@
\advance\R@c \ifdim\R@c<\z@-\fi 16sp \divide\R@c\KK@
\advance\U@c \ifdim\U@c<\z@-\fi 16sp \divide\U@c\KK@
\repeat@
\intersect@ \leave@}
\xydef\splineintercept@{\Hidden@true\Invisible@true\splineintercept@i}
\xydef\splineintercept@i{\enter@{\pfromthep@}\begingroup
\R@p=\X@c \L@p=\X@p \d@X=\R@p \advance\d@X-\L@p
\D@p=\Y@c \U@p=\Y@p \d@Y=\D@p \advance\d@Y-\U@p
\dimen@=}\KK@ \d@X \edef\sd@X{\expandafter\removePT@\the\d@X}\
\dimen@=}\KK@ \d@Y \edef\sd@Y{\expandafter\removePT@\the\d@Y}\
\dimen@=\sd@Y\L@p \advance\dimen@-=\sd@X\U@p
\edef\sideOffset@{\the\dimen@}\def\closestseg@{1}\
\let\crvconnect@@=\relax \crvconnect@
\let\spline@end=\splinef@end
\def\spline@start{\bgroup \def\prevnext@{+-}\
\def\splineBext@{\let\splineBext@=\splineIrec@ \splineinterceptTest@}\
\def\splineBcast@{\let\splineBcast@=\splinedecast@@\
\def\splineBadvance@{\splineIadvance@@}\splinedecast@@}\
\def\splineBadvance@{\intercepthull@}\
% \def\splinepoint@{\splineIpt@@}\
\global\let\splinepoint@=\splineIpt@@\global\let\splinegoal@=\splinef@pt\global\let\splinefinish@=\splinefinish@@\
\begingroup \splinesegments@\splinesetparams@ \splined@@ \def\postfind@{}\splineIplace@}
\setupDirection@ii \leave@}
\xydef\splineIplace@{\d@X=}\dimen5 \d@Y=}\dimen7 \dimen@=}\splineval@
\expandafter\count@\xycrvptsnum@\relax \ifnum\count@>\tw@ \advance\count@\m@ne \%\advance\count@\m@ne
\advance\dimen@=}\segmentnum@\p@ \advance\dimen@=\p@
\dimen@=}\count@\dimen@ \% \divide\dimen@=}\count@
\end{verbatim}
2.1. CURVE AND SPLINE EXTENSION

\def\noexpand\PLACEf@{{\expandafter\removePT@\the\dimen@}}\%
\X@c=\dimen@i \Y@c=\dimen@n \%
\d@X=\the\d@X \d@Y=\the\d@Y \}
\splinetrace@{PLACE = \PLACEf@:(\the\X@c, \the\Y@c)}\%
\expandafter\edef\expandafter\bthisPLACE@\PLACEf@{}

\xydef@\splineIrec@{\splineinterceptTest@ }
\xydef@\quineIpt@{\edef\tmp@{\endgroup
\def\noexpand\prevnext@{\next}\begingroup}\tmp@ }
\xydef@\quineIadvance@{\edef\tmp@{\endgroup
\def\noexpand\prevnext@{\next}\def\noexpand\howclose@{\howclose@}}%
\tmp@ \splineadvance@@\begingroup}
\xydef@\cubicIpt@{\edef\tmp@{\endgroup
\def\noexpand\prevnext@{\next}\begingroup}\tmp@ }
\xydef@\cubicIadvance@{\edef\tmp@{\egroup
\def\noexpand\prevnext@{\next}\def\noexpand\howclose@{\howclose@}}%
\tmp@ \splineadvance@@\bgroup}
\xydef@\splineIsegment@{\DN@{\splinesegment@\@ne} \%
\if\splinesegment@\squinesegment@\else
\if\splinesegment@\cubicsegment@\else\DN@{\findIsegment@}\fi
\fi
\next@}
\xydef@\findIsegment@{\expandafter\count@\xycrvptsnum@ \count@\z@ \%
\loop@ \advance\count@\@ne \bgroup
\expandafter\splinesegment@\expandafter{\the\count@} \%
\xysidesofinterval@ \%
\findIsegment@i \repeat@ \%
\expandafter\dimen@\prevhowclose@ \relax \%
\ifdim\dimen@=\z@ \edef\closestseg@{\the\count@}\fi \%
\expandafter\splinesegment@\expandafter{\closestseg@}} \%
\xydef@\findIsegment@i{%}
\expandafter\ifx\expandafter\next \expandafter\findIsegment@i \%
\else\expandafter\findIsegment@x\fi \next@ }
\xydef@\findIsegment@ii{%}
\edef\next@{\egroup\def\noexpand\next{\next}}\%
\def\noexpand\prevhowclose@{}\%
\def\noexpand\closestseg@{\closestseg@}\noexpand\iffalse}} \%
\xydef@\findIsegment@y{%}
\edef\next@{\egroup\def\noexpand\next{\next}}\%
\def\noexpand\prevhowclose@{}\%
\def\noexpand\closestseg@{\closestseg@}\noexpand\iftrue}} \%
\xydef@\xysidesofinterval@{\bgroup
\L@p=sd@X \Y@p \advance\L@p-\sd@Y \X@p \advance\L@p\sideOffset@
When the line misses the convex hull of the curve's control points it is still possible to get the point of closest approach.

When the line misses the convex hull of the curve's control points it is still possible to get the point of closest approach.
2.1. CURVE AND SPLINE EXTENSION

This is for straight connections intercepting with a curve. It requires a redefinition of \PLACE@intercept@ to recognise that the intercept is with a curved connection.
To Do: If both connections are curves, only the line joining the end-points of the first connection is used. Find a way to do better than this.

If the line does not separate the end-points then there may be no intersection with the curve. If there is one then either the line is tangential or necessarily there will also be at least one other intersection. A message

perhaps no curve intersection, or many.

is written to the log-file, but a search for an intersection will still be performed and a “sensible” place found on the curve. In the usual case of a single quadratic or cubic segment, the place nearest the line is found and the tangent direction is established.

The following examples show this, and show how to get the place on the line nearest to the curve.

\[ \begin{align*}
\text{\xy *+{A}="A";p+/r5pc/+{0,15}*=+{B}="B",} \\
&\quad,p-<.5pc,2pc>*+{C}="C","A"+<6pc,-.5pc> \\
&\quad,*+{D}="D","A",\{\ar@/._25pt/"B"\} \\
&\quad,?!{"C";"D"**@{-}}*\dir{x}="E" \\
&\quad,+/_2pc/="F";"E"**@{-},?!{"C";"D"} \\
&\quad,*\{\otimes\endxy}
\end{align*} \]

\[ \begin{align*}
\text{\xy *+{A}="A";p+/r5pc/+{0,15}*=+{B}="B",} \\
&\quad,p-<.5pc,2pc>*+{C}="C","A"+<6pc,-.5pc> \\
&\quad,*+{D}="D","A",\{\ar@/._40pt/"B"\} \\
&\quad,?!{"C";"D"**@{-}} \\
&\quad,*\{\otimes\endxy}
\end{align*} \]

The warning message, that no intersection may exist, always occurs inside a grouping so it can safely switch itself off for deeper levels.

When the line meets the convex hull of the current control points, then a closer point exists inside the current portion of the curve. We must “decast” to find it, after storing the closest value known so far, within \prevhowclose@.

\[ \begin{align*}
\text{\xydef@\nointerceptwarning@{\let\nointerceptwarning@=\relax} \\
&\xywarning@{perhaps no curve intersection, or many.} \}} \\
\text{\xydef@\intercepthull@{\nointerceptwarning@} \\
&\expandafter\def\expandafter\prevnext@\expandafter{\next} \\
&\splinetestcvxhull@} \\
\end{align*} \]

\[ \begin{align*}
\text{\xydef@\splineoutsidehull@{\nointerceptwarning@} \\
&\expandafter\def\expandafter\prevnext@\expandafter{\next} \\
&\splinetestcvxhull@} \\
\end{align*} \]

\[ \begin{align*}
\text{\xydef@\splinelinsidehull@{\nointerceptwarning@} \\
&\expandafter\def\expandafter\prevnext@\expandafter{\next} \\
&\splinetestcvxhull@} \\
\end{align*} \]

\[ \begin{align*}
\text{\xydef@\splinelinsidehull@{\nointerceptwarning@} \\
&\expandafter\def\expandafter\prevnext@\expandafter{\next} \\
&\splinetestcvxhull@} \\
\end{align*} \]
If the line does not cross into the convex hull, then we use the distance from the control points to determine whether or not there can be a closer point. The value of $\texttt{prevhowcvxclose}$ indicates the best so far, while $\texttt{howcvxclose}$ temporarily holds the distance to the closest control point. When this is nearer then there will be a closer point on the curve.

For cubic segments it is necessary to look at both control-points off the curve and compare their distances; hence the use of $\texttt{prevhowcvxclose}$, initially set to $\texttt{maxdimen}$.
CHAPTER 2. EXTENSIONS

This macro \splineHadvance@ is used at the end of a \splinedecast@ in order to pass information from out of a subsegment, to help decide whether there is any point in further searching.

Miscellaneous features The command \clearcurve was originally provided to discard the curve information, and return to the graphics state before the curve was read, if this is ever found to be necessary. This will probably be removed.

To Do: Define an \extendcurve as a ⟨postcurve⟩ which allows the curve to be continued smoothly past the endpoint c.

A macro can be used within the modifier, to provide different labels at each control point. For example...
2.1. CURVE AND SPLINE EXTENSION

Two control sequences are provided to access the control points. Firstly \texttt{\texttt{numcontrolpts}} returns the number of them, while \texttt{\texttt{xycontrolpt(num)}} sets \texttt{c} to be the particular control point. Here \texttt{(num)} must be a single digit, grouped integer \texttt{e.g.} \{-15\} or count register containing a numerical value. If larger than \texttt{\texttt{numcontrolpts}} then \texttt{c} becomes the end-point of the curve, while if zero it becomes the starting point. A negative value sets both \texttt{p} and \texttt{c} to be the starting point, whereas other values leave \texttt{p} unchanged.

Storing control point locations:  Upon starting to read the control points for a curve, first set the counter \texttt{\texttt{crv@cnt@}} to be zero. Store the position of \texttt{p} and \texttt{c} and the current base in control sequences \texttt{\texttt{\texttt{cv@start}}}, \texttt{\texttt{\texttt{cv@end}}} and \texttt{\texttt{\texttt{cv@restore}}} for later retrieval and to reinstate the current graphics state. \texttt{\texttt{\texttt{cv@start}}} is a little more complicated in order to retain the existing value of \texttt{p}.
The position of each control point is stored as the expansion of a control sequence whose name encodes its place within the \langle poslist \rangle. Expanding this control sequence sets \textit{c} to be the appropriate position.

Alternate curve styles: There is the potential to do many other things here, by using \texttt{\afterCURVE}, once the list of control points has been read.
were added to change styles.

\def\controlpts{\afterCURVE{\xyctrlpts}\
\readxycurve}
\def\convexhull{\afterCURVE{\xycvxhull}\
\readxycurve}

The real work is done by \xyctrlpts and \xycvxhull. At the point when these macros are expanded the graphics state has the restored $p$ and $c$ to their original values, i.e. the end-points of the curve.

\controlpts: This macro visits in order each of the control points dropping the requisite object. The connection style is not used and the direction does not change. For a loop-counter we use \count@@@.

\begin{verbatim}
\xydef@\xyctrlpts@{\cv@end \cv@restore \def\crvconnect@{\straitconnect@}\ifnum\crv@cnt@>\z@ \count@@@=\@ne \DN@{\loop@\xycontrolpt@{\count@@@}\relax \expandafter\xycvxhulldrop@ \xycrvdrop@ \ifnum\crv@cnt@>\count@@@ \advance\count@@@\@ne \repeat@ \else\DN@{\relax}\fi \next@ } \xylet@\xyctrlpts=\xyctrlpts@}
\end{verbatim}

The token list \crvpts@ stores the following information: current scope, location of $p$ and $c$, number of control points \crv@cnt@ and their locations. It does this by storing \def s for control sequence names, depending on the scope, which expand to this information. Thus by executing \the\crvpts@ at the right level, the information is conveniently recovered via control sequences which do not conflict with anything else, and whose names can be reconstructed.

\Convex Hull: This macro visits in order each of the control points dropping the requisite object. With the previous as $p$ and the current one as $c$, a connection is set in the requisite style. For a loop-counter we use \count@@@.

Use a group \{\expandafter\POS\xycrvconn@\} else the \loop@ gets upset. This does not affect the size of the Xy-pic picture, since the dropped objects are not similarly shrouded.
Also used for the trivial case where there are no control points, hence a straight line between \( p \) and \( c \) is what is requested.

\[
\text{xycvxhull@}{\cv@end \cv@restore}
\addtocrypts@\{\def\cvrconnect@{\straitconnect@}\%
\def\cvrconnect@{\straitconnect@}\%
\ifnum\cvr@cnt@>\z@
\count@@@=\@ne\relax \DN@{\loop@\xycontrolpt@{\count@@@}\relax
\expandafter\xycvxhulldrop@\xycrvdrop@
{\expandafter\connect\xycrvconn@{}}\pfromc@
\ifnum\cvr@cnt@>\count@@@elax \advance\count@@@\@ne\repeat@
\cv@end }%
\else
\ifx\empty\xycrvconn@
edef\xycrvconn@{\noexpand!C\noexpand\dir{\addDASH@{}}}\fi
\DN@{\relax}\fi \next@
{\expandafter\connect\xycrvconn@{}}%
\cv@end \cv@restore }
\xylet@\xycvxhull=\xycvxhull@
\xydef@\xycvxhulldrop@#1#{\drop@{#1}}
\xydef@\savespline@{\splinetrace@{savespline@:}%
\edef\endspline@{\endgroup
\savesplineparams@ \savesplinerefs@
\X@min=\the\X@min\relax \X@max=\the\X@max\relax
\Y@min=\the\Y@min\relax \Y@max=\the\Y@max\relax
\ifInvisible@\noexpand\Invisible@true
\else\noexpand\Invisible@false\fi
}endspline@
\edef\tmp@{\noexpand\addtocrypts@{\savesplineparams@}}\tmp@
\xycontrolpt@{\z@ \cv@end }
\xydef@\savesplineparams@{\%
\noexpand\def\expandafter\noexpand\xysplineedges@{\xysplineedges@}}
\xydef@\savesplinerefs@{\%
\noexpand\def\expandafter\noexpand\xysplineparams@{\xysplineparams@}}
\xydef@\savesplineedges@{\%
\noexpand\def\expandafter\noexpand\xysplineparams@{\xysplineparams@}}
\xydef@\savesplinerefs@{\%
\noexpand\def\expandafter\noexpand\xysplineparams@{\xysplineparams@}}
\xydef@\savesplineedges@{\%
\noexpand\def\expandafter\noexpand\xysplineparams@{\xysplineparams@}}
\xydef@\savesplinerefs@{\%
\noexpand\def\expandafter\noexpand\xysplineparams@{\xysplineparams@}\
\expandafter\noexpand\xysplineedges@{\xysplineedges@}}
}

One new box register is required, for the object used to actually set the curve, ...
2.1. CURVE AND SPLINE EXTENSION

\xynew@{box}\splinebox@
... and 4 dimen registers ...
\xynew@{dimen}\splineval@
... which is for tracking the natural spline parametrization;
\xynew@{dimen}\splinedepth@
... which is for tracking the depth of the recursive algorithms;
\xynew@{dimen}\splinetol@
... which contains the tolerance for the spline, (this can probably be shifted to an ordinary macro);
\xynew@{dimen}\splinelength@
... which accumulates a measure of the length along a spline curve. (Currently this is not being
used, as most things can be done satisfactorily using the spline parametrization only.)

Some new conditionals are also required. These are for testing whether successive places on the
square are sufficiently close together, and whether the objects should be placed or not.
\xynew@{if}\ifsplinefar@
\xynew@{if}\ifsplineplot@ % to plot points or not
Provide some tracing ability, for debugging.
\xydef@\splinetracing{\let\splinetrace@=\W@}
\xydef@\splineignore@#1{}
\xylet@\splinetrace@=\splineignore@

The parameters for the current section of the spline are stored indirectly in \xysplineparams@,
while information concerning the places where the spline crosses the edges of objects at p and c is
stored indirectly in \xysplineedges. A control sequence \readsplineparams@ is used to extract this
information. Each spline type must provide a \getsplineparams@ which puts the information into
easily usable form.
\xydef@\readsplineparams@{%
\expandafter\expandafter\expandafter\getsplineparams@\xysplineparams@
\expandafter\expandafter\expandafter\getsplineedges@safe\xysplineedges@ @@}
\xydef@\getsplineedges@safe#1@@{
\DN@{#1}\
\ifx\next@\empty \DN@{\getsplineedges@safe@#1;#2,0.0pt;0.0pt,}\
\else \DN@{\getsplineedges@safe@#1;#2,#3};#4,}\
\splinetrace@{\getsplineedges@safe@#1;#2,0.0pt;0.0pt,}\
\ifx\next@\empty \DN@{\getsplineedges@safe@#1;#2,0.0pt;0.0pt,}\
\else \DN@{\getsplineedges@safe@#1;#2,3000000000100@}\
\splinetrace@{\getsplineedges@safe@#1;#2,3000000000100@}\
\ifx\next@\empty \DN@{\getsplineedges@safe@#1;#2,0.0pt;0.0pt,}\
\else \DN@{\getsplineedges@safe@#1;#2,3000000000100@}\

The edge locations are stored as x,y where x + 1 = spline-segment number and y or 1 - y =
parameter value of the start, resp. finish, in the segment x + 1.
\xydef@\getsplineedges@safe@#1;#2,3000000000100@{\DN@{#1;#2,3000000000100@}}
\global@dimen0=#1\global@dimen5=#2\relax
\dimen@=#3\multiply@\dimen@\m@ne \advance@\dimen@\splinelength@
\global@dimen3=\dimen@\relax
\ifdim@dimen0>\p@ \xywarning@{edge out-of-range: \the@dimen0}@%
Sometimes \TeX\ will run short of memory when many curves are used without a backend with special support for curves. In that case the following commands, that obey normal \TeX\ groupings, may be helpful:

\begin{verbatim}
\SloppyCurves
\splinetolerance{(dimen)}
\end{verbatim}

allow adjustment of the tolerance used to typeset curves. The first sets tolerance to .8pt, after which \texttt{\splinetolerance{0pt}} resets to the original default of fine curves.

Some useful methods for accurate division by integers.

\begin{verbatim}
\xydef@\dimen@half{\
\advance\dimen@ \ifdim\dimen@<\z@-\fi 1sp \divide\dimen@tw@}
\xydef@halve@dimen#1{\dimen@=#1\relax \dimen@half #1=\dimen@}
\xydef@\dimen@nth#1#2{\dimen@=#2\relax \dimen@ii=#1sp \\advance\dimen@ \ifdim\dimen@<\z@-\fi.5\dimen@ii \divide\dimen@ by#1\relax #2=\dimen@}
\end{verbatim}

Before constructing the curve we first examine the “drop” object, setting it in an \texttt{\xybox}. The size of this object determines the spacing of objects placed along the curve, via the \textit{tolerance} of the spline. This determines the maximum separation of places at which objects will be dropped; \textit{i.e.} when two adjacent places would be separated by more than this amount, the spline algorithm constructs another place on the curve intermediate between these two.

\textbf{Note:} Alter the spline tolerance by adjusting the size of the “drop” object. In particular, a dotted line can be achieved by setting an empty object with non-zero size. If the size is zero then the default tolerance is used. Initially this is .4pt; it may be altered using \texttt{\splinetolerance(dimen)}, where (dimen) must be greater than zero, else the initial tolerance is reset.

\begin{verbatim}
\xydef@\splinetolerance@#1{\dimen@=#1\relax \ifdim\dimen@>\z@ \splinetol@=\dimen@ \else\splinetol@=.4\p@\fi}
\xylet@\splinedefaulttol@=relax
\xydef@\splinetolerance{\splinetolerance@1{\dimen@=\#1\relax \ifdim\dimen@>\z@ \splinetol@=\dimen@ \else\splinetol@=.4\p@\fi}}
\xydef@\SloppyCurves{\splinetolerance{.8\p@}}
\end{verbatim}

Similarly examine the “connect” object. If none has been specified then there is no need to calculate the tangent direction at each place along the curve. This saves on both time and memory requirement.

\textit{Procedure:} (1) If \texttt{\xycurvdrop@} is empty, define \texttt{\splinedrop@} to expand to \texttt{\copy\zerodotbox@}, thus creating a curve constructed from small dots. Otherwise (2a) set the \texttt{\splinebox@} with the speci-
2.1. CURVE AND SPLINE EXTENSION

ified object and (2b) use $\copy\text{splinebox}$ as the expansion of $\text{splinedrop}$. (3) Set $\text{splinetol}$ to be $\sqrt{2} \times$ maximum of width and height+depth of the box.

If $\text{xcrvconn}$ is empty (4) then $\text{splineconn}$ is set to $\text{relax}$. (5) Otherwise it is necessary to reset the direction at each place along the curve before dropping the object specified by $\text{xcrvconn}$.

This stops objects being placed too close together along the spline. It is necessary at the beginning of a segment with a non-zero edge. Otherwise, with a large $\text{splinetolerance}$, the accuracy of the place-finding mechanism results in being so deep in the tree that objects placed at every level on the way up would be much too close.

Each place along the curve is tested according to a $\text{splinetest}$. Normally this is for visibility according to the extent of the objects at the end-points of the curve. When an edge is crossed then
the \splinecontinue routine is called to decide how the behaviour may change e.g. whether the recursion should continue or exit to the top of the existing tree.

\xydef@\splinepoint@@{% 
\splinetrace@{\the\X@c,\the\Y@c}:~\the\splineval@;~\the\splinedepth@}% 
{\splineptest@{\ifsplineplot@\relax\else\aftergroup\splinecontinue@\fi}}
\xydef@\splinepoint@{% 
{\splineptest@{\ifsplineplot@\aftergroup\splinecontinue@\fi}}
\xylet@\splinepoint=\splinepoint@
\xydef@\splinecontinue@{% 
\global\let\splinepoint=\splinepoint@@
\global\let\splinecontinue=\relax
}%
\xylet@\splinecontinue=\splinecontinue@

Points where segments join lie at the top of the recursive tree. They are tested directly against the objects at the endpoints, in order to decide whether or not they are visible.

\xydef@\splineknotpoint@{% \X@p=\X@c \Y@p=\Y@c \cv@start 
\the\Edge@c\@ne \ifInside@\else\aftergroup\splineknotpoint@@\fi}%
\xydef@\splineknotpoint@@{% \X@p=\X@c \Y@p=\Y@c \cv@end 
\the\Edge@c\@ne \ifInside@\else \aftergroup\splinepoint\fi}%
\xylet@\firstsplinepoint@=\splineknotpoint@
\xylet@\lastsplinepoint@=\splineknotpoint@

This routine is used to find where the spline crosses the edges of the objects at $p$ and $c$. First (1) we set up the tests which will determine when the edge has been crossed. Then commence the scan. (2)

\xydef@\splined@@@{% 
\splinetrace@{finding end types}%
\splinetrace@{bstartPLACE@:\bstartPLACE@; bendPLACE@:\bendPLACE@}%
\DN@{\splinescan@}{\ifx\bstartPLACE@{}\thenextii@
\DN@{\bendPLACE@\relax}{\def\bendPLACE@{}\fi}
\else\ifx\bstartPLACE@{}\relax
\else\ifx\bstartPLACE@\undefined
\DN@{\splinescanbreak@}{\ifx\bendPLACE@{}\nextii@
\DN@{\bendPLACE@\relax}{\def\bendPLACE@{}\nextii@
\DN@{}}%
\else\DN@{0.0}{\ifx\bendPLACE@{}\nextii@
\DN@{}}\fi
\fi \% use the saved values
\fi\fi\fi\next@ )% }% (2)
\xydef@\splineStarttest@{% \X@p=\X@c \Y@p=\Y@c \cv@start 
\the\Edge@c\@ne \ifInside@\aftergroup\splineplot@false\else\aftergroup\splineplot@true\fi}}
\xydef@\splineEndtest@{% \X@p=\X@c \Y@p=\Y@c \cv@end 
\the\Edge@c\@ne \ifInside@\aftergroup\splineplot@false\else\aftergroup\splineplot@true\fi}}
\xydef@\splineBreakStarttest@{% \X@p=\X@c \Y@p=\Y@c \cv@brstart 
\the\Edge@c\@ne \ifInside@\aftergroup\splineplot@false\else\aftergroup\splineplot@true\fi}}
\xydef@\splineBreakStarttest@{% \X@p=\X@c \Y@p=\Y@c \cv@brstart 
\the\Edge@c\@ne \ifInside@\aftergroup\splineplot@false\else\aftergroup\splineplot@true\fi}}
2.1. CURVE AND SPLINE EXTENSION

\ifInside@aftergroup\splineplot@false\else\aftergroup\splineplot@true@fi}\
\ifInside@aftergroup\splineBreakEndtest@{{\X@c=\X@c \Y@c=\Y@c \cv@brend \the\Edge@c\one
\ifInside@aftergroup\splineplot@false\else\aftergroup\splineplot@true@fi}\
\ifInside@aftergroup\splineplot@false\else\aftergroup\splineplot@true@fi}}\%\subitem(3)

\xydef@\splineStarttest=\splineStarttest@
\xylet@\splineEndtest=\splineEndtest@
\xydef@\splineSlowStarttest{{\X@c=\X@c \Y@c=\Y@c \cv@start \the\Edge@c\one
\ifInside@aftergroup\splineplot@maybe@@\else\aftergroup\splineplot@true@fi}}\%
\xydef@\splineSlowEndtest{{\X@c=\X@c \Y@c=\Y@c \cv@end \the\Edge@c\one
\ifInside@aftergroup\splineplot@maybe@@\else\aftergroup\splineplot@true@fi}}\%
\xydef@\splineplot@maybe\{
\splineplot@false\}
\xylet@\splineplot@maybe@@=\splineplot@maybe

\xydef@\splinescan@{\bgroup%
\expandafter\def\xysplineedges@={\z@;\z@,\z@;\z@,}\
\bgroup\cv@start
\expandafter\DN@\expandafter{\the\Edge@c}\DNii@{\zeroEdge}\
\if\next@=\nextii@ \gdef\splineedges@={\z@;\z@,}\fi
\else\aftergroup\splinestartScan\fi
\egroup\relax
\bgroup\cv@end
\expandafter\DN@\expandafter{\the\Edge@c}\DNii@{\zeroEdge}\
\if\next@=\nextii@ \edef\segmentnum@{\the\count@}\
splineval@=\z@ \splineRecordEndValue@
\else\aftergroup\splineendScan\fi
\egroup\egroup
\expandafter\testsplineedges@\splineedges@;,@@%
\expandafter\edef\xysplineedges@={\splineedges@}%
\global\let\splineedges@=\relax% (8)
\splinetrace@{edge params: \xysplineedges@})%

\xydef@\testsplineedges@#1;#2,#3;#4,#5@@{\
\DN@{#5}\if\next@=\empty\DNii@{#4}\if\nextii@=\empty
\xynoedgespline@{end}\def\splineedges@={\z@;\z@,\z@;\z@,}\fi
\DNii@{#2}\if\nextii@=\empty
\xynoedgespline@{start}\def\splineedges@={\z@;\z@,\z@;\z@,}\fi
\fi }
\xydef@\xynoedgespline@#1{\xywarning@{#1 edge of spline not found})%}
\xydef@\splinestartScan@{\bgroup % (3)
splinetrace@{scan for start}%
\global\let\splineadvance@=\splineadvance@@% (3)
\let\spline@start=\splinefindStart% (3)
\let\spline@end=\spline@end@@% (3)
\let\splinefinish@=\splinefinish@@
\let\splinewhich=\relax
\global\splinelength@=\z@
% \let\splinerec@=\splineSrec@
def\splinerec@{\let\splinerec@=\splineSrec@ \splinedecast@@}
\edef\splineedges@{}%
\expandafter\let\xysplineedges@\relax
% \bgroup
\xylowtolerance@ \splined@@
\ifx\splineedges@\empty\expandafter\splineslowScan@\fi
\ifx\splineedges@\relax\expandafter\splinenostart@\fi
\egroup }% %(4)
\xydef@\splineslowScan@{%
\let\splineStarttest=\splineSlowStarttest
\let\splinescanStarttest=\relax
\def\splinerec@{\let\splinerec@=\splineSrec@ \splinedecast@@}
\splined@@ }
\xydef@\splineslowEnd@{%
\global\let\splineEndtest=\splineSlowEndtest
\let\splinescanEndtest=\relax
\def\splinerec@{\let\splinerec@=\splineErec@ \splinedecast@@}
\splined@@ }
\xydef@\splinenostart@{%
\def\segmentnum@{1}\splinelength@=\z@ \splineval@=\z@
xnoedgespline@{start}\splineRecordValue@ }
\xydef@\splineendScan@{\bgroup % %(3)<-----------
\bgroup % %(7)
\splinetrace@{scan for end}%
\global\let\splineadvance@=\splineadvance@@
\let\spline@start=\splinefindEnd@
\let\spline@end=\spline@end@@
\let\splinefinish@=\splinefinish@@
\let\splinewhich=\relax
\global\splinelength@=\z@
% \let\splinerec@=\splineErec@
\global\let\splineEndtest=\splineEndtest@
def\splinerec@{\let\splinerec@=\splineErec@ \splinedecast@@}
\expandafter\def\expandafter\prevedges@\expandafter{\splineedges@}%
\xylowtolerance@ \splined@@
\ifx\splineedges@\prevedges@\expandafter\splineslowScan@\fi
\egroup \egroup}
\xydef@\splinescanbreak@{%
\splinetrace@{SCANBREAK}%
gdef\breakedges@{}%
\DN@{}\ifx\next@\bstartPLACE@
\DN@{\bgroup \let\tmp@####1;####2,####3;####4,{%
\global\splineval@=####2,####3;####4,{%%
\expandafter\expandafter\expandafter\tmp@\xysplineedges@
\expandafter\expandafter\expandafter\tmp@\xysplineedges@\relax%
\expandafter\expandafter\expandafter\tmp@\xysplineedges@\relax%

2.1. CURVE AND SPLINE EXTENSION

\begin{verbatim}
def\segmentnum@{1}\splineRecordBreakValue@ \egroup}
\else
\bgroup \cv@start
\expandafter\DN@\expandafter{\the\Edge@c}\DNii@{\zeroEdge}\
\ifx\next@\nextii@ \aftergroup\splinezerostart
\else\aftergroup\splinewidestart\fi
\egroup
\fi \next@
\DN@{1}\ifx\next@\bendPLACE@
\DN@{\bgroup \def\tmp@####1;####2,####3;####4,{%
\global\splineval@=####4\relax}%
\expandafter\expandafter\expandafter\tmp@\xysplineedges@
%% \multiply\splineval@\m@ne \advance\splineval@\p@
%% \def\segmentnum@{1}
\getcrvsegsnum@ \edef\segmentnum@{\the\count@}%
\global\multiply\splineval@\count@
\splineRecordEndBreakValue@ \egroup}
\else
\bgroup \cv@end
\expandafter\DN@\expandafter{\the\Edge@c}\DNii@{\zeroEdge}\
\ifx\next@\nextii@ \aftergroup\splinezeroend
\else\aftergroup\splinewideend\fi
\egroup
\fi \next@
\expandafter\def\expandafter
\xybreakedges@\expandafter{\breakedges@}% %(6)
\xydef@\getcrvsegsnum@{% sets \count@
\count@=\ptsnum@\relax \ifnum\count@>\@ne\@ne\advance\count@\m@ne\fi}
\xydef@\splinezerostart{\DN@{\bgroup
\splinetrace@{\splinezerostart:}%
\dimen@=\bstartPLACE@\p@
\global\splineval@=\dimen@
\splineRecordBreakValue@ \egroup }
\xydef@\splinezeroend{\DN@{\bgroup
\splinetrace@{\splinezeroend:}%
\dimen@=\bendPLACE@\p@
\multiply\dimen@\m@ne \advance\dimen@\p@
\def\segmentnum@{0}\global\splineval@=\dimen@
\splineRecordEndBreakValue@ \egroup }
\xydef@\splinewidestart{\DN@{\bgroup
\splinetrace@{\scan for start}%
\let\splinefbcontinue@=\breakstartcontinue@
\let\splinefp@=\splinefbreakpt
\let\splinef@end=\break@start
\edef\tmp@{\{\bstartPLACE@}\}}%
\end{verbatim}
This finds the first spline segment whose endpoint is beyond the edge of the object.
Finding the end is similar, but the spline is searched in reverse order.

\xydef@splinefindEnd@@\{\bgroup \bgroup
\expandafter\expandafter\expandafter\getsplineparams@\xysplineparams@
\splinetrace@\{params:\xysplineparams@\}%
\splineend@\%
\splinetrace@\{params:\xysplineparams@\}%
\global\let\splineadvance@=\splineadvance@@
\global\let\splinepoint=\splinescanEndtest
\let\splinegoal@=\splineRecordEndValue@ \global\dimen5=\z@\%
\{X@c=\X@p \Y@c=\Y@p \splinescanEndtest \}%% first test initial point.
\}
\xydef@splineend@\{\splineend@@\}
\xydef@splineend@@\{\splineend@@@
\def\postspline@{\d@X=-\d@X \d@Y=-\d@Y}%
\dimen@ii=\X@c \X@c=\X@p \X@p=\dimen@ii
\dimen@ii=\Y@c \Y@c=\Y@p \Y@p=\dimen@ii
\global\dimen5=-\dimen5 \relax
\global\advance\dimen5\splinelength@ }
\xydef@splineend@\{\splineend@@@
\xydef@splineEndtest\{\splineEndtest
\ifsplineplot@
\ifdim\splinedepth@<.0001\p@ \aftergroup\splinefinish@
\else\aftergroup\splinedecast@@ \fi
\else \aftergroup\splineadvance@ \fi
\xydef@splinefindEnd@@\%
\expandafter\crv@cnt@\xycrvptsnum@\relax
\ifnum\crv@cnt@>\tw@
\advance\crv@cnt@\m@ne
\splinetrace@false \splinesegment@\{\crv@cnt@\%
\{X@c=\X@p \Y@c=\Y@p \splinesegmenet test \}%% test end of segment.
\ifsplineplot@\aftergroup\splineplot@true\fi
\else \aftergroup\splineadvance@ \fi
\xydef@splinefindEnd@@\%
\expandafter\crv@cnt@\xycrvptsnum@\relax
\ifnum\crv@cnt@>\tw@
\advance\crv@cnt@\m@ne
\splinetrace@false \splinesegment@\{\crv@cnt@\%
\{X@c=\X@p \Y@c=\Y@p \splinesegmenet test \}%% test end of segment.
\ifsplineplot@\aftergroup\splineplot@true\fi
\else \aftergroup\splineadvance@ \fi
\xydef@splinefindEnd@@\%
\expandafter\crv@cnt@\xycrvptsnum@\relax
\ifnum\crv@cnt@>\tw@
\advance\crv@cnt@\m@ne
\splinetrace@false \splinesegment@\{\crv@cnt@\%
\{X@c=\X@p \Y@c=\Y@p \splinesegmenet test \}%% test end of segment.
\ifsplineplot@\aftergroup\splineplot@true\fi
\else \aftergroup\splineadvance@ \fi

These are the tests, to determine when the edge-point has been found.

\texttt{xdef@splinescanStarttest{%}
\texttt{splinetrace@{SST (\the\X@c,\the\Y@c); \the\splineval@ \the\splineendtest \ifsplineplot@expandafter\splinefinish@\fi}}

\texttt{xdef@splinescanEndtest{%}
\texttt{splinetrace@{SET (\the\X@c,\the\Y@c); \the\splineval@ \the\splineendtest \ifsplineplot@expandafter\splinefinish@\fi}}

\texttt{xdef@splineBrec@{%}
\texttt{\dimen@=\splineval@ \advance\dimen@\splinedepth@ \splinetrace@{BR (\the\Xp@,\the\Yp@); (\the\X@c,\the\Y@c); \the\dimen@,\the\splinedepth@} \ifsplineplot@\aftergroup\splineadvance@ \else \ifdim\splinedepth@<.0001\p@ \aftergroup\splinefinish@ \else\aftergroup\splinedecast@@ \fi \fi}}

\texttt{xdef@splineBSrec@{%}
\texttt{splinetrace@{SR (\the\Xp@,\the\Yp@); (\the\X@c,\the\Y@c); \the\dimen@,\the\splinedepth@:SR} \splineStarttest \ifsplineplot@ \ifdim\splinedepth@<.0001\p@ \aftergroup\splinefinish@ \else\global\advance\splineval@ .5\splinedepth@ \splinetest@ \advance\splineval@-\splinedepth@ \ifsplineplot@\aftergroup\splineadvance@ \else\aftergroup\splinedecast@@ \fi \fi \else \global\advance\splineval@\splinedepth@ \aftergroup\splinefinish@ \fi}}

\texttt{xdef@splineBErec@{%}
\texttt{\dimen@=\splineval@ \advance\dimen@\splinedepth@ \splinetrace@{BE (\the\Xp@,\the\Yp@); (\the\X@c,\the\Y@c); \the\dimen@,\the\splinedepth@} \splineEndtest \ifsplineplot@ \ifdim\splinedepth@<.0001\p@ \aftergroup\splinefinish@ \else\global\advance\splineval@ .5\splinedepth@ \splinetest@ \advance\splineval@-\splinedepth@ \global\advance\splineval@\splinedepth@ \aftergroup\splinefinish@ \fi \else \global\advance\splineval@\splinedepth@ \aftergroup\splinefinish@ \fi}}
2.1. CURVE AND SPLINE EXTENSION

\ifsplineplot\aftergroup\splineadvance\fi
\else\aftergroup\splinedecast\fi
\fi
\else
\ifdim\splinedepth<\p\global\advance\splineval\splinedepth \fi
\aftergroup\splinefinish\fi}}}
\xydef@splinef@breakpt#1{% gobbles |\splinecancel|
\ifdim\splineval>\z\splinetrace{found: val=\the\splineval; (\the\X@c,\the\Y@c)}\%
\else
\splinetrace{found: val=\the\splineval; (\the\X@p,\the\Y@p)}\%
\fi \splinefbcontinue\}
\xydef@splinef@pt@@{%
\ifdim\splineval>\z\splinetrace{found: val=\the\splineval; c:(\the\X@c,\the\Y@c)}\%
\global\dimen@i=\X@c \global\dimen3=\Y@c
\else\splinetrace{found: val=\the\splineval; p:(\the\X@p,\the\Y@p)}\%
\fi
\setsplinedir\global\dimen5=d@X \global\dimen7=d@Y\%
\global\splineval=\splineval\aftergroup\splinefocus\}
\xydef@findbreakwarning#1{%
\xywarning{#1 of break not found: bSTART=\bstartPLACE\, bEND=\bendPLACE}\%
\DN#1\def\DNii{start}\ifx\next@\nextii@ \splineRecordBreakValue\else \splineRecordEndBreakValue\fi\egroup\%
\xydef@breakstartcontinue{%bgroup \cv@start
%\xystatus{SS}%%%%
\expandafter\DN\expandafter{\the\Edge@c}\DNii{\zeroEdge}\%
\ifx\next@\nextii@ \aftergroup\breakstartfound\else \aftergroup\breakstartcontinuei\fi \egroup\%
\xydef@breakstartcontinuei{%
\splinetrace{move to start edge, from (\the\X@c,\the\Y@c): val=\the\splineval}\%
%\global\let\splinetest=\splineStarttest
%\global\let\splinerec=\splineSrec\%
%\global\let\splinegoal=\splineRecordBreakValue\%
%\global\let\splinefinish=\splinefinish@@
%\global\let\splinepoint=\relax
%\global\let\splineend=\checkfoundSbreak%
%\xydef@breakstartfound{%
\splinetrace{found start edge, at (\the\X@c,\the\Y@c): val=\the\splineval}\%
%\global\let\splinetest=\splineStarttest
%\global\let\splinerec=\splineSrec\%
%\gdef\splinerec@{\global\let\splinerec@=\splineSrec \splinedecast}\%
%\global\let\splinepoint=\relax
%\global\let\splineend=\checkfoundSbreak%
%\global\let\splinegoal=\splineRecordBreakValue\%
%\global\let\splinefinish=\splinefinish@@
%\global\let\splinepoint=\relax
\xydef@breakstartfound%
CHAPTER 2. EXTENSIONS

\global\let\splinepoint=\relax
\global\let\splineend=\relax %\checkfoundSbreak@
\global\let\splinegoal=\splineRecordBreakValue@
\global\let\splinefinish=\splinefinish@@
}\}
\xydef@\checkfoundSbreak@{%
  \ifnum\xycrvptsnum<\thr@@\DN@{\findbreakwarning@{start}}%
  \else\DN@{\searchBreakSsegment@}\fi \next@ %
\xydef@\searchBreakSsegment@{%
  \crv@cnt=\segmentnum\ \DN@{}\count=\crv@cnt\n
  \DNii@{}\ifnum\splineval=\z@ \ifnum\crv@cnt=\@ne
    \ifnum\xycrvptsnum=\@ne\count=\z@\DNii@{\egroup}\fi
  \else \DN@{\egroup}\fi
  \else \ifnum\splineval=\m@ne
    \else \advance\crv@cnt=\m@ne \fi \fi
  \ifnum\xycrvptsnum=\relax
    \ifnum\xycrvptsnum=\tw@ \DNii@{\egroup}\fi
  \DN@{\expandafter\splineRecordBreakValue@\nextii@}%
  \else
  \DN@{\splinesegment@{\crv@cnt}\splineSetparams@}
  \splined@@ \egroup\%}
  \fi \next@ }
\xydef@\checkfoundEbreak@{%
  \ifnum\xycrvptsnum<\thr@@\DN@{\findbreakwarning@{end}}%
  \else\DN@{\searchBreakEsegment@}\fi \next@ %
\xydef@\searchBreakEsegment@{%
  \crv@cnt=\segmentnum\ \DN@{}\count=\crv@cnt\n
  \DNii@{}\ifnum\splineval=\z@ \ifnum\crv@cnt=\@ne
    \ifnum\xycrvptsnum=\@ne\count=\z@\DNii@{\egroup}\fi
  \else \DN@{\egroup}\fi
  \else \advance\crv@cnt=\m@ne \fi \fi
  \ifnum\count=\z@
    \ifnum\xycrvptsnum=\tw@ \splineval=\m@ne\p@\DNii@{\egroup}\fi
  \DN@{\expandafter\splineRecordEndBreakValue@\nextii@}\else
  \DN@{\splinesegment@{\crv@cnt}\splinereverse@}
  \read splineparams@ \splined@@ \egroup\%}
  \fi \next@ }
\xydef@\break@start{\egroup \egroup
\splinetrace@{scan for start}\
\global\let\splineadvance=\splineadvance@@ (3)
\let\spline@start=\splinefindBStart@ (3)
\let\spline@end=\egroup
\let\splinefinish=\splinefinish@@
\let\splinewhich=\relax
\global\let\splineadvance=\splineadvance@@
\global\let\splinepoint=\relax
2.1. CURVE AND SPLINE EXTENSION

\let\splinegoal@=\splineRecordBreakValue@
\global\dimen5=\z@ \global\splineLength@=\z@
\let\splineRec@=\splineSrec@
\def\breakedges@{}%
\bgroup \bgroup \splined@@

\xydef@\splineFindBStart@{\bgroup
% \global\splineVal@=\z@ \global\splineDepth@=.5\p@}
\global\splineVal@=\z@ \splineDepth@=.5\p@}
\xystatus@{EE}
\ifx\next@\nextii@ \aftergroup\breakendfound@ \else \aftergroup\breakendcontinue@i \fi \egroup

\xydef@\breakendcontinue@i{%
\splinetrace@{move to end edge, from (\the\X@c,\the\Y@c): val=\the\splineVal@ }%
% \global\let\splineTest@=\splineEndtest
\global\let\splineEndtest=\splineBreakEndtest@
% \global\let\splinepoint=\relax
\global\let\splineEnd=\checkfoundEbreak@
% \global\let\splineGoal@=\splineRecordEndBreakValue@
\global\let\splineGoal@=\splineRecordEBreakValue@
\global\let\splineFinish@=\splineFinish@@
\splineRec@ }%
\xydef@\splineRecordEBreakValue@{%\getcrvsegsnum@
\splinetrace@{\the\count@ space segments, \the\splineVal@}%
% \ifnum\segmentnum@>\z@ \dimen@=\segmentnum@\p@ \else \dimen@=\p@\fi
\ifnum\count@>\@ne \divide\dimen@\count@\fi
% \global\splineVal@=\dimen@
% \splineRecordEndBreakValue@ }
\xydef@\splineRecordSBreakValue@{%\getcrvsegsnum@
\splineTrace@{\the\count@ space segments, \the\splineVal@}%
\ifnum\segmentnum@>\z@ \dimen@=\segmentnum@\p@ \else \dimen@=\p@\fi
% \ifnum\count@>\@ne \divide\dimen@\count@\fi
% \global\splineVal@=\dimen@
% \splineRecordBreakValue@ }
\xydef@\splineRecordBreakValue@{%\getcrvsegsnum@
\splineTrace@{\the\count@ space segments, \the\splineVal@}%
\ifnum\segmentnum@>\z@ \dimen@=\segmentnum@\p@ \else \dimen@=\p@\fi
% \ifnum\count@>\@ne \divide\dimen@\count@\fi
% \global\splineVal@=\dimen@
% \splineRecordBreakValue@ }
\xydef@\breakendfound@{%
\splinetrace@{found end edge, at (\the\X@c,\the\Y@c): val=\the\splineVal@ }%
% \global\let\splineTest@=\splineEndtest
\global\let\splineEndtest=\splineBreakEndtest@
% \global\let\splinepoint=\relax
% \checkfoundEbreak@
% \global\let\splineGoal@=\splineRecordEndBreakValue@
\global\let\splineGoal@=\splineRecordEBreakValue@
\global\let\splineFinish@=\splineFinish@@
}
2932 \xydef\breakend\egroup \egroup
2933 \global\let\splineadvance@=\splineadvance@@
2934 \let\spline@end=\splinefindBEnd@
2935 \let\splinefinish@=\splinefinish@@
2936 \let\splinewhich=\relax
2937 \global\let\splineadvance@=\splineadvance@@
2938 \global\let\splinegoal@=\splineRecordEndBreakValue@
2939 \global\spline@length=\z@
2940 \let\splineerec@=\splineErec@
2941 \global\let\splineEndtest=\splineEndtest@
2942 \bgroup \splined@@ }
2945 \xydef\splinefindBEnd@{\bgroup\bgroup
2946 \splinereverse@ \readsplineparams@
2947 \global\splineval=\z@
2948 This appends the new information to that currently stored in the control sequence referenced by \xysplineparams@. It is temporarily stored globally in \splineparams@, to be later transferred to (the c.s. referenced by) \xysplineparams@ when at the appropriate level of grouping.
2957 \xydef\splineRecordValue@{%
2958 \i\f x\unknown\segmentnum@\relax
2959 \else
2960 \getcrvsegsnum@
2961 \dimen=\splineval@ \advance\dimen@ \segmentnum@\p@
2962 \advance\dimen@-\p@ \dimen@nth\count@\dimen@
2963 \global\splineval@=\dimen@
2964 \fi
2965 \xdef\splineedges@{\splineedges@\the\spline@length@;\the\splineval@,}%
2966 \splinetrace@{found edge: \splineedges@}
2968 \xydef\splineRecordEndValue@{%
2969 \i\f x\unknown\segmentnum@\relax
2970 \else
2971 \getcrvsegsnum@
2972 \dimen=\splineval@ \advance\dimen@ \count@\p@
2973 \advance\dimen@-\segmentnum@\p@ \dimen@nth\count@\dimen@
2974 \global\splineval@=\dimen@
2975 \fi
2976 \xdef\splineedges@{\splineedges@\the\spline@length@;\the\splineval@,}%
2977 \splinetrace@{found edge: \splineedges@}
2979 \xydef\splineRecordBreakValue@{%
2980 \i\f x\unknown\segmentnum@\relax
2981 \else\expandafter\advance\expandafter\splineval@\segmentnum@\p@
2982 \count@=\segmentnum@\relax
2983 \ifnum\count@>0 \advance\splineval@-\p@ \fi\fi
2984 \if\breakedges@\relax\relax\else\i\f x\breakedges@\empty\else
2985 \i\f dim\splineval@<\p@\i\f dim\splineval@\z@
2986 \expandafter\testbreakedges@\breakedges@\empty
2987 \i\f \fi\fi\fi
2988 \xdef\breakedges@{\breakedges@\the\spline@length@;\the\splineval@,}%
2.1. CURVE AND SPLINE EXTENSION

Use the following distance approximation:

\[
\text{dist} = \begin{cases} 
\frac{dX + .5dY}{dX} / dY & \text{if } dY/dX dY < (\sqrt{2} - 1) \times dX \\
\frac{3/4dX + \sqrt{2}/4dY}{dX} & \text{if } (\sqrt{2} - 1) \times dX < dY/dX dY < dX 
\end{cases}
\]

and similarly, interchanging \(dY\) and \(dX\) when \(dY > dX\).

\[ \sqrt{2} \approx 1.41422, \ (\sqrt{2})/2 \approx 0.70711, \ (\sqrt{2})/4 \approx 0.35355, \ 3(\sqrt{2})/4 \approx 1.060665 \]
The macro `\spline@find#1#2` attempts to finds a specific point on a single spline segment. \#1 is the parameter value, in the range \([0,1]\) on that segment; \#2 is a \langle\text{dimen}\rangle\ denoting how much further to slide along the segment. It really only checks whether \#2 has a negative value before passing the information to `\spline@@find#1#2`, which does the actual search.

If \#2 is negative then the spline segment is searched in the reverse direction, starting at its endpoint. This is done by reversing the order of the control points, hence the parameter value found as \(x\) implies that we really want \(1 - x\) on the un-reversed segment. Similarly the resulting values for \d@X and \d@Y must be negated. The instructions to do this are loaded into a macro `\postfind@` which is expanded once the search has been completed. For an unreversed segment `\postfind@` expands to `{}`.
2.1. CURVE AND SPLINE EXTENSION

\ifdim\dimen@ii=\z@ \def\postfind@{}\% 
\else\def\postfind@{}\%
\ifdim\dimen@ii<\z@ \splineforward@ \else \splinebackward@
\read splineparams@ \splineforward@ \reverse orientation: \{(the\X@p, the\Y@p), (the\X@c, the\Y@c)\} \% 
\dimen@=#1\relax \dimen@ii=#2\relax 
multiply \dimen@\m@ne \advance \dimen@\p@ \multiply \dimen@ii\m@ne 
\def\postfind@{\d@X=-the\d@X \d@Y=-the\d@Y \noexpand\reversesplineval@} \%
\fi \fi 
\expandafter\splineforward@ \expandafter{\xysplineparams@} \%
\edef\next@{\noexpand\spline@@find{\the\dimen@}{\the\dimen@ii}{#3}} \%
\next@ }

\xydef@\reversesplineval@{\splineval@=-\splineval@ \advance\splineval@\p@ } \%
\xydef@\spline@@find#1#2#3{\%
\splineforward@ \spline@@find #1 #2 #3\%
\let\splineforward@=\splinefmap@ \global\let\splineadvance@=\splineadvance@@ \dimen5=#1\relax 
\ifdim #2=\z@\relax 
\let\spline@@knot=\spline@@knot \global\splineval@=\z@ \else\ifdim #1=\p@ \relax 
\let\spline@@knot=\spline@@knot \global\splineval@=\z@ \else 
\global\splineval@=\z@ \global\let\splineadvance@=\splineadvance@@ \DN@{#3}\ifx\next@\empty \setsplinetest@\splineval@>{#1}{}{} \%
{\ifdim\splineval@=\dimen5\aftergroup\splineplot@false\fi} \%
\else\DNii@{>}\ifx\next@\nextii@ \setsplineundertest@\splineval@>{#1}{}{} \%
\let\splineforward@=\splineBSmap@ \else \DNii@{<}\ifx\next@\nextii@ \setsplineundertest@\splineval@>{#1}{}{} \%
\let\splineforward@=\splineBmap@ \else \xywarning@{unknown find-mode: #3} \fi \fi \fi \fi 
\else 
\global\splineval@=\z@ \global\let\splineforward@=\spline@@knot \DN@{#3}\ifx\next@\empty \setsplinetest@\splineval@>{#1}{}{} \%
{\ifdim\splineval@=\dimen5\aftergroup\splineplot@false\fi} \%
\else\DNii@{>}\ifx\next@\nextii@ \setsplineundertest@\splineval@>{#1}{}{} \%
\let\splineforward@=\splineBSmap@ \else \DNii@{<}\ifx\next@\nextii@ \setsplineundertest@\splineval@>{#1}{}{} \%
\let\splineforward@=\splineBmap@ \else \xywarning@{unknown find-mode: #3} \fi \fi \fi \fi 
\else 
\global\splineval@=\z@ \def\splineslidetest@@{\%
\setsplinetest@\splinelength@<{#2}{}{}} \%
\let\splinefinish@=\splinefcontinue@ \%
\fi 
\else 
\global\splineval@=\z@ \def\splineslidetest@@{\%
\setsplinetest@\splinelength@<{#2}{}{}} \%
\let\splinefinish@=\splinefcontinue@ 
\fi 
\def\spline@start{\bgroup\xylowtolerance@} \%
\splined@ 
\xydef@\spline@@knot{% 
\let\setsplinedir@=\setsplineknotdir@ 
\splinesetparams@\spline@start 
% \global\splinedepth@=\p@ 
\splinedepth@=\p@ 
% \global\dimen0@=\X@p \global\dimen3=\Y@p 
\splinefinishf@ \spline@end \egroup } 

\xydef@\splinefcontinue@{% 
\splinetrace@{sliding... from (\the\X@c,\the\Y@c): val=\the\splineval@ }% 
\global\let\splinerec@=\splinerec@@ 
\global\splinelength@=\z@ 
\global\splineval@=\splineval@ 
\global\let\splinepoint=\splinefindtest@ 
\global\let\splinegoal@=\splinef@pt 
\global\let\splineadvance@=\splineDadvance@ 
\global\let\splinefinish@=\splinefinish@@ 
\splineslidetest@@ 
}

\xydef@\splinefindtest@{% 
\splinetrace@{SFT (\the\X@c,\the\Y@c); \the\splinelength@, \the\splineval@}% 
{\splineplot@false\splinetest@ 
\ifsplineplot@\expandafter\splinefinishf@\fi}}

\def\setsplineundertest@#1#2#3#4#5{% 
\splinetrace@{setsplineundertest@: #1,#2,#3,#4,#5,} 
\DNii@{\gdef\splinetest@} 
\DN@##1\next{\def\tmp@{{\splineplot@false 
\ifdim#1#2##1\relax #4\else\splineplot@true#5\fi}}} 
\next@#3\relax\next 
\expandafter\nextii@\tmp@ } 

\xydef@\splinef@end{% 
\edef\tmp@{\egroup\splinelength@=\the\splinelength@}\tmp@}% 
\xydef@\splinef@end{% 
\ifdim\splineval@>\z@ 
\splinetrace@{found: val=\the\splineval@;c:(\the\X@c,\the\Y@c)}% 
\global\dimen@i=\X@c \global\dimen3=\Y@c 
\else 
\splinetrace@{found: val=\the\splineval@;p:(\the\X@p,\the\Y@p)}% 
\fi 
\setsplinedir@ \global\dimen5=\d@X \global\dimen7=\d@Y 
\global\splineval@=\splineval@ 
\aftergroup\splinefocus@ } 

\def\setsplinetest@#1#2#3#4#5{% 
\DNii@{\gdef\splinetest@} 
\DN@##1\next{\def\tmp@{\splineplot@false 
\ifdim#1#2##1\relax #4\else\splineplot@true#5\fi}}} 
\next@#3\relax\next 
\expandafter\nextii@\tmp@ }
There are some extra hooks.

For the actual setting, provide hooks which will allow alternative back-ends to be used in the special cases.

The special cases are handled just like the normal case except However the control sequence names provide a place for rebindin to accomodate alternative back-ends.
This establishes the test appropriate to actually setting the spline curve. Global definitions are used. This may not always be necessary!!

**Bug:** the 7.5 pt below should be the \texttt{Step@@} method to be included.
2.1. CURVE AND SPLINE EXTENSION

\next@
\ifdim\dimen5=p0\DN{}\else\DN{\splined}@\fi \next@

\xydef@\emptyspline@{\xywarning@{empty curve subsegment}\\}
\splinetrace@{\bstartPLACE@=\bstartPLACE@, bendPLACE@=\bendPLACE@, \empty segment}\\
\gdef\splinecontinue{\splinecontinue@\\}
\global\let\splineendtest@=\splineplot@false
\global\dimen5=z@}

\splineTrec@ is the initial value for \splinerec@ when a spline is being set. It descends the tree of places on the spline (1) until the required parameter value is found (2a) or is sufficiently close (2b). At this point call \splinecontinue@ and \splinecontinue@ to setup, and proceed with, the actual type-setting.

\xydef@\splineTrec@{\%}
\advance\splineval@=\splinedepth@ % (1)
\ifdim\dimen5=\splineval@ % (2a)
\aftergroup\splinecontinue % (3)
\else \dimen@=\dimen5\advance\dimen@-\splineval@
\ifdim\dimen@<\z@ \dimen@=-\dimen@ \fi
\ifdim\dimen@<.001p@ % possibly too high ? % (2b)
\aftergroup\splinecontinue % (3)
\else
\ifdim\dimen5<\splineval@ \aftergroup\splinecast@@ % (1)
\else \aftergroup\splineadvance@ % (1)
\fi\fi\fi}

Setting the curve uses the current \splinerec@ and sets \splinepoint to \splineplotpt@ to cause objects to be typeset. This must be done via a \gdef rather than a \global\let since it is sometimes necessary to omit the object when it would be too close, see \splineetooclose@.

\xydef@\splinesetting@{\xyFN@\splinesetting@@}
\xydef@\splinesetting@{\splinesetting@@}
\xydef@\splinesetting@@{\splinetrace@{splinesetting@@:}\\
\ifdim\splinedepth@=\z@ \splinedepth@=\p@ \fi
\ifdim\splinedepth@=\p@
\def\splinerec@{\global\let\splinerec@=\splinerec@@\splinecast@@}\\
\else \global\let\splinerec@=\splinerec@@ \fi
\gdef\splinepoint{\splineplotpt@}

This handles, for each point of the curve, whether to place an ⟨object⟩ or horizontal glue.
These handle the “cleaning up” after a point on the curve has been located.
Bézier quadratic splines — squines

Use the registers \A@ and \B@ to store the coordinates of the single control point. The whole curve lies within the convex polygon with vertices at \( p \), \( (p + a)/2 \), \( (c + a)/2 \) and \( c \) where \( a \) denotes the control point. Set \X@max, \X@min, \Y@max and \Y@min to be the extremes of the coordinates of these 4 points. There may not actually be any point on the curve achieving these extremes, but certainly we get pretty close.
The algorithm used for computing coordinates of points on quadratic Bézier splines is essentially that used by D. E. Knuth in the picmac.tex macros\(^1\) (aka gpxmac.tex). It is a recursive de Casteljau algorithm of the “divide and conquer” type. (In Knuth’s macros these types of curves were given the name “squires”. This explains some of the control sequence names used here.)

The differences from Knuth’s algorithm are simply to allow more of the available information to be used at points along the spline. In particular the tangent direction can be calculated and tests can be performed to decide when to break out of the algorithm, rather than letting it run its full course.

The tangent direction is computed from the displacement to the “recursive” control point.

\[
\text{The algorithm used for computing coordinates of points on quadratic Bézier splines is essentially that used by D. E. Knuth in the picmac.tex macros.} \quad (In \ Knuth’s \ macros \ these \ types \ of \ curves \ were \ given \ the \ name \ “squires”. \ This \ explains \ some \ of \ the \ control \ sequence \ names \ used \ here.)
\]

\[
The \ differences \ from \ Knuth’s \ algorithm \ are \ simply \ to \ allow \ more \ of \ the \ available \ information \ to \ be \ used \ at \ points \ along \ the \ spline. \ In \ particular \ the \ tangent \ direction \ can \ be \ calculated \ and \ tests \ can \ be \ performed \ to \ decide \ when \ to \ break \ out \ of \ the \ algorithm, \ rather \ than \ letting \ it \ run \ its \ full \ course.
\]

\[
\text{The tangent direction is computed from the displacement to the “recursive” control point.}
\]

\(^1\)http://www-cs-faculty.stanford.edu/~uno/papers/picmac.tex.gz.
2.1. CURVE AND SPLINE EXTENSION

We need a way to access the information in \splineparams.

\xydef@\getsquineparams@#1,#2,#3,#4,#5,#6,#7,{\%
\splinelength@=#1\relax\X@p=#2\relax\Y@p=#3\relax
\A@=#4\relax\B@=#5\relax\X@c=#6\relax\Y@c=#7\relax
}\%
\xydef@\squinereverse@{\readsplineparams@
\edef\xysplineparams@{\the\z@,\the\X@p,\the\Y@p,\the\A@,\the\B@,\the\X@c,\the\Y@c,}}%

Start the picture by (1) setting a box, as usual. (2) Initialize global variables; these must be global since they are used to store information which must be preserved outside the grouping which is vital to the recursive nature of the algorithm. (3) Move horizontally to the starting point at p. (4) Begin by handling the starting point; i.e. decide whether it is to be plotted or not.

\xydef@\squine@start@{\setboxz@h\bgroup% (1)
\global\splinelength@=\z@ \global\dimen@i=\z@ \global\dimen3=\z@ % (2)
\kern\X@p % (3)
\{\squinesetparams@ \X@c=\X@p \Y@c=\Y@p \firstsplinepoint@ % (4)
\bgroup \}
\xydef@\squinesetparams@{% 
\global\dimen0i=\X@p \global\dimen3=\Y@p
\L@c=\A@ \U@c=\B@ \R@c=\L@c \D@c=\U@c \%
\xydef@\squined@{% 
\edef\xysplineparams@{\the\splinelength@,\%
\the\X@c,\the\Y@c,\the\A@,\the\B@,\the\X@p,\the\Y@p,}}%

% \global\splineparams@=
% \global\dimen0i=\X@p \global\dimen3=\Y@p
% \L@c=\A@ \U@c=\B@ \R@c=\L@c \D@c=\U@c \%
\xydef@\squined@{% 
\edef\xysplineparams@{\the\z@,\the\X@p,\the\Y@p,\the\A@,\the\B@,\the\X@c,\the\Y@c,}}
\xydef@\squined@{% 
\global\splineparams@=\p@ 
\splinedepth@=\p@
Bez'sier cubic splines

Use the registers \A@, \B@, \dimen3 and \dimen5 to store coordinates of the two control points, denoted l and r say. The whole curve lies within the convex polygon p, (p + l)/2, (c + r)/2 and c where the vertices are not necessarily in this order. Set \X@max, \X@min, \Y@max and \Y@min to be the extremes of the coordinates of these 4 points. There may not actually be any point on the curve achieving these extremes, but certainly we get pretty close.
The recursive algorithm for cubic Bézier splines is similar to the quadratic one. Now there are two "recursive" control points to be calculated upon each subdivision.

On the \(p\)-side:

\[
\begin{align*}
\tilde{p}_p &= \frac{1}{2} (p + l) \\
\tilde{l}_p &= \frac{1}{4} (p + 2l + r) \\
\tilde{r}_p &= \frac{1}{8} (p + 3l + 3r + c)
\end{align*}
\]

while on the \(c\)-side

\[
\begin{align*}
\tilde{p}_c &= \frac{1}{8} (p + 3l + 3r + c) \\
\tilde{l}_c &= \frac{1}{4} (l + 2r + c) \\
\tilde{r}_c &= \frac{1}{2} (r + c) \\
\tilde{c}_c &= c
\end{align*}
\]

Notice that \(\tilde{p}_p = \tilde{p}_c\) and that the tangents match there.
The tangent direction is computed from the displacement to the "recursive" control point.

The tangent direction is computed from the displacement to the “recursive” control point.
2.1. CURVE AND SPLINE EXTENSION

We need a way to access the information in \splineparams.

\xydef@\getcubicparams@#1,#2,#3,#4,#5,#6,#7,#8,#9,{
\splinelength@=#1\relax\X@p=#2\relax\Y@p=#3\relax\L@c=#4\relax
\U@c=#5\relax\R@c=#6\relax\D@c=#7\relax\X@c=#8\relax\Y@c=#9\relax
}\xydef@\cubicinfo@{
\expandafter\removePT@\the\X@p\space
\expandafter\removePT@\the\Y@p\space
}\xydef@\cubicreverse@{
\expandafter\edef\xysplineparams@{\the\splinelength@,}
\expandafter\edef\xysplineparams@{\the\X@c,\the\Y@c,\the\R@c,\the\D@c,\the\L@c,\the\U@c,\the\X@p,\the\Y@p,}}
\xydef@\cubicreverse@@{
\splinereverse@@@
\dimen@ii=\L@c \L@c=\R@c \R@c=\dimen@ii

\xydef@\setcubiccoarsedir@{%
\d@X=\X@c \advance\d@X-\X@p \d@Y=\Y@c \advance\d@Y-\Y@p
\global\dimen5=\d@X \global\dimen7=\d@Y
\splinetrace@{dir:(\the\d@X,\the\d@Y),\the\Direction; depth:\the\splinedepth@}%
}
\xydef@\cubicinfo@{
\expandafter\removePT@\the\X@p\space
\expandafter\removePT@\the\Y@p\space
\expandafter\removePT@\the\L@c\space
\expandafter\removePT@\the\U@c\space
\expandafter\removePT@\the\R@c\space
\expandafter\removePT@\the\D@c\space
\expandafter\removePT@\the\X@c\space
\expandafter\removePT@\the\Y@c\space
\expandafter\removePT@\the\dimen5\space\space
\expandafter\removePT@\the\dimen7\space
}
\xydef@\cubicreverse@{
\expandafter\edef\xysplineparams@{\the\splinelength@,}
\expandafter\edef\xysplineparams@{\the\X@c,\the\Y@c,\the\R@c,\the\D@c,\the\L@c,\the\U@c,\the\X@p,\the\Y@p,}}
\xydef@\cubicreverse@@{%
\splinereverse@@@
\dimen@ii=\L@c \L@c=\R@c \R@c=\dimen@ii

\xydef@\setcubiczerodir@{%
\d@X=\L@c \advance\d@X-\X@p \d@Y=\U@c \advance\d@Y-\Y@p
\ifdim\zz@\splineval@ \DN@{\setcubiczerodir@}\
\else \DN@{\cubicfinedir@}\fi\next@}
\expandafter\edef\xysplineparams@{\the\splinelength@,}
\expandafter\edef\xysplineparams@{\the\X@c,\the\Y@c,\the\R@c,\the\D@c,\the\L@c,\the\U@c,\the\X@p,\the\Y@p,}}
\expandafter\edef\xysplineparams@{\the\splinelength@,}
\expandafter\edef\xysplineparams@{\the\X@c,\the\Y@c,\the\R@c,\the\D@c,\the\L@c,\the\U@c,\the\X@p,\the\Y@p,}}
\xydef@\cubicinfo@{
\expandafter\removePT@\the\X@p\space
\expandafter\removePT@\the\Y@p\space
\expandafter\removePT@\the\L@c\space
\expandafter\removePT@\the\U@c\space
\expandafter\removePT@\the\R@c\space
\expandafter\removePT@\the\D@c\space
\expandafter\removePT@\the\X@c\space
\expandafter\removePT@\the\Y@c\space
}\xydef@\cubicreverse@{\readsplineparams@}
\expandafter\edef\xysplineparams@{\the\splinelength@,}
\expandafter\edef\xysplineparams@{\the\X@c,\the\Y@c,\the\R@c,\the\D@c,\the\L@c,\the\U@c,\the\X@p,\the\Y@p,}}
\xydef@\cubicreverse@@{%
\splinereverse@@@
\dimen@ii=\L@c \L@c=\R@c \R@c=\dimen@ii
Start the picture by (1) setting a box, as usual. (2) Initialize global variables; these must be global since they are used to store information which must be preserved outside the grouping which is vital to the recursive nature of the algorithm. (3) Move horizontally to the starting point at $p$. (4) Begin by handling the starting point; i.e. decide whether it is to be plotted or not.

B-splines

The cases of 3 and 4 control points have some special simplifying features.
2.1. CURVE AND SPLINE EXTENSION

\let\getsplineparams@=\getbsplineparams@
\let\spline@start@=\cubic@start@
\let\splinedecast@@=\cubicdecast@
\let\splinerec@@=\cubicrec@
\let\splineIpt@@=\cubicIpt@
\let\splineIadvance@@=\cubicIadvance@
\let\splined@@=\bsplined@@
\let\splineinfo@=\cubicinfo@
\let\setsplinedir@=\setcubicdir@
\let\setsplineknotdir@=\setcubicknotdir@
\let\splinereverse@=\cubicreverse@
\let\splineDadvance@@@=\cubicDadvance@@@
\let\splinesetparams@=\cubicsetparams@
\global\let\lastbspline@=\lastbspline@@
\global\let\middlebspline@=\middlebspline@@
}

\xydef@\bsplined@@{%
\global\splinedepth@=\p@
\splinedepth@=\p@
\global\splineval@=\z@
\spline@start
\global\dimen@i=\X@p \global\dimen3=\Y@p
\splineinfo@ \spline@end }

\xydef@\bsplineconnect@{\splineconnect@ \crvconnect@@ }
\let\splinesegment@=\bsplinesegment@ )%
\def\getbsplineparams@{\getcubicparams@}
\def\dobspline@{\xysplineparams@ \scanbspline@ \firstbspline@ }%

Registers \texttt{\dimen3} and \texttt{\dimen5} are used globally, since the standard local registers are already used.

\def\segmentnum@{1}\xycontrolpt@\z@ \X@p=\X@c \Y@p=\Y@c
\splinetrace@{0: \the\X@p, \the\Y@p}%
\xycontrolpt@\one \A@=\X@c \B@=\Y@c
\splinetrace@{1: \the\X@c, \the\Y@c}%
\xycontrolpt@\tw@
\splinetrace@{2: \the\X@c, \the\Y@c}%
\dimen@=\X@c \advance\dimen@A@ \dimen@half
\global\dimen3=\dimen@
\dimen@=\Y@c \advance\dimen@B@ \dimen@half
\global\dimen5=\dimen@
\xycontrolpt@{3}%
\splinetrace@{3: \the\X@c, \the\Y@c}%
\expandafter\count@\xycrvcnt@relax %%% very important to \relax
\ifnum\count@=3\relax
\advance\X@c-\A@ \dimen@nth4\X@c \divide\X@c by4
2.1. CURVE AND SPLINE EXTENSION

\enter@\{\X@p=\the\X@c \Y@p=\the\Y@c \crv@cnt@=\segmentnum@\relax
\noexpand\middlebspline@\%
\else
\advance\X@c-\A@ \dimen@ nth4\X@c \% divide \X@c by 4
\advance\X@c \dimen3 \relax
\advance\Y@c-\B@ \dimen@ nth4\Y@c \% divide \Y@c by 4
\advance\Y@c \dimen5 \relax
\enter@\{\X@p=\the\X@c \Y@p=\the\Y@c \noexpand\lastbspline@\%
\fi \adjustmaxmin@
\bsplined@ \splineset@ \leave@ }
\xylet@\middlebspline@=\middlebspline@@
\xydef@\scanbspline@{\splined@@@}
\xydef@\bsplined@{\cubied@}
\expandafter\ifx\xysplineedges@\relax\relax\DN@{\%}
\else\DN@{\adjustbsplineedges@}\fi \next@ }
\xydef@\adjustforsegments@{\getcrvsegnums@}
\ifnum\count@>\@one
\splinetrace@{adjust for \the\count@ space segments:}
\the\dimen5, \the\dimen7\%
\multiply\dimen5 by \count@ \relax \dimen@=\segmentnum@ \p@ 
\ifnum\dimen5>\dimen@ \relax \dimen5=\p@
\else \advance\dimen5-\segmentnum@ \p@ \advance\dimen5 by \p@ \fi
\ifnum\dimen5<\z@ \relax \dimen5=\z@ \relax \fi
\multiply\dimen7 by \count@ \relax
\dimen@=\segmentnum@ \p@ \% advance \dimen@-\segmentnum@ \p@
\ifnum\dimen7<\dimen@ \advance\dimen@-\p@
\ifnum\dimen7>\dimen@ \advance\dimen@-\dimen@
\else \dimen7=\z@ \fi
\else \dimen7=\p@ \fi
\splinetrace@{adjusted for \the\count@ space segments:}
\the\dimen5, \the\dimen7\%
\fi}
\xydef@\adjustbsplineedges@{\bgroup \% dimen5 and dimen7 are NOT global here
\splinetrace@{** adjusting edges **}\
\readsplineparams@ \adjustforsegments@
\% \count@@=\dimen7/\divide\count@@/\p@
\% \advance\dimen7-\count@@/\p@ \advance\count@@/\@one
\% \count@=\dimen5/\divide\count@/\p@
\% \advance\dimen5-\count@@/\p@ \advance\count@@/\@one
\splinetrace@{params: \xysplineparams@}\
\splinetrace@{segment \segmentnum@:<\the\dimen5,\the\dimen7>: \xysplineedges@}\
\% \expandafter\ifnum\segmentnum@=\count@
\% \else
\% \expandafter\ifnum\segmentnum@<\count@ \dimen5=\p@
\% \else \dimen5=\z@ \fi \fi \fi
\% \expandafter\ifnum\segmentnum@=\count@@
\% \multiply\dimen7 by \m@ne \advance\dimen7 by \p@
These macros select the correct Bézier control points for each segment of the spline. This is needed for finding places on the constructed curves.

In general there are 5 types of segment: first, second, middle, penultimate, final. The conversion from B-spline to Bézier is slightly different for each type. For middle segments the Bézier control points are determined in the following way:

\[
X^{(1)}_{B_i} = \frac{1}{3} \left( 2X^{(i)} + X^{(i+1)} \right)
\]
\[
X^{(2)}_{B_i} = \frac{1}{3} \left( X^{(i)} + 2X^{(i+1)} \right)
\]
\[
X^{(0)}_{B_i} = \frac{1}{6} \left( X^{(i-1)} + 4X^{(i)} + X^{(i+1)} \right)
\]
\[
= \frac{1}{6} \left( X^{(i-1)} + 7X^{(1)}_{B_i} - 2X^{(2)}_{B_i} \right)
\]
\[
X^{(3)}_{B_i} = \frac{1}{6} \left( X^{(i)} + 4X^{(i+1)} + X^{(i+2)} \right)
\]
\[
= \frac{1}{6} \left( X^{(i+2)} + 7X^{(2)}_{B_i} - 2X^{(1)}_{B_i} \right)
\]
2.1. CURVE AND SPLINE EXTENSION

\[ X_{B_2}^{(0)} = \frac{1}{4} \left( X^{(1)} + 4X_{B_2}^{(1)} - X_{B_2}^{(2)} \right) \]
\[ X_{B_{n-1}}^{(3)} = \frac{1}{4} \left( X^{(3)} + 4X_{B_{n-1}}^{(2)} - X_{B_{n-1}}^{(1)} \right) \]

For the first segment the first three Bézier control points are always determined in the same way:

\[ X_{B_1}^{(0)} = X^{(0)} \]
\[ X_{B_1}^{(1)} = X^{(1)} \]
\[ X_{B_1}^{(2)} = \frac{1}{2} \left( X_{B_1}^{(1)} + X^{(2)} \right) \]

3+ segments

\[ X_{B_1}^{(3)} = \frac{1}{6} \left( 7X_{B_1}^{(2)} - 2X_{B_1}^{(1)} + X^{(3)} \right) \]

2 segments

\[ X_{B_1}^{(3)} = \frac{1}{4} \left( 4X_{B_1}^{(2)} - X_{B_1}^{(1)} + X^{(3)} \right) \]
The last segment is determined symmetrically from the final four control points:

\[
\begin{align*}
    X_{B_n}^{(3)} &= X^{(n+2)}  & X_{B_n}^{(2)} &= X^{(n+1)}  & X_{B_n}^{(1)} &= \frac{1}{2} (X_{B_n}^{(2)} + X^{(n)}) \\
    3+ \text{ segments} \quad X_{B_n}^{(0)} &= \frac{1}{6} (X^{(n-1)} - 2X_{B_n}^{(2)} + 7X_{B_n}^{(1)}) \\
    2 \text{ segments} \quad X_{B_n}^{(0)} &= \frac{1}{4} (X^{(1)} - X_{B_n}^{(2)} + 4X_{B_n}^{(1)})
\end{align*}
\]
2.1. CURVE AND SPLINE EXTENSION

This is the switching-yard.

Here are the differences for 3 control points.

Here are the differences for 4 control points.
2.1.2 Circles and Ellipses

Here we describe the means to specify circles of arbitrary radius, drawn with arbitrary line styles. When large-sized objects are used they are regularly spaced around the circle. Similarly ellipses may be specified, but only those having major/minor axes aligned in the standard directions; spacing of objects is no longer regular, but is bunched toward the narrower ends.

Such a circle or ellipse is specified using...

\[ \text{\texttt{xycircle}(vector)\{\text{style}\}} \]

where the components of the \texttt{vector} determine the lengths of the axis for the ellipse; thus giving a circle when equal. The \texttt{style} can be any \texttt{conn}, as in 3.2 that works with curved arrows—many do. Alternatively \texttt{style} can be any \texttt{object}, which will be placed equally-spaced about the circle at a separation to snugly fit the \texttt{object}s. If \texttt{empty} then a solid circle or ellipse is drawn.

To Do: Recognize special \texttt{style}s; e.g. \texttt{for} dotted with \texttt{zerodot}, = for a doubled circle — alter the radius, draw two circles; > for double dotted (perhaps use \texttt{doubled@}, \texttt{tripled@}; > and < for chevrons; any others?

To use any of these special symbols as the \texttt{object} for \texttt{style} then enclose it within extra braces, e.g. \texttt{xycircle<20pt>{{>}}}. The curves are not truly circular or elliptical, but are approximations given by cubic Bézier segments. Hence the \texttt{xycurve} feature must be loaded.

The circles are constructed from four Bézier cubic curves, one for each quadrant of the circle or circle. To do this it is sufficient to establish the control points for each cubic segment. This is straightforward, using the following “magic number”, given a square basis such that the desired circle is the unit circle, or rectangular basis for which the “unit circle” is the desired circle.

\[ \text{\texttt{xydef}@Circmagic}@\{0.5517847\} \]
\[ \text{\texttt{xylet}@Circmagic}=	ext{\texttt{Circmagic}@} \]
\[ \text{\texttt{xydef}@twoPi}@\{6.2831852\} \]
\[ \text{\texttt{xydef}@fullPi}@\{3.1415926\} \]
\[ \text{\texttt{xydef}@halfPi}@\{1.5707963\} \]

This is the magic number, exactly given by \( \frac{\sqrt{385}}{12}(\sqrt{385} - 13) \), that helps construct the Bézier cubic curve that best approximates a quadrant of a circle. It does so with remarkable accuracy, differing by at most .5% of the radius at any angle; the average deviation along the whole quadrant being less than .13%. The basic \texttt{object} defined here is \texttt{circle}. 

\[ \text{\texttt{xynew}@\texttt{dimen}\texttt{L}@} \]
2.1. CURVE AND SPLINE EXTENSION

\xydef@\xycircle#1#{\hbox\bgroup\afterVECTORorEMPTY{%\xy@@{\R@=\X@c \L@=\Y@c}\xycircle@}{\xy@@{\R@=\R@c \L@=\L@c}\xycircle@}#1@}
\xydef@\xycircle@#1@#2{\DN@{#1}\ifx\next@\empty\def\onlyQuad@{}\else \count@=#1\relax
\ifnum\count@<5 \advance\count@-3\relax
\ifnum\count@<\z@ \advance\count@ 4\relax\edef\onlyQuad@{\the\count@}%\else\xyerror@{illegal circle <radius>: must be <vector> or <empty>}{}i\fi \xy@@{\def\circleSTYLE@{#2}}\def\circleSTYLE@{#2}\xycircle@i}
\xydef@\xycircle@i{\hbox{\vbox{\vskip\L@\hbox to2\R@{\hfill \buildcircle@ \hfill}\vskip\L@}}%\L@c=\R@ \R@c=\R@ \D@c=\L@ \U@c=\L@ \def\Leftness@{.5}\def\Upness@{.5}\def\Drop@@{\styledboxz@}\def\Connect@@{\straight@\relax}\Edge@c={\circleEdge}\OBJECT@x}
\xydef@\adjustMinMaxcirc@#1#2{\dimen@=\X@c \advance\dimen@#1\relax\ifdim\dimen@>\X@max \X@max=\dimen@\fi\dimen@=\X@c \advance\dimen@-#1\relax\ifdim\dimen@<\X@min \X@min=\dimen@\fi\dimen@=\Y@c \advance\dimen@#2\relax\ifdim\dimen@>\Y@max \Y@max=\dimen@\fi\dimen@=\Y@c \advance\dimen@-#2\relax\ifdim\dimen@<\Y@min \Y@min=\dimen@\fi}
\xydef@\buildcircle@{\save@\cubicCircleControls@@\ifx\circleSTYLE@\empty\DN@{\solidcircle@{}}\else \expandafter\DN@\expandafter{\addDASH@{}}\relax\fi\if\next@\circleSTYLE@\DN@{\expandafter{\expandafter{\addDASH@{}}}}\else\DN@{\circledobjects@}\fi\fi
\vfromcartesian@@1,0@\czeroEdge@\idfromc@{0@c}\vfromcartesian@@0,1@\czeroEdge@\idfromc@{1@c}\vfromcartesian@@-1,0@\czeroEdge@\idfromc@{2@c}\vfromcartesian@@0,-1@\czeroEdge@\idfromc@{3@c}\vfromcartesian@@1,\Circmagic@@@\czeroEdge@\idfromc@{1@m}\vfromcartesian@@\Circmagic@@,1@\czeroEdge@\idfromc@{2@m}\vfromcartesian@@-\Circmagic@@,1@\czeroEdge@\idfromc@{3@m}\vfromcartesian@@-1,\Circmagic@@@\czeroEdge@\idfromc@{4@m}\vfromcartesian@@-\Circmagic@@,-1@\czeroEdge@\idfromc@{5@m}\vfromcartesian@@\Circmagic@@,-\Circmagic@@@\czeroEdge@\idfromc@{6@m}\vfromcartesian@@\Circmagic@@,\Circmagic@@@\czeroEdge@\idfromc@{7@m}\vfromcartesian@@\Circmagic@@,\Circmagic@@@\czeroEdge@\idfromc@{8@m}}
\xydef@\doCircleQuadrant@@#1#2{\save@\cubicCircleControls@@\if\next@\circleSTYLE@\DN@{\\vfrom\circledobjects@}\fi\next@\leave@}
\xydef@\cubicCircleControls@@{\X@origin=\z@ \Y@origin=\z@\X@xbase=\R@ \Y@xbase=\z@ \X@ybase=\z@ \Y@ybase=\L@\vfrom\circledobjects@0\1,0@\czeroEdge@\idfromc@{0@c}\vfrom\circledobjects@0\0,1@\czeroEdge@\idfromc@{1@c}\vfrom\circledobjects@-1,0@\czeroEdge@\idfromc@{2@c}\vfrom\circledobjects@0,-1@\czeroEdge@\idfromc@{3@c}\vfrom\circledobjects@1,\Circmagic@@@\czeroEdge@\idfromc@{1@m}\vfrom\circledobjects@\Circmagic@@,\Circmagic@@@\czeroEdge@\idfromc@{2@m}\vfrom\circledobjects@\Circmagic@@,\Circmagic@@@\czeroEdge@\idfromc@{3@m}\vfrom\circledobjects@\Circmagic@@,-\Circmagic@@@\czeroEdge@\idfromc@{4@m}\vfrom\circledobjects@\Circmagic@@,-\Circmagic@@@\czeroEdge@\idfromc@{5@m}\vfrom\circledobjects@-\Circmagic@@,\Circmagic@@@\czeroEdge@\idfromc@{6@m}\vfrom\circledobjects@-\Circmagic@@,-\Circmagic@@@\czeroEdge@\idfromc@{7@m}\vfrom\circledobjects@-\Circmagic@@,\Circmagic@@@\czeroEdge@\idfromc@{8@m}}
\xydef@\doCircleQuadrant@@#1#2{\save@\ifcase\next@\relax\doCircleQuadrant@@@{0@c}{1@m}{2@m}{1@c}{#1}\or\doCircleQuadrant@@@{1@c}{3@m}{4@m}{2@c}{#1}\or\doCircleQuadrant@@@{2@c}{5@m}{6@m}{3@c}{#1}\or\doCircleQuadrant@@@{3@c}{7@m}{8@m}{0@c}{#1}\or\doCircleQuadrant@@@{0@c}{1@m}{2@m}{1@c}{#1}\or...}
CHAPTER 2. EXTENSIONS

The hook \texttt{\dosolidcircle@@} is provided so that back-ends may provide an alternative method to draw the circles/ellipses. Note that the token following \texttt{\solidcircle@} will be a group representing the ⟨style⟩ to be used.

\texttt{\xydef\solidcircle@\texttt{\dosolidcircle@@}}

This places objects equally spaced around a circle, according to angular position.  

\textbf{Bug:} objects are not equally spaced around ellipses constructed this way. To get regular spacing around ellipses, in terms of arc-length say, is much more difficult, both theoretically and practically.

\textbf{To Do:} the size of the resulting ⟨object⟩ does not take into account the size of the ⟨object⟩ dropped around the circle.
2.1. CURVE AND SPLINE EXTENSION

\placeCircQuadrant@{2}\%
\advance\dimen@-\p\advance\dimen@-\circleseparation@\p\%
\edef\circleplace@{\expandafter\removePT@\the\dimen@}\%
\placeCircQuadrant@{3}\%
\advance\dimen@-\p\advance\dimen@-\circleseparation@\p\%
\edef\circleplace@{\expandafter\removePT@\the\dimen@}\%
\placeCircQuadrant@{4}\}
\xydef@\placeCircQuadrant@#1{\%
\let\bstartPLACE@=\relax \doCircleQuadrant@@{}#1\%
\loop@ 
\expandafter\dimen@\circleplace@\p\%
\advance\dimen@-\circleseparation@\p\%
\edef\circleplace@{\expandafter\removePT@\the\dimen@}\%
\ifdim\dimen@<\p\ %\placeCircObject@ 
\repeat@ }
\xydef@\placeCircObject@{\begingroup 
\crvconnect@@ \Creset@@ \Invisible@false 
\expandafter\splinealong@\expandafter{\circleplace@} \%
\expandafter\drop\circleSTYLE@{}\endgroup }
\xydef@\buildcircle@{\save@ 
% \cubicCircleControls@@ 
\DNii@{\circledobjects@} \%
\ifx\circleSTYLE@\empty \DNii@{\solidcircle@{}}\%
\else \expandafter\DN@\expandafter{\addDASH@{}\relax 
\ifx\next@\circleSTYLE@ \DNii@{\solidcircle@{}}\%
\else \expandafter\DN@\expandafter{\addEQ@{}}
\ifx\next@\circleSTYLE@ \expandafter\circletemplate@{\addEQ@{}}{2}\%
\else \expandafter\DN@\expandafter{\addEQ@{3}}
\ifx\next@\circleSTYLE@ \expandafter\expandafter\expandafter\DN@\expandafter{\expandafter{\addEQ@{2}}}%
\expandafter\DN@\expandafter{\addEQ@{3}}% 
\expandafter\DN@\expandafter{\addEQ@{}}% 
\expandafter\DN@\expandafter{\addEQ@{2}}% 
\expandafter\DN@\expandafter{\addEQ@{3}}% 
\expandafter\DN@\expandafter{\addEQ@{}}% 
\expandafter\DN@\expandafter{\addEQ@{2}}% 
\expandafter\DN@\expandafter{\addEQ@{3}}% 
\expandafter\DN@\expandafter{\addEQ@{}}% 
\expandafter\DN@\expandafter{\addEQ@{2}}% 
\expandafter\DN@\expandafter{\addEQ@{3}}% 
\expandafter\DN@\expandafter{\addEQ@{}}% 
}
2.1. CURVE AND SPLINE EXTENSION

2.1.3 Quadratic Splines

Quadratic Bézier splines, as distinct from cubic Bézier splines, are constructed from parabolic arcs, using 'control points' to determine the tangents where successive arcs are joined.

Various implementations of such curves exist. The one adopted here is consistent with the \texttt{xfig} drawing utility and \texttt{tpic} implementations. These have the property of beginning and ending with straight segments, half the length to the corresponding adjacent control-point. Furthermore the midpoint between successive control-points lies on the spline, with the line joining the control-points being tangent there.

Such curves are specified, either as an \texttt{⟨decor⟩} or as an \texttt{⟨object⟩}, using...

\begin{verbatim}
\texttt{qspline{⟨style⟩}}
\end{verbatim}
where the start and end of the curve are at \( p \) and \( c \) respectively. The control-points are taken from the current stack, see 1.3o. If this stack is empty then a straight line is constructed.

The following example compares the quadratic spline with the gentler curving B-spline having the same control points, using \texttt{\crvs}.

\[ P \quad + \quad C \]

\[
\xy /r1.5pc/;++{P};p@,(+(2,2)++{+}++{+},+(2,-2)++{+}++{0})@+(2,2)++{+}++{0},+(2,0)++{C}="C","C",**\crvs{.}\)
\]

If the current stack is empty, simply pass everything to \texttt{\crvs@}, as if the request originated from a \texttt{\crvs} or \texttt{\ar}. Otherwise we must build up the segments of the spline.

The \#1 parameter to \texttt{\xy@spline@i} is to allow alternative implementations; currently it is ignored. When \#1 is indeed empty,
2.1. CURVE AND SPLINE EXTENSION

The initial and final points in a segment are stored as \( \mathbf{p} \) and \( \mathbf{c} \) respectively. Where needed, the control is in \( \mathbf{m} \).

The end & Log

\% $Log: xycurve.doc,v $ 
\% Revision 3.12 2011/03/14 20:14:00 krisrose 
\% Preparing for release 3.8.6. 
\% 
\% Revision 3.11 2010/06/10 18:45:50 krisrose 
\% Ross Moore’s e-mail address updated. 
\% Many obsolete files degraded to Historic. 
\% 
\% Revision 3.10 2010/05/06 17:46:29 krisrose 
\% 
\% Revision 3.9 2010/05/06 03:48:05 krisrose 
\% Fixed missing references. 
\%
CHAPTER 2. EXTENSIONS

% Revision 3.8 2010/04/16 06:06:52 krisrose
% Preparing for a new release...
% Revision 3.7 1999/02/16 15:12:50 krisrose
% Interim release (Y&Y fonts now free).
% Revision 3.6 1998/03/06 01:28:05 krisrose
% Releasing (with Y&Y fonts).
% Revision 3.4 1997/05/18 01:13:24 ross
% Essential bugfixes.
% Revision 3.3 1996/12/18 09:01:45 ross
% major revisions for the new BREAK methods
% spline edge-finding is more robust
% spline-breaks fully implemented
% intersections of curve with straight connections implemented
% improved tracings
% adjusted methods for styles
% Revision 3.2 1995/09/19 18:20:20 ross
% Bug fix release.
% Revision 3.1 1995/09/05 20:36:33 ross
% Release!
% Revision 3.0 1995/07/07 20:13:19 ross
% Major release w/new User’s Guide!
% Revision 2.13 1995/07/05 07:58:43 ross
% Ready for v3 release?
% Revision 2.12 1994/10/25 03:01:14 ross
% Final 3beta release [bug fixes & AMS-LaTeX fitting].
% Revision 2.12 1994/09/05 08:22:11 ross
% incorporates some speed-ups, extra documentation
% Revision 2.11 1994/07/05 09:27:52 ross
% Minor fixes; use 2.11 kernel stack code.
% Revision 2.9 1994/06/09 14:50:54 ross
% Release 3beta.
% Revision 2.8 1994/04/08 10:36:40 ross
% Second 3alpha release.
% Revision 2.7 1994/03/08 02:06:01 kris
% Release 3alpha.
2.2 Frame and Bracket extension

Vers. 3.14 by Kristoffer H. Rose ⟨krisrose@tug.org⟩

The frame extension provides a variety of ways to put frames in \text{Xy-pic} pictures.

Header:

The frames are \text{Xy-pic} ⟨object⟩s on the form

\texttt{\frm{⟨frame⟩}}

\begin{verbatim}
\frm{⟨frame⟩}
\end{verbatim}

to be used in ⟨positions⟩: Dropping a frame with *\ldots\frm{⟨frame⟩} will frame the \textit{c} object; connecting with **\ldots\frm{\ldots⟨frame⟩} will frame the result of \textit{c}.p.
**XY-pic interface**  The frames are integrated in XY-pic as (object)s as follows. They generate a dummy object, then define the \Drop@@ action to place the requested frame. The \Connect@@ operation is defined to place the requested frame around the current and the previous object (using the \cmerge@ command of the kernel).

\begin{verbatim}
\xydef@{\frm@{\hbox{}\dimen@=\L@c \advance\dimen@\R@c \wdz@=\dimen@}
  \ht\z@=\U@c \dp\z@=\D@c \dimen@=\U@c \advance\dimen@\D@c
  \ifdim\wdz@=\z@ \def\Leftness@{0}\else \quotient@\Leftness@\L@c\wdz@ \fi
  \ifdim\dimen@=\z@ \def\Upness@{0}\else \quotient@\Upness@\U@c\dimen@ \fi
  \ifdim\wdz@=\z@ \def\Leftness@{.5}\else \quotient@\Leftness@\L@c\wdz@ \fi
  \ifdim\dimen@=\z@ \def\Upness@{.5}\else \quotient@\Upness@\U@c\dimen@ \fi
  \expandafter\Edge@c\expandafter{\prevEdge@@}\let\frmradius@@=\z@
  \def\Drop@@{}%
  \def\Connect@@{\xy@@{\save@\cmerge@\Y@p\X@p\D@p\U@p\L@p\R@p\Drop@@}\restore}%
  \xyFN@{\frm@i}
\end{verbatim}

The main command is \frm@ which looks up if a custom control sequence is available for the requested frame or whether the generic `curve along the edge' code should be invoked.

\begin{verbatim}
\xydef@{\frm@{#1#2}{\DNii@{frm#1{#2}}%}
  \expandafter\let\expandafter\next@\csname\codeof\nextii@\endcsname
  \ifx\next@\relax \DNii@{frm{#2}}%\expandafter\let\expandafter\next@\csname\codeof\nextii@\endcsname
  \ifx\next@\relax \let\next@=\frm@generic \fi\fi \next@}
\xydef@{\frm@generic{\xyerror@{No generic frames yet!}{}}}
\end{verbatim}

This extra command is used to define \Drop@@ for each frame such that \frmradius@@ is available.

\begin{verbatim}
\xydef@{\frmDrop@{#1}{\ifx\frmradius@@\z@ \addtoDrop@@{\let\frmradius@@=\z@ #1}%
  \else \expandafter\addtoDrop@@\expandafter{\expandafter\def\expandafter\frmradius@@\expandafter{\frmradius@@}#1}\fi}
\end{verbatim}

Below we distinguish between ‘ordinary’ frames, ‘brackets’ and ‘fills’; last we present how some
2.2. FRAME AND BRACKET EXTENSION

Frames can be added to other objects using object modifier (shape)s.

2.2.1 Frames

Figure 2.2 shows the possible frames and the applicable (modifier)s with reference to the notes below.

Notes

2.2a. The \frm{} frame is a dummy useful for not putting a frame on something, e.g., in macros that take a (frame) argument.

\xydefcsname@{frm{}}{}
\xyletcsnamecsname@{frm[]}{frm{}}
\xyletcsnamecsname@{frm[o]}{frm{}}

2.2b. Rectangular frames include \frm{.}, \frm{-}, \frm{=}, \frm{--}, \frm{==}, and \frm{o-}. They all make rectangular frames that essentially trace the border of a rectangle-shaped object. The (frame)s \frm{-} and \frm{=} allow an optional corner radius that rounds the corners of the frame with quarter circles of the specified radius. This is not allowed for the other frames—the \frm{o-} frame always gives rounded corners of the same size as the used dashes (when \xydashfont is the default one then these are 5pt in radius).

Exercise 2.4: How do you think the author typeset the following?

\begin{center}
\begin{tikzpicture}
\draw (0,0) rectangle (1,1);
\draw (0,0) rectangle (0.5,0.5);
\end{tikzpicture}
\end{center}

The commands still hack away with rules and stuff... \frm{} just fills the edges of the object rectangle border with dots using dots as in dotted connections in the kernel.

The \frm{variant}{-} and \frm{variant}{=} set a single/double frame that just surrounds the current object; if a (radius) is given then it should be a <⟨dimen⟩> where the ⟨dimen⟩ will be used for the corner radius.

The commands for \frm{-} and \frm{=} are quite similar in that they just call \framed@ one or two times, respectively, with the corner radius and \(L, R, D, U\) extents reflecting the size of the
These are overlayed with the \texttt{frm{.}} frame above to show the way they are centered on the object.

Figure 2.2: Plain (frame)s.

Figure 2.3: Bracket (frame)s.
frame. \texttt{framed@} then sets the horizontal and vertical fill commands to generate rules of the right length, then calls \texttt{framed@@} where the real work is done.

\begin{verbatim}
\xdef\csname frm-{\string-}\endcsname{\framed@\frmradius@@}
\xlet\csname frm[\string-]\endcsname{frm-{}}
\xdef\frm@\string{\let zcor=ramezerodot@@
\let hfill=rm@solidh@@ \let vfill=rm@solidv@@
\framed@@}
\xlet\framesetthick@relax
\xdef\frmradius@@{
\leader\hrule height\B@ depth\B@ hfill}
\xdef\frmradius@@{
\leader\vrule width2\B@
\vfill}
\xdef\csname frm{=\string-}\endcsname{\framed@\frmradius@@
\advance\L@c-2\p@ \advance\R@c-2\p@ \advance\U@c-2\p@
\ifdim\dimen@>2\p@ \advance\dimen@-2\p@ \else \dimen@=\z@\fi
\framed@\dimen@}
\xlet\csname frm[\string-]{=}\endcsname{frm{=}}
\end{verbatim}

When the line extension is also loaded then we use the line thickness for frames:

\begin{verbatim}
\edef\framesetthick@{line} \edef\frmradius@@{.5\xylinethick@}
\edef\frmradius@@\bgroup\framesetthick@line
\hbox{\framesetthick@line} \vrule height.5\B@ depth.5\B@ width\B@}
\withoption{line}{%}
\let\framezerodot@@\frm@thickc@@ \let\framesetthick@=\framesetthick@line
\framed@@
\end{verbatim}

\texttt{framed@@} is where we build a box with the sides of the frame displaced appropriately. To DoGeneralise this to handle any directional!

\begin{verbatim}
% Procedure:
% 0 setup hrulefill and vrulefill as appropriate...
% 1 Lower R to be less than half of both (U+D) and (L+R).
% 2 Start vbox at (X-L,Y-D) except center overshoot.
% 3 Generate corner CO4 at R; \texttt{w[A0]} := -1/2 width(corner);
% r[B0] := 1/2 rulewidth; h[\texttt{dimen@ii}] := U+D+2w;
% 4 Row 1: hbox to \texttt{W{kern w CO4 hrulefill CO3 kern w}}.
% 5 Row 2: hbox to \texttt{W{kern-r vbox to h\{vrulefill\} hfill vbox to h\{vrulefill\} kern-r}}.
% 6 Row 3: hbox to \texttt{W{kern w CO1 hrulefill CO2 kern w}}.

\edef\framed@@@{\setboxz@h{\R@=#1\relax
\dimen@=\L@c \advance\dimen@
\R@c \dimen@ii=\U@c \advance\dimen@ii
\D@c %1
\ifdim.5\dimen@<\R@ \R@=.5\dimen@ fi
\ifdim.5\dimen@ii<\R@ \R@=.5\dimen@ii fi
\A@=\X@c \advance\A@-\L@c \B@=\Y@c \advance\B@-\D@c %2
\ifdim\R@<\p@ else \cirrestrict\fi
\dimen@=\R@ \advance\dimen@-\L@c \advance\dimen@-\R@c
\ifdim\dimen@z@ \advance\A@-.5\dimen@ fi
\ifdim\dimen@=\R@ \advance\dimen@-\U@c \advance\dimen@-\D@c
\ifdim\dimen@z@ \advance\B@-.5\dimen@ fi
\kern\A@ \raise\B@ vbox to \dimen@ii{framed@body@@@}%
\end{verbatim}
The dashed frames ‘dash’ with the dash used for dashed lines.
2.2. FRAME AND BRACKET EXTENSION

2.2c. The frame \( \texttt{\textbackslash frm{,}} \) puts a shade, built from rules, into the picture beneath the (assumed rectangular) object, thereby giving the illusion of ‘lifting’ it; \( \texttt{\textbackslash frm{\langle \text{dimen} \rangle{,}}} \) makes this shade \( \langle \text{dimen} \rangle \) deep.

\( \texttt{\textbackslash frm{\langle \text{dimen} \rangle{,}}} \) combines a \( \texttt{\textbackslash frm{-}} \) with a \( \texttt{\textbackslash frm{,}} \).

A black rule is just that, a shade is two rules placed under and left of the rectangular object outline.

\begin{verbatim}
2.2c. The frame \texttt{\textbackslash frm{,}} puts a shade, built from rules, into the picture beneath the (assumed rectangular) object, thereby giving the illusion of ‘lifting’ it; \texttt{\textbackslash frm{\langle \text{dimen} \rangle{,}}} makes this shade \langle \text{dimen} \rangle deep.
\texttt{\textbackslash frm{\langle \text{dimen} \rangle{,}}} combines a \texttt{\textbackslash frm{-}} with a \texttt{\textbackslash frm{,}}.

A black rule is just that, a shade is two rules placed under and left of the rectangular object outline.
\end{verbatim}
The \texttt{\blackened} option is subsumed within a more general \texttt{\filled}; see below.

2.2d. Circles done with \texttt{\frame{o}} have radius as \((R + L)/2\) and with \texttt{\frame<(\text{dimen})>{o}} have radius as the \((\text{dimen})\); \texttt{\frame{oo}} makes a double circle with the outermost circle being the same as that of \texttt{\frame{o}}.

\textbf{Exercise 2.5:} What is the difference between \texttt{*\cir{}} and \texttt{*\frame{o}}? \hfill (p. 578)

The code here is repetitive and could be improved. The only purpose of \texttt{\let\framehfill=\frm@dashh@@} etc. is to serve as a marker for the required style: solid/dashed/dotted.
2.2e. Ellipses specified using $\texttt{\textbackslash frm\{e\}}$ have axis lengths $\left(\frac{R + L}{2}\right)$ and $\left(\frac{U + D}{2}\right)$, while those with $\texttt{\textbackslash frm<\textbackslash dimen,\textbackslash dimen>\{e\}}$ use the given lengths for the axes. $\texttt{\textbackslash frm\{ee\}}$ makes a double ellipse with outermost ellipse being the same as that of $\texttt{\textbackslash frm\{e\}}$.

Without special support to render the ellipses, either via a ⟨driver⟩ or using the \texttt{arc} feature, the ellipse will be drawn as a circle of radius approximately the average of the major and minor axes.
2.2. FRAME AND BRACKET EXTENSION

To Do: Allow (frame variant)s like those used for directionals, i.e., \frm2\{-\} should be the same as \frm\{=\}. Add \frm\{o,\} and more brackets.

2.2.2 Brackets

The possible brackets are shown in figure 2.3 with notes below.

Notes

2.2f. Braces are just the standard plain \TeX{} large braces inserted correctly in \textsc{Xy-pic} pictures with the ‘nib’ aligned with the reference point of the object they brace.

Exercise 2.6: How do you think the author typeset the following?

\[ \begin{array}{c}
\text{A} \\
\text{B}
\end{array} \]

They just use the pieces of plain \TeX{} brace delimiters at \texttt{\Bigg} size and those of \texttt{\overbrace} and \texttt{\underbrace}.

The horizontal ones mimic the plain \TeX{} ones quite closely:
The inserted extra .5pt at (1) is the author's responsibility...

The vertical ones repeat the above for the vertical brace extension characters:
An ad hoc correction to compensate for the ‘undershoot’ of top/bottom segments is added at (1).

2.2g. Parenthesis are like braces except they have no nib and thus do not depend on where the reference point of $c$ is.

The horizontal ones repeat the braces only without a nib:

The vertical ones are very like braces including the ad hoc correction.
Bug: The brackets above require that the computer modern cmex font is loaded in TeX font position 3.

2.2.3 Filled regions

In addition to the above there is a special frame that “fills” the inside of the current object with ink: \frm{*} and \frm{**}; the latter is intended for emphasizing and thus “strokes” the outline, using the thinnest black line available on the printer or output device; furthermore it moits the actual filling in case this would obscure further text typeset on top. Some alteration to the shape is possible, using *\frm<dimen>{*}. Hence rectangular, oval, circular and elliptical shapes can be specified for filling. The following examples illustrate this in each case:

<table>
<thead>
<tr>
<th>(object)</th>
<th>\frm{*}</th>
<th>\frm{**}</th>
<th>\frm&lt;6pt&gt;**</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Example" /></td>
<td><img src="image" alt="Example" /></td>
<td><img src="image" alt="Example" /></td>
<td><img src="image" alt="Example" /></td>
</tr>
</tbody>
</table>

However, filling non-rectangular shapes will result in a rectangle unless a driver is used that supports arbitrary filling. With some drivers the above fills will thus all be identical, as rectangular.

When the previous ⟨object⟩ has \rectangleEdge then the shape is either oval or rectangular, depending on whether a ⟨dimen⟩ has been specified, using the ⟨dimen⟩ as radius for the corners, but not exceeding half the shorter side-length.

When the current ⟨object⟩ has \circleEdge then the region is elliptical, using the extents R+L and U+D as the major/minor axes, or circular if a ⟨dimen⟩ is specified. The ⟨dimen⟩ will become the radius after reducing to a maximum of half the longest side of the enclosing rectangle. With \zeroEdge presume rectangular, or circular if a ⟨dimen⟩ is supplied.

The default for \frm{**} is to just make a plain frame.
2.2. FRAME AND BRACKET EXTENSION

The default implementation uses only \texttt{bblackened}, in all cases. The distinction between \texttt{\textbackslash frm{\*}} and \texttt{\textbackslash frm{\**}} can be ignored.

Alternative implementations may use these, in which the parameter \texttt{#1} is the contents of a \texttt{\vbox} filling to the height, depth and width of the region to be filled.

2.2.4 Framing as object modifier

In addition, frames may be accessed using the special \texttt{[F⟨frame⟩]} object modifier \langle shape⟩s that will add the desired \langle frame⟩ to the current object. The frame appropriate to the edge of the object will be chosen (presently either rectangular or elliptical).

If shape modifiers need to be applied to the \langle frame⟩ alone then they can be included using : as separator. Thus \texttt{[F-:red]} will make a red frame (provided the color extension is active, of course). Additionally the variant of frames using \texttt{⟨dimen⟩} can be accessed by specifying \texttt{[...:<⟨dimen⟩>]}.

\begin{verbatim}
\xydefcsname@{shape [F...]}#1{\xyFN@\Fshape@#1:@}
\end{verbatim}
The following are used to ensure that the current styles, after typesetting, are those of the ⟨object⟩, rather than the ⟨frame⟩:

\xydef@\saveframestyles@{%  
\expandafter\def\expandafter\afterframe@preStyle@\expandafter{\preXY@style@}  
\expandafter\def\expandafter\afterframe@postStyle@\expandafter{\postXY@style@}  
\xydef@\restoreframestyles@{%  
\expandafter\gdef\expandafter\preXY@style@\expandafter{\afterframe@preStyle@}  
\expandafter\gdef\expandafter\postXY@style@\expandafter{\afterframe@postStyle@}  
}\def\afterframe@preStyle@{}\def\afterframe@postStyle@{}}

Here are some simple examples using this feature.

\xy **<1.5pt>[F**:/white]++[F**:/red]  
\txt{text with background}  ,+!D+/d1pc/,*++[F**:/black][white]  
\txt\bf{bold white on black}\endxy

Notice that when multiple frame-modifiers are used, the frames are actually placed in reverse order, so that earlier ones are printed on top of later ones.

To Do: The frame option is not quite complete yet: some new frames and several new brackets should be added.
2.2.5 Using curves for frames

If the curve option is loaded, then circular and elliptical frames of arbitrary radius can be constructed, by specifying \UseCurvedFrames. This can be negated by \UseFontFrames. Both of these commands obey normal \TeX\ grouping. Furthermore, dotted and dashed frames now have a regular spacing of their constituent objects. The usual warnings about memory requirements for large numbers of curves apply here also.

Use the \xycircle setup in xycurve.doc to implement circular and elliptical frames.

Finally, the frame extension is added to the ⟨driver⟩ system:

Finally, the frame extension is added to the \⟨driver\⟩ system:

End & log
CHAPTER 2. EXTENSIONS

% $Log: xyframe.doc,v $
% Revision 3.14 2012/05/24 00:30:38 krisrose
% Release 3.8.8 with xyframes fix by Norbert Preining.
% %
% Revision 3.13 2011/03/14 20:14:00 krisrose
% Preparing for release 3.8.6.
% %
% Revision 3.12 2010/07/27 09:49:34 krisrose
% Started xyling (and address updates).
% %
% Revision 3.11 2010/06/10 18:45:50 krisrose
% Reference to GPL by URL.
% %
% Revision 3.10 2010/04/27 05:08:37 krisrose
% Elliptic frame adjustment suggested by Daniel.
% %
% Revision 3.9 2010/04/16 06:06:52 krisrose
% Preparing for a new release...
% %
% Revision 3.8 2010/04/13 08:44:32 krisrose
% Old xydiff patches applied.
% %
% Revision 3.7 1999/02/16 15:12:50 krisrose
% Interim release (Y&Y fonts now free).
% %
% Revision 3.6 1998/03/06 01:28:05 krisrose
% Releasing (with Y&Y fonts).
% %
% Revision 3.4 1997/05/18 01:14:25 krisrose
% Essential bugfixes.
% %
% Revision 3.3 1996/12/19 03:31:56 krisrose
% Maintenance release
% %
% Revision 3.2 1995/09/19 18:22:27 kris
% Bug fix release.
% %
% Revision 3.1 1995/09/05 20:31:32 kris
% Releasing!
% %
% Revision 3.0 1995/07/07 20:14:21 kris
% Major release w/new User’s Guide!
% %
% Revision 2.14 1995/07/05 22:11:03 kris
% Buglets...
% %
% Revision 2.13 1995/07/04 15:11:17 kris
% Ready to release v3?
2.3  More Tips extension

Vers. 3.11 by Kristoffer H. Rose (kris@diku.dk)

This extension provides several additional styles of ‘tips’ for use (primarily) as arrow heads, and makes it possible to define customised tips. This is used to support tips that mimic the style of the Computer Modern fonts\(^2\) by Knuth (see [8] and [6, appendix F]) and of the Euler math fonts distributed by the AMS.

Header:

\[
\text{%% $Id: xytips.doc,v 3.11 2013/10/06 01:12:08 krisrose Exp $}
\]
\[
\text{%% Xy-pic ‘More Tips’ extension.}
\]
\[
\text{%% Copyright (c) 1992-1996 Kristoffer H. Rose <kris@diku.dk>}
\]
\[
\text{%% This file is part of the Xy-pic package for graphs and diagrams in TeX.}
\]
\[
\text{%% Copyright (c) 1991-2011 Kristoffer H. Rose <kris@diku.dk>}
\]
\[
\text{%% The Xy-pic package is free software; you can redistribute it and/or modify}
\]
\[
\text{%% it under the terms of the GNU General Public License as published by the}
\]
\[
\text{%% Free Software Foundation; either version 2 of the License, or (at your}
\]
\[
\text{%% option) any later version.}
\]
\[
\text{%% The Xy-pic package is distributed in the hope that it will be useful, but}
\]
\[
\text{%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY}
\]
\[
\text{%% or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License}
\]
\[
\text{%% for more details.}
\]

\(^2\)This function was earlier supported by the \texttt{cmtip} extension which is still included in the distribution but is now obsolete.
You should have received a copy of the GNU General Public License along with this package; if not, see http://www.gnu.org/licenses/.

\ifx\xyloaded\undefined \input xy \fi
\xyprovide{tips}{More Tips extension}{\stripRCS$Revision: 3.11 $}%
\{Kristoffer H.``Rose}{kris@diku.dk}%
\{IBM T.~J.\ Watson Research Center, P.O.~Box 704, Yorktown Heights, NY 10598 (USA)}

First save the XY-commands that may need restoring.

\xylet\tip@xy=\tip@
\xylet\atip@xy=\atip@
\xylet\btip@xy=\btip@
\xylet\Tip@xy=\Tip@
\xylet\Ttip@xy=\Ttip@
\xylet\tipjot@xy=\tipjot@

Next declare the fonts (initially the original cm style).

\xyfont@\xy@@atfont=xycmat10
\xyfont@\xy@@btfont=xycmbt10
\xyfont@\xy@@alfont=xyluat10
\xyfont@\xy@@blfont=xylubt10

Arrow tip font selection is done with the command

\SelectTips{(family)}{(size)}

\xydef@\tipfamily@{cm}
\xydef@\tipsize@@{10}
\xydef@\SelectTips#1#2{%
  \DN@{#1}\ifx\next@\empty\else \def\tipfamily@{#1}\fi
  \DN@{#2}\ifx\next@\empty\else \def\tipsize@@{#2}\fi
  \csname tipfamily \tipfamily@\endcsname \ignorespaces}

where the (family) and (size) should be selected from the following table.

<table>
<thead>
<tr>
<th>Family</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>xy</td>
<td>→⇒</td>
<td>→⇒</td>
<td>→⇒</td>
</tr>
<tr>
<td>cm</td>
<td>→⇒</td>
<td>→⇒</td>
<td>→⇒</td>
</tr>
<tr>
<td>eu</td>
<td>→⇒</td>
<td>→⇒</td>
<td>→⇒</td>
</tr>
<tr>
<td>lu</td>
<td>→⇒</td>
<td>→⇒</td>
<td>→⇒</td>
</tr>
</tbody>
</table>

You can also include a \TeX scaling in your font (size), if you have the corresponding font, like \SelectTips{cm}{10 scaled 2000}. Also see the line extension for more ways to scale your tips.

The table is really encoded in control sequences \tipfamily (family) that look very much alike since all the currently existing combinations are merely alternative fonts. This is likely to change, however, ...

First cm, i.e., computer modern: these are simplified by the fact that we can use the normal double arrows.
2.3. MORE TIPS EXTENSION

Next Euler tips: the single tips merely reuse the cm code except for the slightly smaller distance between double tips; the double arrows have to be recoded to mimic the steeper Euler appearance.

Finally, the lu fonts, contributed by Jeremy Gibbons. Uses the fonts for the arrow tip family lu (which will expand out to xyluat10 and xylubt10, at 10pt).

Once a selection is made, the following commands are available:

\UseTips activate selected tips
\NoTips deactivate
They are local and thus can be switched on and/or off for individual pictures using the \TeX grouping mechanism, \textit{e.g.},
\SelectTips{cm}{10}
\xy*{} \ar @{*\UseTips\dir_{<<}}-*\NoTips\dir{>}
\endxy
will typeset

\begin{xy}
(20,5)*{}
\end{xy}

regardless of which tips are used otherwise in the document.

### 2.3.1 End & log

\xyendinput

\% $Log: xytips.doc,v $  
\% Revision 3.11 2013/10/06 01:12:08 krisrose  
\% Backpatch for 3.8.9...  
\%  
\% Revision 3.10 2013/10/02 02:04:28 krisrose  
\% Release 3.8.9 including Barr's diagxy feature.  
\%  
\% Revision 3.9 2011/03/14 20:14:00 krisrose  
\% Preparing for release 3.8.6.  
\%  
\% Revision 3.8 2010/07/27 09:49:34 krisrose  
\% Started xyling (and address updates).  
\%  
\% Revision 3.7 2010/06/10 18:45:50 krisrose  
\% Reference to GPL by URL.  
\%  
\% Revision 3.6 2010/04/17 14:45:48 krisrose  
\% Generate and extract Type1 fonts.  
\%  
\% Revision 3.5 2010/04/17 04:19:41 krisrose  
\% Integrated xylu tips by Jeremy Gibbons.  
\%  
\% Revision 3.4 2010/04/16 06:06:52 krisrose  
\% Preparing for a new release...  
\%  
\% Revision 3.3 1996/12/19 04:12:13 krisrose  
\% New for this maintenance release.  
\%  
2.4 Line styles extension

Vers. 3.10 by Ross Moore (ross.moore@mq.edu.au)

This extension provides the ability to request various effects related to the appearance of straight lines; e.g. thickness, non-standard dashing, and colour.

Header:

This file is part of the Xy-pic package for graphs and diagrams in TeX. See the companion README and INSTALL files for further information.

The Xy-pic package is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

The Xy-pic package is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this package; if not, see http://www.gnu.org/licenses/.
These are effects which are not normally available within \TeX. Instead they require a suitable ‘back-end’ option to provide the necessary \special commands, or extra fonts, together with appropriate commands to implement the effects. Thus

Using this extension will have no effect on the output unless used with a backend that explicitly supports it.

The extension provides special effects that can be used with any \Xy-pic \langle object \rangle, by defining \langle \text{shape} \rangle modifiers. The modification is local to the \langle object \rangle currently being built, so will have no effect if this object is never actually used.

**Adjusting line thickness**

The following table lists the modifiers primarily to alter the thickness of lines used by \Xy-pic. They come in two types — either a single keyword, or using the key-character | with the following text parsed.

<table>
<thead>
<tr>
<th>Modifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[thicker]</td>
<td>double line thickness</td>
</tr>
<tr>
<td>[thinner]</td>
<td>halve line thickness</td>
</tr>
<tr>
<td>[</td>
<td>&lt;\text{num}&gt;]</td>
</tr>
<tr>
<td>[</td>
<td>&lt;\text{dimen}&gt;]</td>
</tr>
<tr>
<td>[</td>
<td>&lt;\text{dimen}&gt;]</td>
</tr>
<tr>
<td>[</td>
<td>=\langle \text{word} \rangle]</td>
</tr>
<tr>
<td>[</td>
<td>*]</td>
</tr>
<tr>
<td>[butt]</td>
<td>butt cap at ends</td>
</tr>
<tr>
<td>[roundcap]</td>
<td>round cap at ends</td>
</tr>
<tr>
<td>[projcap]</td>
<td>projecting square cap.</td>
</tr>
</tbody>
</table>

Later settings of the linewidth override earlier settings; multiple calls to [thicker] and [thinner] compound, but the other variants set an absolute thickness. The line-thickness specification affects arrow-tips as well as the thickness of straight lines and curves. Three kinds of line-caps are available; they are discussed below in the section on ‘poly-lines’.

Using the PostScript back-end, the size of the arrow-head grows aesthetically with the thickness of the line used to draw it. This growth varies as the square-root of the thickness; thus for very thick lines (20+ times normal) the arrowhead begins to merge with the stem.

Load the style extension to establish the necessary infra-structure.
Implementation  Record the line thickness locally in a dimen register. The initial value is read from the xydash10 font. Each time \xylinethick@ is changed, its previous value is stored as the expansion of \xyprevwidth@@, in case this is needed by a specific back-end.

The diagram in figure 2.4, page 236, uses different line-thicknesses and colours.

Poly-lines  By a ‘poly-line’ we mean a path built from straight line segments having no gaps where each segment abuts the next. The poly-line could be the edges of a polygon, either closed or open if the end-points are different. The reason for considering a poly-line as a separate ⟨object⟩, rather than simply as a ⟨path⟩ built from straight lines, becomes apparent only when the lines have appreciable thickness. Then there are several standard ways to fashion the ‘joins’ (where segments meet). Also the shape of the ‘caps’ at
either end of the poly-line can be altered.

The following modifiers are used to determine the shapes of the line ‘caps’ and ‘joins’:

<table>
<thead>
<tr>
<th>Modifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[</td>
<td>J\langle val\rangle]</td>
</tr>
<tr>
<td>[\text{mitre}]</td>
<td>mitre-join, same as [</td>
</tr>
<tr>
<td>[\text{roundjoin}]</td>
<td>round join, same as [</td>
</tr>
<tr>
<td>[\text{bevel}]</td>
<td>bevel-join, same as [</td>
</tr>
<tr>
<td>[\text{butt}]</td>
<td>“butt” cap, same as [</td>
</tr>
<tr>
<td>[\text{roundcap}]</td>
<td>round cap, same as [</td>
</tr>
<tr>
<td>[\text{projcap}]</td>
<td>“projecting square” cap, same as [</td>
</tr>
<tr>
<td>[\text{miter}\langle\text{num}\rangle]</td>
<td>set mitrelimit to (\langle\text{num}\rangle \geq 1)</td>
</tr>
</tbody>
</table>

These effects are currently implemented only with the POSTSCRIPT back-end or when using \texttt{\xypolyline} (described below) with a POSTSCRIPT \langle driver\rangle. In this case the ‘cap’ setting can be applied to any segment, straight or curved, whether part of a poly-line or not; however the ‘join’ setting applies only to poly-lines. Arrow-tips are not affected. The defaults are to use round joins and round-cap ends.

Adjusting the miter-limit affects how far miters are allowed to protrude when two wide lines meet at small angles. The \(\langle\text{num}\rangle\) is in units of the line-thickness. Higher values mean using bevels only at smaller angles, while the value of 1 is equivalent to using bevels at all angles. The default miter-limit is 10.

Current values are stored in case these are needed within nested portions of diagrams.

\texttt{\xypolyline}

\texttt{\xydef@\xypolylinecap@\{2\}}
\texttt{\xydef@\xypolylinejoin@\{2\}}
\texttt{\xydef@\xypolylinemiter@\{10\}}

Recognise keywords for ‘line-caps’ and ‘line-joins’:

\texttt{\xydefcsname@{style \{bevel\}}\{\xyshape@bevel\}}
\texttt{\xydefcsname@{style \{roundjoin\}}\{\xyshape@rdjoin\}}
\texttt{\xydefcsname@{style \{miter\}}\{\xyshape@miter\}}
\texttt{\xydefcsname@{style \{butt\}}\{\xyshape@butt\}}
\texttt{\xydefcsname@{style \{roundcap\}}\{\xyshape@rdcap\}}
\texttt{\xydefcsname@{style \{projcap\}}\{\xyshape@projcap\}}

\texttt{\xydef@\xyshape@butt@\{\xysetlinecap\}}
\texttt{\xydef@\xyshape@rdcap@\{\xysetlinecap\}}
\texttt{\xydef@\xyshape@projcap@\{\xysetlinecap\}}
\texttt{\xydef@\xyshape@miter@\{\xysetlinejoin\}}
\texttt{\xydef@\xyshape@rdjoin@\{\xysetlinejoin\}}
\texttt{\xydef@\xyshape@projjoin@\{\xysetlinejoin\}}
\texttt{\xydef@\xyshape@bevel@\{\xysetlinejoin\}}
These use the more sophisticated approach that is easier to adjust for different (driver)s.

Implementation of line-caps:

Implementation of line-joins:
Implementation of miter-limit:

The path taken by the 'poly-line' this is read as the list of positions in the current 'stack', ignoring size extents. The macro \xypolyline is used as a ⟨decor⟩; it reads the positions from the stack, but leaves the stack intact for later use.

Implementation of 'poly-lines'.

The following diagram illustrates the use of line-thickness, line-joins and line-caps with poly-lines. It contains an example of each of the styles.

\begin{center}
\begin{tikzpicture}
  \draw[very thick] (0,0) -- (1,0) node[midway,right] {A} -- (1,1) node[midway,right] {B} -- (0,1) node[midway,right] {A} -- cycle;
\end{tikzpicture}
\end{center}
\xycompileto{poly}\
{/r4pc/:,*[*]<5pt>}[thicker]\xybox{\%
++(3,2){}="X"
@={p+CU,p+LU,p+LD,p+RD,p+RU,p+CU}
,{0*[miter]\xypolyline{}}
,{\xypolyline{**},0i0}
,"X",++(2.5,1.5){}="X"
@=!{CU,LU,LD,RD,RU,CU}
,\xypolyline{[roundjoin]}\xypolyline{}}
,{0*[gray]\xypolyline{*},0i0}
,"X",++(2,1){}="X"
,\xypolyline{[bevel]}\xypolyline{0i0}
,"X"-(.7,0)***\txt\LARGE{A}="a"
,"X"+(.7,0)***\txt\LARGE{B}="b"
,{\ar@{-}*@{[butt][thinner]}"a";"b"<1pc>}
,{\ar@{-}*@{[roundcap][thinner]}"a";"b"}
,{\ar@{-}*@{[projcap][thinner]}"a";"b"<-1pc>}
})

Note the use of \xypolyline{...} to apply style-modifiers to a polyline. The @=!.. method for loading the stack gives equivalent results to using @={p+..}, since \xypolyline ignores the edge extents of each ⟨pos⟩ in the stack.

Note also that the argument #1 to \xypolyline affects what is typeset. Allowable arguments are:

<table>
<thead>
<tr>
<th>Argument</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>\xypolyline{}</td>
<td>solid line</td>
</tr>
<tr>
<td>\xypolyline{..}</td>
<td>dotted line</td>
</tr>
<tr>
<td>\xypolyline{-}</td>
<td>dashed line</td>
</tr>
<tr>
<td>\xypolyline{*}</td>
<td>fill enclosed polygon</td>
</tr>
<tr>
<td>\xypolyline{?}</td>
<td>fill enclosed polygon using even-odd rule</td>
</tr>
<tr>
<td>\xypolyline{(*)}</td>
<td>use \dir{*} for lines</td>
</tr>
<tr>
<td>\xypolyline{&lt;toks&gt;}}</td>
<td>using \dir{&lt;toks&gt;}</td>
</tr>
</tbody>
</table>

The latter cases one has **\dir{...} being used to connect the vertices of the polyline, with {(*)} being needed to get **\dir{(*)}. Similarly **\dir is used when a ⟨driver⟩ is not available to specifically support polylines; in particular the two ‘fill’ options * and ? will result in a dotted polygon outline the region intended to be filled.

In all cases it is up to the user to load the stack before calling \xypolyline{...}. A particularly common case is the outline of an existing Xy-pic ⟨object⟩, as in the example above. Future extensions to \frm will provide a simplified mechanism whereby the user need not call \xypolyline explicitly for such effects.

\xylet@\xypolyline@Special=\eat@
\xylet@\xypolyfill@Special=\eat@
\xylet@\xypolyeofill@Special=\eat@
\xylet@\xypolydot@Special=\eat@
\xylet@\xypolydash@Special=\eat@
\xydef@\UnloadpolySpecials@%\let\xypolyline@Special=\eat@
\let\xypolyfill@Special=\eat@
\let\xypolyeofill@Special=\eat@
CHAPTER 2. EXTENSIONS

\let\xypolyeofill@Special=\eat@
\let\xypolydot@Special=\eat@
\let\xypolydash@Special=\eat@  
\}
\xydef@\xy@polystyle@#1{%
  \let\poly@style@@=\xypolyline@Special
  \DNii@{#1}\ifx\nextii@\empty
  \else\DN@{*}\ifx\next@\nextii@ \let\poly@style@@=\xypolyfill@Special
  \else\DN@{?}\ifx\next@\nextii@ \let\poly@style@@=\xypolyeofill@Special
  \else\expandafter\DN@\expandafter{\addDOT@{}}\ifx\next@\nextii@
  \let\poly@style@@=\xypolydot@Special
  \else\expandafter\DN@\expandafter{\addDASH@{}}\ifx\next@\nextii@
  \let\poly@style@@=\xypolydash@Special
  \else\DN@{{*}}\ifx\next@\nextii@ \def\poly@style@@{\xypolystraight@{*}}\%
  \else \def\poly@style@@{\xypolystraight@{#1}}\%
  \fi\fi\fi\fi\fi\fi }
\xydef@\xydopoly@style{\expandafter\poly@style@@\expandafter{\xyps@list}}
\xydef@\xynopolystyle@#1#2{\xyundefinedLine@{polyline}{{#1}}@@}
\newcommand{\xydef@}{\xy@polystyle@@=\xypolynospec@
\xydef@\UninstallPolylines@{\UnloadpolySpecials@ \let\xy@polystyle@@=\xypolynospec@}}
\newcommand{\xydef@}{\xyundefinedLine@{#1#2}{\xy@polystyle@@=\xypolynospec@}}
\newcommand{\xydef@}{\xynopolystyle@#1#2{\xyundefinedLine@{polyline}{{#1}}@@}}

When there is no special support this handles the various cases using ordinary thin line types.

\newcommand{\xydef@}{\xypolynospec@#1{\DNii@{#1}%
  \ifx\nextii@\empty\def\poly@style@@{\xypolystraight@{-}}\%
  \else\DN@{.}\ifx\next@\nextii@ \def\poly@style@@{\xypolystraight@{.}}\%
  \else\DN@{-}\ifx\next@\nextii@ \def\poly@style@@{\xypolystraight@{--}}\%
  \else\DN@{*}\ifx\next@\nextii@ \def\poly@style@@{\xypolystraight@{.}}\%
  \else\DN@{?}\ifx\next@\nextii@ \def\poly@style@@{\xypolystraight@{.}}\%
  \else\DN@{{*}}\ifx\next@\nextii@ \def\poly@style@@{\xypolystraight@{*}}\%
  \else \def\poly@style@@{\xypolystraight@{#1}}\%
  \fi\fi\fi\fi\fi\fi }
\newcommand{\xydef@}{\xypolystraight@#1#2{%
  \enter@{\pfromthep@ \basefromthebase@ \cfromthec@}\
  \edef\smapp@@{\s@bot}\csname S@0\endcsname\czeroEdge@\pfromc@\def\smapp@@{0}\
  \smapxy@i{\czeroEdge@\enter@{\pfromthec@}{\connect@0{dir{#1}}\leave@}\leave@ } }
\newcommand{\xydef@}{\xypolystraight@#1#2{\enter@{\pfromthep@ \basefromthebase@ \cfromthec@}\
  \edef\smapp@@{s@bot}\csname S@0\endcsname\czeroEdge@\pfromc@\def\smapp@@{0}{\smapp@@\
  \cmapx@{\czeroEdge@\enter@{\pfromthec@}{\connect@0{dir{#1}}\leave@}\leave@ } }
\newcommand{\UninstallPolylines@}{\UninstallPolyline@{\UninstallPolylineSpecials@ \let\xy@polystyle@@=\xypolynospec@}}

Initially poly-lines are uninstalled, until loaded by a \langle driver \rangle.

\newcommand{\xydef@}{\xy@polystyle@@=\xypolynospec@}
\newcommand{\xydef@}{\UninstallPolylines@{}}
\newcommand{\xynolinewidth@}{
\edef\smap@@{s@bot}\cmapx@{\czeroEdge@\enter@{\pfromthec@}{\connect@0{dir{#1}}\leave@}\leave@ } }

\newcommand{\xydef@}{\xyundefinedLine@{polyline}{{#1}}@@
\fi\fi\fi\fi\fi\fi }
\xydef@\xydopoly@style{\expandafter\poly@style@@\expandafter{\xyps@list}}
\xydef@\xynopolystyle@#1#2{\xyundefinedLine@{polyline}{{#1}}@@}

When there is no special support this handles the various cases using ordinary thin line types.

\newcommand{\xydef@}{\xypolynospec@#1{\DNii@{#1}%
  \ifx\nextii@\empty\def\poly@style@@{\xypolystraight@{-}}\%
  \else\DN@{.}\ifx\next@\nextii@ \def\poly@style@@{\xypolystraight@{.}}\%
  \else\DN@{-}\ifx\next@\nextii@ \def\poly@style@@{\xypolystraight@{--}}\%
  \else\DN@{*}\ifx\next@\nextii@ \def\poly@style@@{\xypolystraight@{.}}\%
  \else\DN@{?}\ifx\next@\nextii@ \def\poly@style@@{\xypolystraight@{.}}\%
  \else\DN@{{*}}\ifx\next@\nextii@ \def\poly@style@@{\xypolystraight@{*}}\%
  \else \def\poly@style@@{\xypolystraight@{#1}}\%
  \fi\fi\fi\fi\fi\fi }
\newcommand{\xydef@}{\xypolystraight@#1#2{%
  \enter@{\pfromthep@ \basefromthebase@ \cfromthec@}\
  \edef\smapp@@{\s@bot}\csname S@0\endcsname\czeroEdge@\pfromc@\def\smapp@@{0}\
  \smapxy@i{\czeroEdge@\enter@{\pfromthec@}{\connect@0{dir{#1}}\leave@}\leave@ } }
\newcommand{\UninstallPolylines@}{\UninstallPolyline@{\UninstallPolylineSpecials@ \let\xy@polystyle@@=\xypolynospec@}}

Initially poly-lines are uninstalled, until loaded by a \langle driver \rangle.

\newcommand{\xydef@}{\xy@polystyle@@=\xypolynospec@}
\newcommand{\xydef@}{\UninstallPolylines@{}}
\newcommand{\xynolinewidth@}{
\edef\smap@@{s@bot}\csname S@0\endcsname\czeroEdge@\pfromc@\def\smapp@@{0}{\smapp@@\
  \cmapx@{\czeroEdge@\enter@{\pfromthec@}{\connect@0{dir{#1}}\leave@}\leave@ } }

\newcommand{\xydef@}{\xyundefinedLine@{polyline}{{#1}}@@
\fi\fi\fi\fi\fi\fi }
\xydef@\xydopoly@style{\expandafter\poly@style@@\expandafter{\xyps@list}}
\xydef@\xynopolystyle@#1#2{\xyundefinedLine@{polyline}{{#1}}@@}

When there is no special support this handles the various cases using ordinary thin line types.

\newcommand{\xydef@}{\xypolynospec@#1{\DNii@{#1}%
  \ifx\nextii@\empty\def\poly@style@@{\xypolystraight@{-}}\%
  \else\DN@{.}\ifx\next@\nextii@ \def\poly@style@@{\xypolystraight@{.}}\%
  \else\DN@{-}\ifx\next@\nextii@ \def\poly@style@@{\xypolystraight@{--}}\%
  \else\DN@{*}\ifx\next@\nextii@ \def\poly@style@@{\xypolystraight@{.}}\%
  \else\DN@{?}\ifx\next@\nextii@ \def\poly@style@@{\xypolystraight@{.}}\%
  \else\DN@{{*}}\ifx\next@\nextii@ \def\poly@style@@{\xypolystraight@{*}}\%
  \else \def\poly@style@@{\xypolystraight@{#1}}\%
  \fi\fi\fi\fi\fi\fi }
\newcommand{\xydef@}{\xypolystraight@#1#2{%
  \enter@{\pfromthep@ \basefromthebase@ \cfromthec@}\
  \edef\smapp@@{\s@bot}\csname S@0\endcsname\czeroEdge@\pfromc@\def\smapp@@{0}\
  \smapxy@i{\czeroEdge@\enter@{\pfromthec@}{\connect@0{dir{#1}}\leave@}\leave@ } }
\newcommand{\UninstallPolylines@}{\UninstallPolyline@{\UninstallPolylineSpecials@ \let\xy@polystyle@@=\xypolynospec@}}

Initially poly-lines are uninstalled, until loaded by a \langle driver \rangle.

\newcommand{\xydef@}{\xy@polystyle@@=\xypolynospec@}
\newcommand{\xydef@}{\UninstallPolylines@{}}
\newcommand{\xynolinewidth@}{
\edef\smap@@{s@bot}\csname S@0\endcsname\czeroEdge@\pfromc@\def\smapp@@{0}{\smapp@@\
  \cmapx@{\czeroEdge@\enter@{\pfromthec@}{\connect@0{dir{#1}}\leave@}\leave@ } }

\newcommand{\xydef@}{\xyundefinedLine@{polyline}{{#1}}@@
\fi\fi\fi\fi\fi\fi }
\xydef@\xydopoly@style{\expandafter\poly@style@@\expandafter{\xyps@list}}
\xydef@\xynopolystyle@#1#2{\xyundefinedLine@{polyline}{{#1}}@@}

When there is no special support this handles the various cases using ordinary thin line types.
2.4. LINE STYLES EXTENSION

Nothing further can be done unless allowed by a specific ⟨driver⟩, which must establish an alternative expansions to be bound to \xylinewidth@@ and the other hooks.

This is a possible alternative for \xylinewidth@@. Indeed it is used with the POSTSCRIPT support, which then redefines just \xylineSpecial@@.

These need to be rebound by a back-end which supports variation in the line-thickness.
CHAPTER 2. EXTENSIONS

\xylet@xylineSpecial@=\eat@

The end & Log

\xyendinput

\% $Log: xyline.doc,v $  
\% Revision 3.10  2011/03/14  20:14:00  krisrose  
\% Preparing for release 3.8.6.  
\%  
\% Revision 3.9  2010/06/10  18:45:50  krisrose  
\% Reference to GPL by URL.  
\%  
\% Revision 3.8  2010/05/06  17:46:30  krisrose  
\% Ross Moore’s e-mail address updated.  
\% Many obsolete files degraded to Historic.  
\%  
\% Revision 3.7  2010/04/16  06:06:52  krisrose  
\% Preparing for a new release...  
\%  
\% Revision 3.6  1998/03/06  01:28:05  ross  
\% adjustments for styles now within the kernel code  
\% fixed problem with line-widths  
\%  
\% Revision 3.3  1996/12/18  09:22:29  ross  
\% Bug fix release.  
\%  
\% Revision 3.2  1995/09/19  18:21:41  ross  
\%  
\% Revision 3.1  1995/09/05  20:36:33  ross  
\%  
\% Revision 3.0  1995/07/07  20:13:19  ross  
\% Major release w/new User’s Guide!  
\%  
\% Revision 2.13  1995/07/04  15:04:51  ross  
\% Ready for release of v3.  
\%  
\% Revision 2.12  1994/10/25  03:01:14  ross  
\% Final 3beta release [bug fixes & AMS-LaTeX fitting].  
\%  
\% Revision 2.11  1994/07/05  09:27:49  ross  
\% fixed documentation bug  
\%  
\% Revision 2.10  1994/06/15  12:46:03  ross  
\% Second release 3beta.  
\% Colour and line style saving works; label colouring bug fixed.  
\%  
\% Revision 2.9  1994/06/09  14:39:49  ross
2.5 Rotate and Scale extension

Vers. 3.8 by Ross Moore (ross.moore@mq.edu.au)

This extension provides the ability to request that any object be displayed rotated at any angle as well as scaled in various ways.

Header:

These are effects which are not normally available within TeX. Instead they require a suitable \special commands, or extra fonts, together with appropriate commands to implement the effects. Thus

Using this extension will have no effect on the output unless used with a backend that explicitly supports it.
The extension provides special effects that can be used with any \texttt{Xy-pic} \texttt{(object)} by defining \texttt{[(shape)]} modifiers. The modification is local to the \texttt{(object)} currently being built, so will have no effect if this object is never actually used.

The following table lists the modifiers that have so far been defined. They come in two types – either a single keyword, or a key-character with the following text treated as a single argument.

<table>
<thead>
<tr>
<th>Modifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{[@]}</td>
<td>align with current direction</td>
</tr>
<tr>
<td>\texttt{[@\langle direction\rangle]}</td>
<td>align to \texttt{(direction)}</td>
</tr>
<tr>
<td>\texttt{[@!\langle number\rangle]}</td>
<td>rotate \texttt{(number)} degrees</td>
</tr>
<tr>
<td>\texttt{[*\langle number\rangle]}</td>
<td>scale by \texttt{(number)}</td>
</tr>
<tr>
<td>\texttt{[*\langle num\rangle_x,\langle num\rangle_y]}</td>
<td>scale \texttt{x} and \texttt{y} separately</td>
</tr>
<tr>
<td>\texttt{[left]}</td>
<td>rotate anticlockwise by 90°</td>
</tr>
<tr>
<td>\texttt{[right]}</td>
<td>rotate (clockwise) by 90°</td>
</tr>
<tr>
<td>\texttt{[flip]}</td>
<td>rotate by 180°; same as \texttt{[*-1,-1]}</td>
</tr>
<tr>
<td>\texttt{[dblsize]}</td>
<td>scale to double size</td>
</tr>
<tr>
<td>\texttt{[halfsize]}</td>
<td>scale to half size</td>
</tr>
</tbody>
</table>

These \texttt{[(shape)]} modifiers specify transformations of the \texttt{(object)} currently being built. If the object has a rectangle edge then the size of the rectangle is transformed to enclose the transformed object; with a circle edge the radius is altered appropriately.

**To Do:** There should be an \texttt{[\@*]} form which repeats the set of transformations applied to the last object which has been transformed, possibly in an earlier diagram. Also an \texttt{[\@=\langle word\rangle]} form allowing a set of transformations to be saved and used later, simply by specifying \texttt{[(\langle word\rangle)]]}.

Each successive transformation acts upon the result of all previous. One consequence of this is that the order of the shape modifiers can make a significant difference in appearance—in general, transformations do not commute. Even successive rotations can give different sized rectangles if taken in the reverse order.

Sometimes this change of size is not desirable. The following commands are provided to modify this behaviour.

\begin{verbatim}
\NoResizing prevents size adjustment
\UseResizing restores size adjustments
\end{verbatim}

\begin{verbatim}
\xydef\NoResizing{\xyNoResizing@}
\xydef\UseResizing{\xyUseResizing@}
\xydef\xyNoResizing@{%
\global\let\origxyrescale@=\xyrescale@
\global\let\origxyrotSizeMod@=\xyrotateSizeMod@
\global\let\xyNoResizing@=\xyNoResizing@@
\global\let\xyUseResizing@=\xyUseResizing@@
\xyNoResizing@@ }
\xydef\xyNoResizing@@{%
\global\let\xyrescale@=\xyignorescale@
\global\let\xyrotateSizeMod@=\xyignoreSizeMod@
}
\xylet\xyUseResizing@=relax
\xydef\xyUseResizing@@{%
\global\let\xyrescale@=\origxyrescale@
\end{verbatim}
The \NoResizing command is also useful to have at the beginning of a document being typeset using a driver that cannot support scaling effects, in particular when applied to whole diagrams. In any case an unscaled version will result, but now the spacing and positioning will be appropriate to the unscaled rather than the scaled size.

Interface  Recognize the (shape) characters and keywords.

Each style modifier adds code to one, or both, of two global lists. These contents of these lists, called \preXYtransform@@ and \postXYtransform@@. An implementation must define macros which are to be bound to the control sequence names \preXYtransform@ and \postXYtransform@, which are prepended and appended to the \Drop@@ and Connect@@ methods for the current ⟨object⟩.

Also \Drop@@ and \Connect@@ must be modified, if this has not already been done as indicated by \@transform@ having expansion \relax.

The transformations must be implemented by emulating the adjoint coordinate transformations. This requires that code be added to \preXYtransform@@ in reverse order to the order of occurrence of the (shape) modifiers. The code is added to \postXYtransform@@ in natural order, so that each addition to \preXYtransform@@ can be closed off, if necessary, in a correctly nested sequence.
Figure 2.4: Rotations, scalings, and flips

Bug: The reference point does not move correctly in POSTSCRIPT when a shift modifier has been applied before a rotation. The object prints outside of its bounding rectangleEdge. Fix this!!

Global macros are used, so that the same styles can be reused by successive objects without having to re-interpret ⟨shape⟩ modifiers, as described next.

Saving transformations Once specified for an ⟨object⟩, the collection of transformations can be assigned a name, via [⟩⟨name⟩]. The ⟨name⟩ here is any collection of letters or other characters up to the closing ]. Subsequent use of [⟩⟨name⟩], with the same or other ⟨objects⟩s, will re-establish the saved transformations, acting on the new ⟨object⟩ with initial reference point appropriate to this ⟨object⟩.

Note: Such namings are global and permanent. They are intended to allow a particular set of transformations to be easily repeated for different objects inside various pictures and diagrams within the same document.

Scaling and Scaled Text The ⟨shape⟩ modifier can contain either a single scale factor, or a pair indicating different factors in the x- and y-directions. Negative values are allowed, to obtain reflections in the coordinate axes, but not zero.
2.5. ROTATE AND SCALE EXTENSION

\xydef\xyscalechars@#1,#2,#3@@%\DN@{\xyrescale@(#1,#2)}%\dimen@=#1\p@\ifdim\dimen@>\z@ \else \ifdim\dimen@<\z@ \else \DN@{\xywarning@{invalid scaling factors #1, #2}}%\fi\fi \dimen@=#2\p@\ifdim\dimen@>\z@ \else \ifdim\dimen@<\z@ \else \DN@{\xywarning@{invalid scaling factors #1, #2}}%\fi\fi \next@ }\xydef\xyrescale@(#1,#2){\setboxz@h{\xyscale@@{#1}{#2}\dimen@=\ifx-#1\R@c\advance\dimen@-\R@c \dimen@=-\dimen@\else#1\L@c \advance\dimen@-\L@c\fi\kern\dimen@\boxz@\kern-\dimen@}\R@c=\if-#1\L@c\else#1\R@c\fi\L@c=\if-#1\R@c\else#1\L@c\fi\U@c=\if-#2\D@c\else#2\U@c\fi\D@c=\if-#2\U@c\else#2\D@c\fi} \xydef\xyignorescale@(#1,#2){\xywarning@{Ignoring rescale: #1,#2}\setboxz@h{\xyscale@@{#1}{#2}\boxz@}}% Some particular scalings.
\xydef\xyshape@halfsize@{\xyrescale@(.5,.5)} \xydef\xyshape@dbsize@{\xyrescale@(2,2)} \xydef\xyshape@flip@{\xyrescale@(-1,-1)} \xydef\xyshape@vflip@{\xyrescale@(1,-1)} \xydef\xyshape@hflip@{\xyrescale@(-1,1)}

Rotation and Rotated Text  Within \[@\ldots\] the \ldots are parsed as a \langle(direction)\rangle locally, based on the current direction. The value of count register \Direction contains the information to determine the requested direction. When no \langle(direction)\rangle is parsed then \[@\] requests a rotation to align with the current direction.

The special sequence \[@!\ldots\] is provided to pass an angle directly to the back-end. The \texttt{Xy-pic} size and shape of the \langle(object)\rangle with \texttt{rectangleEdge} is unchanged, even though the printed form may appear rotated. This is a feature that must be implemented especially by the back-end. For example, using the \texttt{PostScript} back-end, \[@!45\] will show the object rotated by 45° inside a box of the size of the unrotated object.

\xydef\xyrotatechar@{\bgroup\afterDIRECTIONorEMPTY{\xyrotateSizeMod@}{\xyFN@\xySpecialRotate@i}}\xydef\xySpecialRotate@i{\ifx \space@\next \expandafter\DN@\space{\xyFN@\xySpecialRotate@i}\else\ifx @\next \let\next@=\xyrotateSizeMod@\else\addEQ@\ifx \next\expandafter\xySpecialRotate@@\expandafter\addEQ@!\next\DN@!\next@{\egroup\getSpecialRotate@}\else\ifx *\next \DN@*@@{\egroup\transformDrop@}\else \DN@{\xySpecialRotate@@}%}
CHAPTER 2. EXTENSIONS

To Do: Provide example of repeated, named transformation.

Installation Initially there is no support for the actual effects. They will be processed according to size/shape requirements, but the actual rotation or re-sizing is omitted. A \langle\text{driver}\rangle file is required to implement the effects. It need to define alternative expansions to which \texttt{\xyRotate@@#1}, \texttt{\xySpecialRotate@@#1@@}, \texttt{\doSpecialRotate@@#1@@} and \texttt{\xyscale@@#1#2} can be bound.

This adjusts the sizes of \texttt{\L@c}, \texttt{\R@c}, \texttt{\U@c} and \texttt{\D@c} appropriately.

Some particular rotations.

Some particular rotations.
Reflections  Reflections can be specified by a combination of rotation and a flip — either [hflip] or [vflip].

Shear transformations  To Do: Provide the structure to support these; then implement it in PostScript.

The following diagram requires various options to be loaded and installed.

Example  The diagram in figure 2.4 illustrates many of the effects described above as well as some additional ones defined by the color and rotate extensions.

Exercise 2.7:  Suggest the code used by the author to typeset figure 2.4. (p.578)
The actual code is given in the solution to the exercise. Use it as a test of the capabilities of your DVI-driver. The labels should fit snugly inside the accompanying rectangles, rotated and flipped appropriately.

Bug:  This figure also uses colours, alters line-thickness and includes some PostScript drawing. The colours may print as shades of gray, with the line from A to B being thicker than normal. The wider band sloping downwards may have different width and length according to the DVI-driver used; this depends on the coordinate system used by the driver, when ‘raw’ PostScript code is included.

The end & Log

\xyendinput
2.6 Colour extension

Vers. 3.11 by Ross Moore ⟨ross.moore@mq.edu.au⟩

This extension provides the ability to request that any object be displayed in a particular colour. It requires a suitable ‘driver’ option to provide the necessary \special commands to implement the effects. Thus

```
Using this extension will have no effect on the output unless used with a dvi-driver that explicitly supports it.
```

Header:

1  %% $Id: xycolor.doc,v 3.11 2011/03/14 20:14:00 krisrose Exp $
2  %%
3  %% Xy-pic ‘‘Colour extension’’ option.
4  %% Copyright (c) 1993-1996 Ross Moore <ross.moore@mq.edu.au>
5  %%
6  %% This file is part of the Xy-pic package for graphs and diagrams in TeX.
7  %% See the companion README and INSTALL files for further information.
8  %% Copyright (c) 1991-2011 Kristoffer H. Rose <krisrose@tug.org>
9  %%
2.6. COLOUR EXTENSION

<table>
<thead>
<tr>
<th>Colour Name</th>
<th>Colour Name</th>
<th>Colour Name</th>
<th>Colour Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>GreenYellow</td>
<td>Yellow</td>
<td>Goldenrod</td>
<td>Peach</td>
</tr>
<tr>
<td>Dandelion</td>
<td>Apricot</td>
<td>Peach</td>
<td>Orange</td>
</tr>
<tr>
<td>Melon</td>
<td>YellowOrange</td>
<td>Orange</td>
<td>RedOrange</td>
</tr>
<tr>
<td>BurntOrange</td>
<td>Bittersweet</td>
<td>RedOrange</td>
<td>RubineRed</td>
</tr>
<tr>
<td>Mahogany</td>
<td>Maroon</td>
<td>BrickRed</td>
<td>Bittersweet</td>
</tr>
<tr>
<td>Red</td>
<td>OrangeRed</td>
<td>Fuchsia</td>
<td>Mahogany</td>
</tr>
<tr>
<td>WildStrawberry</td>
<td>Salmon</td>
<td>CarnationPink</td>
<td>WildStrawberry</td>
</tr>
<tr>
<td>Magenta</td>
<td>VioletRed</td>
<td>Rhodamine</td>
<td>Magenta</td>
</tr>
<tr>
<td>Mulberry</td>
<td>RedViolet</td>
<td>Fuchsia</td>
<td>Salmon</td>
</tr>
<tr>
<td>Lavender</td>
<td>Thistle</td>
<td>Orchid</td>
<td>Mulberry</td>
</tr>
<tr>
<td>DarkOrchid</td>
<td>Purple</td>
<td>Plum</td>
<td>Lavender</td>
</tr>
<tr>
<td>Violet</td>
<td>RoyalPurple</td>
<td>BlueViolet</td>
<td>DarkOrchid</td>
</tr>
<tr>
<td>Periwinkle</td>
<td>CadetBlue</td>
<td>CornflowerBlue</td>
<td>Blue</td>
</tr>
<tr>
<td>MidnightBlue</td>
<td>NavyBlue</td>
<td>RoyalBlue</td>
<td>Blue</td>
</tr>
<tr>
<td>Blue</td>
<td>Cerulean</td>
<td>Cyan</td>
<td>MidnightBlue</td>
</tr>
<tr>
<td>ProcessBlue</td>
<td>SkyBlue</td>
<td>Turquoise</td>
<td>Blue</td>
</tr>
<tr>
<td>TealBlue</td>
<td>Aquamarine</td>
<td>BlueGreen</td>
<td>ProcessBlue</td>
</tr>
<tr>
<td>Emerald</td>
<td>JungleGreen</td>
<td>SeaGreen</td>
<td>TealBlue</td>
</tr>
<tr>
<td>Green</td>
<td>ForestGreen</td>
<td>PineGreen</td>
<td>Emerald</td>
</tr>
<tr>
<td>LimeGreen</td>
<td>YellowGreen</td>
<td>SpringGreen</td>
<td>Green</td>
</tr>
<tr>
<td>OliveGreen</td>
<td>RawSienna</td>
<td>Sevia</td>
<td>LimeGreen</td>
</tr>
<tr>
<td>Brown</td>
<td>Tan</td>
<td>Gray</td>
<td>OliveGreen</td>
</tr>
<tr>
<td>Black</td>
<td>White</td>
<td></td>
<td>Brown</td>
</tr>
</tbody>
</table>

Figure 2.5: Colour names after \UseCrayolaColors.

All effects defined in the color extension can be implemented using most POSTSCRIPT (driver)s, and some non-POSTSCRIPT ones, loaded using \xyoption{xyps}. This file provides ‘generic’ code which is known to work correctly with most drivers. For driver-specific variations, consult the appropriate (driver) file.

Colours are specified as a ⟨shape⟩ modifier which gives the name of the colour requested. It is
applied to the whole of the current ⟨object⟩ whether this be text, an Xy-pic line, curve or arrow-tip, or a composite object such as a matrix or the complete picture. However some DVI drivers may not be able to support the colour in all of these cases.

\begin{tabular}{ll}
\langle colour name\rangle & use named colour \\
\texttt{newxycolor}\langle name\rangle\{\langle code\rangle\} & define colour \\
\texttt{UseCrayolaColors} & load colour names (shown in figure 2.5) \\
\end{tabular}

If the DVI-driver cannot support colour then a request for colour only produces a warning message in the log file. After two such messages subsequent requests are ignored completely.

\setlength{\parindent}{0ex}
\begin{verbatim}
\newdef\xycolorwarning@@{\xycolormessage@
\gdef\xycolorwarning@{\xycolormessage@
\xywarning@{...no further colour warnings will be given.}}%
\global\let\xycolorwarning@=\relax }
\xylet@\xycolorwarning@=\xycolorwarning@@
\newdef\xycolormessage@{\xywarning@{Current driver does not support colour.}}%
\xydef@\xycolormessage@{\%
\xywarning@{Current driver does not support colour.}}%
\end{verbatim}

There are two methods, perhaps used simultaneously, for handling requests for colour. The \texttt{\xylocalColor@} is the normal method, colouring whatever objects \TeX typesets from fonts. The more specialized \texttt{\xycolor@} is used with the \texttt{PostScript} back-end. This is necessary since some ⟨driver⟩s assume that the current colour should be ‘black’ when the contents of a \texttt{\special} is interpreted, e.g. within \texttt{PostScript}.

Thus a ⟨driver⟩ which supports both colour and \texttt{PostScript} must define a method to handle \texttt{\xylocalColor@} but should leave alone \texttt{\xycolor@}. The latter method will be adjusted appropriately when a \texttt{PostScript} ⟨driver⟩ is installed.

\setlength{\parindent}{0ex}
\begin{verbatim}
\newdef\xyNoColor@#1{}%
\xydef@\xycolor@@=\xyNoColor@
\newdef\xycolor@={% \\
\let\xylocalColor@=\xyNoColor@@ \\
\let\xycolors@=\xyNoColor@@ \\
\let\xycolorwarning@=\xycolorwarning@@}
\xydef@\xyNoColor@@#1#2{\xycolorwarning@}
\newdef\xycolor@={\xycolor@@}
\xylet@\xycolor@=\xyNoColor@@
\newdef\xycolor@={% \\
\let\xylocalColor@=\xyNoColor@@ \\
\let\xycolor@=\xyNoColor@@ \\
\let\xycolorwarning@=\xycolorwarning@@}
\xylet@\xycolor@=\UnLoadColor@@
\xyaddunsupport{color}{\UnLoadColor@@\relax}
\end{verbatim}

This next macro is to inherit a default colour from the surrounding document, perhaps using one already defined by \texttt{I\TeX2ε}.

\setlength{\parindent}{0ex}
\begin{verbatim}
\ifx\current@color\undefined \xydef@\xycolor@={\current@color}\fi
\else \xydef@\xycolor@={\current@color}\fi
\%\xydef@\xycm@th@={\hbox\bgroup\dimen@=.55ex \checkxycolor@ \xinside@}
\%\xydef@\xcn@m@th@={\hbox\bgroup\dimen@=\z@ \checkxycolor@ \xinside@}
\%\xydef@\checkxycolor@={\ifx\xycolor@\empty\else
\% \expandafter\imposexycolor@\fi}
\%\xydef@\imposexycolor@={% \\
\end{verbatim}
A commonly used method is to maintain a colour-stack. The following commands are for use with such a setup, however they should only be called from code installed from a driver-file. Different drivers may do things differently; e.g. dvips uses a single \special via: \def\xycolor@push@@#1{\special{color push #1}}\ix butTextures uses two: \xdef\xycolor@push#1{\special{color push}\special{color #1}}. Since the colour-stack idea was proposed by Tomas Rokicki, the dvips method is the default behaviour.

\def\xystackcolor@{
\DN@##1##2{
\DN@##1##2{\egroup}
\DN@##1##2{\特种}
\DN@##1##2{\特种}{\no@}{\ifnextii\Connect@@}{\else\DN@{\xystackcolor@i}{\fi}{\fi}{\next@}}}
\DN@{#2}{\ifx\next@\empty}
\DN@{#1}{\DN@{\xycolor@push(#1)}{\DN@{\xycolor@pop}}}\ix\DN@{#2}{\DN@{\xycolor@push(#2 #1)}{\DN@{\xycolor@pop}}}{\fi}{\toks@={\egroup}{\let\xy@style@=\relax}{\def\Drop@@}}{\expandafter\addtotoks@}{\expandafter{\next@}{\def\Connect@@}}{\expandafter\addtotoks@}{\expandafter{\next@}}{\the\toks@}
\DN@{#2}{\DN@{\xycolor@push@@#1}{\special{color push #1}}}
\DN@{#2}{\DN@{\xycolor@pop}{\special{color pop}}}
\DN@{#2}{\DN@{\xycolor@push}{\special{color push}}}

Named colours and colour models New colour names are created with \newxycolor, taking two arguments. Firstly a name for the colour is given, followed by the code which will ultimately be passed to the output device in order to specify the colour. If the current driver cannot support colour, or grayscale shading, then the new name will be recognised, but ignored during typesetting.

\DN@{#2}{\DN@{\newxycolor#1#2}{\newxycolor{#1}{#2}{#3}{#4}{#5}}}
\DN@{#2}{\DN@{\newxycolor#1#2#3#4}{\newxycolor{#1}{#2}{#3}{#4}{#5}}}
\DN@{#2}{\DN@{\newxycolor#1#2#3#4#5}{\newxycolor{#1}{#2}{#3}{#4}{#5}}}

For PostScript devices, the Xy-ps PostScript dictionary defines operators rgb, cmyk and gray corresponding to the standard RGB and CMYK colour models and grayscale shadings. Colours and shades are described as: rgb or cmyk or s gray, where the parameters are numbers in the range 0 ≤ r, g, b, c, m, y, k, s ≤ 1. The operators link to the built-in colour models or, in the case of cmyk for earlier versions of PostScript, give a simple emulation in terms of the RGB model.

Existing color names can be overridden using \newxycolor@#1#2\relax.
Standard colour names which are defined automatically correspond to the three primaries red, green, blue and their complements cyan, magenta, yellow as well as three extreme grayscale shades black, white, gray.

Note: The driver file must call \texttt{xystandardcolors@} before these colours become accessible. This is to allow any \texttt{(driver)-specific} definitions to be active when these colours are defined.

{}\begin{verbatim}
\def\newxystyle{\xylocalColor@{}{#3}\xycolor@{#3}}%
\else
\DN@{\newxystyle{#1}{\xylocalColor@{#2}{#3}\xycolor@{#2 #3}}}%
\else \DN@{}
\fi \next@elax}
\newxystyle{\xylocalColor@{black}{grey}{}\relax}{% Textures reverses gray-scales
\newxystyle{white}{0.}{gray}{}\relax % This is normal for PostScript
\newxystyle{black}{0.0.0.}{rgb}{} % avoid the hassles.
\newxystyle{white}{1.1.1.}{rgb}{} %
\newxystyle{gray}{.5}{gray}{}\relax
\newxystyle{grey}{.5}{gray}{}\relax
\newxystyle{red}{1.0.0.}{rgb}{}\relax
\newxystyle{green}{0.1.0.}{rgb}{}\relax
\newxystyle{blue}{0.0.1.}{rgb}{}\relax
\newxystyle{cyan}{0.1.1.}{rgb}{}\relax
\newxystyle{magenta}{1.0.1.}{rgb}{}\relax
\newxystyle{yellow}{1.1.0.}{rgb}{}\relax
\xuncatcodes }
\end{verbatim}

Note: The driver file must call \texttt{xystandardcolors@} before these colours become accessible. This is to allow any \texttt{(driver)-specific} definitions to be active when these colours are defined.

Saving colour and styles When styles are saved using [+\texttt{(word)}], see , then the current colour setting (if any) is saved also. Subsequent use of [\texttt{(word)}] recovers the colour and accompanying line-style settings.

Further colour names are defined by the command \texttt{\UseCrayolaColours} that loads the \texttt{crayon} option, in which more colours are defined. Consult the file \texttt{xyps-col.doc} for the colours and their specifications in the RGB or CMYK models.

\texttt{\UseCrayolaColours\xyspec{\texttt{xyps-col}}\xuncatcodes}\%
This option provides the command to install definitions for the 68 colours recognised by name by Tomas Rokicki's dvips driver \cite{Rokicki1991}. This command must be called from a ⟨driver⟩-file which can actually support the colours.

The colour definitions themselves are in the file \texttt{xyps-col.tex}. They are not loaded directly here since it is not known which ⟨driver⟩ is to be used. Different ⟨driver⟩s may redefine the \texttt{\xynewcolor@} macro to set the colour by name or colour-model.
CHAPTER 2. EXTENSIONS

% Ross Moore's e-mail address updated.
% Many obsolete files degraded to Historic.
% Revision 3.4  2010/04/16 06:06:52 krisrose
% Preparing for a new release...
% Revision 3.3  1996/12/18 08:58:16 ross
% checked in with -k by krisrose at 1996/12/18 14:17:11
% Revision 3.3  1996/12/18 08:58:16 ross
% cosmetic updates
% Revision 3.1  1995/09/05 20:36:33 ross
% Release!
% Revision 3.0  1995/07/07 20:13:19 ross
% Major release w/new User's Guide!
% Revision 2.13  1995/07/05 07:58:43 ross
% Ready for v3 release?
#
% NEW for version 3.1 by Ross Moore  1995/03/18.
%

POSTSCRIPT colour  When POSTSCRIPT driver support is available, the method for handling colour within diagrams may be different. The reason for this is that some dvi-drivers reset the colour to black before placing the contents of the \special. To counter this, the POSTSCRIPT code maintains its own colour-stack. Support is established within a separate file xyps-c.tex, loaded from the appropriate ⟨driver⟩-files.

\xydef@\xyBEcolorcheck@{%if\xycolor@\xyNoColor@\else
\xywarning@{PostScript colour support should be OK.}\fi }%
\xywithoption{ps}{%
\xyinputorelse@{xyps-c}{\xyerror@{Could not load xyps-c}{}{}}%}
\xycatcodes

The end & Log

\xyendinput
% $Log: xycolor.doc,v $
% Revision 3.11  2011/03/14 20:14:00 krisrose
% Preparing for release 3.8.6.
% Revision 3.10  2010/06/10 18:45:50 krisrose
% Reference to GPL by URL.
% Revision 3.9  2010/05/21 15:33:51 krisrose
% Table layout.
2.6. COLOUR EXTENSION

% Revision 3.8 2010/05/17 23:29:21 krisrose
% Experiment: generate all the Type1 fonts with METAPOST.
%
% Revision 3.7 2010/05/14 01:12:16 krisrose
% Figure fixes.
%
% Revision 3.6 2010/05/14 00:22:18 krisrose
% Manual fixes.
%
% Revision 3.5 2010/05/06 17:46:29 krisrose
% Ross Moore's e-mail address updated.
% Many obsolete files degraded to Historic.
%
% Revision 3.4 2010/04/16 06:06:52 krisrose
% Preparing for a new release...
%
% Revision 3.3 1996/12/19 03:50:08 ross
% Maintenance release.
%
% Revision 3.3 1996/12/18 08:56:49 ross
% minor updates
%
% Revision 3.2 1995/09/19 18:20:20 ross
% Bug fix release.
%
% Revision 3.1 1995/09/05 20:36:33 ross
% Release!
%
% Revision 3.0 1995/07/07 20:13:19 ross
% Major release w/new User's Guide!
%
% Revision 2.14 1995/07/06 02:56:02 kris
% Buglets...
%
% Revision 2.13 1995/07/04 15:04:51 ross
% Ready for release of v3.
%
% Revision 2.10 1994/06/15 12:46:03 ross
% Second release 3beta.
% Colour and line style saving works; label colouring bug fixed.
%
% Revision 2.9 1994/06/09 14:39:49 ross
% Release 3beta.
%
% NEW for version 2.9 by by Ross Moore.
2.7 Pattern and Tile extension

Vers. 3.8 by Ross Moore (ross.moore@mq.edu.au)

This extension provides the ability to request that a filled region be tiled using a particular pattern. This is an effect not normally available within TeX. Instead it requires a suitable \textit{driver} option to provide the necessary \texttt{special} commands, together with any extra commands needed to implement the effects. Thus

Using this extension will have no effect on the output unless used with a dvi-driver that explicitly supports it.

All effects defined in the \texttt{tile} extension can be implemented using most \texttt{PostScript} (driver)s, loaded as \texttt{\xyoption{(driver)}}.

\section*{Header:}

\begin{verbatim}
%% $Id: xytile.doc,v 3.8 2011/03/14 20:14:00 krisrose Exp $
%% Xy-pic ``Pattern and Tile extension'' option.
%% Copyright (c) 1993-1997 Ross Moore <ross.moore@mq.edu.au>
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-2011 Kristoffer H. Rose <krisrose@tug.org>
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License
%% for more details.
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, see http://www.gnu.org/licenses/.
\if\xyloaded\undefined \input xy \fi
\xyprovide{tile}{Pattern and Tile extension}{\stripRCS$Revision: 3.8 $}
{Ross Moore}{ross.moore@mq.edu.au}
{Mathematics Department, Macquarie University, NSW~2109, Australia}
\end{verbatim}

Like using color, tiling with patterns is an instance of a special style, since use of a pattern has no effect on the size or shape of the \langle object\rangle being typeset. It only affects how the \langle object\rangle will appear on the printed page or on-screen.

\textbf{Patterns} Patterns are specified as a \langle shape\rangle modifier, similar to the way colours are specified by name. The pattern is applied to the whole of the current \langle object\rangle whether this be text, an \texttt{Xy-pic} line,
curve or arrow-tip, or a composite object such as a matrix or the complete picture. However some
DVI-drivers may not support use of patterns in all cases.

If the current DVI-driver cannot support patterns then a request for one simply produces a warning
message in the log file. After two such messages subsequent requests are ignored completely.

```
\xdef\xypatternwarning@@{\xypatternmessage@}
\gdef\xypatternwarning@{\xypatternmessage@}
\warning@{...no further pattern warnings will be given.}\
\global\let\xypatternwarning@=\relax
\xylet@\xypatternwarning@=\xypatternwarning@@
\xdef\xypatternmessage@{\warning@{Current driver does not support patterns.}}
\warning@{Current driver does not support patterns.}}
```

This allows new patterns to be specified from raw data. Make sure the usual catcodes are in place
before the data is tokenised.

```
\xdef\newxypattern{\begingroup\xycatcodes\newxypattern@i}
\xdef\newxypattern@i#1#2{\edef\next@{\endgroup
\noexpand\newxyPattern{#1}{#2}{#3}{#4}}\next@}
\xdef\newxyPattern#1#2#3#4{\newxypattern@{#1}{#2}{#3}{#4}}
```

Existing pattern names can be overridden using \newxypattern@#1#2\relax.

```
\xdef\newxypattern@#1#2#3#4#5{%
\expandafter\let\expandafter\next@\csname shape [#1]\endcsname
\ifx\next@\relax
\DN@{#4}\ifx\next@\empty
\DN@{#2}\ifx\next@\empty
\DN@{\newxystyle{#1}{\xylocalpattern@{}{#3}\xypattern@{#3}}}%
\else
\DN@{\newxystyle{#1}{\xylocalpattern@{#2}{#3}\xypattern@{#2 #3}}}%
\fi
\else\DN@{#2#3}\ifx\next@\empty
```
Although pattern data may be specified directly using \texttt{\newxypattern}, it is more usual to load it from a \langle file\rangle in which many patterns are defined by name, each on a separate line. By convention such files always end in \texttt{.xyp} (XY-pattern) so no extension should be specified. The pattern is then requested using either the name supplied in the file or by an alias. Once \texttt{\UsePatternFile} has been used, then a null \langle file\rangle argument to the other commands will still find patterns in the default file. The default remains in effect for the current level of \LaTeX grouping.

For example, the following picture

\begin{verbatim}
\ AliasPattern{bricks}{mac12}{xymacpat}
\ AliasPattern{bars}{mac08}{xymacpat}
\ xy **<5pc,3.1pc>{},*{[bricks]}\frm{**}
  ,**<2.5pc>[o]{},*{[bars]}\frm{**}
\end{verbatim}

uses ‘filled’ frames from the \texttt{\frame} feature:
2.7. PATTERN AND TILE EXTENSION

There are two methods, perhaps used simultaneously, for handling requests for a pattern. The `\localpattern` is the normal method, patterning whatever objects \TeX{} typesets from fonts or rules. The more specialized `\pattern` is used with a PostScript \langle driver \rangle.

Thus a \langle driver \rangle which supports both pattern and PostScript must define a method to handle `\localpattern` but should leave alone `\pattern`. The latter method will be adjusted appropriately when a PostScript \langle driver \rangle is installed.

```latex
\def\xyNopattern@#1{}
\let\xylocalpattern@=\xyNopattern@
\let\xypattern@=\xypattern@@
\let\xylocalpattern@=\xyNopattern@@
```

This is to facilitate unloading patterns when the \langle driver \rangle is changed to one that does not support this feature. This is the default state, until an appropriate \langle driver \rangle is loaded.

Pattern data  
A region is tiled using copies of a single ‘cell’ regularly placed so as to seamlessly tile the entire region. The \langle data \rangle appearing as an argument to `\newxypattern` is ultimately passed to the dvi-driver.

The simplest form of pattern data is: \langle num \rangle \langle Hex-data \rangle, where the data is a 16-character string of Hexadecimal digits; i.e. 0–9, A–F. Each Hex-digit equates to 4 binary bits, so this data contains 64 bits representing pixels in an 8 × 8 array. The \langle num \rangle is an integer counting the number of ‘0’s among the 64 bits. Taken as a fraction of 64, this number or its complement, represents the average density of ‘on’ pixels within a single cell of the pattern. Drivers unable to provide the fine detail of a pattern may simply use this number, or its complement, as a gray-level or part of a colour specification for the whole region to be tiled.

A desirable set of standard patterns could be specified to be always loaded. If such a set emerges indeed as being frequently desirable then it will be included here.

```latex
\def\xystandardpatterns@{}%
\new@{read}{\xypatread@}
\def\defaultpattfile@{}
\def\xygetfilepatterns@#1{\DN@{#1}%
  \ifx\next@\empty
    \ifx\defaultpattfile@\empty \xywarning@{No pattern file specified.}%
    \else \DN@{\expandafter\openpattfile@\expandafter{\defaultpattfile@}}\fi
  \else \DN@{\openpattfile@{#1}}\fi \next@ }
\def\openpattfile@#1{\openin\xypatread@=#1.xyp %
  \ifeof\xypatread@\xywarning@{No patterns in file: #1.xyp}%
  \closein\xypatread@
```
The character ! is used to delimit comments within pattern-data files.

The file `xymacpat.xyp` contains defining data for the 38 standard patterns available with the Macintosh Operating system. Figure 2.6 displays all these patterns.

Rotating and Resizing Patterns Some implementations of patterns are sufficiently versatile to allow extra parameters to affect the way the pattern data is interpreted. PostScript is one such implementation in which it is possible to rotate the whole pattern and even to expand or contract the sizes of the basic cell.

Due to the raster nature of output devices, not all such requests can be guaranteed to produce aesthetic results on all devices. In practice only rotations through specific angles (e.g. 30°, 45°, 60°) and particular scaling ratios can be reliably used. Thus there is no sophisticated interface provided by XY-pic to access these features. However the ‘PostScript escape’ mechanism does allow a form of
access, when a POSTSCRIPT (driver) is handling pattern requests.

Special POSTSCRIPT operators \texttt{pa} and \texttt{pf} set the pattern angle (normally 0) and ‘frequency’ measured in \textit{cells per inch}. Hence, when used as an \langle \text{object} \rangle-modifier, \texttt{[! 30 pa 18.75 pq]} rotates the pattern by 30° clockwise and uses a smaller pattern cell (larger frequency). The default frequency of 12.5 = 300/(8 \times 3) means that each pixel in a pattern cell corresponds, on a device of resolution 300dpi, to a 3 \times 3 square of device pixels; on such a device 18.75 uses 2 \times 2 squares.

At 300dpi a frequency of 9.375 = 300/(8 \times 4) uses 4 \times 4 squares. These match the natural size for pixels on a 75dpi screen and are pretty close for 72dpi screens. Though appropriate for screen displays, these are ‘too chunky’ for high quality printed work. Doubling the frequency is too fine for some patterns, hence the intermediate choice of 12.5 as default. In order for printed output to match the screen view, a POSTSCRIPT operator \texttt{macfreq} has been defined to facilitate requests for 9.375, via \texttt{[!macfreq]}.

The next diagram displays changes to the frequency.

\begin{verbatim}
9.375 12.5 18.75 37.5
\end{verbatim}

\textbf{Saving patterns:} When styles are saved using \langle \text{word} \rangle, see note 1.4k of §1.4, then the current pattern (if any) is also saved. Subsequent use of \langle \text{word} \rangle recovers the pattern as well as colour and line-style settings. This includes any explicit variations applied using the “Style Escape” mechanism.

Here is a variation of an earlier example, with extra effects.

\begin{verbatim}
\UsePatternFile{xymacpat}\\\AliasPattern{bricks}{mac12}{}\\\LoadPattern{mac28}{}\\\LoadPattern{mac05}{}\\\xy *=0[! macfreq -45 pa][mac28][|=Bars]{}\\**,<12pc,4pc>[|][bricks][frm{**}]\\,-<3.5pc,0pt>,**,<2.65pc>[o],*[Bars][frm{**}]\\,*[thicker][frm{o},<6pc,0pt>]\\,**<5pc,2.7pc>[|][mac05][frm{**}]*,frm{-,}\\,*[white]\txt\Large\bf\sf{Kilroy\ was\ here}\\\endxy
\end{verbatim}

Add this extension to the driver-tables as “unload”, unless an already-loaded driver can support it:

\begin{verbatim}
\DN@{\xyaddunsupport{tile}\Unloadpattern@}\\\ifx\xyeverywithoption@tile@undefined\\\else\ifx\xyeverywithoption@tile@empty\\\else\\\DN@{\xysetup@@{\xywithoption{tile}@\xydriversloaded@@}}\\\fi\fi\next@
\end{verbatim}
2.8 Import graphics extension

Vers. 3.13 by Ross Moore (ross.moore@mq.edu.au)

This feature provides the ability to easily add labels and annotations to graphics prepared outside \TeX\ or \LaTeX. An \texttt{XY-pic} graphics environment is established whose coordinates match that within the contents of the imported graphic, making it easy to specify exactly where a label should be placed, or an arrow drawn to highlight a particular feature.

Header:

\begin{verbatim}
\%% $Id: xyimport.doc,v 3.13 2011/03/14 20:14:00 krisrose Exp $
\end{verbatim}
A command \texttt{\textbackslash xyimport} is defined which is used, in conjunction with imported graphics, to establish a coordinate system appropriate to the particular graphics. This enables \texttt{\langle pos\rangle}itions within the graphic to be easily located, either for labelling or adding extra embellishing features. It is used in either of the follow ways:

\begin{verbatim}
\xyimport(width,height){\langle graphic\rangle}
\xyimport(width,height)(x-off,y-off){\langle graphic\rangle}
\end{verbatim}

Normally the \texttt{\langle graphic\rangle} will be a box containing a graphic imported using the commands from packages such as \texttt{\textbackslash graphics}, \texttt{\textbackslash epsf} or \texttt{\textbackslash epsfig}, or using other commands provided by the local \TeX{} implementation. However the \texttt{\langle graphic\rangle} could be \textit{any} balanced \TeX{} material whatsoever; provided it occupies non-zero size, both vertically and horizontally.

The \texttt{width} and \texttt{height} are \texttt{\langle number\rangle}s given in the coordinate system for the \texttt{contents of the \langle graphic\rangle}. These are not dimensions, but coordinate-lengths, using the units appropriate to the picture displayed by \texttt{\langle graphic\rangle}.

When provided, \texttt{\langle x-off,y-off\rangle} give the distance in coordinate units from bottom-left corner to where the origin of coordinates should be located, usually within area covered by the \texttt{\langle graphic\rangle}. Usually the negatives of these numbers will give the coordinate location of the bottom-left corner of the \texttt{\langle graphic\rangle}. If no offsets are supplied then the origin is presumed to lie at the bottom-left corner.
CHAPTER 2. EXTENSIONS

Framed contents of graphics file.

Rational points on the elliptic curve:
\[ x^3 + y^3 = 7 \]

Figure 2.7: Importing a graphic for labelling.

```
\xydef@{\xyextern@(#1,#2){\hbox\bgroup\R@c=#1\p@ \U@c=#2\p@\xyextern@i}}
\xydef@{\xyextern@i#1#{\DN@{#1}\
\ifx\next@\empty \L@c=\z@ \D@c=\z@ \DN@{\xyextern@x}\%\
\else \DN@{\xyextern@ii#1}\fi \next@ }}
\xydef@{\xyextern@ii(#1,#2){\L@c=#1\p@ \D@c=#2\p@\xyimportdefault@x}}
\xydef@{\xyimportdefault@{3}}
\xydef@{\xyexportwarning@#1#2#3{\xywarning@{\%\
#1 for import has zero \#2; using \xyimportdefault@#3 default}}}\n\xydef@{\xyextern@x#1{\toks@={\egroup\L@c=}\setboxz@h{#1}\
\dimen@ii=\L@c \advance\dimen@ii\R@c \dimen@=\wdz@\n\ifdim\dimen@=\z@ \xyexportwarning@{graphic}{width}{cm}\
\dimen@=\xyimportdefault@ cm\fi\n\ifdim\dimen@ii=\z@ \xyexportwarning@{coords}{width}{unit}\
\dimen@ii=\xyimportdefault@ cm\fi\n\dimen@=\ht\z@ \advance\dimen@\dp\z@ \n\ifdim\dimen@=\z@ \xyexportwarning@{graphic}{height}{cm}\
\dimen@=\xyimportdefault@ cm\fi\n\ifdim\dimen@ii=\z@ \xyexportwarning@{coords}{height}{unit}\
\dimen@ii=\xyimportdefault@ cm\fi\n\ifdim\dimen@=\ht\z@ \advance\dimen@\dp\z@ \n\ifdim\dimen@ii=\z@ \xyexportwarning@{graphic}{depth}{cm}\
\dimen@ii=\xyimportdefault@ cm\fi
```

Normally the \texttt{\textbackslash xyimport} command is used at the beginning of an \texttt{\textbackslash xy..\textbackslash endxy} environment. It is not necessary to give any basis setup, for this is deduced by measuring the dimensions of the \texttt{\langle graphic\rangle} and using the supplied \textit{width}, \textit{height} and offsets. The \texttt{\langle graphic\rangle} itself defines named \texttt{\langle pos\rangle} called "\texttt{import}" , located at the origin and having appropriate extents to describe the area covered by the \texttt{\langle graphic\rangle}. This makes it particularly easy to surround the \texttt{\langle graphic\rangle} with a frame, as on the left side of figure 2.7, or to draw axes passing through the origin.

Here is the code used to apply the labelling in figure 2.7:

\begin{verbatim}
\def\ellipA{\resizebox{6cm}{!}{\includegraphics{import1.eps}}}
\xy
\xyimport(3.7,3.7)(1.4,1.4){\ellipA}*rm{-}
,(!D+<2pc,-1pc>**!U)*+!U{\text{Framed contents of graphics file.}}
\endxy
\quad
\xy
\xyimport(3.7,3.7)(1.4,1.4){\ellipA}
,(!D+<2pc,-1pc>**!U)*+!U{\text{Rational points on the elliptic curve: } x^3+y^3=7}$
,(1,0)**!U{\langle 1 \rangle},(-1,0)**!U{\langle -1 \rangle}
,(0,1)**!R{\langle 1 \rangle},(0,-1)**!R{\langle -1 \rangle}
,(2,-1)**!RU{\langle P \rangle},(-2,-1)**!RU{\langle -P \rangle}
,(1.3333,1.6667)**!UR{\langle -2P \rangle}
,(1.6667,1.3333)**!DL{\langle -2P \rangle}
,(-.5,1.9)**!DL{\langle 3P \rangle},(1.9,-.5)**!DL{\langle -3P \rangle}
,(-1,2.3)**!DL{\langle \infty \rangle}@*0{},\ar{(-2,2)}
,(5.2,3.3)**!DL{\langle \infty \rangle}@*0{},\ar{(-2,2)}
,(2.3,-1)**!DL{\langle \infty \rangle}@*0{},\ar{(2.2,-2)}
\endxy
\end{verbatim}

This example uses the $\LaTeX$ standard graphics package to import the graphics file \texttt{import1.eps}; other packages could have been used instead. e.g. \texttt{epsfig, epsf}, or the \texttt{\picture} or \texttt{\illustration} commands in Textures on the Macintosh.

The only possible problems that can occur are when the graphics package is loaded after $\texttt{\textbackslash xy-pic}$ has been loaded. Generally it is advisable to have $\texttt{\textbackslash xy-pic}$ loading \textit{after} all other macro packages.
CHAPTER 2. EXTENSIONS

% $Log: xyimport.doc,v $
% Revision 3.13  2011/03/14 20:14:00  krisrose
% Preparing for release 3.8.6.
%
% Revision 3.12  2010/06/10 18:45:50  krisrose
% Reference to GPL by URL.
%
% Revision 3.11  2010/05/14 01:12:16  krisrose
% Figure fixes.
%
% Revision 3.10  2010/05/06 17:46:30  krisrose
% Ross Moore's e-mail address updated.
% Many obsolete files degraded to Historic.
%
% Revision 3.9  2010/05/04 08:23:00  krisrose
% Updating documentation to use dvipdfmx.
%
% Revision 3.8  2010/04/26 05:56:57  krisrose
% Link fixes in progress...
%
% Revision 3.7  2010/04/16 06:06:52  krisrose
% Preparing for a new release...
%
% Revision 3.6  1998/03/06 01:28:05  krisrose
% Releasing (with Y&Y fonts).
%
% Revision 3.4  1997/05/18 01:13:24  ross
% Essential bugfixes.
%
% Revision 3.3  1996/12/18 09:19:22  ross
% no changes
%
% Revision 3.1  1995/09/05 20:36:33  ross
% Release!
%
% Revision 3.0  1995/07/07 20:13:19  ross
% Major release w/new User’s Guide!
%
% Revision 2.14  1995/07/05 22:10:51  kris
% Buglets...
%
% Revision 2.13  1995/07/04 15:04:51  ross
% Ready for release of v3.
%
% NEW for version 3.0 by by Ross Moore.
2.9 Movie Storyboard extension

Vers. 3.9 by Kristoffer H. Rose (krisrose@tug.org)

This extension interprets the \scene primitive of the movie class, setting the progress indicators to dummy values. The following assumes that you are familiar with the movie class.

Header:

The size of the frame is determined by the command

\MovieSetup{width=\textit{width},height=\textit{height},...}

(the ... indicate the other arguments required by the movie class but silently ignored by the Xy-pic movie extension).

Note: This extension still experimental and subject to change. The only documentation is in the movie.cls source file.

The implementation mimics the aspects of the keyval package needed here, namely reading the height and width parameters.
The progress macros all default to
\xydef@\theScene{0}
\xydef@\theF{0}
\xydef@\theFr{1}
\xydef@\F#1{#1(0)}
\xydef@\Fr#1{#1(1)}

We need to interpret and ignore the arguments to \scene except the * which is used on the last scene since this means that we should output the scene with progress values corresponding to “The End”.
\xynew@if\iflastframe@
\xynew@{count}\m@scene
\DN@{\ifx*\next \lastframe@true \DN@{*{\xyFN@\scene@}}
\else \let\next@=\scene@ \fi \next@}
\xyFN@\next@
\xydef@\scene@{%
\ifx[\next \DN@[##1]{\scene@i}\else \let\next@=\scene@i \fi \next@}
\long\def\scene@i#1{{%
% \gdef\m@caption{Scene \the\m@scene. #1}}%
\edef\theScene{\the\m@scene}%
\def\theF{0}\def\theFr{1}%
\def\F##1{##1(0)}\def\Fr##1{##1(1)}%
\scene@frame{#1}%
%
\iflastframe@
\gdef\m@caption{The End.}%
\def\theF{1}\def\theFr{0}%
\def\F##1{##1(1)}\def\Fr##1{##1(0)}%
\scene@frame{#1}%
\fi
}\ignorespaces

\def\scene@frame#1{\vbox{\null
\dimen@=\m@width \advance\dimen@2\xydashw@
\hrule width\dimen@
\hbox to\dimen@{\hrule width\xydashw@ \vfil}
\vbox to\m@height{\hsize=\m@width \null\vfil#1}\vfil\null}%
\vfilnull%}
\hfill\vfilnull%}
\hrule width\xydashw@%}
\vfilnull%}
\hrule width\dimen@
\setbox0=\hbox{\null
2.10. POSTSCRIPT BACKEND

XY-ps is a ‘back-end’ which provides XY-pic with the ability to produce DVI files that use POSTSCRIPT \specials for drawing rather than the XY-pic fonts.

In particular this makes it possible to print XY-pic DVI files on systems which do not have the ability to load the special fonts. The penalty is that the generated DVI files will only function with one particular DVI driver program. Hence whenever XY-ps is activated it will warn the user:

```
XY-pic Warning: The produced DVI file is not portable: It contains POSTSCRIPT \specials for (one particular) driver
```

A more complete discussion of the pros and cons of using this backend is included below.

Header:
```
%% $Id: xyps.doc,v 3.12 2011/05/27 04:51:17 krisrose Exp $
```
2.10.1 Choosing the DVI-driver

Including \texttt{\textbf{\textbackslash xyoption\{ps\}}} within the document preamble, tells Xy-pic that the \texttt{POSTSCRIPT} alternative to the fonts should be used, provided the means to do this is also specified. This is done by also specifying a dvi-driver which is capable of recognising and interpreting \texttt{\textbackslash special} commands. Although the file \texttt{xyps.tex} is read when the option request is encountered, the macros contained therein will have no effect until an appropriate driver has also been loaded.

With \LaTeXe both the backend and driver may be specified, along with other options, via a single \texttt{\textbf{\usepackage}} command, see [4, page 317]; e.g.

\begin{verbatim}
\usepackage[ps,textures,color,arrow]{xy}
\end{verbatim}

The rebindings necessary to support \texttt{POSTSCRIPT} are not effected until the \texttt{\begin{document}} command is encountered. This means that an alternative driver may be selected, by another \texttt{\textbf{\textbackslash xyoption\{(driver)\}}}, at any time until the \texttt{\begin{document}}. Only the macros relevant to last named \texttt{(driver)} will actually be installed.

The following table describes available support for \texttt{POSTSCRIPT} drivers. Please consult the individual driver sections in part 4 for the exact current list. For each \texttt{(driver)} there is a corresponding file named \texttt{xy\{\textbackslash (driver)\}.tex} which defines the necessary macros, as well as a documentation file named...
2.10. POSTSCRIPT BACKEND

xy⟨driver⟩.doc. The spelling is all lower-case, designed to be both descriptive and unique for the 1st 8 characters of the file names.

<table>
<thead>
<tr>
<th>⟨driver⟩</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dvips</td>
<td>Tomas Rokicki’s DVIPS</td>
</tr>
<tr>
<td>dvips</td>
<td>Karl Berry’s dvipsk</td>
</tr>
<tr>
<td>dvips</td>
<td>Thomas Kiffe’s dvips for Macintosh</td>
</tr>
<tr>
<td>textures</td>
<td>Blue Sky Research’s TEXTURES v1.7+</td>
</tr>
<tr>
<td>16textures</td>
<td>Blue Sky Research’s TEXTURES v1.6</td>
</tr>
<tr>
<td>oztex</td>
<td>Andrew Trevorrow’s OzTEX v1.8+</td>
</tr>
<tr>
<td>17oztex</td>
<td>Andrew Trevorrow’s OzTEX v1.7</td>
</tr>
</tbody>
</table>

Other DVI-drivers may also work using one of these files, if they use conventions similar to dvips, OzTEX or TEXTURES. Alternatively it should not be too difficult to write the files required, using these as a basis indicating the type of information needed to support the specific \special commands. Anyone attempting to do this should inform the author and convey a successful implementation to him for inclusion within the XY-pic distribution.

**Note:** In some previous versions of XY-pic the POSTSCRIPT backend and driver were loaded simultaneously by a command of the form \UsePSspecials{⟨driver⟩}. For backward-compatibility these commands should still work, but now loading the latest version of the given ⟨driver⟩. However their future use is discouraged in favour of the option-loading mechanism, via \xyoption{⟨driver⟩}. This latter mechanism is more flexible, both in handling upgrades of the actual driver support and in being extensible to support more general forms of \special commands.

Once activated the POSTSCRIPT backend can be turned off and on again at will, using the user following commands:

\NoPSspecials cancels POSTSCRIPT
\UsePSspecials {} restores POSTSCRIPT

These obey normal TeX scoping rules for environments; hence it is sufficient to specify \NoPSspecials within an environment or grouping. Use of POSTSCRIPT will be restored upon exiting from the environment.

The default level of POSTSCRIPT is [4], see below.

170 \xydef@\UsePSspecials\{\DNii@{{[4]}}\}\xyFN@\UsePSspecials@i
171 \xydef@\UsePSspecials@i{%
172 \ifx \space@\next \expandafter\DN@\space{\xyFN@\UsePSspecials@i}%
173 \else\ifx \[\next \DN@[[#1]{\DNii@{{[[#1]}}}\xyFN@\UsePSspecials@i]%
174 \else\ifcat 5\noexpand\next \DN@#1{\DNii@{{[#1]}}\xyFN@\UsePSspecials@i}%
175 \else\ifx\bgroup\next \DN@{\expandafter\UsePS@specials\nextii@}%
176 \else \DN@{\UsePS@specials{[4]}%}
177 \fi\fi\fi\fi \next@ }

**To Do:** There is redundancy between here and \UsePSspecials@. The value of \setupxyPS@ POSTSCRIPT level could be set here rather than later in \PSspecials@.

185 \xydef@\UsePS@specials#1#2{%
186 \ifx\empty\whichPSspecials@
187 \DN@[#2]\ifx\next@\empty
188 \else
189 \expandafter\let\expandafter\next@csname#2@\endcsname
190 \ifx\next@relax \DN@{\xyerror{PostScript special for ‘#2’ not supported}{}}%
191 \else
192 \expandafter\let\expandafter\next@csname#2@\endcsname
193 \ifx\next@relax \DN@{\UsePSspecials@{#2}{}}%
The following control sequences will also load the corresponding driver. They are retained for backwards compatibility only; they may be phased out of future releases:

\UseDVIPSspecials dvips
\UsePostScriptSpecials dvips
\UseTexturesPSspecials Textures
\UseTexturesSpecials Textures

First the driver is specified by \UsePSspecials{(driver)}. This causes a special driver file to be read. Use of fonts is restored at any point by calling \NoPSspecials, after which use of PostScript...
is re-instated by simply \UsePSspecials, without need of an argument. The commands described
above for specific drivers are simply aliases for \UsePSspecials@{(driver)}.

Once the new bindings have been setup, then the use of \specials is governed by the value of the
conditional \ifPSspecials@.

\xydef\loadPSdriver@#1#2{\xyinputorelse@{xy#1}\%
{\xyrequire@{#2}\expandafter@\ifx\csname xy#2\loaded\else \%
{\xyerror@{Unable to load xy#1.tex for #2 driver}@{}}\fi}}
\xydef\dvips@{\loadPSdriver@{ps-dvi}\{dvips\}}
\xydef\Textures@{\loadPSdriver@{ps-txt}\{Textures\}}
\xydef\OzTeX@{\loadPSdriver@{ps-oz}\{OzTeX\}}
\xydef\dvitops@{\loadPSdriver@{ps-dto}\{dvitops\}}
\xydef\dvipsone@{\loadPSdriver@{ps-one}\{dvipsone\}}
\xydef\dviiwindo@{\loadPSdriver@{ps-wdo}\{dviwindo\}}
\xydef\dviwin@{\loadPSdriver@{ps-win}\{dviwin\}}
\xydef\pubps@{\loadPSdriver@{ps-pub}\{pubps\}}

The driver file contains definitions which are specific to the particular driver. Note that some
drivers may not be able to support all of the PostScript effects that can be requested from within
\Xy-pic. When an unsupported effect is encountered, it is simply ignored; a warning message will be
produced unless too many such messages have already been issued.

\global\newif\ifPSspecials@
%\xydef\UsePSspecials@#1{\gdef\whichPSspecials@{#1}\xyFN@\PSspecials@}
\global\newif\ifPSspecials@
\xydef\UsePSspecials@#1{\def\whichPSspecials@{#1}\DN@\Textures@\ifx\next@\whichPSspecials@
\DN@#1{\xyrequire@{textures}\xyps@options}\%
{\else}\DN@\dvips@\ifx\next@\whichPSspecials@
\DN@#1{\xyrequire@{dvips}\xyps@options}\%
{\else}\DN@\OzTeX@\ifx\next@\whichPSspecials@
\DN@#1{\xyrequire@{oztex}\xyps@options}\%
{\else}\DN@\dvitops@\ifx\next@\whichPSspecials@
\DN@#1{\xyrequire@{dvitops}\xyps@options}\%
{\else}\DN@{\xyrequire@{ps}\xyrequire@{#1}\xyrequire@{\xyuncatcodes } }
\fi\fi\fi \next@ }
\xydef\xyPSwarning@#1{\xywarning@{PostScript switched #1}}
%
Adding new drivers  Other drivers can be added by including an appropriate control-sequence name for the driver at this point. The purpose of an expansion such as \texttt{\textbackslash xydef \textbackslash OzTeX\{\noexpand\OzTeX\}} is so that this name always refers to a unique token. The macro \texttt{\textbackslash PSspecials@@} defined below, detects this token and then calls up the appropriate macro to make the necessary bindings.

Thus adding a new driver involves 5 steps:

1. Define a new control sequence, as just explained.

2. Define appropriately named macros to generate the desired \texttt{\special}s. (See the existing ones for examples of what is needed here.)

3. Define a macro that will perform the appropriate bindings of the different classes of \texttt{\special}. (See the existing ones for examples of what is needed here.)

4. Determine how frequently the \texttt{XY-ps} dictionary must be included within the PostScript file. Once (at the beginning) is the ideal, however it may be necessary to include it with every page—this is the case with \textsc{Textures}. Also if the dictionary can be loaded as a header or prolog, determine whether this can be done only once or must be for each page. Also check whether a specific name is required, as with \textsc{OzT\v{E}X}.

5. Check to see whether the \texttt{\textbackslash dumpPSdict} macro correctly writes the dictionary to a file. The \texttt{\textbackslash endlinechar} can be important here.

\texttt{XY-ps} works by rebinding existing control sequence names, in the \texttt{XY-pic} kernel and extensions, to have new expansions. These new expansions may eventually typeset nothing, or at most an empty box; instead they use a \texttt{\special} command to place \texttt{PostScript} code directly into the \texttt{.dvi} file (or \texttt{DVI2} resources in the case of \textsc{Textures} on the Macintosh). The new expansions alter the \textsc{T\v{E}X} processing, often simplifying it considerably hence leading to savings in both time and memory requirement.

It should not be possible to mix \texttt{\special}s intended for different drivers. Thus the first use of \texttt{\textbackslash PSspecials@} will establish which driver is required then rebinds \texttt{\UsePSspecials@} so that no other driver can be used; subsequent attempts to use \texttt{\UsePSspecials@} simply result in the same driver being reinstated.

An optional argument to the \texttt{\UsePSspecials} command allows for some control over precisely when \texttt{PostScript} \texttt{\special}s will be used. Similarly \texttt{\UseDVIPSspecials} and \texttt{\UseTexturesSpecials}, etc. can take an optional argument.

For example, \texttt{\UsePSspecials[1]} replaces only font characters with a \texttt{PostScript} drawing of the same character. Both the \texttt{PostScript} and the bitmapped fonts should produce (virtually) identical printed images. This is primarily a test mode.
\UsePSspecials[2] also only replaces font characters but such that the number of possible directions is 8192, so that arrowheads turns and hooks fit better.

\UsePSspecials[3] draws straight and dotted lines from a single \special command, and similarly for circle segments. The printed output should be identical to that obtained with \UsePostScriptSpecials[2], but the size of the .ps file should be smaller.

\UsePSspecials[4] is the default level; all PostScript is turned on. Dotted curves use equally spaced dots, dashed curves have curved dashes; even dashed lines are better.

\UsePSspecials[0] does no rebinding of fonts at all. It allows the special effects, such as rotation, scaling, colour, etc defined in extensions, to be implemented while using the Xy-pic fonts.

\xydef@\setupxyPSlevel0@{\relax}
\xydef@\setxyPSlevel@#1{\ifcase#1%
  \gdef\setupxyPS@{\setupxyPSlevel10@}%
  \or\gdef\setupxyPS@{\setupxyPSlevel1A@}%
  \or\gdef\setupxyPS@{\setupxyPSlevel1B@}%
  \or\gdef\setupxyPS@{\setupxyPSlevel1C@}%
  \or\gdef\setupxyPS@{\setupxyPSlevel1D@}%
  \else\gdef\setupxyPS@{\setupxyPSlevel1D@}\fi }
\xywarnifdefined\setupxyPS@
\gdef\setupxyPS@{\setupxyPSlevelD@ \gdef\setupxyPS@{\Ps specials@true}}
\xydef@\PSincrease@#1{\xywarning@{The PS level may only increase: #1 is already active}}

2.10.2 Why use PostScript

At some sites users have difficulty installing the extra fonts used by XY-pic. The .tfm files can always be installed locally but it may be necessary for the .pk bitmap fonts (or the .mf METAFONT fonts) to be installed globally, by the system administrator, for printing to work correctly. If PostScript is available then XY-ps allows this latter step to be bypassed.

Note: with XY-ps it is still necessary to have the .tfm font metric files correctly installed, as these contain information vital for correct typesetting.

Other advantages obtained from using XY-ps are the following:

- Circles and circle segments can be set for arbitrary radii.

- solid lines are straighter and cleaner. They are no longer typeset as a collection of fixed-sized segments (drawn from the xydash10 font). Previously special placement algorithms were required, to construct lines of arbitrary length in up to 8192 distinct directions, from the 128 characters in the font. These algorithms are no longer required, giving improved TeX processing, as well as having smooth lines of arbitrary length and direction limited only by the resolution of the PostScript device.

- The range of possible angles of directionals is greatly increased. For arrow tips, hooks, and turns, there are now 8192 possible orientations rather than just the 128 contained in the xyatip10, xybtip10, and xybsql10 fonts.

- Spline curves are smoother. True dotted and dashed versions are now possible, using equally spaced segments which are themselves curved.
XY-pic enables special effects such as variable line thickness, gray-level and colour. Also, rotation of text and (portions of) diagrams is now supported with some drivers. Similarly whole diagrams can be scaled up or down to fit a given area on the printed page.

Some of the above advantages are significant, but they come at a price. Known disadvantages of using XY-ps include the following:

- A DVI file with specials for a particular PostScript driver can only be previewed if a previewer is available that supports exactly the same \special format. A separate PostScript previewer will usually be required.

However recent versions of xdvi support viewing of PostScript using either the GhostScript program or via “Display PostScript”. The PostScript produced by XY-ps can be viewed this way

- DVI files created using XY-ps in fact lose their “device-independence”. So please do not distribute DVI files with PostScript specials—send either the \TeX source code, expecting the recipient to have XY-pic, or send a (compressed) PostScript file.

The latter comment applies to files in which any special ‘back-end’ support is required, not just to PostScript. Of course it can be ignored when you know the configuration available to the intended recipient.

PostScript header file: With some DVI-drivers it is more efficient to have the PostScript commands that XY-ps needs loaded initially from a separate “header” file. To use this facility the following commands are available...

\begin{verbatim}
\UsePSheader {}
\UsePSheader {<filename>}
dumpPSdict {<filename>}
\xyPSdefaultdict
\end{verbatim}

Normally it is sufficient to invoke \UsePSheader{}, which invokes the default name of xy38dict.pro, referring to the current version of XY-pic. The optional (filename) allows a different file to be used. Placing \dumpPSdict{..} within the document preamble causes the dictionary to be written to the supplied (filename).

\begin{verbatim}
611 \newif\ifUsePSdict
612 \ifx\undefined\UsePSdict\true\else\DN@{}\fi \next@
613 \ifx\undefined\xyPSdefaultdict
614 \xydef\xyPSdefaultdict{%
615 \DN@##1.##2.##3@{\gdef\xyPSdictname{xy##1##2dict.pro}}%
616 \expandafter\next@\xyversion.0}%
617 \fi
618 \ifx\undefined\UsePSheader@
619 \xydef\UsePSheader@{%
620 \DN@(#1)\ifx\next@\empty
621 % \ifx\xyPSdictname\undefined\xyPSdefaultdict\fi
622 \else \gdef\xyPSdictname{#1}\fi
623 \ifx\xyPSdictname\undefined\xyPSdefaultdict\fi
624 \gdef\xyHeaderValue%
\end{verbatim}
It includes a reference to the PostScript file `\xyPSdictname.^^J}%
`\UsePSdict@true\%
\let\UsePShader=\UsePShader@
\fi

See the documentation for the specific driver to establish where the dictionary file should be located on any particular \TeX system. Usually it is sufficient to have a copy in the current working directory. Invoking the command `\dumpPSdict{} will place a copy of the requisite file, having the default name, in the current directory. This file will be used as the dictionary for the current processing, provided it is on the correct directory path, so that the driver can locate it when needed. Consult your local system administrator if you experience difficulties.

The `dump' cannot be performed until a driver has been installed. This ensures that the correct `end-of-line' character is used.

\xydef@\dumpPSdict#1{\DN@{#1}\ifx\next@\empty
\ifx\undefined\xyPSdictname\xyPSdefaultdict\fi
\else\gdef\xyPSdictname{#1}\fi
\def\dumpPSdict@@{\writePSdict@@}\
\ifx\xydriversselected@@\empty\DN@{\xysetup@@\null@xy@ps}\
\else \DN@{\installxyps@x}\fi \next@}

Multiple instances of `\UsePShader and `\dumpPSdict are possible, only the last will determine which file is used for the current document. The command `\xyPSdefaultdict reverts to the default name.

Allowing an arbitrary ⟨filename⟩ with `\UsePShader is for flexibility, to accommodate systems that may impose special requirements on the filenames of files to be used as PostScript header files. Oz\TeX is one such. It also caters for advanced users of XY-pic who may wish to experiment with customised PostScript to obtain new effects.

2.10.3 Hooking into XY-pic

This next macro streamlines the rebinding process.

\gdef\xyPSalternative@#1#2{\%
\expandafter\global\expandafter\let\csname origxy#2@\endcsname=#1\%
\xdef#1{\noexpand\ifPSspecials@
\noexpand\expandafter\expandafter
\noexpand\csname xyPS#2@\endcsname
\noexpand\else
\noexpand\expandafter\expandafter
\noexpand\csname origxy#2@\endcsname
\noexpand\fi}}%

The commands `\setupxyPSlevelA@, `\setupxyPSlevelB@, `\setupxyPSlevelC@ and `\setupxyPSlevelD@ actually perform the rebindings. Each may be called precisely once, and each requires all lower levels are also set.

\xydef@\setupxyPSlevelA@{\xyPStips@s}%
\xyPSalternative@{\Tip@@}{Tip}%
\xyPSalternative@{\Tip@@eu}{Tip@eu}%
\xyPSalternative@{\hook@@}{hook}%
\xyPSalternative@{\ahook@@}{ahook}%
The bindings are not performed until \setupxyPS@ is called.

2.10.4 Kernel improvements

Directionals:

These macros standardise the way a character is described in PostScript. First give the \Direction code then the operator name, which is just a simple character string. The modifiers \xyPSfont@ and \xyPSsemifont@ are empty in all but the most primitive level of PostScript usage. When non-empty they restrict to using only angles corresponding to actual font characters.
These macros provide PostScript code to round a \texttt{Direction} code to that corresponding to the nearest font character code, for Directional and semi-Directional fonts respectively.

\texttt{\textbackslash xydef@\textbackslash xyPSfont@\{f \}}%\{\texttt{xyfont} \}
\texttt{\textbackslash xydef@\textbackslash xyPSsemifont@\{fs \}}%\{\texttt{xysemifont} \}

Before any binding we save the original expansions of control-sequences whose names will be subject to rebinding. We give these first for each font.

tips  Arrow heads in \texttt{\textbackslash yatipfont} and \texttt{\textbackslash ybtipfont}

The tips are all set as zero-sized characters:

\texttt{\textbackslash xydef@\textbackslash xyPStip@\{\textbackslash xyPSchar@\{t\}\}}%\{\texttt{\textbackslash xyPSchar@\{tip\}\}}
\texttt{\textbackslash xydef@\texttt{\textbackslash xyPSatip@\{\textbackslash xyPSchar@\{a\}\}}%\{\texttt{\textbackslash xyPSchar@\{atip\}\}}
\texttt{\textbackslash xydef@\texttt{\textbackslash xyPSbtip@\{\textbackslash xyPSchar@\{b\}\}}%\{\texttt{\textbackslash xyPSchar@\{btip\}\}}
\texttt{\textbackslash xydef@\texttt{\textbackslash xyPStip@\{\textbackslash xyPSchar@\{/XT \{t\}\}}%\{\texttt{\textbackslash xyPSchar@\{Tip\}\}}
\texttt{\textbackslash xydef@\texttt{\textbackslash xyPSTip@\{\texttt{\textbackslash setbox\{h\}}\}}%\{\texttt{\textbackslash Kern\{L\}}\}
\texttt{\textbackslash reverse\textbackslash Direction@\{\texttt{\textbackslash line\{w\}}\}}%\{\texttt{\textbackslash HT \{z\}}%\{\texttt{\textbackslash DP \{z\}}%\{\texttt{\textbackslash KT \{z\}}%\{\texttt{\textbackslash xyPSchar@\{/Xt \{t\}\}}%\{\texttt{\textbackslash xyPSchar@\{Ttip\}\}}
\texttt{\textbackslash xydef@\texttt{\textbackslash xyPStip@\{\texttt{\textbackslash xyPSchar@\{/ET \{t\}\}}%\{\texttt{\textbackslash xyPSchar@\{Tip\}\}}
\texttt{\textbackslash xydef@\texttt{\textbackslash xyPSTip@\{\texttt{\textbackslash setbox\{h\}}\}}%\{\texttt{\textbackslash Kern\{L\}}\}
\texttt{\textbackslash reverse\textbackslash Direction@\{\texttt{\textbackslash line\{w\}}\}}%\{\texttt{\textbackslash HT \{z\}}%\{\texttt{\textbackslash DP \{z\}}%\{\texttt{\textbackslash KT \{z\}}%\{\texttt{\textbackslash xyPSchar@\{/ET \{t\}\}}%\{\texttt{\textbackslash xyPSchar@\{Ttip\}\}}
\texttt{\xydef@\xyPStips@\{\xyPSalternative@\{\xyPSatip@\} \xyPSalternative@\{\xyPSbtip@\} \xyPSalternative@\{\xyPSabtip@\}\}}
\texttt{\xydef@\xyPStip@\{\xyPSchar@\{ct\}\} \xyPSalternative@\{\xyPSatip@\} \xyPSalternative@\{\xyPSbtip@\} \xyPSalternative@\{\xyPSabtip@\}\}}

The mechanism to handle Computer Modern tips is a little different. Here we must rebind \texttt{\textbackslash xycm@} to avoid any calls to font names. When \texttt{\textbackslash PSspecialstrue} then we expand a macro which will look at the following token to decide which type of tip is required after \texttt{\textbackslash tip@x}.

\texttt{\%\textbackslash xydef@\texttt{\textbackslash xyPScmtips@\{\xyPSalternative\{\texttt{\textbackslash xycm@}\{cm\}\}}}\%\}
\texttt{\%\textbackslash xydef@\texttt{\textbackslash xyPScm@\{\texttt{\textbackslash tip@x\}}\{\texttt{\textbackslash xyPScm@\}}\%\}
\texttt{\%\textbackslash xydef@\texttt{\textbackslash xyPScm@\{\texttt{\textbackslash tip@x\}}\{\texttt{\textbackslash xyPScm@\}}\%\}
\texttt{\%\textbackslash xydef@\texttt{\textbackslash xyPScmtip@\{\texttt{\textbackslash xyPSchar@\{ct\}\}}}\%\}
\texttt{\%\textbackslash xydef@\texttt{\textbackslash xyPSmatip@\{\texttt{\textbackslash xyPSchar@\{ca\}\}}}\%\}
\texttt{\%\textbackslash xydef@\texttt{\textbackslash xyPSmbtip@\{\texttt{\textbackslash xyPSchar@\{cb\}\}}}\%\}

From version 3.3 the font coding has changed, for greater flexibility and consistency. In particular \texttt{\textbackslash xycm@} is no longer defined.

\texttt{\textbackslash xydef@\texttt{\textbackslash xyPStips@\{\texttt{\textbackslash xyPSalternative\{\texttt{\textbackslash atip@\}}\{\texttt{\textbackslash atip@\}\}}}\%\}
\texttt{\textbackslash xydef@\texttt{\textbackslash xyPSalternative\{\texttt{\textbackslash btip@\}}\{\texttt{\textbackslash btip@\}}}\%\}
\texttt{\textbackslash xydef@\texttt{\textbackslash xyPSalternative\{\texttt{\textbackslash tip@\}}\{\texttt{\textbackslash abtip@\}}}\%\}
\texttt{\textbackslash xydef@\texttt{\textbackslash xyPSwhichatip@\{\texttt{\textbackslash xyPSchar@\{ct\}\}}}\%
\texttt{\textbackslash xydef@\texttt{\textbackslash xyPSwhichbtip@\{\texttt{\textbackslash xyPSchar@\{cb\}\}}}\%
\texttt{\textbackslash expandafter@\texttt{\textbackslash xyPSchar@\{expanded\}}}\%
\texttt{\textbackslash expandafter@\texttt{\textbackslash xyPSchar@\{expanded\}}}\%
hooks, turns and squiggles: squiggles from: \xybsqlfont

The hooks are zero-sized characters...

\xydef@\xyPShook@{\xyPSchar@{h}}%{\xyPSchar@{hook}}
\xydef@\xyPSahook@{\xyPSchar@{ha}}%{\xyPSchar@{ahook}}
\xydef@\xyPSbhook@{\xyPSchar@{hb}}%{\xyPSchar@{bhook}}

...so are the turns...

\xydef@\xyPSaturn@{\xyPSchar@{ta}}%{\xyPSchar@{aturn}}
\xydef@\xyPSbturn@{\xyPSchar@{tb}}%{\xyPSchar@{bturn}}

This handles squiggles as individual characters. Squiggled connections should be handled sepa-
ately, e.g. by having \xyPSsquiggle@ set the connection by something like \edef\Connect@@{\noexpand\xyPSsquiggled@ \the\Direction}.

\xydef@\xyPSsquiggle@{\xyPSchar@{g}}%{\xyPSchar@{squigl}}

To Do: define composite \xyPSsquiggled@
To Do: define composite \xyPSdashsquiggled@

dashes and stoppers: using characters from: \xydashfont

A stopper contributes zero size. However a dash gets its size from the italic correction in the
\xydashfont, accessed here using \origxyline@@.

\xydef@\xyPSstopper@{\xyPSchar@{st}}%{\xyPSchar@{stopper}}
\%\xydef@\xyPSflipt@{\xyPSchar@{true dash}}
\%\xydef@\xyPSflipt@{\xyPSchar@{false dash}}
\% \DN@{\xyPSsemich@{true dash}}
\% \DN@{\xyPSsemich@{false dash}}
\% \ifnum\SemiDirectionChar=31
\% \ifdim\d@Y<\z@ \ifx\xyPSsemifont@\empty\xyPSflipt@i

2.10. POSTSCRIPT BACKEND

Circles

full circles and circle segments: \xycircfont

This handles the cases where a font character is called using \circhar#1.

\xydef@\xyPScirchar@#1{\count@=#1\edef\tmp@{\the\count@}\
  \ifcase#1% 
  \or{-180}{-135}{\tmp@}\kern\partroottwo\R@
  \or{-135}{-90}{\tmp@}\kern\halfroottwo\R@
  \or{-90}{-45}{\tmp@}\kern\halfroottwo\R@
  \or{-45}0{\tmp@}\kern\partroottwo\R@
  \or0{45}{\tmp@}\kern\partroottwo\R@
  \or{45}{90}{\tmp@}\kern\halfroottwo\R@
  \or{90}{135}{\tmp@}\kern\halfroottwo\R@
  \or{135}{180}{\tmp@}\kern\partroottwo\R@
  \fi}

\xydef@\xyPScirrest@{\relax}%

This replaces just the font character, called using \circhar@@#1.

\xydef@\xyPScirchar@@#1{\relax\count@=#1\edef\tmp@{\the\count@}\
  \expandafter\xyPScirchar@@@\ifcase#1% 
  {-180}{-135}{\tmp@}\kern\partroottwo\R@
  {-135}{-90}{\tmp@}\kern\halfroottwo\R@
  {-90}{-45}{\tmp@}\kern\halfroottwo\R@
  {-45}0{\tmp@}\kern\partroottwo\R@
  0{45}{\tmp@}\kern\partroottwo\R@
  {45}{90}{\tmp@}\kern\halfroottwo\R@
  {90}{135}{\tmp@}\kern\halfroottwo\R@
  {135}{180}{\tmp@}\kern\partroottwo\R@
  \fi}

\xydef@\xyPScirchar@@@#1#2#3{\edef\tmp@{#1\space#2\space\
  \expandafter\removePT@\the\R@\space #3\space c}\
  \PSmacro@{\tmp@}}

This next macro will allow for more general circle segments to be done in POSTSCRIPT. The two
parameters are the starting angle and finishing angle respectively, measured anti-clockwise.

\xydef@\xyPScir@#1#2{\edef\tmp@{\expandafter\removePT@\the\R@}\
  \PSmessage{\tmp@\space #1 \#2 circ}\PSmacro@{\tmp@\space #1 \#2 circ}}

This gives full circles and circle segments built from quarter turns.
solid, dashed and dotted; without using segments.

This is based on \straight@. It is used by all three types of line, taking an appropriate macro as the parameter #1.

This is not needed unless styles are used, hence xyps-s is loaded.
Here is how solid lines are done.

\xydef\xyPSsolid@{\straight\xyPSsolidSpread}
\xydef\xyPSsolidSpread@{(solid)}
\xydef\xyPSdottedSpread@{(dotted)}
\xydef\xyPSdashed@{(dashed)}

This replaces the \texttt{Xy-pic} \texttt{\textbackslash dash@} to give dashed lines:

\xydef\xyPSdashed@{(line@ \def\Connect@@{(dashed)\xyPSdashedSpread})}
\xydef\xyPSdashedSpread@{(dashed)\xyPSSpread}

This replaces the \texttt{Xy-pic} \texttt{\textbackslash point@} to give dotted lines. \texttt{Xy-pic} defines \texttt{\textbackslash zerodot} to be a small centred square with side length $\xydashw$. The PostScript substitute is a round dot of this radius.

\xydef\xyPSzerodot@{(dot)\xyPSzerodot@\egroup}
\xydef\xyPSdotted@{(dotted)\xyPSSpread}

Frames

These are used for dashed frames.

This may do circular frames.

Curves

\xydef\xyPScubic@#1{\readsplineparams@}
\xydef\xyPSsquine@#1{\readsplineparams@}

\texttt{Frames}

\texttt{Curves}
CHAPTER 2. EXTENSIONS

\xydef@\xyPSsplinesolid@{\ifx\splineinfo@\squineinfo@\expandafter\xyPSsquine@\else\expandafter\xyPScubic@\fi{}}
\xydef@\xyPSsplinedashed@{\ifx\splineinfo@\squineinfo@\expandafter\xyPSsquine@\else\expandafter\xyPScubic@\fi{d}}
\xydef@\xyPSsplinedotted@{\ifx\splineinfo@\squineinfo@\expandafter\xyPSsquine@\else\expandafter\xyPScubic@\fi{t}}
\expandafter\xyendinput

The end & Log
\xyendinput

\% $Log: xyps.doc,v $
\% Revision 3.12 2011/05/27 04:51:17 krisrose
\% Ready to release.
\% 
\% Revision 3.11 2011/03/14 20:14:00 krisrose
\% Preparing for release 3.8.6.
\% 
\% Revision 3.10 2010/06/10 18:45:50 krisrose
\% Reference to GPL by URL.
\% 
\% Revision 3.9 2010/05/06 17:46:30 krisrose
\% Ross Moore’s e-mail address updated.
\% Many obsolete files degraded to Historic.
\% 
\% Revision 3.8 2010/04/16 06:06:52 krisrose
\% Preparing for a new release...
\% 
\% Revision 3.7 1999/02/16 15:12:50 krisrose
\% Interim release (Y&Y fonts now free).
\% 
\% Revision 3.4 1997/05/18 01:13:24 ross
\% Essential bugfixes.
\% 
\% Revision 3.3 1996/12/19 03:50:08 ross
\% Maintenance release.
\% 
\% Revision 3.3 1996/12/18 09:31:46 ross
\% revised interface to straight connections
\% revised interface for tips, handles arbitrary scaling
\% driver/option-loading now uses \xyrequire
\% 
\% Revision 3.2 1995/09/19 18:21:41 ross
\% Bug fix release.
\% 
\% Revision 3.1 1995/09/05 20:36:33 ross
\% Release!
\% 
\% Revision 3.0 1995/07/07 20:13:19 ross
2.11 TPIC backend

Vers. 3.7 by Ross Moore (ross.moore@mq.edu.au)

This option allows the Xy-pic fonts to be replaced by TPIC \specials, when used with a dvi-driver capable of supporting them. Extra capabilities include smoother lines, evenly spaced dotted/dashed curves, variable line-widths, gray-scale fills of circles, ellipses and polygonal regions.

Header:
1 % $Id: xytpic.doc,v 3.7 2011/03/14 20:14:00 krisrose Exp$
2 %
3 % Xy-pic ‘‘tpic’’ TPIC backend, for lines, curves and fills.
4 % Copyright (c) 1995-1996 Ross Moore <ross.moore@mq.edu.au>
5 %
6 % This file is part of the Xy-pic package for graphs and diagrams in TeX.
Use of \texttt{tpic} \texttt{specials} offers an alternative to the \texttt{Xy-pic} fonts. However they require a \texttt{dvi}-driver that is capable of recognizing and interpreting them. One such viewer is \texttt{xdvik}, Karl Berry’s modification to the \texttt{xdvi} viewer on \texttt{UNIX} systems running \texttt{X}-windows or a derivative. \texttt{dvipsk}, Karl Berry’s modification to \texttt{dvips} also handles \texttt{tpic} \texttt{specials}, so \texttt{xdvik/dvipsk} is an good combination for quality screen-display and \texttt{PostScript} printing.

Once loaded using \texttt{xyoption(tpic)}, with an appropriate \texttt{⟨driver⟩} also specified either already or subsequently, the following commands are available to turn the \texttt{TPIC} backend off/on.

\begin{Verbatim}
\NoTPICspecials \hspace{1em} \text{turns off TPIC specials.}
\UseTPICspecials \hspace{1em} \text{reinstates TPIC specials.}
\end{Verbatim}

There is a limit to the number of points allowable in a path. For paths constructed by \texttt{Xy-pic}, which includes spline curves, when the limit is reached the path is automatically flushed and a new path commenced. The following command can be used to customise this limit—initially set at 300 for use with \texttt{XDVI}—to suit alternative \texttt{⟨driver⟩}s.

\begin{Verbatim}
\maxTPICpoints{⟨num⟩} \hspace{1em} \text{set maximum for paths}
\end{Verbatim}

\footnote{\texttt{UNIX} is a trademark of Bell Labs.}
Each object involving \texttt{TPIC} \texttt{\special}s must set the style, using \texttt{\xytpic@set@} to store a macro as the expansion of \texttt{\xytpic@style@@}. Currently this is global, since a TPIC object must be completed once it has been started — perhaps this is not necessary.

Note that \texttt{\xytpic@set@} uses \texttt{\def}, but has a \texttt{\noexpand} in its expansion; this forces expansion of the argument, when there is one, to the control sequence passed as the first token following \texttt{\xytpic@set@}.

Here are the actual \texttt{\special}s that get placed in the dvi-file.

Some effects may require a TPIC \texttt{\special} to be placed both before and after the ⟨object⟩.

TPIC uses units of ‘milli-inches’ for coordinates, but inches for dot-separations and dash-length.
re-bindings  This is similar to the mechanism used by the PostScript back-end to rebind “hooks” to incorporate a switch according to the current value of \ifTPICspecials@.

 Allow only one instance of \setupxyTPIC@; thereafter simply do \reloadxyTPIC@ to turn on anything that has been turned off.
2.11. TPIC BACKEND

\TPICspecials@false
\let\setupxyTPIC@=\reloadxyTPIC@
\xydef@\activateTPIC@{%
  \xywithoption{line}{\xysetup@\installTPICline@}%
  \xywithoption{curve}{\xysetup@\installTPICcurves@}%
  \xywithoption{frame}{\xysetup@\installTPICframes@}%
  \TPICspecials@true
}\xydef@\reloadxyTPIC@{\xywarning@{TPIC specials back on}\activateTPIC@ }

\xydef@\xytpic@solid@{%
  ifInvisible@ \let\next@=\no@@
  else \DN@{\xyTPICstraight@{\xytpic@solid@@ \xytpic@line}}\fi
  \next@
}\xydef@\xytpic@dashed@{\line@
  \def\Connect@@{\xyTPICstraight@{\xytpic@dashed@@ \xytpic@line}}}
\xydef@\xytpic@point@{\xytpic@dot@@\xytpic@fp\egroup
  Invisible@false \Hidden@false \def\Leftness@{.5}\def\Upness@{.5}\ctipEdge@
  \def\Drop@@{\styledboxz@}\
  \def\Connect@@{\xyTPICstraight@{\xytpic@dotted@@ \xytpic@line}}}
\xydef@\xyTPICstraight@#1{\setupDirection@
  \edef\Creset@@{\cfromthec@ \pfromthep@ \DirectionfromtheDirection@}%
  \DN@##1##2{\def\checkoverlap@@{%
    \ifdim##1\X@p>##1\X@c \let\next@=\relax \fi
    \ifdim##2\Y@p>##2\Y@c \let\next@=\relax \fi}\
    \edef\nextii@{{\sd@X}{\sd@Y}}\expandafter\next@
extii@
    \noCshavep@@ \edef\Cshavep@@{\pfromthep@ \resetDirection@}%
    \noCshavec@@ \edef\Cshavec@@{\cfromthec@ \resetDirection@}%
  \ifHidden@else
    \ifdim\Y@c>Y@max \Y@max=\Y@c \fi \ifdim\Y@p>Y@max \Y@max=\Y@p \fi
    \ifdim\Y@c<Y@min \Y@min=\Y@c \fi \ifdim\Y@p<Y@min \Y@min=\Y@p \fi
    \ifdim\X@c>X@max \X@max=\X@c \fi \ifdim\X@p>X@max \X@max=\X@p \fi
    \ifdim\X@c<X@min \X@min=\X@c \fi \ifdim\X@p<X@min \X@min=\X@p \fi
  \fi}
\ifInvisible@\let\next@=\relax \else
    \DN@{\setboxz@h{\kern\X@c \raise\Y@c\hbox{#1}}}\
    \ht@z@=\z@ \wd@z@=\z@ \dp@z@=\z@ {\Drop@@}}%}
\fi
\checkoverlap@@
\ifdim\d@X=\z@ \ifdim\d@Y=\z@ \DN@{\relax}fi fi \next@
\def\Cslidep@@{\noCslidep@@}\def\Cslidec@@{\noCslidec@@}\
\def\Calong@@{\noCalong@@}\Creset@@

straight lines

spline curves Of the curves defined in the xycurve extension, only solid spline curves are supported. This is done by treating the spline as a polygon (poly-line) with many segments. The dotted or dashed variants do not work correctly.

Implementations of TPIC draw dashed polygons such that the start and finish of each segment is
solid. Since these segments can be very short, the effect is simply to create a solid line. Similarly the
shortness of the segments tends to give nothing at all for large portions of a dotted curve. What is
needed is an implementation whereby the on/off nature of a dashed or dotted polygon is determined
by the accumulated length, not the length along just the current segment.

\xydef@\xytpic@splinesolid@{\xyTPICspline@\xytpic@fp }
\xydef@\xytpic@splinedashed@{\xytpic@in\xydashl@\expandafter\xyTPICspline@
\expandafter\{\expandafter\xytpic@da\expandafter{\next@}\}}
\xydef@\xytpic@splinedotted@{\xytpic@in{.5\xydashl@}\expandafter\xyTPICspline@
\expandafter\{\expandafter\xytpic@dt\expandafter{\next@}\}}
\xydef@\xyTPICspline@#1{\setbox\splinebox@=\copy\voidb@x %
\SloppyCurves %\splinetol@=\xydashl@ % \xytpic@ip
\global\xytpic@cnt=\@ne \xytpic@set@{#1}%
\def\splineextra@@{\xyTPICspline@dot}%
\def\xycrvdrop@{ }\def\xycrvconn@{\splineset@@ \xytpic@style@@}
\xydef@\xyTPIClinewidth@#1{\xytpic@pt{#1}\edef\next@{\the\count@}}
\xydef@\installTPICline@{\let\xylinewidth@@=\xylinewidth@ %
\let\xylineSpecial@@=\xytpic@pn
\let\transxyline@@=\xyTPIClinewidth@}
\xydef@\uninstallTPICline@{\UnloadLine@}%
\xydef@\xydoTPICcircle@#1{\DN@{#1}%
\ifx\next@\empty\DN@{\xyTPICfullcircle@}%
\else\expandafter\DNii@\expandafter{\addDASH@{}}%
\ifx\next@\nextii@\DN@{\L@=\R@ \xyTPICfullcircle@}%
\else\DN@{\dosolidcircle@{#1}}\fi\fi \next@ }
\xydef@\xyTPICfullcircle@{\xyTPICpartcircle@{00}0\twoPi@}
\xydef@\xyTPICpartcircle@#1#2#3{\xytpic@pt{\R@}\edef\next@{{\nextii@}{\the\count@}{#2}{#3}}%
\expandafter{\xyTPICpartcircle@{#1}}\expandafter{\fi}\next@ }
\xydef@\xyTPICfullcircle@{\xyTPICpartcircle@{00}0\twoPi@}
\xydef@\xyTPICpartcircle@#1#2#3{%}
\xytpic@pt{\R@}\expandafter{\DNii@}\expandafter{\{\addDASH@{}}%
\ifx\next@\nextii@\DN@{\L@=\R@ \xyTPICfullcircle@}%
\else\DN@{\dosolidcircle@{#1}}\fi\fi \next@ }
\xydef@\qartPi@{0.7853981}
\xydef@\tartPi@{2.3561944}
\xydef@\xytpic@circhar@#1{\xytpic@width@/0\ifcase#1%
\xytpic@pt{\R@}\edef\next@{\the\count@}}
This replaces just the font character, called using \circhar@@#1.
\xydef@\xytpic@circhar@#1{\xytpic@width@/0\ifcase#1%
\xytpic@pt{\R@}\edef\next@{\the\count@}}

\textbf{circles and ellipses} Here we catch circles and ellipses specified using the \xycircle command from
the xycurve extension. Since TPIC supports only solid circles, we catch it only when the (style) is
either {} or {-}.
\xydef@\xydoTPICcircle@#1{\DN@{#1}%
\ifx\next@\empty\DN@{\xyTPICfullcircle@}%
\else\expandafter\DNii@\expandafter{\addDASH@{}}%
\ifx\next@\nextii@\DN@{\L@=\R@ \xyTPICfullcircle@}%
\else\DN@{\dosolidcircle@{#1}}\fi\fi \next@ }
\xydef@\xyTPICfullcircle@{\xyTPICpartcircle@{00}0\twoPi@}
\xydef@\xyTPICpartcircle@#1#2#3{%}
\xytpic@pt{\R@}\expandafter{\DNii@}\expandafter{\{\addDASH@{}}%
\ifx\next@\nextii@\DN@{\L@=\R@ \xyTPICfullcircle@}%
\else\DN@{\dosolidcircle@{#1}}\fi\fi \next@ }
\xydef@\qartPi@{0.7853981}
\xydef@\tartPi@{2.3561944}
\xydef@\xytpic@circhar@#1{\xytpic@width@/0\ifcase#1%
\xytpic@pt{\R@}\edef\next@{\the\count@}}
\let\dosolidcircle@@=\xydoTPICcircle@
\let\xy@spline@ii@@=\xyTPIC@spline@ii@
}
\xydef@\uninstallTPICcurves@{%
\let\dosolidcircle@@=\dosolidcircle@
\let\xy@spline@ii@@=\xy@spline@ii@}
}
\xydef@\uninstallTPICframes@{\UnloadFrames@}
\xyendinput

% $Log: xytpic.doc,v $
% Revision 3.7 2011/03/14 20:14:00 krisrose
% Preparing for release 3.8.6.
%
% Revision 3.6 2010/06/10 18:45:50 krisrose
% Reference to GPL by URL.
%
% Revision 3.5 2010/05/06 17:46:30 krisrose
% Ross Moore’s e-mail address updated.
% Many obsolete files degraded to Historic.
%
% Revision 3.4 2010/04/16 06:06:52 krisrose
% Preparing for a new release...
%
% Revision 3.3 1996/12/18 09:50:38 ross
% checked in with -k by krisrose at 1996/12/18 14:17:11
%
% Revision 3.3 1996/12/18 09:50:38 ross
% adjusted Drop@@ for styles
% minor improvements to file-loading commands
%
% Revision 3.2 1995/09/19 18:21:41 ross
% Bug fix release.
%
% Revision 3.1 1995/09/05 20:36:33 ross
% Release!
%
% Revision 3.0 1995/07/07 20:13:19 ross
% Major release w/new User’s Guide!
%
% Revision 2.13 1995/07/04 15:04:51 ross
% Ready for release of v3.
%
% NEW for version 3.1 by Ross Moore 1995/03/18.
2.12. **EM-TEX BACKEND**

Vers. 3.7 by Ross Moore (ross.moore@mq.edu.au)

Eberhard Mathe’s em-T\TeX{} implementation provides a suite of \special commands to facilitate the drawing of lines, both on-screen and with various printing devices. This ‘back-end’ extension allows the lines in Xy-pic diagrams to be drawn using these methods.

Note that this extension does not have to be used with em-T\TeX{}. Better results may be obtained using the PostScript back-end and dvips ⟨driver⟩, since a version of dvips is available for em-T\TeX{}. However, in particular for screen previewing purposes, it may be convenient to use this back-end. Furthermore note that dvips is capable of supporting em-T\TeX\specials.

Header:

\input xy
\input xyemtex

\ifx\xypd\undefined
\input xy\fi

\input xyemtex

\noemspecials
\usespecials

Once loaded using \xypd, with an appropriate ⟨driver⟩ also specified either already or subsequently, the following commands are available to turn the em-T\TeX{} back-end off/on.

\noemspecials turns off em-T\TeX{} specials.
\usespecials reinstates em-T\TeX{} specials.
CHAPTER 2. EXTENSIONS

\newif{ifEMspecials}
\def\UseEMspecials{%true}
\def\NoEMspecials{%false}

\def\xyemtex@lineto{\xyEM@special{lineto}}
\def\xyemtex@moveto{\xyEM@special{moveto}}
\def\xyemtex@line#1{\xyEM@special{line #1}}
\def\xyemtex@point#1{\xyEM@special{point #1}}
\def\xyemtex@pen#1{\addtostyletoks@{\xyEM@special{linewidth #1}}}

The \special{\text{s}} These place the actual \special{\text{s}} and allow tracing:

\catcode`:=12 \gdef\next#1{\special{em:#1}}
\let\xyEM@special=\next
\catcode`:=12 \gdef\next#1{\xymath{EM:#1 (\the\X@p,\the\Y@p), (\the\X@c,\the\Y@c)}\special{em:#1}}
\let\xyEM@noisy=\next
\def\NoisyEMTeX{%let\xyEM@special=\xyEM@noisy}

Currently these are not used by \text{Xy-pic}.

re-bindings This is similar to the mechanism used by the \textsc{PostScript} back-end to rebind “hooks”
to incorporate a switch according to the current value of \texttt{ifEMspecials}.

\def\EMalternative@#1#2{%\expandafter\global\expandafter\let\csname origxy#2@\endcsname=#1\xdef#1{\noexpand\ifEMspecials@\noexpand\expandafter\expandafter\noexpand\csname xyemtex@#2@\endcsname\noexpand\else\noexpand\expandafter\expandafter\noexpand\csname origxy#2@\endcsname\noexpand\fi}}

Allow only one instance of \texttt{setupxyEM@}; thereafter simply do \texttt{reloadxyEM@} to turn on anything
that has been turned off.

\def\xywarn@{using \texttt{em-\textsc{TeX}} specials}%\EMalternative@{\solid@}{solid}%
\EMalternative@{\point@}{point}%
\EMalternative@{\splinesolid@}{splinesolid}%
\let\setupxyEM@=%\activateEM@

\let\xywarning@{EM specials turned off}%
\xywithoption{curve}{\xystep@\uninstallEMcurves@ }%
\xywithoption{line}{\xystep@\uninstallEMline@ }%
\xywithoption{frame}{\xystep@\uninstallEMframes@ }%
\EMspecials@false
\let\setupxyEM@=%\activateEM@
2.12. EM-TEX BACKEND

\edef\activateEM@{
% \xywithoption{line}{\xysetup@\installEMline@}%
% \xywithoption{curve}{\xysetup@\installEMcurves@}%
% \xywithoption{frame}{\xysetup@\installEMframes@}%
\EMspecials@true }

\edef\reloadxyEM@{\xywarning@{EM specials back on}\activateEM@ }

\edef\xyemtex@solid@{
\ifInvisible@ \let\next@=\no@@ \else \DN@{\xyEMstraight@}\fi
\next@ }

\edef\xyEMstraight@{
\setupDirection@\edef\Creset@@{\cfromthec@ \pfromthep@ \DirectionfromtheDirection@}%
\DN@##1##2{\def\checkoverlap@@{%\ifdim##1\X@c>##1\X@p \let\next@=\relax \fi
\ifdim##2\Y@c>##2\Y@p \let\next@=\relax \fi}%
\edef\nextii@{{\sd@X}{\sd@Y}}\expandafter\next@\nextii@
\noCshavep@@ \edef\Cshavep@@{\pfromthep@ \resetDirection@}%
\noCshavec@@ \edef\Cshavec@@{\cfromthec@ \resetDirection@}%
\ifHidden@\else
\ifdim\Y@c>\Y@max \Y@max=\Y@c \fi \ifdim\Y@p>\Y@max \Y@max=\Y@p \fi
\ifdim\X@c>\X@max \X@max=\X@c \fi \ifdim\X@p>\X@max \X@max=\X@p \fi
\ifdim\Y@c<\Y@min \Y@min=\Y@c \fi \ifdim\Y@p<\Y@min \Y@min=\Y@p \fi
\ifdim\X@c<\X@min \X@min=\X@c \fi \ifdim\X@p<\X@min \X@min=\X@p \fi \fi
\ifInvisible@\let\next@=\relax \else
\DN@{%\setboxz@h{\kern\X@p \raise\Y@p\hbox{\xyemtex@moveto}}%
\ht\z@=\z@ \wd\z@=\z@ \dp\z@=\z@ \% \setboxz@h{\boxz@ \kern\X@c \raise\Y@c\hbox{\xyemtex@lineto}}%
\ht\z@=\z@ \wd\z@=\z@ \dp\z@=\z@ \{\Drop@@\}%
\fi
\checkoverlap@@\ifdim\d@X=\z@ \ifdim\d@Y=\z@ \DN@{\relax}\fi\fi \next@
\def\Cslidep@@{\noCslidep@@}\def\Cslidec@@{\noCslidec@@}\def\Calong@@{\noCalong@@}\Creset@@
}

\edef\xyEMlinewidth@#1{\dimen@=#1\relax\edef\next@{\the\dimen@}}

\edef\xyemtex@solid@{%
\ifInvisible@ \let\next@=\no@@ \else
\DN@{\xyEMstraight@}
\next@
\fi

Of the curves defined in the \texttt{xycurve} extension, only solid spline curves are supported. This is done by treating the spline as a polygon (poly-line) with many segments.

\edef\xyemtex@splinesolid@{
\setbox\splinebox@=\copy\voidb@x
\multiply\splinetol@3\edef\splineextra@@{\xyemtex@moveto\gdef\splineextra@@{\xyemtex@lineto}}%
\edef\xycrvdrop@{ }\edef\xycrvconn@{ }\splineset@@
}

\edef\xyEMlinewidth@#1{%
\dimen@=#1\relax\edef\next@{\the\dimen@}
\next@
}

\edef\xyEMlinewidth@{1}{\dimen@=#1\relax\edef\next@{\the\dimen@}}

\edef\installEMline@{%

straight lines

spline curves Of the curves defined in the \texttt{xycurve} extension, only solid spline curves are supported. This is done by treating the spline as a polygon (poly-line) with many segments.
CHAPTER 2. EXTENSIONS

\let\xylinewidth@=\xylinewidth@
\let\xylineSpecial@=\xyemtex@pen
\let\transxyline@=\xymathwidth@ }
\xydef@\uninstallEMline@{\UnLoadLine@\relax}\
\xydef@\UseEMlinewidth@{\installEMline@}%
\xydef@\NoEMlinewidth@{\uninstallEMline@}%

line width

The end & Log

\xyendinput

% $Log: xyemtex.doc,v $
% Revision 3.7 2011/03/14 20:14:00 krisrose
% Preparing for release 3.8.6.
%
% Revision 3.6 2010/06/10 18:45:50 krisrose
% Reference to GPL by URL.
%
% Revision 3.5 2010/05/06 17:46:30 krisrose
% Ross Moore’s e-mail address updated.
% Many obsolete files degraded to Historic.
%
% Revision 3.4 2010/04/16 06:06:52 krisrose
% Preparing for a new release...
%
% Revision 3.3 1996/12/18 09:18:00 ross
% checked in with -k by krisrose at 1996/12/18 14:17:11
%
% Revision 3.3 1996/12/18 09:18:00 ross
% fixed bug affecting line-widths
%
% Revision 3.1 1995/09/05 20:28:57 ross
% Releasing version 3.1!
%
% Revision 3.0 1995/07/07 20:13:19 ross
% Major release w/new User’s Guide!
%
% Revision 2.13 1995/07/04 15:04:51 ross
% Ready for release of v3.
%
% NEW for version 3.

2.13 Necula’s extensions

Vers. 3.4 by George C. Necula (necula@cs.cmu.edu)

Header:
This option contains two extensions of the \texttt{Xy-pic} kernel: A way to expand \LaTeX{} macros in object (modifier)s, and a way to specify arbitrary polygons as the (shape) of an object.

\section*{2.13.1 Expansion}

The special syntax \texttt{e|\langle macros\rangle|} is introduced in an object (modifier)s and (coord)inates. It expands the given \LaTeX{} (macros) (with \texttt{\edef}) before reinterpretation as a (modifier) of (coord), respectively.

\begin{verbatim}
def expandbeforenext@#1{\% \\
  \DN@ e|#1|{\edef\tmp@{#1}\expandafter\xyFN@\expandafter\OBJECT@\tmp@}{

\xylet@\xy@oldOBJECT@=\OBJECT@ \\
\xydef@\xy@newOBJECT@{\% \\
  \ifx \space@\next \expandafter\DN@\space\xyFN@\xy@newOBJECT@\%gobble spaces \\
  \else\ifx e\next \\
  \DN@{\xy@oldOBJECT@}\fi\fi \\
  \expandafter\OBJECT@ \\
  \else \\
  \DN@{\xy@oldOBJECT@}\fi\fi \\
  \let\OBJECT@=\xy@newOBJECT@
\end{verbatim}
\xylet@\xy@oldCOORD@=\COORD@
\xydef@\xy@newCOORD@{%
  \ifx \space@\next \expandafter\DN@\space@{\xyFN@\xy@newCOORD@}% gobble spaces
  \else\ifx e\next
  \expandbeforenext@\COORD@
  \else
  \DN@{\xy@oldCOORD@}\fi\fi
  \let\COORD@=\xy@newCOORD@
}

This code may become part of the \texttt{Xy-pic} kernel at a certain point.

2.13.2 Polygon shapes

A polygon \langle shape \rangle is specified as

\[ P: \langle pos \rangle, \ldots, \langle pos \rangle \]

where \[ P: p_1, \ldots, p_n \] denotes the shape obtained by tracking the edge with each \( p_i \) a position relative to the object reference point. \langle vector \rangle s and \langle corner \rangle s can be used directly; otherwise use \texttt{-p} to get the relative position.

\textbf{Note:} Do not use \{\} or [] in the \langle pos \rangle tions.

\textbf{Bug:} The algorithm assumes that the reference point is always inside the polygon.

It is possible to frame polygons is also possible.

% Define the polygon as a stylechar so that it is deferred
\xydefcsname@{*stylechar@P@}#1{\Pshape@#1@}
\xynew@{count}\c@poly@count \c@poly@count=\z@ % Polygon identifiers
\xydef@\Pshape@:#1@{% Strip the mandatory :
  \addtotoks@{%
  \xy@showthe c{Before parse}\xy@showthe p{Before parse}{%
  \xy@showdim{Before parse}{%
  \def\poly@list@{0}{%
  \poly@parse #1,\relax,%Sets the poly@list
  \def\poly@cache@{0}{%
  \edef\poly@id@{\the\c@poly@count@}{% Polygon identifiers
  \global\advance\c@poly@count@\@ne
  \poly@setEdge
  % Save the old values of Lc for shifting
  \edef\poly@saveLcshape@{\the\L@c@}{%
  \polygonEdge@Outer% Set U,R,D,L
  \poly@setEdge
  % Now shift the object to keep it centered
  \dimen@=\poly@saveLcshape@{\the\L@c@}{%
  \polygonEdge@Outer% Set U,R,D,L
  \poly@setEdge
  % Set the edge
  \% A polygonal edge is represented as \polygonEdge x1,y1;x2,y2;\ldots;xn,yn@id@
  % where xi,yi define the vertices of the polygonal edge as displacement from
  % the object reference point (xn=x1,yn=y1), and id is a numerical id used for
% caching the result of the last intersection operation. The cached values for
% polygon id are stored in the global macro poly@cache<id>.
\xydef\poly@setEdge{\expandafter\poly@setEdge@\poly@list\poly@id}
\xydef\poly@setEdge@#1@#2{\Edge@c={\polygonEdge#1@#2@}%
\expandafter\xdef\csname poly@cache#2%
endcsname{\poly@cache}}
% Define the polygon framing operations
\xydefcsname@{frm[P]{-}}{\expandafter\draw@polyframe\the\Edge@c{-}}
\xydefcsname@{frm[P]{.}}{\expandafter\draw@polyframe\the\Edge@c{.}}
\xydefcsname@{frm[P]{=}}{\expandafter\draw@polyframe\the\Edge@c{=}}
\xydef\draw@polyframe\polygonEdge#1@#2@#3{%
def\poly@dir{#3}%
def\poly@list{#1}@%
def\poly@id{#2}%
edef\poly@cache{\csname poly@cache\poly@id\endcsname}{%
draw@polygon}
% A polygon is maintained as a list of relative positions, as follows:
% poly@list = Xdimen_0,Ydimen_0;...;Xdimen_n,Ydimen_n @
% The list is closed, i.e., the first and last element coincide
% Each polygon has an id storred in \poly@id
% Each polygon caches the result of the last intersection operation
% as d@x,d@y,x1,y1,x2,y2 where the first two values define the slope
% and the two sets of points are the two intersection points
% These values are storred in \poly@cache<id>.
%
% Helper function for dealing with polygon lists
% Map over a polygon list. Make sure you define
% \let\poly@map@next=\poly@map before. Then, you can define
% \let\poly@map@next=\poly@map@stop if you want to stop the traversal
\xydef\poly@empty{}
\xydef\poly@map#1,#2,#3;#4@{#1{#2}{#3}%
def\tmp@{#4}%
\ifx \poly@empty\tmp@ \else
\poly@map@next\poly@map@stop \else
\poly@map@next#1#4@%
\fi}
\xydef\poly@map@stop{}
\xydef\poly@mapExpand#1#2{\edef\tmp@{\noexpand\poly@map\noexpand #1#2}%
tmp@}
% Parse a polygon definition
\xydef\poly@parse #1,{#1[#2,#3;#4}@%
#1[#2]{#3}%
def\tmp@{#4}%
\ifx \poly@empty\tmp@ \else
\poly@map@next#1#4@%
\fi}
\xydef\poly@map@stop#1@{}
\xydef\poly@mapExpand#1#2{\edef\tmp@{\noexpand\poly@map\noexpand #1#2}%
tmp@}
% Parse a polygon definition
\xydef\poly@parse #1,{%
\ifx #1\relax %Done. Copy the head of the list at its end
\poly@close
\let\next@=\relax
\else
  % Save everything
  \save@
  % Process the position using POS
  \POS #1%
  % Now add the new c to the list
  \edef\tmp@{{\the\X@c}{\the\Y@c}}%
  \expandafter\poly@append\tmp@
\restore
  \let\next@=\poly@parse % continue parsing
\fi
\next@

% Append to the list of polygon points
\xydef@\poly@append#1#2{\expandafter\poly@append@\poly@list{#1,#2;}}
\xydef@\poly@append@#1@#2{\global\def\poly@list{#1#2@}}

% Close the polygon
\xydef@\poly@close{\expandafter\poly@close@\poly@list}
\xydef@\poly@close@#1,#2;#3@{\poly@append{#1}{#2}}

% Draw a polygon at the current location
\xydef@\draw@polygon{% First separate the head
  % \W@{Drawing polygon \poly@list}\xy@showthe c{ Centered at}%
  \save@
  % Set the origin to point at the reference point
  \X@origin=\X@c \Y@origin=\Y@c
  \poly@setp % Set p at the beginning of the polygon and set poly@rest
  % Zero the edges
  \U@c=\z@ \R@c=\z@ \D@c=\z@ \L@c=\z@
  \U@p=\z@ \R@p=\z@ \D@p=\z@ \L@p=\z@
  \Edge@c={\zeroEdge} \Edge@p={\zeroEdge}%
  \let\poly@map@next=\poly@map
  \poly@mapExpand\poly@drawseg\poly@rest
\restore

% Set p to the first element. Requires the origin to be
% at the reference point
\xydef@\poly@setp{\expandafter\poly@setp@\poly@list}
\xydef@\poly@setp@#1,#2;#3%;
  \X%p=\#1\advance\X%p\X@origin \Y%p=\#2\advance\Y%p\Y@origin%
  \global\def\poly@rest{#3}@}
\xydef@\poly@drawseg#1#2{%
  \dimen@=#1\X@c=\the\dimen@\advance\X@c \X@origin
  \dimen@=#2\Y@c=\the\dimen@\advance\Y@c \Y@origin%
  \W@{Next segment is at offset #1,#2 and absolute \the\X@c,\the\Y@c}%
  \expandafter\connect@\expandafter\dir\poly@dir%
  \X%p=\X@c \Y%p=\Y@c%
% Polygonal edge
2.13. NECULA'S EXTENSIONS

% Requires c to be the reference point
\xydef\polygonEdge1@2@3{%
  \edef\poly@list{#1@}%
  \edef\poly@id{#2}%
  \edef\poly@cache{\csname polygon poly@cache\poly@id\endcsname}%
  \ifcase#3\relax
    \DN@{\polygonEdge@Inters%
      \ifpoly@badinters%
        \xyerror{Could not find intersection for polygon}\fi
      \}}%0
    \or \DN@{\polygonEdge@Under}%1
    \or \DN@{\polygonEdge@Dist}%2
    \or \DN@{\rectangleProp}%3 I do not understand Prop!
    \or \DN@{\polygonEdge@Inner}%4
    \or \DN@{\polygonEdge@Outer}%5
    \else \DN@{}\fi
  \next@
}
\newif\ifpoly@badinters
\xydef\polygonEdge@Inters{%
  \W@{Edge intersection with (\the\X@c, \the\Y@c) \to (\the\X@p,\the\Y@p)}%
  \W@{for polygon: \poly@id}%
  \W@{with list: \poly@list}%
  \W@{with cache: \poly@cache}%
  \Check the cache first
  \ifx\poly@cache\poly@empty
    \poly@intersdoit
  \else
    \expandafter\poly@intersprobecache\poly@cache @%
  \fi
  \xy@showthe c{Inters res}%
}
\xydef\polygon@cachehit#1#2{%
  \W@{intersection point found in cache}%
  \X@c=#1\Y@c=#2}
\xydef\polygon@cachehitdisable#1#2{\poly@intersdoit}
\let\poly@cachehit=\poly@cachehitdiasable% Uncomment this line to disable $\
\xydef\polygon@intersprobecache#1,#2,#3,#4,#5@{%
  \dimen@=#1\advance\dimen@-\d@X
  \ifdim\zz@\dimen@\advance\dimen@-\d@Y
    \poly@cachehit{#3}{#4}
  \else
    \dimen@=#2\advance\dimen@-\d@X
    \poly@cachehit{#3}{#4}
    \else
    \poly@cachehit{#3}{#4}
\end{verbatim}
\poly@intersdoit
\fi
\else
\poly@intersdoit
\fi
}

% Computes the intersection between the line cp and the edge.
% Both intersection points are stored in the cache, the one in the
% direction towards p first.
\xydef@\poly@intersdoit{% 
% \W@{ intersection not in cache. Computing it}%
% Save A@ and B@ because we cannot change them
\edef\polyoldA@{\the\A@}\edef\polyoldB@{\the\B@}%
% Initialize the intersection points
\def\poly@intersneg{}\def\poly@interspos{}%
\save@
\poly@setorigin % Set origin at the beginning of the polygon and
% set poly@rest
\def\zeroDivide@{\poly@badinterstrue}% Handle the division by zero except.
\let\poly@map@next=\poly@map
\poly@mapExpand\poly@interseg\poly@rest
\ifx\poly@intersneg\poly@empty
\poly@badinterstrue
\else
\ifx\poly@interspos\poly@empty
\poly@badinterstrue
\else
\poly@badintersfalse
\fi
\fi
\restore
\A@=\polyoldA@\B@=\polyoldB@ % Restore A@ and B@
\if\poly@badinters \else
\edef\tmp@[\poly@interspos,\poly@intersneg]@%
\expandafter\poly@intersfinish\tmp@
\fi
}

\xydef@\poly@intersfinish#1,#2,#3@{%
\X@c=#1\Y@c=#2\relax
\xdef\poly@cache{\the\d@X,\the\d@Y,#1,#2,#3}%
\poly@setEdge
%
\xydef@\poly@intersfinish#1,#2,#3@% 
\X@c=#1\Y@c=#2\relax
\xdef\poly@cache{\the\d@X,\the\d@Y,#1,#2,#3}%
\poly@setEdge
% Set origin to the first point. Define poly@rest
% Requires c to be set to the reference point
\xydef@\poly@intersdoit{\expandafter\poly@setorigin\poly@list}
\xydef@\poly@intersdoit@#1,#2,#3@{%
\X@c=\poly@origin\X@c=\poly@origin\Y@c=\poly@origin\Y@c%
\global\def\poly@rest{#3}@}
% Compute one intersection with a given edge
% c is set to the reference point, p is set to the end of the ray
% origin is set to the start of the segment
\xydef@poly@interseg#1#2{% 
  % Default is no intersection
  \W@{ Intersection with edge (\the\X@origin,\the\Y@origin) -> (#1,#2)}% 
  % Compute the absolute values
  \dimen@=#1\advance\dimen@\X@c \edef\poly@saveXcinters{\the\dimen@}%
  \dimen@=#2\advance\dimen@\Y@c \edef\poly@saveYcinters{\the\dimen@}%
  \save@
  \poly@badintersfalse
  % Set R@c and U@c to the distance to end of segment
  \R@c=\poly@saveXcinters \advance\R@c -\X@origin
  \U@c=\poly@saveYcinters \advance\U@c -\Y@origin
  % Now call intersect to set X@c and Y@c to the intersection point
  \intersect@
  \A@=\X@c \B@=\Y@c % Save result
  \restore
  \ifpoly@badinters
    % \W@{ \space failed after intersect}%
  \else
    % \W@{ \space after intersect\% (\the\A@,\the\B@)}%
    % Verify that it is on the segment from origin to (#1, #2)
    \poly@isonseg\X@origin\Y@origin\A@\B@\poly@saveXcinters\poly@saveYcinters
    \ifpoly@badinters
      % \W@{ \space failed after seg check}%
    \else
      % Verify that (A,B) is on the ray from c to p
      \poly@isonray\X@c\Y@c\A@\B@\X@p\Y@p
      \ifpoly@badinters
        % It is on the negative ray
        \W@{ \space a negative intersection}%
        \edef\poly@intersneg{\the\A@,\the\B@}%
      \else
        % \W@{ \space a positive intersection}%
        \edef\poly@interspos{\the\A@,\the\B@}%
      \fi
    \fi
  \fi
  % Set the origin to the start of the next segment
  \X@origin=\poly@saveXcinters\Y@origin=\poly@saveYcinters
}\}

% Computes the distance from reference point to the intersection
\xydef@polygonEdge@Dist{\xyerror@{Dist is not yet implemented for polygons}}
% Checks that \#1 <= \#2 <= \#3 or that \#3 <= \#2 <= \#1
% Sets ifpoly@badinters otherwise
% ifpoly@closedrange decides whether the second inequality is also checked
% All checks are done with a precision of 100*almostz@ =~ 5000sp = 0.08pt
\newif\ifpoly@closedrange
\xydef\poly@isinrange#1#2#3{%}
  \ifpoly@badinters \else
    \dimen@=#1\dimen@i=#2\dimen@ii=#3\relax
    \W{check if in \ifpoly@closedrange closed\else open\fi\space
      range 1=\the\dimen@,2=\the\dimen@i,3=\the\dimen@ii}\
    \advance\dimen@i -\dimen@ \advance\dimen@ii -\dimen@ 
    \ifdim\dimen@ii<0pt\relax
      \ifdim\dimen@i>100\almostz@
poly@badinterstrue
    \fi
    \dimen@i=-\dimen@i \dimen@ii=-\dimen@ii
  \else
    \ifdim\dimen@i<-100\almostz@\relax
      \poly@badinterstrue
    \fi
  \fi
  \ifpoly@closedrange
    \advance\dimen@ii 100\almostz@
    \ifdim\dimen@i>\dimen@ii
      \poly@badinterstrue
    \fi
  \fi
  \W{ failed}\else\W{ succeeded}\fi
\fi
%
% Checks that (#3,#4) is on a segment defined by (#1,#2)->(#5,#6)
\xydef\poly@isonseg#1#2#3#4#5#6{%}
  % Check X first
  \poly@closedrangetrue
  \poly@isinrange{#1}{#3}{#5}
  \poly@isinrange{#2}{#4}{#6}
}
%
% Checks that (#3,#4) is on a ray defined by (#1,#2)->(#5,#6)
\xydef\poly@isonray#1#2#3#4#5#6{%}
  % Check X first
  \poly@closedrangefalse
  \poly@isinrange{#1}{#3}{#5}
  \poly@isinrange{#2}{#4}{#6}
}
%
% Test whether (Xp,Yp) is inside the polygon (or on the edge)
% Sets \ifInside@ accordingly
\xydef\polygonEdge@Under{% 
% \W@{}\W@{Edge Under with c=(\the\X@c,\the\Y@c) and p=(\the\X@p,\the\Y@p)}% 
% \W@{ for polygon: \poly@list}% 
% Save A@ and B@ 
\edef\polysaveA@under{\the\A@}\edef\polysaveB@under{\the\B@}% 
% Save X@c and Y@c 
\edef\poly@saveXcUnder{\the\X@c}\edef\poly@saveYcUnder{\the\Y@c}% 
% Compute the intersection 
\polygonEdge@Inters 
\ifpoly@badinters% p is very close to c 
  \Inside@true 
\else 
  \A@=\X@c\B@=\Y@c 
  % Restore c 
  \X@c=\poly@saveXcUnder\Y@c=\poly@saveYcUnder 
  % Now verify that the intersection point is on the ray c->p 
  \poly@isonseg\X@c\Y@c\X@p\Y@p\A@\B@ 
  \ifpoly@badinters \Inside@false \else \Inside@true\fi 
  \A@=\polysaveA@under\B@=\polysaveB@under 
\fi 
% \ifInside@\W@{->inside}\else\W@{->outside}\fi 
% 
% Compute the inner rectangle 
% 
% \xydef\polygonEdge@Inner{% 
% \W@{}\W@{Edge Inner with (\the\X@c, \the\Y@c) -> (\the\X@p,\the\Y@p)}% 
% \W@{ for polygon: \poly@list}% 
% \W@{ with cache: \poly@cache}% 
% Save everything except c 
\enter@{\basefromthebase@ \pfromthep@ \DirectionfromtheDirection@}% 
% Save c in Lc,Dc 
\L@c=\X@c \D@c=\Y@c 
% Compute the two intersection points 
\polygonEdge@Inters 
\ifpoly@badinters 
  \czeroEdge@ 
\else 
  % Save the result in Rc,Uc and in X@c,Y@c 
  \expandafter\poly@getinterspoints\poly@cache 0% 
  % Now compute the inner rectangle centered 
  \ifdim\X@c>R@c 
    \L@c=\R@c \R@c=\X@c 
  \else 
    \L@c=\X@c 
\fi 
  \X@c=0.5\L@c \advance\X@c 0.5 \R@c=\X@c \L@c=\R@c 
  \ifdim\Y@c>U@c 
    \D@c=U@c \U@c=\Y@c
\else
  \D@c=\Y@c
\fi
\Y@c=0.5\D@c \advance\Y@c 0.5\U@c \advance\U@c -\Y@c \D@c=\U@c
\Edge@c={\rectangleEdge}"
\fi
\leave@ % \xy@showthe c{After inner}"
\xydef@\poly@getinterspoints#1,#2,#3,#4,#5,#6{%
  \R@c=#3\U@c=#4\X@c=#5\Y@c=#6}
%
% Compute the outer rectangle (set Uc,Dc,Lc,Rc and Edgec).
% Does not change Xc,YC
%
% \xydef@\polygonEdge@Outer{%
% \W@{Computing outer for \poly@list}\xy@showthe c{Before outer}"
% \W@{Computing outer for \poly@list}\xy@showthe c{Before outer}"
% Save everything except c
\enter@{\basefromthebase@ \pfromthep@ \DirectionfromtheDirection@}"
  \czeroEdge@ % Zero out c
  \let\poly@map@next=\poly@map
  \poly@mapExpand\poly@findextent\poly@list
  \Edge@c={\rectangleEdge}"
\leave@
% \xy@showthe c{After outer}
}
\xydef@\poly@findextent#1#2{
% \xy@showthe c{Before find extent}"
% \W@{ extx=#1, exty=#2}"
  \dimen@=#1\dimen@=\the\dimen@% it fails if I remove the second assign
  \ifdim\dimen@>\R@c \R@c=\dimen@ \fi
  \ifdim -\dimen@>\L@c \L@c=-\dimen@ \fi
  \dimen@=#2\dimen@=\the\dimen@% it fails if I remove the second assign
  \ifdim\dimen@>\U@c \U@c=\dimen@ \fi
  \ifdim -\dimen@>\D@c \D@c=-\dimen@ \fi
% \xy@showthe c{ After extent}"
}
%
% Change to account for polynomial shapes, in addition to circular ones
%
\xydef@\Fshape@#1:{\def\whichframe@@{{#1}}\let\whichoptions@@=\empty
  \DN@{{}}\ifx\whichframe@@\next@ \def\whichframe@@{{-}}\fi
  \expandafter\xyFN@\expandafter\Fshape@whichframe\the\Edge@c}
\xydef@\Fshape@whichframe{%
  \if\circleEdge 
def\whichframe@@{[o]\whichframe@@}%
  \DN@#1{\xyFN@\Fshape@i}%
  \else
    \DN@ #1{\xyFN@\Fshape@i}%
  \fi
  \xydef@\circleEdge 
%
\if\next\circleEdge
  \edef\whichframe@@{[o]\whichframe@@}%
  \DN@ #1{\xyFN@\Fshape@i}%
  \else
    \DN@ #1{\xyFN@\Fshape@i}%
  \fi
\xydef@\polygonEdge
2.14. LATEX PICTURE EXTENSION

This extension provides replacement commands for the LaTeX picture environment commands line and vector. At the moment this option requires \LaTeX.  

\edef\whichframe@\{[P]\whichframe@\}%  
\DN@\polygonEdge##1##2\{\xyFN@\Fshape@i}%  
\else  
\DN@##1\{\xyFN@\Fshape@i}%  
\fi\fi  
\next@  
)  
\xyendinput

**Bug:** This code should be merged with the ‘frame’ and ‘poly’ options. The example at the end of §3.11 illustrates the extensions.

2.14 LaTeX Picture extension

**Vers. 3.6 by Kristoffer H. Rose** (krisrose@tug.org)

This extension provides replacement commands for the LaTeX picture environment commands line and vector. At the moment this option requires \LaTeX. 

%% $Id: xypicture.doc,v 3.6 2011/03/14 20:14:00 krisrose Exp $  
%%  
%% Xy-pic ‘‘LaTeX Picture Mode’’ option.  
%% Copyright (c) 199802011 Kristoffer H. Rose <krisrose@tug.org>  
%%  
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.  
%% See the companion README and INSTALL files for further information.  
%% Copyright (c) 1991-1998 Kristoffer H. Rose <krisrose@tug.org>  
%%  
%% The Xy-pic package is free software; you can redistribute it and/or modify  
%% it under the terms of the GNU General Public License as published by the  
%% Free Software Foundation; either version 2 of the License, or (at your  
%% option) any later version.  
%%  
%% The Xy-pic package is distributed in the hope that it will be useful, but  
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
Next define some internal parameters.

```
\xynew@{dimen}\xyp@xsize
\xynew@{dimen}\xyp@ysize
\xylet@\xyp@arrow=\relax
{\xyuncatcodes\makeatletter
\gdef\xyp@tail{}
\gdef\xyp@mid{-}
\gdef\xyp@head{>}
}
```

Finally the actual command.

```
\xydef@\xyvector(#1,#2)#3{% 
\xyp@xsize#3\relax
\ifnum#1<\z@\multiply\xyp@xsize\m@ne\fi
\xyp@ysize\xyp@xsize\relax
\ifnum#1=\z@
 \xyp@xsize\z@
 \ifnum#2<\z@\multiply\xyp@ysize\m@ne\fi
 \else
 \multiply\xyp@ysize#2\relax\divide\xyp@ysize#1\relax
 \fi
\edef\xyp@arrow{\noexpand\arAT{\xyp@tail\xyp@mid\xyp@head}}%
\ifnum#1<\z@
 \makebox(\xyp@xsize,\xyp@ysize){%
 \begin{xy} 0;<\unitlength,\z@>:%
 \xyp@arrow(\xyp@xsize,\xyp@ysize)
 \end{xy}}%
 \else
 \xyp@arrow(\xyp@xsize,\xyp@ysize)\relax
 \fi
}
```

Then the option ends:

```
\xyendinput
```

Finally xypicture.doc is maintained using RCS and thus contains the following revision log:

```
% $Log: xypicture.doc,v $
% Revision 3.6  2011/03/14 20:14:00  krisrose
```
% Preparing for release 3.8.6.
%
% Revision 3.5  2010/06/10 18:45:50  krisrose
% Reference to GPL by URL.
%
% Revision 3.4  2010/04/17 14:45:48  krisrose
% Generate and extract Type1 fonts.
%
% Revision 3.3  2010/04/17 04:19:41  krisrose
% Integrated xylu tips by Jeremy Gibbons.
%
% Revision 3.2  2010/04/16 06:06:52  krisrose
% Preparing for a new release...
%
% Revision 3.1  2010/04/13 08:10:26  krisrose
% Up to date with Kris’ development directory.
%
% Based on xypicture.doc 3.3 1996/12/19 03:31:56
Chapter 3

Features

This chapter describes the options that support facilities that can be obtained using the kernel and extensions yet are much easier to obtain using the provided special syntax.

3.1 All features

Vers. 3.8 by Kristoffer H. Rose (krisrose@tug.org)

As a special convenience, this feature loads a subset of Xy-pic, namely the extensions: curve (cf. §2.1), frame (§2.2), tips (§2.3), line (§2.4), rotate (§2.5), color (§2.6), and the following features: matrix (§3.5), arrow (§3.3), and graph (§3.6).

The name ‘all’ hints at the fact that these were all the available options at the time ‘all’ was added.
Here we go:

\xyrequire{curve}
\xyrequire{frame}
\xyrequire{cmtip}
\xyrequire{line}
\xyrequire{rotate}
\xyrequire{color}
\xyrequire{matrix}
\xyrequire{arrow}
\xyrequire{graph}

That is all.

\xyendinput

% $Log: xyall.doc,v $
% Revision 3.8 2011/03/14 20:14:00 krisrose
% Preparing for release 3.8.6.
% 
% Revision 3.7 2010/07/27 09:49:34 krisrose
% Started xyling (and address updates).
% 
% Revision 3.6 2010/06/10 18:45:49 krisrose
% Reference to GPL by URL.
% 
% Revision 3.5 2010/05/17 23:29:21 krisrose
% Experiment: generate all the Type1 fonts with METAPOST.
% 
% Revision 3.4 2010/04/16 06:06:52 krisrose
% Preparing for a new release...
% 
% Revision 3.3 1996/12/19 03:31:56 krisrose
% Maintenance release
% 
% Revision 3.0 1995/07/07 20:14:21 kris
% Major release w/new User’s Guide!
% 
% Revision 2.13 1995/07/04 15:11:17 kris
% Ready to release v3?
% 
% Revision 2.12 1994/10/25 11:34:25 kris
% Interim release just before v3 [works with AMS-LaTeX 1.2]...
% 
% Revision 2.11 1994/07/05 10:37:32 kris
% Third 3beta release [bug fixes].
% 
% Experimental graph feature included (for ECCT-94 presentation).
% 
% Revision 2.9 1994/06/09 14:59:19 kris
3.2 Dummy option

Vers. 3.7 by Kristoffer H. Rose (krisrose@tug.org)

This option is provided as a template for new options, it provides neither features nor extensions but it does count how many times it is requested. Even though the option does nothing it still has a standard \texttt{Xy-pic} header:

```latex
\% $Id: xydummy.doc,v 3.7 2011/03/14 20:14:00 krisrose Exp$
\%\%
\% \texttt{Xy-pic} `Dummy' option.
\% Copyright (c) 1993-1996 Kristoffer H. Rose <krisrose@tug.org>
\%\%
\% This file is part of the \texttt{Xy-pic} package for graphs and diagrams in \texttt{TeX}.
\% See the companion \texttt{README} and \texttt{INSTALL} files for further information.
\% Copyright (c) 1991-2011 Kristoffer H. Rose <krisrose@tug.org>
\%\%
\% The \texttt{Xy-pic} package is free software; you can redistribute it and/or modify
\% it under the terms of the GNU General Public License as published by the
\% Free Software Foundation; either version 2 of the License, or (at your
\% option) any later version.
\%\%
\% The \texttt{Xy-pic} package is distributed in the hope that it will be useful, but
\% \textsc{without any warranty}; without even the implied warranty of \textsc{merchantability}
\% or \textsc{fitness for a particular purpose}. See the GNU General Public License
\% for more details.
\%
\% You should have received a copy of the GNU General Public License along
\% with this package; if not, see http://www.gnu.org/licenses/.
\%\%
```

Next follows two declarations that are used to count the number of invocations:

\new@{count}\xydummyrequires@@

This is used by the option to output a message every time it is loaded with \texttt{\textbackslash xyoption} or \texttt{\textbackslash xyrequire}:

\everyrequest{dummy}{\global\advance\xydummyrequires@@\@ne
\texttt{The \textasciitilde \textquoteleft\textasciitilde dummy\textquoteleft\textasciitilde option was used \textbackslash the\textbackslash xydummyrequires@@\textbackslash space times!}}

(\texttt{\textbackslash W@} is an \texttt{Xy-pic} primitive that outputs its argument.)

Then the option ends:
Finally \textit{xyzdummy.doc} is maintained using RCS and thus contains the following revision log:

\verbatiminput{xyzdummy.doc}

\section{3.3 Arrow and Path feature}

\textit{Vers. 3.9 by Kristoffer H. Rose (krisrose@tug.org)}

This feature provides \textsc{Xy-pic} with the arrow paradigm presented in [13].

\textbf{Note:} \texttt{PATH} command incompatibly changed for version 3.3 (the \texttt{ar} command is unaffected).

The basic concept introduced is the \textit{path}: a connection that \textit{starts} from \texttt{c} (the current object), \textit{ends} at a specified object, and may be split into several \textit{segments} between intermediate specified objects that can be individually labelled, change style, have breaks, etc.

\S 3.3.1 is about the \texttt{PATH} primitive, including the syntax of paths, and \S 3.3.2 is about the \texttt{ar} customisation of paths to draw arrows using \textsc{Xy-pic} directional objects.

\textbf{Header:}

\verbatiminput{xyarrow.doc}

\verbatiminput{xyendinput}
3.3. ARROW AND PATH FEATURE

The fundamental commands of this feature are \PATH and \afterPATH that will parse the (path) according to the grammar in figure 3.1 with notes below.

Parsing: \afterPATH sets up \afterPATH@ and initialise all the actions (see note 3.3a below) before invoking the actual (path) parser. \PATH is just a dummy interface.
CHAPTER 3. FEATURES

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>\PATH (path)</td>
<td>interpret (path)</td>
</tr>
<tr>
<td>\afterPATH{ &lt;decor&gt; } (path)</td>
<td>interpret (path) and then run &lt;decor&gt;</td>
</tr>
<tr>
<td>(path) → ~ (action) { (stuff) } (path)</td>
<td>set (action)\textsuperscript{3.3a} to (stuff)</td>
</tr>
<tr>
<td></td>
<td>~ (which) { (labels) } (path)</td>
</tr>
<tr>
<td></td>
<td>~ { (stuff) } (path)</td>
</tr>
<tr>
<td></td>
<td>` (segment) (path)</td>
</tr>
<tr>
<td></td>
<td>` (turn) (segment) (path)</td>
</tr>
<tr>
<td></td>
<td>` (segment)</td>
</tr>
<tr>
<td>(turn) → (diag) (turnradius)</td>
<td>1/4 turn\textsuperscript{3.3f} starting in (diag)</td>
</tr>
<tr>
<td></td>
<td>(cir) (turnradius)</td>
</tr>
<tr>
<td>(turnradius) → (empty)</td>
<td>use default turn radius</td>
</tr>
<tr>
<td></td>
<td>/ (dimen)</td>
</tr>
<tr>
<td>(segment) → (path-pos) (slide) (labels)</td>
<td>segment\textsuperscript{3.3e} with (slide) and (labels)</td>
</tr>
<tr>
<td>(slide) → (empty)</td>
<td>&lt; (dimen) &gt;</td>
</tr>
<tr>
<td>(labels) → ~ (anchor) (it) (alias) (labels)</td>
<td>label with (it)\textsuperscript{3.3i} \textit{above} (anchor)</td>
</tr>
<tr>
<td></td>
<td>~ (anchor) (it) (alias) (labels)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>(anchor) → ~ (anchor)</td>
<td>(place)</td>
</tr>
<tr>
<td>(it) → (digit)</td>
<td>(letter)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>(alias) → (empty)</td>
<td>=&quot;(id)&quot;</td>
</tr>
</tbody>
</table>

Figure 3.1: (path)s

\PATH is the parser for (path): it eats the (path) until it fails and then calls the ‘failure continuation’ which will eventually become \texttt{empty} which ends the parsing. \textbf{Hack:} The ‘ifPATHsingle’ switch switches the (segment) parsing \texttt{off}; this is used by the graph feature to parse stand-alone (slide) (labels) sequences.
3.3. ARROW AND PATH FEATURE

\PATHsetting@ just reads an ⟨action⟩ or ⟨which⟩ determining which action stuff to set and then does it—we treat the failure continuation as an action here even though it is not, strictly speaking. The four actions are explained in the appropriate notes below.

Next the setup for straight segments including the last: they set \PATHinit@@ to initialise, \PATHextra@@ to do what is needed after the = action, and finally \PATHpost@@ is set to any operations to be done after the entire segment is typeset but before the next is read. \PATHcontinue@@ is used internally to distinguish the ordinary segments from the last.

\PATHstraight@
Setup for turning segments is in note 3.3f.

A ⟨segment⟩ is interpreted as follows after \( p \) has been set to the previous end object, \( c \) to the ⟨pos⟩ given at the start of the segment, and \( \text{PATHslide}@{} \) to the slide:

1. Expand \( \text{PATHinit}@{} \) (for straight segments this just sets up the direction and applies the ⟨slide⟩).

2. Expand \( \text{PATHaction}={} \langle \text{stuff} \rangle \) as set using \( ~={} \langle \text{stuff} \rangle \).

3. Sets \( \text{PATHcontinue}@{} \) to any continuation set with \( ~={} \langle \text{stuff} \rangle \).

4. Construct list of ⟨labels⟩ specified by the user through \( ~={} \langle \text{which} \rangle \) setup. Those applicable to the present segment are inserted before the user’s ⟨labels⟩ in the sequence \( <> \).

5. Store \( p, c \) as \textit{start,end} of segment.

6. Interpret ⟨labels⟩ (see below), including the ones added by the \( ~={} \langle \text{when} \rangle \ldots \langle \text{setup} \rangle \).

\( \text{PATHsegment}@{} \) performs this except the last two points:
\expandafter\def\expandafter\PATHlabels@@\expandafter{\the\toks@}\	oks@={}\expandafter\xyFN@\expandafter\PATHlabels@\PATHlabels@@}

\PATHlabels@ parses all \langle labels \rangle of a \langle segment \rangle. This sets \PATHlabelit@@ to the operation building the label in question. Then some parsing stores the tokens \langle \rangle for each - in the \langle anchor \rangle and finally passes control to the kernel (place) parser with these tokens as the head. This then in turn calls \PATHit@ that parses \langle it \rangle and applies the chosen operation. After the last we continue with the accumulated subsegment actions.

\expandafter\def\expandafter\PATHlabels@{\%\begin{grouping \toks@={}\PATHanchor@i}\begin{grouping}

The individual operations bound to \PATHlabelit@@ are discussed as appropriate in the notes.

After the last break and label we typeset the last piece of the connection.
Bug: The order of the breaks determines the order of the subsegments. This is maybe a feature.

Notes

3.3a. An \langle action\rangle can be either of the characters =/. The associated \langle stuff\rangle is saved and used to call

\PATHaction\langle action\rangle\{\langle stuff\rangle\}

before and after each segment (including all \langle labels\rangle) for = and /, respectively.

The default \PATHaction macro just expands to “\POS \langle stuff\rangle \relax” thus \langle stuff\rangle should be of the form \langle pos\rangle \langle decor\rangle. The user can redefine this—in fact the \ar command described in §3.3.2 below is little more than a special \PATHaction command and a clever defaulting mechanism.

Here is the default \PATHaction:

\xydef\PATHaction@default#1#2\{\xy\{\PATHaction#1\{#2\}\}\POS#2\relax\}
\xylet\PATHaction=\PATHaction@default

It is called using \PATHaction@ to expand the action control sequences back to their \langle stuff\rangle first:

\xydef\PATHaction@1#2\{\expandafter\PATHaction\expandafter\#1\expandafter\#2\}

3.3b. It is possible to include a number of default \langle labels\rangle before the \langle labels\rangle of the actual \langle segment\rangle are interpreted, using \~\langle which\rangle\{\langle labels\rangle\}. The specified \langle which\rangle determines for which segments the indicated \langle labels\rangle should be prefixed as follows:

<table>
<thead>
<tr>
<th>(which)</th>
<th>applied to...</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>next segment only</td>
</tr>
<tr>
<td>&gt;</td>
<td>last segment only</td>
</tr>
<tr>
<td>=</td>
<td>every segment</td>
</tr>
</tbody>
</table>

(when several apply to the same segment they are inserted in the sequence <\,+>).

This is useful to draw connections with a ‘center marker’ in particular with arrows, \textit{e.g.}, the ‘mapsto’ example explained below can be changed into a ‘breakto’ example: typing

\xy**{0}\PATH
\~={**\dir{-}}
\~>{|*>\dir{>}}
\~+{|*\dir{/}}
'(10,1)**{1} '。(20,-2)**{2} (30,0)**{3}
\endxy

will typeset

\[ 0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \]
3.3. ARROW AND PATH FEATURE

Note, however, that what goes into \(
\{\ldots\}
\) is \langle labels \rangle and thus not a \langle pos \rangle – it is not an action in the sense explained above.

3.3c. Specifying \(
\{\text{stuff}\}\) will set the “failure continuation” to \langle stuff \rangle. This will be inserted when the last \langle segment \rangle is expected—it can even replace it or add more \langle segment \rangle s, i.e.,

\[
\begin{xy}
*+{0} \PATH \xymatrix{ \big({\bullet} \ar@{-}[r] & 1 \big) \ar@{-}[r] & 2 \ar@{-}[r] & 3}
\end{xy}
\]

is equivalent to

\[
\begin{xy}
*+{0} \PATH \xymatrix{ \big({\bullet} \ar@{-}[r] & 1 \big) \ar@{-}[r] & \big(20,-2\big) \ar@{-}[r] & 2 \ar@{-}[r] & (30,0) \ar@{-}[r] & 3}
\end{xy}
\]

typesetting

0 \longrightarrow 1 \longrightarrow 2 \longrightarrow 3

because when \endxy is seen then the parser knows that the next symbol is neither of the characters \(\sim\) and hence that the last \langle segment \rangle is to be expected. Instead, however, the failure continuation is inserted and parsed, and the \langle path \rangle is finished by the inserted material.

Failure continuations can be nested:

\[
\begin{xy}
*+{0} \PATH \xymatrix{ \big({\bullet} \ar@{-}[r] & \big(30,0\big) \ar@{-}[r] & \big(20,-2\big) \ar@{-}[r] & 2 \ar@{-}[r] & (30,0) \ar@{-}[r] & 3}
\end{xy}
\]

will also typeset the connected digits.

3.3d. A “straight segment” is interpreted as follows:

1. First \(p\) is set to the end object of the previous segment (for the first segment this is \(c\) just before the path command) and \(c\) is set to the \langle pos \rangle starting the \langle segment \rangle, and the current \langle slide \rangle is applied.
2. Then the = and < segment actions are expanded (in that sequence) and the < action is cleared. The resulting \(p\) and \(c\) become the start and end object of the segment.
3. Then all \langle labels \rangle (starting with the ones defined as described in note 3.3b below).

The code expanding the actions is part of the parsing above.

3.3e. A segment is a part of a \langle path \rangle between a previous and a new target given as a \langle path-pos \rangle: normally this is just a \langle pos \rangle as described in §1.3 but it can be changed to something else by changing the control sequence \PATHafterPOS to be something other than \afterPOS.

\begin{xymatrix}{\texttt{xylet}@\PATHafterPOS@default=\afterPOS \texttt{xylet}@\PATHafterPOS=\PATHafterPOS@default} \end{xymatrix}

3.3f. A turning segment is one that does not go all the way to the given \langle pos \rangle but only as far as required to make a turn towards it. The \(c\) is set to the actual turn object after a turning segment such that subsequent turning or other segments will start from there, in particular the last segment (which is always straight) can be used to finish a winding line.

What the turn looks like is determined by the \langle turn \rangle form:
CHAPTER 3. FEATURES

Nothing between the \< and the \<pos> is interpreted the same as giving just the \<diag> last used out of a turn.

Specifying a single \<diag> \d is the same as specifying either of the \<cir>cles \d^ or \d_\ _, depending on whether the specified \<pos> has its center ‘above’ or ‘below’ the line from \p in the \<diag>onal direction.

When a full explicit \<cir>cle is available then the corresponding \<cir>cle object is placed such that its ingoing direction is a continuation of a straight connection from \p and the outgoing direction points such that a following straight (or last) segment will connect it to \c (with the same slide).

Here is an example using all forms of \<turn>s:

was typeset by

\xy <4pc,0pc>:(0,0)
**\txt{base}="base"
\PATH \-{**dir{-}?>*\dir{}}
 ' l (-1,-1)*{A} \^a
 ' (1,-1)*{B} \^b
 '_ul (1, 0)*{C} \^c
'ul\^l "base" \^d
"base" \^e
\endxy

**Bug**: Turns are only really reasonable for paths that use straight lines like the one above.

**Note**: Always write a valid \<pos> after a \<turn>, otherwise any following ^ or _ labels can confuse the parser. So if you intend the ^r in ^r to be a label then write ',^r, using a dummy \<pos>ition.

We need to keep track of the current in and out \<diag> directions as well as the current radius.

\xydef@\PATHlastout@@{3}

The startup routine is just a selection of the two kinds of \<turn>:

\xydef@\PATHturn@{\afterCIRorDIAG\PATHturn@cir\PATHturn@diag}

In both cases the trick is to set up a \PATHinit@@ method that typesets the right circle segment at the right location. The simplest one is when there is a full \<cir> available because that should just be used directly. The only complication is that we forbid 180° turns:

\xydef@\PATHturn@cir\{\toks@={\xy@{%
\count@=\CIRin@@ \ifnum\count@<4\else\advance\count@-4\fi
\count@@=\CIRout@@ \ifnum\count@@<4\else\advance\count@@-4\fi
\ifnum\count@=\count@@ \xyerror@{<turn> cannot be half or full}{% You asked for a <turn>ed segment with parallel in- and out-direction.^^J%
This is not allowed because it is not possible to position it uniquely.}}\fi\}%%
3.3. ARROW AND PATH FEATURE

Procedure for \(\text{\langle diag\rangle\ turns}\): Builds initial part of \(\text{\PATHinit}\) that computes the missing \(\text{orient}\) and \(\text{out}\) from \(\text{in}\) and the \(c\) passed through the segment. This initial segment should set \(\text{orient}\) as follows:

\[ \text{orient} = \text{sign of } \langle p \rangle \cdot \langle \text{in} \rangle, \]

\[ \text{set } \text{out} \text{ as for the } \langle \text{cir} \rangle \text{cle in orient}. \]

\[ \text{i.e., as the sign of the inner product } \langle p \rangle \cdot \langle \text{in} \rangle. \]

\[ \text{set } \text{out} \text{ as for the } \langle \text{cir} \rangle \text{cle in orient}. \]

\(\langle p \rangle\) is where \(\text{\PATHinit}\) is defined to do the following:

1. Setup the default \(\text{in}, \text{orient}, \text{and } \text{out}\) as defined by the call from the above macros. Set \(\text{radius}\) parameter to the current \(\text{turnradius}\) value. The commands in \(\text{\toks@}\) are run to do any last-minute fixing of the defaults.

2. The segment will leave \(p\) in direction \(\text{in}\) and pretend to be meeting \(c\) in direction \(\text{out}\). Thus the center of the \(\langle \text{cir} \rangle \text{cle object to be used for the actual } \langle \text{turn} \rangle \text{ object is located at the intersection of the line through } p + \langle \text{radius}\rangle \langle \text{in}\rangle \text{ in direction } \langle \text{in}\rangle \text{ and the line through } c + \langle \text{radius}\rangle \langle \text{out}\rangle \text{ in direction } \langle \text{out}\rangle. \]

3. Here we compute \(p + R\langle \text{in}\rangle\) using \(R = \langle \text{radius}\rangle\) if this is the initial turn and \(R = \langle \text{radius}\rangle \pm \langle \text{slide}\rangle\) if it is a continuation turn.

4. Save \(p\) and \(\text{base}\) for later.

5. Compute \(c + \langle \text{radius}\rangle \langle \text{out}\rangle\) and use the kernel \(\text{\intersect}\) primitive to compute the turn circle segment center.

6. However, if the intersection point lies on the ‘wrong side’ of \(p\), i.e., if \(\langle p \rangle \cdot \langle \text{in}\rangle\) is negative, then we should not use the intersection point but the point as close to \(p\) as possible because the connection from \(p\) to the turn should at least have zero length—in fact we set it to have \(8sp\) length to make sure that a sugsegment is typeset!

7. Modify \(\text{radius}\) to use for the actual circle segment by the \text{slide}: either positively (for \text{orient } _\circlearrowleft\) or negatively (for \text{orient } _\circlearrowright), and drop the circle segment (bypassing \text{\cir} to use the internal settings from above).

8. Store in \(\text{\PATHpostpos}\) the true end of the circle segment and code to prevent the application of the \text{slide} for the start of the next \(\langle \text{segment}\rangle\).

9. Move \(c\) to true beginning of the circle segment.

10. Restore saved \(p\) and \(\text{base}\), and slide \(p\) finally.
3.3. ARROW AND PATH FEATURE

The final macro takes care of the v2.6 format for changing \textit{turnradius}: using \textbackslash\langle\textit{dimen}\rangle right after the \langle\textit{turn}\rangle.

The default used for \textit{turnradius} can be set by the operation

```
\texttt{\textbackslash turnradius \langle add op \rangle \{ \langle dimen\rangle \}}
```

that works like the kernel \texttt{\textbackslash objectmargin etc.} commands; it defaults to 10pt.

Exercise 3.1: Typeset \begin{center} \includegraphics[width=0.2\textwidth]{figure.png} \end{center} using \langle\textit{turn}\rangle s. (p.579)

3.3g. The last segment is exactly as a straight one except that the > action (if any) is executed (and cleared) just after the < action.

(The code implementing this is merged with the parsing.)

3.3h. “Sliding” a segment means moving each of the \textit{p}, \textit{c} objects in the direction perpendicular to the current direction at each.

(The code implementing this is merged with the parsing, using the \texttt{\textbackslash PATHslide@@} control sequence to hold the current slide.)
3.3i. Labelling means that \(\langle \text{it} \rangle\) is dropped relative to the current segment using a ** \(\langle \text{pos} \rangle\)** action as one of the actions—typically the = action is used for this (see note 3.3d for the details). The only difference between ^ and _ is that they shift the label in the ^ respectively _ direction; for straight segments it is placed in the “superscript” or “subscript” position.

The operations used by \PATHlabels to put labels above and below (with ^ and _) are trivial, taking two arguments and placing the label immediately; for convenience the \drop head are named separately such that they may be redefined:

```latex
\xydef@\PATHlabelabove@#1#2{\droplabel@\belowDirection@{#1}{#2}% \\
\let\afteraliases@@=\empty \xyFN@\PATHlabelalias@}
\xydef@\PATHlabelbelow@#1#2{\droplabel@\aboveDirection@{#1}{#2}\
\let\afteraliases@@=\empty \xyFN@\PATHlabelalias@}
\xylet@\PATHlabelabove@@=\PATHlabelabove@
\xylet@\PATHlabelbelow@@=\PATHlabelbelow@
\xydef@\droplabel@#1#2#3{\xy@@{\enter@\DirectionfromtheDirection@}\
\DN@{#2}\ifx\next@\empty \expandafter\xy@@ix@\expandafter{\PATHitshape@@\labelbox{#3}}% \else \expandafter\xy@@ix@\expandafter{\PATHitshape@@#2{#3}}\fi
\xy@@{\setbox\z@=\expandafter\object\the\toks9 %}
\advance\L@c\labelmargin@ \advance\R@c\labelmargin@ \advance\D@c\labelmargin@ \advance\U@c\labelmargin@
\setboxz@h{\kern\labelmargin@\boxz@\kern\labelmargin@}
\ht\z@=\U@c \dp\z@=\D@c \dimen@=\L@c \advance\dimen@\R@c \wd\z@=\dimen@ 
\drop@{\box\lastobjectbox@}{%}
\leave@}}
\xydef@\PATHlabelbreak@#1#2{% \\
\DN@{#2}\ifx\next@\empty \drop\labelbox{#2}\else \drop#1{#2}\fi}
\def\afteraliases@@{\xy@@\Cbreak@@}
```

These emulate the effect of the \(\langle \text{modi}\text{fier} \rangle\)s \!C+<2\labelmargin\> \!P \langle \text{perp} \rangle \langle \text{unperp} \rangle where \langle \text{perp} \rangle, \langle \text{unperp} \rangle are ^, _ for ^-labels and _, ^ for _-labels. The last hack using \lastobjectbox aims at both setting that right and setting the label as an object (such that modifiers take effect) even though its placement is peculiar.

Labels will be separated from the connection by the labelmargin that you can set with the operation

```
\labelmargin \langle \text{add op} \rangle \{\langle \text{dimen} \rangle\}
```

that works like the kernel \objectmargin command; in fact labelmargin defaults to use objectmargin if not set.

```latex
\xylet@\labelmargin@=\objectmargin@
\xydef@\labelmargin{\afterADDOP{\Addop@\labelmargin@}}
```

3.3j. Breaking means to “slice a hole” in the connection and insert \(\langle \text{it} \rangle\) there. This is realized by typesetting the connection in question in subsegments, one leading to the break and one continuing after the break as described in notes 3.3a and 3.3d.
3.3. ARROW AND PATH FEATURE

The special control sequence \hole is provided to make it easy to make an empty break.

Subsegments are typeset using the \Cbreak@@ method.

3.3k. Unless (it) is a full-fledged \langle object \rangle (by using the * form), it is typeset using a \labelbox object (initially similar to \objectbox of basic XY-pic but using \labelstyle for the style).

Remark: You can only omit the {}s around single letters, digits, and control sequences.

A label is an object like any other in the $XY$-picture. Inserting an \langle alias \rangle ="\langle id \rangle" saves the label object as "\langle id \rangle" for later reference.

Exercise 3.2: Typeset

\[
\begin{array}{c}
\text{label} \\
\hline
\end{array}
\]

\[\text{A} \]

3.3.2 Arrows

Arrows are paths with a particularly easy syntax for setting up arrows with tail, stem, and head in the style of \[13\]. This is provided by a single \langle decor \rangle ation the syntax of which is described in figure 3.2 (with the added convention that a raised '*' means 0 or more repetitions of the preceding nonterminal).

Parsing: The \ar command effectively translates the \langle arrow \rangle into a \PATH command that initialises the tail, stem, and head of it as required to get the desired \langle arrow \rangle.

The translation of the \langle arrow \rangle is saved in the following variables:
\texttt{\ar} stores the defaults; it loads an intermediate command to avoid problems when used as first command in compiled matrix entry...

\texttt{\xydef@{\relax\arSAFE}}

\texttt{\xydef@{\relax\arSAFE{\%}}}

\texttt{\let\arvariant@@=\empty}

\texttt{\edef\arstem@@={\arvariant@@{-}}}

\texttt{\edef\arhead@@={\arvariant@@{>}}}

\texttt{\def\armodifiers@@{}}

\texttt{\def\arafterPOS@@{}}

\texttt{\def\arlabels@@{}}

\texttt{\def\arinit@@{}}

\texttt{\def\arexit@@{}}

\texttt{\let\PATHlabelabove@@=} \texttt{\PATHlabelabove@}

\texttt{\let\PATHlabelbelow@@=} \texttt{\PATHlabelbelow@}

\texttt{\xyFN@\ar@}

\texttt{\ar@} is where we parse the sequence of \texttt{⟨form⟩s} and continue with typesetting the arrow afterwards.

\texttt{\xydef@{\relax\ar@{\%}}}

\texttt{\ifx \space@ \next \expandafter\DN@\space{\xyFN@\ar@}{\space@}}

\texttt{\else \addAT@\next \addAT@\DN@{\xyFN@\ar@\form}{\space@}}

\texttt{\else\ifx \next \DN@ \\\{\ar@anchor\}\\}

\texttt{\else\ifx \next \DN@ \\{\ar@anchor^{-}\}\\}

\texttt{\else\ifx \next \DN@ \\{\ar@anchor_{\!}\}\\}

\texttt{\else \let\next@=\ar@x \fi\fi\fi\fi\fi\fi\fi\fi\fi

\texttt{\Note: Makes use of the \textit{new compilation principle} to parse a \langle place\rangle vacuously...}

\texttt{\xydef@{\addtoarinit@\#1{\%}}}

\texttt{\expandafter\def\expandafter\arinit@@\expandafter{\arinit@@ #1}}

\texttt{\xydef@{\addtoarlabels@\#1{\%}}}

\texttt{\expandafter\def\expandafter\arlabels@@\expandafter{\arlabels@@ #1}}

\texttt{\xydef@{\ar@anchor\#1{\begingroup}}}

\texttt{\def\PATHlabelit@@##1##2{}}

\texttt{\let\PATHlabelabove@@=} \texttt{\PATHlabelabove@}

\texttt{\let\PATHlabelbelow@@=} \texttt{\PATHlabelbelow@}

\texttt{\xyFN@\ar@}}
3.3. ARROW AND PATH FEATURE

\ar@form handles one (form):
\begin{verbatim}
\xydef@ar@form{% 
  \def\defarstem@#1{\edef\arstem@@{\arvariant@@{\string#1}}% 
    \DNii@##1##2\next@##1##2\next@}\
  \edef\next@##1{\def\noexpand\artail@@{\arvariant@@{##1}}}% 
  \expandafter\artail@@\
  \edef\next@##1{\def\noexpand\arhead@@{\arvariant@@{##1}}}% 
  \expandafter\arhead@@}
\end{verbatim}

\ar@i sets the remaining defaults and then allows \ar@ii-iv to pick up any tail, stem, and head using a $ as terminator.

\xydef@\ar@style{%
\ifx \bgroup\next \def\artail@@{{}}%
\edef\arstem@@{\arvariant@@{-}}\edef\arhead@@{\arvariant@@{>}}%
\expandafter\ar@i
\else \resetvariant@\artail@@ \resetvariant@\arstem@@ \resetvariant@\arhead@@
\expandafter\xyFN@\expandafter\ar@ \fi}
\xydef@\resetvariant@#1{%
\DN@##1\ifx \next@\empty
\DN@{\expandafter\nextii@\expandafter{##1}}%
\else \DN@{\nextii@{##1}}%
\fi \next@}
\DNii@##1##2{\def#1{##1{##2}}}
\expandafter\next@#1}
\xydef@\ar@i#1{\DN@{#1}%
\ifx \next@\empty \edef\arstem@@{\arvariant@@{}}\edef\arhead@@{\arvariant@@{}}%
\DN@{\xyFN@\ar@}%
\else
\let\arcomponent@@=\ar@ii \let\arcomponenttype@@=\artip@
\DN@{\xyFN@\arcomponent@#1$}%
\fi \next@}
\xydef@\ar@ii{%
\ifx $\next \let\next@=\ar@iv
\else \expandafter\def\expandafter\artail@@\expandafter{\the\toks@}%
\let\arcomponent@@=\ar@iii \let\arcomponenttype@@=\arconn@
\DN@{\xyFN@\arcomponent@#1}%
\fi \next@}
\xydef@\ar@iii{%
\expandafter\edef\expandafter\arstem@@\expandafter{\the\toks@}
\resetvariant@\artail@@
\let\arcomponent@@=\ar@iv \let\arcomponenttype@@=\artip@
\xyFN@\arcomponent@}
\xydef@\ar@iv{%
\expandafter\edef\expandafter\arhead@@\expandafter{\the\toks@}
\ifx $\next \DN@{\xyFN@\ar@}%
\else \xyerror@{illegal <arrow>: \meaning\next\space not valid}{}
\fi \next@}

\ar@x is the final macro that passes control to the ⟨path⟩ interpreter. It is a bit clumsy because we must pass the right character tokens; it could definitely be more efficient by calling ⟨path⟩ internals directly...

\xydef@\ar@x{%
\let\arsavedPATHafterPOS@@=\PATHafterPOS \let\PATHafterPOS=\arafterPOS@
\toks@={\ar@PATH}%
\expandafter\addtotoks@\expandafter{\expandafter{\artail@@}}%
\expandafter\addtotoks@\expandafter{\expandafter{\arstem@@}}%
\expandafter\addtotoks@\expandafter{\expandafter{\armodifiers@@}}%
\expandafter\addtotoks@\expandafter{\expandafter{\arinit@@}}%
\expandafter\addtotoks@\expandafter{\expandafter{\arexit@@}}%
3.3. ARROW AND PATH FEATURE

⟨tip⟩s and ⟨conn⟩ectors are interpreted by these that leave any ⟨variant⟩ in \toks@ii and the sequence of ⟨tipchar⟩s or ⟨connchar⟩s characters in \toks@:

The work is really done in \arcomponent@ that builds \toks@ with something conforming to the \arcomponenttype@@ and then passes control to \arcomponent@@:

\xydef@\arcomponent@{% 
  \ifx ^ \next \toks@ii={^} \DN@ {^\xyFN@\arcomponent@i} \else\ifx _ \next \toks@ii={_} \DN@ {_\xyFN@\arcomponent@i} \else\ifx 1 \next \toks@ii={1} \DN@ {1\xyFN@\arcomponent@i} \else\ifx 2 \next \toks@ii={2} \DN@ {2\xyFN@\arcomponent@i} \else\ifx 3 \next \toks@ii={3} \DN@ {3\xyFN@\arcomponent@i} \else\ifx \bgroup \next \expandafter\toks@ii\expandafter{\arvariant@@} \let\next@=\arcomponent@i \else\ifx * \next \DN@*\#\#1\#\#1\{\arcomponent@i\#\#1\} \else \expandafter\adddotoks@ \expandafter{\arcomponent@i\#\#1\#\#1\{\arvariant@@\toks@=}{} \let\next@=\arcomponenttype@@ \fi\fi\fi\fi\fi\fi
  \fi\fi\fi\fi\fi\fi

\xydef@\arcomponent@i1{% \toks@={#1}\xydef@\arcomponent@x}
\xydef@\arcomponent@i2{% \toks@={#1}#{#2}\xyFN@\arcomponent@@}
\xydef@\artip@{% 
  \addGT@\ifx \next \addGT@\adddotoks@ \addGT@\DN@{\xyFN@\artip@} \else\addLT@\ifx \next \adddotoks@ \addLT@\DN@{\xyFN@\artip@} \else\expandafter\adddotoks@\expandafter{\DN@{\xyFN@\artip@} \next\adddotoks@\DN@{\xyFN@\artip@} \fi\fi\fi

\expandafter\addtotoks@\expandafter{\expandafter{\arlabels@@}}% #8
\addtotoks@{\afterar@@} % #9
\DNii@\expandafter{\the\toks@} \toks@={} %
%\let\x=\expandafter \x \x \x \DNii@ \x \x {\nextii@} \W@ {RUN: \codeof \nextii@} %
%\nextii@

\xyuncatcodes \gdef\next#1#2#3#4#5#6#7#8#9{%
  \def\next{%
    \afterPATH{#9} ~={#6\preconnect#5#3#2} ~/{#7} ~<{<\h#5\dir#1} ~>{>\h#5\dir#4} ~+{#8} }
} %\next

\xylet@\ar@PATH=\next
\xydef@\arafterPOS@#1{%
  \arsavedPATHafterPOS@@ {\let\PATHafterPOS=\arsavedPATHafterPOS@@ \DN@{#1}\expandafter\next@\arafterPOS@@}
\xylet@\arsavedPATHafterPOS@@=\relax

⟨tip⟩s and ⟨conn⟩ectors are interpreted by these that leave any ⟨variant⟩ in \toks@ii and the sequence of ⟨tipchar⟩s or ⟨connchar⟩s characters in \toks@:

The work is really done in \arcomponent@ that builds \toks@ with something conforming to the \arcomponenttype@@ and then passes control to \arcomponent@@:
CHAPTER 3. FEATURES

Notes

3.3m. Building an ⟨arrow⟩ is simply using the specified directionals (using \dir of §1.6.1) to build a path: the first ⟨tip⟩ becomes the arrow tail of the arrow, the ⟨connection⟩ in the middle becomes the arrow stem, and the second ⟨tip⟩ becomes the arrow head. If a ⟨variant⟩ is given before the { then that variant \dir is used for all three. For example,

\xy\ar @^{(->} (20,7)\endxy

typesets

Exercise 3.3: Typeset these arrows:

\begin{center}
\begin{tikzpicture}
\node (A) at (0,0) {$A$};
\node (A') at (1.5,1.5) {$A'$};
\node (A'') at (1.5,3) {$A''$};
\node (A''') at (1.5,4.5) {$A'''$};
\node (B) at (3,0) {$B$};
\node (B') at (4.5,1.5) {$B'$};
\node (B'') at (4.5,3) {$B''$};
\node (B''') at (4.5,4.5) {$B'''$};
\draw[->] (A) -- (A');
\draw[->] (A) -- (A'');
\draw[->] (A) -- (A''');
\draw[->] (A') -- (B');
\draw[->] (A'') -- (B'');
\draw[->] (A''') -- (B''');
\end{tikzpicture}
\end{center}

(p.579)

The above is a flexible scheme when used in conjunction with the kernel \newdir to define all sorts of arrowheads and -tails. For example,

\newdir{|>}{!/4.5pt/\dir{|}*:(1,-.2)\dir^\rightarrow}
3.3. ARROW AND PATH FEATURE

\( *:(1,+.2)\dir_{>}{>} \)

defines a new arrow tip that makes

\begin{xy}
(0,0)*+{A} \\
\ar @{=} (20,3)*+{B} \\
\end{xy}

typeset

\[ A \longrightarrow B \]

Notice that the fact that the directional uses only \langle tipchar \rangle characters means that it blends naturally with the existing tips.

**Exercise 3.4:** Often tips used as ‘tails’ have their ink on the wrong side of the point where they are placed. Fortunately space is also a \langle tipchar \rangle so we can define \( \dir\{>\} \) to generate a ‘tail’ arrow. Do this such that

\begin{xy}
(0,0)*+{A}="a", (20,3)*+{B}="b" \\
\ar @{>->} "a";"b" < 2pt> \\
\ar @{ >->} "a";"b" <-2pt> \\
\end{xy}

typesets

\[ A \longleftarrow B \]

(p.579)

3.3n. Specifying a \langle dir \rangle as a \langle tip \rangle or \langle conn \rangle means that \( \dir\langle dir\rangle \) is used for that \langle tip \rangle or \langle conn \rangle. For example,

\begin{xy}
\ar @{{}{+}>} (20,7) \\
\end{xy}

typesets

When using this you must specify a \{ \} dummy \langle dir\ranglectional in order to ignore one of the tail, stem, or tip components, e.g.,

\begin{xy}
\ar @{{}{+}>} (20,7) \\
\end{xy}

typesets

In particular \( *\langle object \rangle \) is a \langle dir \rangle so any \langle object \rangle can be used for either of the tail, stem, or head component:

\begin{xy}
\ar @{*{x}*{y}*{z}} (20,7) \\
\end{xy}

typesets

\[ xyuyuyuyuyyz \]

**Note:** A \( * \) introduces an \langle object \rangle whereas the directional \( *\star \) is typeset by the \langle dir \rangle \{*\}.
Exercise 3.5: Typeset

using only one \ar command.

(p.579)

3.3o. Curving the arrow using /d\ell/, where d is a \langle direction\rangle and \ell a \langle dimen\ranglesion, makes the stem a curve which is similar to a straight line but has had its center point ‘dragged’ the distance \ell in d:

was typeset by

\xy
\POS (0,10) \cir<2pt>{} ="a"
, (20,-10)*\cir<2pt>{} ="b"
\POS"a" \ar @/^1ex/ "b"\uparrow
\POS"a" \ar @/_1ex/ "b"\downarrow
\POS (20,10) \cir<2pt>{} ="a"
, (40,-10)*\cir<2pt>{} ="b"
\POS"a" \ar @/u1ex/ "b"\u
\POS"a" \ar @/d1ex/ "b"\d
\endxy

\ell defaults to .5pc if omitted.

This is really just a shorthand for curving using the more general form described next: @/d\ell/ is the same as @\langle**{} ?+/d \ell \rangle/ which makes the (quadratic) curve pass through the point defined by the \langle pos \rangle **{} ?+/d\ell.

3.3p. Using \langle d_1, d_2 \rangle where d_1, d_2 are simple \langle direction\rangles (as described in note 1.4l except it is not possible to use (\rangle)s will typeset the arrow curved such that it leaves the source in direction d_1 and enters the target from direction d_2.

Exercise 3.6: Typeset

\begin{xy}
\POS (0,10) \cir<2pt>{} ="a"
, (20,-10)*\cir<2pt>{} ="b"
\POS"a" \ar @/^1ex/ "b"\uparrow
\POS"a" \ar @/_1ex/ "b"\downarrow
\POS (20,10) \cir<2pt>{} ="a"
, (40,-10)*\cir<2pt>{} ="b"
\POS"a" \ar @/u1ex/ "b"\u
\POS"a" \ar @/d1ex/ "b"\d
\endxy

\ell defaults to .5pc if omitted.

This is really just a shorthand for curving using the more general form described next: @/d\ell/ is the same as @\langle**{} ?+/d \ell \rangle/ which makes the (quadratic) curve pass through the point defined by the \langle pos \rangle **{} ?+/d\ell.

3.3p. Using \langle d_1, d_2 \rangle where d_1, d_2 are simple \langle direction\rangles (as described in note 1.4l except it is not possible to use (\rangle)s will typeset the arrow curved such that it leaves the source in direction d_1 and enters the target from direction d_2.

Exercise 3.6: Typeset

\begin{xy}
\POS (0,10) \cir<2pt>{} ="a"
, (20,-10)*\cir<2pt>{} ="b"
\POS"a" \ar @/^1ex/ "b"\uparrow
\POS"a" \ar @/_1ex/ "b"\downarrow
\POS (20,10) \cir<2pt>{} ="a"
, (40,-10)*\cir<2pt>{} ="b"
\POS"a" \ar @/u1ex/ "b"\u
\POS"a" \ar @/d1ex/ "b"\d
\endxy

\ell defaults to .5pc if omitted.

This is really just a shorthand for curving using the more general form described next: @/d\ell/ is the same as @\langle**{} ?+/d \ell \rangle/ which makes the (quadratic) curve pass through the point defined by the \langle pos \rangle **{} ?+/d\ell.

3.3p. Using \langle d_1, d_2 \rangle where d_1, d_2 are simple \langle direction\rangles (as described in note 1.4l except it is not possible to use (\rangle)s will typeset the arrow curved such that it leaves the source in direction d_1 and enters the target from direction d_2.
3.3. ARROW AND PATH FEATURE

3.3q. The final curve form is the most general one: \(\langle\text{control point lists}\rangle\) sets the control points explicitly to the ones in the \(\langle\text{control point lists}\rangle\) (where they should be separated by ,). See the curve extension described in §2.1 for the way the control points are used; when the control points list is parsed \(p\) is the source and \(c\) the target of the arrow.

Curving simply uses \texttt{\textbackslash crvi} instead of \texttt{\textbackslash dir}. The only tricky bit is to put the control points on the stack right.

3.3r. \[ \ldots \] and \*{\ldots} formations define what object \(\langle\text{modifier}\rangle\)s should be used when building objects that are part of the arrow. This is mostly useful in conjunction with extensions that define additional \[\langle\text{shape}\rangle\] modifiers, e.g., if a \[\text{red}\] \(\langle\text{modifier}\rangle\) changes the colour of an object to red then \[\text{red}\] will make the entire arrow red; similarly if it is desired to make and entire arrow invisible then \*{\ldots} can be used.

3.3s. \(\langle D\rangle\) will slide (each segment of) the arrow the dimension \(D\) as explained in note 3.3h.
3.3. @? reverse the meaning of ‘above’ and ‘below’ for this particular arrow.

All the features of ⟨\path⟩s described above are available for arrows.

End & log
3.4. TWO-CELL FEATURE

3.4 Two-cell feature

Vers. 3.7 by Ross Moore (ross.moore@mq.edu.au)

This feature is designed to facilitate the typesetting of curved arrows, either singly or in pairs, together with labels on each part and between. The intended mathematical usage is for typesetting categorical "2-cell" morphisms and "pasting diagrams", for which special features are provided. These features also allow attractive non-mathematical effects.

Header:

% $Id: xy2cell.doc,v 3.8 2011/03/15 12:49:57 kris Exp $
% $Id: xy2cell.doc,v 3.7 2011/03/14 20:14:00 krisrose Exp $
% Xy-pic "2-cell" feature.
% Copyright (c) 1993-1996 Ross Moore <ross.moore@mq.edu.au>
% This file is part of the Xy-pic package for graphs and diagrams in TeX.
% See the companion README and INSTALL files for further information.
% Copyright (c) 1991-2011 Kristoffer H. Rose <krisrose@tug.org>
% The Xy-pic package is free software; you can redistribute it and/or modify
% it under the terms of the GNU General Public License as published by the
% Free Software Foundation; either version 2 of the License, or (at your
% option) any later version.
% The Xy-pic package is distributed in the hope that it will be useful, but
% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
% or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License
% for more details.
The 2-cell feature makes use of facilities from the ‘curve’ extension which is therefore automatically loaded.

3.4.1 Typesetting 2-cells in Diagrams

Categorical “2-cell” morphisms are used in the study of tensor categories and elsewhere. The morphisms are displayed as a pair of curved arrows, symmetrically placed, together with an orientation indicated by a short broad arrow, or Arrow. Labels may be placed on all three components.

\message{two-cells,}

\begin{diagram}
A \rtwocell^f_g & B
\end{diagram}

\begin{diagram}
A \ruppertwocell^f{\alpha} \rlowertwocell_h{\beta} & B
\end{diagram}

These categorical diagrams frequently have a matrix-like layout, as with commutative diagrams. To facilitate this there are control sequences of the form: \rtwocell, \ultwocell, \xtwocell, ... analogous to the names defined in xyv2 for use in diagrams produced using xymatrix. As this involves the definition of 21 new control sequences, many of which may never be used, these are not defined immediately upon loading xy2cell. Instead the user must first specify \UseTwocells.

As in the second example above, just the upper or lower curved arrow may be set using control sequences of the form \..uppertwocell and \..lowertwocell. These together with the \..compositemap family, in which two abutting arrows are set with an empty object at the join, allow for the construction of complicated “pasting diagrams” (see figure 3.3 for an example).
The following initialise the families of control sequences for use in matrix diagrams.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\UseTwocells</td>
<td>two curves</td>
</tr>
<tr>
<td>\UseHalfTwocells</td>
<td>one curve</td>
</tr>
<tr>
<td>\UseCompositeMaps</td>
<td>2 arrows, end-to-end</td>
</tr>
<tr>
<td>\UseAllTwocells</td>
<td>(all the above)</td>
</tr>
</tbody>
</table>

\xydef\@UseTwocells{\definesupermorphism{twocell}}{% 
\xy{start of 2-cell}{\begingroup}{twocell@}}

\xydef\@UseHalfTwocells{% 
\definesupermorphism{uppertwocell}}{% 
\xy{start of 2-cell}{\begingroup}{uppertwocell@}}

\xydef\@UseCompositeMaps{% 
\definesupermorphism{compositemap}}{% 
\xy{start composite-map}{\begingroup}{compositemap@}}

\xydef\@UseAllTwocells{% 
\UseTwocells \UseHalfTwocells \UseCompositeMaps}

The families of connections are defined using the utility macro...\definesupermorphism which defines control sequences for morphisms between neighbouring cells, and next-to-neighbouring cells.

More distant cells use \xtwocell, \xcompositemap, etc. with \langle hop\rangle s e.g. \xtwocell[1,3] and \xtwocell[l1d].

\xydef\supermorphism#1[#2]#3{\def\afterMORPHISM{[#2]#3}#1}
\xydef\afterMORPHISM{ }
\xydef\definesupermorphism#1#2{% 
\expandafter\def\csname x#1\endcsname{\supermorphism{#2}}\%
\expandafter\def\csname u#1\endcsname{\supermorphism{#2}[-1,0]{}}\%
\expandafter\def\csname d#1\endcsname{\supermorphism{#2}[1,0]{}}\%
\expandafter\def\csname l#1\endcsname{\supermorphism{#2}[0,-1]{}}\%
\expandafter\def\csname r#1\endcsname{\supermorphism{#2}[0,1]{}}\%
\expandafter\def\csname uu#1\endcsname{\supermorphism{#2}[-2,0]{}}\%
\expandafter\def\csname dd#1\endcsname{\supermorphism{#2}[2,0]{}}\%
\expandafter\def\csname ll#1\endcsname{\supermorphism{#2}[0,-2]{}}\%
\expandafter\def\csname rr#1\endcsname{\supermorphism{#2}[0,2]{}}\%
\expandafter\def\csname ur#1\endcsname{\supermorphism{#2}[-1,1]{}}\%
\expandafter\def\csname ul#1\endcsname{\supermorphism{#2}[-1,-1]{}}\%
\expandafter\def\csname dr#1\endcsname{\supermorphism{#2}[1,1]{}}\%
\expandafter\def\csname dl#1\endcsname{\supermorphism{#2}[1,-1]{}}\%
\expandafter\def\csname uur#1\endcsname{\supermorphism{#2}[-2,1]{}}\%
\expandafter\def\csname ull#1\endcsname{\supermorphism{#2}[-2,-1]{}}\%
\expandafter\def\csname urr#1\endcsname{\supermorphism{#2}[-1,2]{}}\%
Alternatively 2-cells can be set directly in XY-pictures without using the matrix feature. In this case the above commands are not needed. This is described in §3.4.6.

Furthermore a new directional \dir{=>} can be used to place an “Arrow” anywhere in a picture, after the direction has been established appropriately. It is used with all of the 2-cell types.

\newdir{=>}{{!/5pt/\dir{=}!/2.5pt/\dir{=}*!/-5pt/\dir2{>}}

\xydef@\arrowobject#1{\def\Arrowobject@{#1}}
\xydef@\Arrowobject@{\dir{=>}}

Default positions: These set up default values for the parameters which specify the position of each component as well as the flags to indicate what type of 2-cell is to be typeset.

\toks6  position of control point, normal to line \( \overline{PC} \) at midpoint
\toks5  position of central Arrow
\toks7  position of ^ label
\toks8  position of _ label
\toks3  flag for number of pieces...
  f  full: i.e. both curved arrows
  o  one: \uppertwocell or \lowertwocell
  t  two maps: \compositemap

\@ means to calculate from the value in \toks6, see \twocell@@@.

\xydef@\twocell@{\toks6={3.5}\toks5={@}\toks7={@}\toks8={@}\toks3={f}%%
\twocell@ }
\xydef@\uppertwocell@{\toks6={7}\toks5={@}\toks7={@}\toks8={@}\toks3={o}%%
\twocell@ }
\xydef@\lowertwocell@{\toks6={-7}\toks5={@}\toks7={@}\toks8={@}\toks3={o}%%
\twocell@ }
\xydef@\compositemap@{\toks6={3.5}\toks5={@}\toks7={@}\toks8={@}\toks3={t}%%
\twocell@ }

This collects together all the relevant data, and resolves the default positions.

\xydef@\twocell@@@{%
\expandafter{\def\the\toks6={@}}
% default position of label on upper arrow,
% relative to where it meets the perpendicular bisector
% \expandafter{\ifx\the\toks7\empty\relax\the\toks6=}\else
\expandafter{\@\the\toks7\relax\expandafter{\dimen@\the\toks6}p@
3.4. TWO-CELL FEATURE

\begin{verbatim}
\edef\tmp@{\expandafter{\removePT0\the\dimen0}!!}
\edef\tmp@{\expandafter{\twocell@@@@}{\the\toks6}{\the\toks5}{\the\toks7}{\the\toks8}{\the\toks3}{\the\toks2}{\the\toks1}{\the\toks0}{\the\toks4}!!}.\afterMORPHISM}
\end{verbatim}

% possible modifications for module maps
% pass all the parameters to \twocell@@@@

\def\tmp@{\begin{groupbox}{\twocell@}
{\the\toks6}\{\the\toks5\}
{\the\toks7}\{\the\toks8}\{\the\toks3\}
{\the\toks1}\{\the\toks2}\{\the\toks0}\{\the\toks4\}!!].\afterMORPHISM\}
\fi
\toks@={\endgroup}\afterMORPHISM
\expandafter\def\expandafter\next\expandafter{\expandafter\def\expandafter\twocellhead@\expandafter{\twocellhead@}}\afterMORPHISM
\expandafter\addtotoks@\expandafter{\next@}\afterMORPHISM
\expandafter\def\expandafter\next\expandafter{\expandafter\def\expandafter\twocelltail@\expandafter{\twocelltail@}}\afterMORPHISM
\expandafter\addtotoks@\expandafter{\next@}\afterMORPHISM
\expandafter\addtotoks@\expandafter{\expandafter\twocellPATH\tmp@}\afterMORPHISM
\the\toks@ }\afterMORPHISM

See below for the possible forms of \twocellPATH.

Labels are placed labels on the upper and lower arrows, more correctly ‘anti-clockwise’ and ‘clockwise’, using ^ and __. These are entirely optional with the following token, or grouping, giving the contents of the label. When used with \..compositemap the ^ and _ specify labels for the first and second arrows, respectively.

\xydef@\twocellstyle{\scriptstyle}\afterMORPHISM
\xydef@\droptwocelllabel1@1{\xyFN\droptwocelllabel1@1!}\afterMORPHISM
\xydef@\droptwocelllabel@\ifx*\next\DN@*{\droptwocelldrop@}\afterMORPHISM
\else\DN@#1!{\droptwocelltext@{#1}}\afterMORPHISM
\fi \next@ }\afterMORPHISM
{\xyuncatcodes \gdef\next#1{\drop+!C{\twocellstyle \#1}}\afterMORPHISM
\xylet@\droptwocelltext@={\next}\afterMORPHISM
\xydef@\droptwocelldrop@1!{\bgroupph\let\xy@=\oxy@\afterMORPHISM
\let\objectstyle=\twocellstyle \drop#1\afterMORPHISM
\edef\tmp@{\egroup \X@min=\the\X@min \X@max=\the\X@max \Y@min=\the\Y@min \Y@max=\the\Y@max}\tmp@ }\afterMORPHISM

Normally the label is balanced text, set in \TeX’s math mode, with \twocellstyle setting the style. The default definition is given by \ldots

\def\twocellstyle{\scriptstyle}\afterMORPHISM
This can be altered using \def in versions of \TeX\ or \redefine in \LaTeX. However labels are not restricted to being simply text boxes. Any effect obtainable using the XY-pic kernel language can be set within an \xybox and used as a label. Alternatively if the first character in the label is * then the label is set as an XY-pic ⟨object⟩, as if with \drop<object> or *<object> in the kernel language. The current direction is tangential to the curved arrows. Extra braces are needed to get a * as the label, as in ^{{*}} or _{{}*}.

The position of a label normal to the tangential direction can also be altered by nudging (see below). Although it is possible to specify multiple labels, only the last usage of each of ^ and _ is actually set, previous specifications being ignored.

Similarly a label for the central Arrow must be given, after the other labels, by enclosing it within braces \ldots. An empty group {} gives an empty label; this is necessary to avoid misinterpretation of subsequent tokens. As above if the first character is * then the label is set as an XY-pic ⟨object⟩, the current direction being along the Arrow.
3.4. TWO-CELL FEATURE

Parsing: Open a new group so the following registers can be used locally... \toks@=\toks0, \toks1, ..., \toks8 the final parameter set is stored briefly in \tmp@ before being used.

OPTIONS:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; &lt;num&gt;</td>
<td>determines position of curves control point</td>
</tr>
<tr>
<td>&lt;0&gt;</td>
<td>uses a single solid line,</td>
</tr>
<tr>
<td>\omit</td>
<td>causes only the central Arrow (with label) to be set</td>
</tr>
<tr>
<td>\omit</td>
<td>omits the curved arrows, but allows their labels</td>
</tr>
<tr>
<td>^</td>
<td>place “above” label (more correctly “anti-clockwise”)</td>
</tr>
<tr>
<td>_</td>
<td>place “below” label (more correctly “clockwise”)</td>
</tr>
<tr>
<td>~</td>
<td>change the arrowhead</td>
</tr>
<tr>
<td>~`</td>
<td>change the arrowtail</td>
</tr>
</tbody>
</table>

Label data e.g. ^{#1} _{#1} is stored temporarily in \toks@ then passed via \tw@cell@@ to \tw@cell@@ for further parsing, as: \the\toks@ @

~` and ~ change the values of \twocellhead@ and \twocelltail@ Normally \twocelltail@ is {}, so ~` also changes a flag to indicate that \twocelltail@ is indeed required.

When the next token is a grouping {...} then it gives the label for the central Arrow. No other label specifications may follow.

Default head and tail ornaments...

452 \xydef@twocellhead#1{\def	wocellhead@{#1}}
453 \xydef@twocelltail#1{\def	wocelltail@{#1}}
454 \xydef@twocellhead@{\dir{>}}
455 \xydef@twocelltail@{}

clear \toks.. registers for the new connection labels.

461 \xywarnifdefined\twocellmod@
462 \xydef@\tw@cell@\begingroup
463 \toks@={{}\toks1={{}\toks2={{}\toks3={}}\toks4={}}}%
464 \def\twocellmod@{}\xyFN@\tw@cell@@}
465 \xydef@\tw@cell@@{%
466 \ifx\space@\next\expandafter\DN@\space{\xyFN@\tw@cell@@%
467 \else\expandafter\DN@\space{\xyFN@\tw@cell@@%
468 \else\ifx\DN@\#1\{\toks@={\#1}\nudgepos@71\tw@cell@@%
469 \else\ifx\DN@\#1\{\toks@={\#1}\nudgepos@82\tw@cell@@%
470 \else\addLT@\ifx\next\addLT@\DN@\#1\{%
471 \toks6={\#1}\ifx\omit\relax\toks3={t}\toks5={t}\fi\xyFN@\tw@cell@@%
472 \else\ifx\omit\DN@\omit\omitarrows@\xyFN@\tw@cell@@%
473 \else\ifx\next\DN@\xyFN@\whichCurveObject@@%
474 \else\DN@{\xyFN@\tw@cell@@\empty @}%
475 \fi\fi\fi\fi\fi\fi\fi%
476 \next@ }
477 \xydef@\whichCurveObject@@%
478 \ifx\space@\next\expandafter\DN@\space{\xyFN@\whichCurveObject@@%
479 \else\ifx\next\DN@\#1{\xy@\{\#1\}\{\xyFN@\tw@cell@@%
480 \else\ifx\next\DN@\#1{\xy@\{\#1\}\{\xyFN@\tw@cell@@%
481 \else\addRQ@\ifx\next\addRQ@\DN@\#1%
482 \xy@\{\#1\}\{\def\twocellhead@{\#1}\xyFN@\tw@cell@@%
483 \xy@\{\#1\}\{\def\twocellhead@{\#1}\xyFN@\tw@cell@@%
3.4.2 Standard Options

The orientation of the central Arrow may be reversed, turned into an equality, or omitted altogether. In each case a label may still be specified, so in effect the Arrow may be replaced by anything at all.

These effects are specified by the first token in the central label, which thus has the form: {⟨tok⟩⟨label⟩} where ⟨tok⟩ may be one of ...

\begin{itemize}
  \item \_ Arrow points clockwise
  \item ^ Arrow points anti-clockwise
  \item = no tip, denotes equality
  \item \omit no Arrow at all.
\end{itemize}

When none of these occurs then the default of _ is assumed. If the label itself starts with one of these characters then specify _ explicitly, or enclose the label within a group {...}. See Extra Options 1, for more values of ⟨tok⟩. Also note that * has a special role when used as the first character; however it is considered to be part of the ⟨label⟩, see above.

OPTIONS for the central Arrow:

\begin{itemize}
  \item c l ignore spaces
  \item ^ points anti-clockwise
  \item _ points clockwise (default)
  \item = no \Tip, equality
  \item \omit no Arrow at all, just squine(s) or composed arrows
  \item ' no Arrow, opposite squine reversed
  \item ' no Arrow, forward squine reversed
  \item " no Arrow, tips on both ends
  \item ! no Arrow, no tips on squines
\end{itemize}
the position of the Arrow is determined by “nudging” (see below) e.g. \{^<\text{num}>\text{label}\}, where the actual dimen is \text{<num>\text{xydashl@}}

\begin{equation}
\begin{tikzpicture}
\path (0,0) -- node[above]{$\alpha$} (1,0) -- node[above]{$\beta$} (2,0) -- node[above]{$\gamma$} (3,0) -- node[above]{$\psi$} (4,0);
\end{tikzpicture}
\end{equation}

\xymatrixcolsep{4pc}
\xymatrixrowsep{1pc}
\begin{diagram}
A \rtwocell{_{\alpha}} & B \rtwocell{^{\beta}} & C \rtwocell{=\gamma} & D \rtwocell{\omit\Psi} & E \end{diagram}

3.4.3 Nudging

Positions of all labels may be adjusted, as can the amount of curvature for the curved arrows. The way this is done is by specifying a “nudge” factor (\text{num}) at the beginning of the label. Here (\text{num}) is a number which specifies the actual position of the label in units of \text{\text{xydashl@}} (the length of a single dash, normally 5pt) except with \text{\text{\compositemap}}, see below. Movement is constrained to the perpendicular bisector of the line \text{cp}. When nudging the label for the central Arrow it is the whole Arrow which is moved, along with its label.

Curvature of the arrows themselves is altered by a nudge of the form \text{\text{\rtwocell}}(\text{\text{num}}). The separation of the arrows, along the bisector, is set to be (\text{\text{num}})\text{\text{\xydashl@}}. When (\text{num}) is zero, that is \text{\text{\rtwocell}}<0>..., the result is a single straight arrow, its mid-point being the origin for nudging.
labels. A negative value for \(\langle \text{num} \rangle\) is also acceptable; but check the orientation on the Arrow and which of \(\wedge\) and \(\_\) correspond to which component.

The origin for nudging labels is where the arrow crosses the bisector. Positive nudges move the label outwards while negative nudges move towards \(\overrightarrow{pc}\) and possibly beyond. The default position of a label is on the outside, with edge at the origin.

The origin for nudging the Arrow is at the midpoint of \(\overrightarrow{pc}\). A positive nudge moves in the clockwise direction. This will be the direction of the arrowhead, unless it has been reversed using \(\wedge\).

Labels on a \texttt{\ldots compositemap} are placed relative to the midpoint of the component arrows. Nudges are in units of 1pt. Movement is in the usual \texttt{Xy-pic} \texttt{above} and \texttt{below} directions, such that a positive nudge is always outside the triangle formed by the arrows and line \(\overrightarrow{pc}\).

The special nudge value \(\langle \text{\omit} \rangle\) typesets just the Arrow, omitting the curved arrows entirely. When used with labels, the nudge value \(\langle \text{\omit} \rangle\) causes the following label to be ignored.

**Exercise 3.7:** Give code to typeset figure 3.3. Such code is relatively straight-forward, using “nudging” and \texttt{\omit} to help position the arrows, curves and Arrows. It also uses an *excursion*, as described below in the subsection *Extra Options 3*. (p. 580)

**NUDGE** position of labels
\[\begin{align*}
\texttt{\xywarnifdefined\whichnudge} \\
\texttt{\xywarnifdefined\whichinfo} \\
\texttt{\xywarnifdefined\afternudge} \\
\texttt{\xydef@\nudgepos@#1#2#3\{\def\whichnudge{\toks\#1}\def\whichinfo{\toks\#2}\%
\def\afternudge\{\#3\}\expandafter\xyFN@\expandafter\nudgepos@@\the\toks@ @}
\end{align*}\]

Parsing: \(\langle \text{num} \rangle \langle \text{label} \rangle \), \(\langle \text{num} \rangle = \text{nudge amount}, \text{as multiple of} \ \texttt{\xydashl@}\).

**3.4.4 Extra Options**

The following features are useful in non-mathematical applications.

1. **no Arrow**

   This is determined by special values for \(\langle \text{tok} \rangle\) as the first (or only) character in the central label, as in the above description of the standard switches.
3.4. TWO-CELL FEATURE

The central Arrow is omitted, leaving symmetrically placed curved connections with arrowheads at the specified ends. A label can be placed where the Arrow would have been.

If a special arrowhead is specified using \'{..} (see Extra Options 2, below) then this will be used instead of the standard \dir{>}.

\begin{verbatim}
\xymatrixcolsep{5pc}
\text{\textcolor{white}{diagram}}
\text{\relax\txt{Clouds }\rtwocell<10>}
\text{\hbox{\tiny evaporation}}
\text{\hbox{\tiny precipitation}}
\text{\bf{H_2 O}}
\text{\relax\txt{Oceans}}
\text{\enddiagram}
\end{verbatim}

\begin{verbatim}
\xymatrixcolsep{5pc}
\text{\textcolor{white}{diagram}}
\text{\relax\txt{Mathematics }\rtwocell_{\text{experiment}}^{\text{theory}}}
\text{\relax\txt{Physics}}
\text{\enddiagram}
\end{verbatim}

2. Changing Tips and Module Maps

The following commands are provided for specifying the (object) to be used when typesetting various parts of the twocells.
These commands set the object to be used for all subsequent 2-cells at the same level of \LaTeX\ grouping. \texttt{\textbackslash curveobject} specifies both of the upper- and lower-curve objects. For some of these there is also a way to change the object for the current 2-cell only. This requires a \texttt{\textasciitilde\{switch\}} which is described below, except for the \texttt{\textbackslash..curveobject} types, which are discussed in Extra Options 4.

These effects are specified by placing switches after the \texttt{\textbackslash twocell} control sequence, \textit{e.g.} \texttt{\textbackslash rtwocell switches labels\ldots}. Each switch is either a single token \texttt{\langle tok\rangle}, or a \texttt{\textasciitilde\{tok\}} with a single argument: \texttt{\textasciitilde\{tok\}\{arg\}}. Possibilities are listed in the following table, in which \texttt{\{..\}} denotes the need for an argument.

<table>
<thead>
<tr>
<th>command</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>\modmapobject{\langle object\rangle}</td>
<td>\dir{}</td>
</tr>
<tr>
<td>\twocellhead{\langle object\rangle}</td>
<td>\dir{&gt;}</td>
</tr>
<tr>
<td>\twocelltail{\langle object\rangle}</td>
<td>\dir{}</td>
</tr>
<tr>
<td>\arrowobject{\langle object\rangle}</td>
<td>\dir{=&gt;}</td>
</tr>
<tr>
<td>\curveobject{\langle object\rangle}</td>
<td>{}</td>
</tr>
<tr>
<td>\uppercurveobject{\langle object\rangle}</td>
<td>{}</td>
</tr>
<tr>
<td>\lowercurveobject{\langle object\rangle}</td>
<td>{}</td>
</tr>
</tbody>
</table>

Here we discuss the use of \texttt{\textasciitilde}, \texttt{\textasciitilde\{..\}}, \texttt{\textasciitilde^\{} and \texttt{\textasciitilde_\{}. The description of \texttt{\textasciitilde^\{} and \texttt{\textasciitilde_\{} is given in Extra Options 4.

The default module map indicator places a single dash crossing the arrow at right-angles, located roughly midway along the actual printed portion of the arrow, whether curved or straight. This takes into account the sizes of the objects being connected, thereby giving an aesthetic result when these sizes differ markedly. This also works with \texttt{\textbackslash..compositemap} where an indicator is placed on each arrow. The actual object can be changed using \texttt{\modmapobject}.

Any of the standard X\textsc{y}-pic tips may be used for arrow-heads. This is done using \texttt{\textasciitilde\{dir\}} for example \texttt{\textasciitilde\{dir\{\textasciitilde\{\textasciitilde\}\}} gives double-headed arrows. Similarly \texttt{\textasciitilde\{..\}} can be used to place an arrow-tail. Normally the arrow-tail is \texttt{\textasciitilde}, so is not placed; but if a non-empty tail has been specified then it will be placed, using \texttt{\drop}. No guarantee is offered for the desired result being obtained when an arrow-tail is mixed with the features of Extra Options 1.

\[
\xymatrixcolsep{5pc}
\xymatrix{
  & M \\
  P^f S \ar@{-}[uu] \ar@{-}[rr] & & S \\
  & M'
}
\modmapobject\{\objectbox\{\otimes\}\}
\xymatrixcolsep{5pc}
\diagram
3.4. TWO-CELL Feature

\[ \text{Prtwocell}^{-1} \{ \text{dir} \} \}^{-1} \{ \text{dir} \} \}
\{<1.5>M\}_{<1.5>M'}=f \text{ & S } \|
\text{enddiagram}

switch for module maps... places decoration called \modmapobject half way along arrows

\[ \text{xydef@modmap@} \%
\expandafter\ifx\the\toks4@=!\toks4@={!}\%
\else\expandafter\ifx\the\toks4@=H\toks4@={H}\%
\else\expandafter\ifx\the\toks4@=V\toks4@={V}\fi\fi\fi\fi\]
\expandafter\DN@\expandafter{\the\toks6@}%
\ifx\next@\empty\relax\else\expandafter\moddefs@\fi

also adjust curvature and label positions

\[ \text{xydef@moddefs@} \%
\expandafter\dimen@\the\toks6@\p@\%
\ifdim\dimen@=\z@\relax
\expandafter\dimen@\the\toks7@\p@\%
\edef\tmp@{\noexpand\toks7@={\expandafter\removePT@\the\dimen@}}\tmp@
\expandafter\dimen@\the\toks8@\p@\%
\ifdim\dimen@<\z@\advance\dimen@-.5\p@\else\advance\dimen@.5\p@\fi
\edef\tmp@{\noexpand\toks8@={\expandafter\removePT@\the\dimen@}}\tmp@
\else
\ifdim\dimen@<\z@\advance\dimen@-.5\p@\else\advance\dimen@.5\p@\fi
\edef\tmp@{\noexpand\toks7@={\expandafter\removePT@\the\dimen@}}\tmp@
\expandafter\dimen@\the\toks5@\p@\%
\ifdim\dimen@<\z@\advance\dimen@-.5\p@\else\advance\dimen@.5\p@\fi
\edef\tmp@{\noexpand\toks5@={\expandafter\removePT@\the\dimen@}}\tmp@
\expandafter\dimen@\the\toks7@\p@\%
\ifdim\dimen@<\z@\advance\dimen@-.5\p@\else\advance\dimen@.5\p@\fi
\edef\tmp@{\noexpand\toks7@={\expandafter\removePT@\the\dimen@}}\tmp@
\expandafter\dimendef@\expandafter{\the\toks6@}%
\edef\tmp@{\noexpand\toks6@={\expandafter\removePT@\the\dimen@}}\tmp@
\expandafter\dimen@\the\toks7@\p@\%
\ifdim\dimen@<\z@\advance\dimen@-.5\p@\else\advance\dimen@.5\p@\fi
\edef\tmp@{\noexpand\toks7@={\expandafter\removePT@\the\dimen@}}\tmp@
\expandafter\dimen@\the\toks8@\p@\%
\ifdim\dimen@<\z@\advance\dimen@-.5\p@\else\advance\dimen@.5\p@\fi
\edef\tmp@{\noexpand\toks8@={\expandafter\removePT@\the\dimen@}}\tmp@
\fi \}%
\%\def\moddefs@{}\%

default object is \stop, user can change this with \modmapobject#1

\[ \text{xydef@modmapobject@#1{\def@modmapobject@{#1}}} \]
\[ \text{xydef@modmapobject@{\dir{}{}}} \]
\[ \text{xydef@modmapobject@{}{\if@mod@@relax\expandafter\empty\text{}\}} \]
\[ \text{else}\expandafter\modmapobject@{\fi} \]
3. Excursions

Syntax for \texttt{\textbackslash xcompositemap} and \texttt{\textbackslash ..twocell} types is a little different to what might be expected from that for \texttt{\textbackslash xto}, \texttt{\textbackslash xline}, etc. For example,

\begin{verbatim}
\xtwocell[(hop)]{(displace)}...
\end{verbatim}

connects to the \texttt{\langle pos\rangle} displaced by \texttt{\langle displace\rangle} from the relative cell location specified by \texttt{\langle hop\rangle}. The displacement can be any string of valid \texttt{XY-pic} commands, but they must be enclosed within a group \{\ldots\}. When the cell location is the target, a null grouping \{\} must be given.

When used with the \texttt{\langle \textbackslash omit\rangle} nudge, such excursions allow a labelled Arrow to be placed anywhere within an \texttt{XY-pic} diagram; furthermore the Arrow can be oriented to point in any direction.

The empty object gives the default of closely spaced tiny dots, resulting in a “smooth” curve.

The following macro is used in conjunction with compiling diagrams, to include the global parameters in the compiled file.

\begin{verbatim}
\xydef\save\twocell objects{%
\xy@{\twocell objects}%
\expandafter\uppercurveobject\expandafter{\ucurveObject@}%
\expandafter\lowercurveobject\expandafter{\dcurveObject@}%
\expandafter\modmapobject\expandafter{\modmapobject@}%
\expandafter\twocellhead\expandafter{\twocellhead@}%
\expandafter\twocelltail\expandafter{\twocelltail@}%
\expandafter\arrowobject\expandafter{\Arrowobject@}%
}
\end{verbatim}

4. Fancy curves

By specifying \texttt{\textbackslash curveobject} an arbitrary object may be used to construct the curved arrows. Indeed with a \texttt{\textbackslash ..twocell} different objects can be used with the upper and lower curves by specifying \texttt{\uppercurveobject} and \texttt{\lowercurveobject}.

These specifications apply to all 2-cells subsequently constructed at the same level of \texttt{\TeX} grouping. Alternatively using a \texttt{\~}-switch, as in \textit{Extra Options 2}, allows such a specification for a single 2-cell or curved part.

Objects used to construct curves can be of two types. Either a single \texttt{\langle object\rangle} is set once, with copies placed along the curve. Alternatively a directional object can be aligned with the tangent along the curve. In this case use a specification takes the form:

\begin{verbatim}
\curveobject{\langle spacer\rangle\textasciitilde\ast\langle object\rangle}.
\end{verbatim}

Here \texttt{\langle spacer\rangle} may be any \texttt{\langle object\rangle} of non-zero size. Typically it is empty space, \textit{e.g.} \texttt{\textasciitilde\langle dimen\rangle\{\}}.

\textbf{Exercise 3.8:} Give code to typeset the following diagrams.
3.4. TWO-CELL FEATURE

3.4.5 After Parsing...

Values for `twocellPath`, depends on the mode of picture/diagram as to whether the $c$ and $p$ are known yet, or still have to be read.

```latex
\xydef@{twocell@path.[#1].{% 
\expandafter\ifx\csname Q@@c\endcsname\relax
\DN@{\enter@{\cfromthec@ \pfromthep@ \basefromthebase@}\
\xy@{}{\expandafter\edef\csname Q@@\endcsname{\cfromthec@}}% 
\afterPOS{\xy@{}{\expandafter\edef\csname Q@@c\endcsname{\cfromthec@}}% 
\xy@{}{\cfromid@{@}\expandafter\edef\csname Q@@p\endcsname{\cfromthec@}}% 
\xy@{twocell #1}{#1}\leave@\aftertwoCELL};p,}% 
\else
\DN@{\enter@{\cfromthec@ \pfromthep@ \basefromthebase@}\
\xy@{twocell #1}{#1}\leave@\aftertwoCELL}% 
\fi \next@ }
\xydef@{twocellpath.[#1].{% 
\expandafter\edef\csname Q@@p\endcsname{\cfromthec@}\
\xy@{}{\enter@{\pfromthep@}\
\xy@{}{\enter@{\cfromthec@ \pfromthep@ \basefromthebase@}\
\enter@{\pfromthep@}}% 
\afterPOS{\xy@{}{\leave@\setupDirection@}#1\restore\restore
\xy@{end of 2-cell}\xysaveMinMax@ }%}
\xydef@{xysaveMinMax@{\edef\tmp@{\endgroup 
\noexpand\ifdim\X@max<\the\X@max \X@max=\the\X@max
\noexpand\fi fi 
\noexpand\ifdim\X@min>\the\X@min \X@min=\the\X@min
\noexpand\fi fi 
\noexpand\ifdim\Y@max<\the\Y@max \Y@max=\the\Y@max
\noexpand\fi fi 
\noexpand\ifdim\Y@min>\the\Y@min \Y@min=\the\Y@min
\noexpand\fi fi 
\tmp@ } }
```
It remains to collect everything together and place it on the queue.

3.4.6 2-cells in general $\mathbf{X}\mathbf{Y}$-pictures

Two-cells can also be set directly within any $\mathbf{X}\mathbf{Y}$-picture, without the matrix feature, using either \texttt{\textbackslash drop} or \texttt{\textbackslash connect}.

\begin{verbatim}
\let\xystatus=\xystatus@
\xydef@\twocell{\hbox\bgROUP\@twocell}
\xydef@\uppertwocell{\hbox\bgROUP\@uppertwocell}
\xydef@\lowertwocell{\hbox\bgROUP\@lowertwocell}
\xydef@\compositemap{\hbox\bgROUP\@compositemap}
\xydef@\twocellll#1#{\hbox\bgROUP\xy@\save\save@\twocellll[#1]}
\xydef@\uppertwocelll#1#{\hbox\bgROUP\xy@\save\save@\uppertwocellll[#1]}
\xydef@\lowertwocelll#1#{\hbox\bgROUP\xy@\save\save@\lowertwocellll[#1]}
\xydef@\compositemapp#1#{\hbox\bgROUP\xy@\save\save@\compositemapp[#1]}
\xydef@\@compositemap#1#2{\afterPOS{\@twocelll\xyFN@\compositemap@#1{#2}}}
\xydef@\@twocelll@{\afterPOS{\@twocelll\xyFN@\twocelll@}}
\def\myPOS#1{\POS}
\def\goVia#1{\afterPOS{\connect#1\myPOS}}
\end{verbatim}

\begin{verbatim}
\def\aftertwoCELL{\twocell@@DONE}
\def\aftertwoCELL{\twocell@@DONE \xy@\restore \leave@}
\end{verbatim}

\begin{verbatim}
\let\xystatus=\xystatus@
\xydef@\twocell{\hbox\bgROUP\@twocell}
\xydef@\uppertwocell{\hbox\bgROUP\@uppertwocell}
\xydef@\lowertwocell{\hbox\bgROUP\@lowertwocell}
\xydef@\compositemap{\hbox\bgROUP\@compositemap}
\xystatus@{end: }
\end{verbatim}

\begin{verbatim}
\def\myPOS{\POS}
\end{verbatim}

\begin{verbatim}
\xy
++{A}="A",+<1cm,1.5cm>++{B}="B",
+<2.0cm,0pt>++{C}="C",
\end{verbatim}
3.4. TWO-CELL FEATURE

345

+<1cm,-1.5cm>*+{D}="D",
"A";\goVia{\uppertwocell^\alpha{}}"B"{}
;\goVia{\twocell^\zeta_\xi{\gamma}}"C"{}
;\goVia{\compositemap{}}"D"{},
"A";\goVia{\lowertwocell{}}"D"{}
\endxy
ζ
α

1B
!)

A

γ


ξ



(

6C
v~

(

8D

The code shown is a compact way to place a chain of 2-cells within a picture. It illustrates
a standard technique for using \afterPOS to find a ⟨pos⟩ to be used for part of a picture, then
subsequently reuse it. Also it is possible to use \drop or ⟨decor⟩s to specify the 2-cells, giving the
same picture.
\xy *+{A}="A",+<1cm,1.5cm>*+{B}="B",
+<2cm,0pt>*+{C}="C",
+<1cm,-1.5cm>*+{D}="D",
"A";"B",{\uppertwocell^\alpha{}},
"B";"C",{\twocell^\zeta_\xi{\gamma}},
"C"; \afterPOS{\drop\compositemap{}}"D"
\POS "A";
\afterPOS{\drop\lowertwocell{}}"D"
\endxy
The \connect variant is usually preferable as this maintains the size of the object at c, while the
\drop variant leaves a rectangular object having p and c on opposite sides.
This is used outside of diagrams.
1337
1338
1339
1340
1341
1342
1343
1344

\xydef@\@twocell@#1{\ifx\next[\DN@[{\toks@={#1}\xyFN@\@@twocell@}%
\else\DN@{#1{}}\fi
%%
% \idfromc@{@c}\swap@\idfromc@{@p}\swap@
\expandafter\edef\csname Q@@c\endcsname{\cfromthec@}\swap@
\expandafter\edef\csname Q@@p\endcsname{\cfromthec@}\swap@
\let\twocellPATH=\twocell@path
\def\aftertwoCELL{\twocell@DONE}\next@}
This hack is necessary, since \DN@[##1]{#1##1} will strip braces.

1350
1351

\xydef@\@@twocell@{\ifx\next\bgroup\DN@##1]{\the\toks@{##1}}%
\else\DN@##1]{\the\toks@##1}\fi\next@}

To Do: These are supposed to allow \dir\...twocell{...} to work. At present it seems to be
impossible to do this.
1361
1362
1363

% \expandafter\xydefcsname@\expandafter{\codeof\twocell}{\@twocell}
% \expandafter\xydefcsname@\expandafter{\codeof\uppertwocell}{\@uppertwocell}
% \expandafter\xydefcsname@\expandafter{\codeof\lowertwocell}{\@lowertwocell}


...since the contents of \{..\} is variable.

Finish off the box, set up the object size, establish the \Drop@@ and \Connect@@ methods.

modiﬁed \Drop@@ and \Connect@@

\twocell@@ is the main switching engine, taking 9 parameters determining which pieces to set and where to position them. This includes curvature of the squines, whether to draw two squines or only one, or to use straight lines, whether to have a central Arrow and its orientation, contents of labels and their positions.

The current p and c are the extents of the connection these are stored as "@p" and "@c" for repeated use;
3.4. TWO-CELL FEATURE

#1 determines location of control point "m"
#2 location of \(\Rightarrow\), as multiple of \xydashl@ from "m"
#3 location of _ label (source arrow)
#4 location of ^ label (target arrow)
#5 flag for how many squines to draw & which tips
#6 label for source arrow
#7 label for target arrow
#8 label for \(\Rightarrow\) (2-cell morphism)
#9 determines orientation of 2-cell

The following tokens may appear as parameters in #9: !, ^, _, =, A, V, H
3.4. TWO-CELL FEATURE

\advance\dimen@\ifdim\dimen@<90\fi.5\p\fi
\edef\next@{\expandafter\removePT\the\dimen@}%
\expandafter\vfromslide@i\expandafter{\next@\xydashl0}\leave@
\droptwocelllabel@{#6}%
\fi\fi\fi

\DN@{#1}\ifx\next@\empty\relax\else
\ifx #5b\relax % tip at start only
\enter@ {\pfromthep@ }\Creset@@ \PLACEedgep@
\Calong@@{0}\czeroEdge@ \leave@
\edef\next@{\codeof\twocelltail@}%
\ifx\next@\empty \DN@{\dir{>}}%
\else
\reverseDirection@ \DNii@##1##{\drop@{##1}}%
\expandafter\nextii@\twocellhead@{}
\reverseDirection@
\fi
\else
\DNii@##1##{\drop@{##1}}\expandafter\drop\twocelltail@{}%
\fi
\else\ifx #5@\relax % no tips at all
\else % tip at end...
\enter@ {\pfromthep@ }\Creset@@ \PLACEedgec@
\Calong@@{1}\czeroEdge@ \leave@
\edef\next@{\codeof\twocellhead@}%
\ifx\next@\empty\drop@{\dir{>}}%
\else
\DNii@##1##{\drop@{##1}}\expandafter\nextii@\twocellhead@{}
\fi
\else
\DNii@##1##{\drop@{##1}}\expandafter\drop\twocelltail@{}%
\fi
\else\ifx #5B\relax % ... and also at start.
\enter@ {\pfromthep@ }\Creset@@ \PLACEedgep@
\Calong@@{0}\czeroEdge@ \leave@
\edef\next@{\codeof\twocelltail@}%
\ifx\next@\empty \DN@{\dir{>}}%
\else
\reverseDirection@\DNii@##1##{\drop@{##1}}%
\expandafter\nextii@\twocellhead@{}
\reverseDirection@
\fi
\else
\DNii@##1##{\drop@{##1}}\expandafter\drop\twocelltail@{}%
\fi
\else\ifx #5/\relax
\else \edef\next@{\codeof\twocelltail@}%
\ifx\next@\empty\relax\else
\enter@ {\pfromthep@ }\Creset@@ \PLACEedgep@
\Calong@@{0}\czeroEdge@ \leave@
\DNii@##1##{\drop@{##1}}\expandafter\drop\twocelltail@{}%
\fi
\else \edef\next@{\codeof\twocelltail@}%
\ifx\next@\empty\relax\else
\enter@ {\pfromthep@ }\Creset@@ \PLACEedgep@
\Calong@@{0}\czeroEdge@ \leave@
\DNii@##1##{\drop@{##1}}\expandafter\drop\twocelltail@{}%
\fi
3.4. TWO-CELL FEATURE

\DNii@{}\iffalse\nextii@\DNii@{\relax}\else
\ifx @#4\relax\DNii@{\relax}\else
\DNii@{\enter@\pfromthep@ \Creset@@
\Calong@@{.5}\czerolong@ \leave@
\enter@\cplusthec@@
\enter@\DirectionfromtheDirection@ \begingroup
\DNii@{#1}\ifx\nextii@\empty\dimen@z@ \else\dimen@=#1\p@\fi
\ifx #5t\relax \ifdim\dimen@<\z@ \belowDirection@ \xydashl@ \fi
\else \ifdim\dimen@<\z@ \aboveDirection@ \xydashl@ \fi
\else \ifdim\dimen@<\z@ \aboveDirection@ \xydashl@ \fi
\else \belowDirection@ \xydashl@ \fi
\if\dimen@=#4\p@ \if@mod@ \relax
\advance\dimen@-\p@ \edef\next@{\expandafter\removePT@\the\dimen@}\
\edef\next@{\expandafter\nwslide@i@\expandafter{\next@}}\leave@
\droptwocelllabel@{#7}\
\fi\fi\fi \next@
% \DNii@{#1}\iffalse\nextii@\DNii@{\relax}\else
\ifx #5b\relax % tip at start only
\enter@{\pfromthep@} \Creset@@ \PLACEedgec@
\Calong@@{1}\czerolong@ \leave@
\edef\next@{\codeof\twocelltail@}\relax
\ifx\nextii@\empty \DNii@{\dir{>}}\relax
\ifx\nextii@\twocellhead@\drop@{\dir{>}}\relax
\else
\DNii@##1##{\drop@{##1}}\expandafter\nextii@\twocellhead@{}
\reverseDirection@ \DNii@##1##{\drop@{##1}}\expandafter\nextii@\twocelltail@{}
\else
\DNii@##1##{\drop@{##1}}\expandafter\nextii@\twocelltail@{}
\fi
\fi
\else
\DNii@##1##{\drop@{##1}}\expandafter\nextii@\twocellhead@{}
\reverseDirection@ \DNii@##1##{\drop@{##1}}\expandafter\nextii@\twocellhead@{}
\reverseDirection@
\fi
\else\iffalse\nextii@\DNii@{\relax}\else
\ifx #5@\relax % no tips at all
\else % tip at end...
\enter@\pfromthep@ \Creset@@ \PLACEedgec@
\Calong@@{1}\czerolong@ \leave@
\fi
\fi
\edef\next@{%codeof/twocellhead@}%
\ifx\next@{empty}drop@{dir}{>}%
\else
\DNii@##1##{drop@{##1}}\expandafter\nextii@{twocellhead@}{}
\fi
\ifx #5B\relax % ... and also at start.
enter@ {pfromthep@}Creset@@PLACEedgep@
Calong@@0\czeroEdge@ leave@
\edef\next@{%codeof/twocelltail@}%
\ifx\next@{empty}DN@{dir{>}}%
\ifx\next@twocellhead@drop@{dir}{<}%
\reverseDirection@DNii@##1##{drop@{##1}}%
\expandafter\nextii@{twocellhead@}{reverseDirection@}
\fi\fi\else
\DNii@##1##{drop@{##1}}\expandafter\nextii@{twocelltail@}{}
\fi\fi\else
\edef\next@{%codeof/twocelltail@}%
\ifx\next@{empty}relax\else
enter@ {pfromthep@}Creset@@PLACEedgep@
Calong@@0\czeroEdge@ leave@
\DNii@##1##{drop@{##1}}\expandafter\nextii@{twocelltail@}{}
\fi\fi\fi
\fi\fi\fi
\DN@{#1}\ifx\next@{empty}\dimen@=\z@\else\dimen@=#1\p@\fi
\ifdim\dimen@<\z@ cfromid@{@m2}\swap@cfromid@{@m1}%
\else cfromid@{@m1}\swap@cfromid@{@m2}%
\fi \no@@ cfromid@{@m}%
\ifx\Arrowtok@{empty}
\else\DN@{_}\edef\next@{%codeof/\Arrowtok@}
\ifx\next@\Arrowtok@
enter@{pfromthep@ cplusthec@}\dimen@=#2\xydashl@
\enter@DirectionfromtheDirection@ \begingroup
\expandafter\vfromslide@i\expandafter{\the\dimen@}@\czeroEdge@
\leave@ idfromc@{@m}%
\DNii@##1##{drop@{##1}}\expandafter\nextii@{Arrowobject@}{}
\drop@{dir}{=>}%
\else\DN@{^}\edef\Arrowtok@{%codeof/\next@}%
\enter@{pfromthep@ cplusthec@}\dimen@=#2\xydashl@
\endgroup
\DN@{\_}\edef\next@{%codeof/\next@}\iffalse\Arrowtok@\else\DN@{^}\edef\next@{%codeof/\next@}\fi
\DN@{\_}\edef\Arrowtok@{%codeof/\next@}\iffalse\Arrowtok@\else\DN@{^}\edef\next@{%codeof/\next@}\fi
\enter@{pfromthep@ cplusthec@}\dimen@=#2\xydashl@
\enter@DirectionfromtheDirection@ \begingroup
\expandafter\vfromslide@i\expandafter{\the\dimen@}@\czeroEdge@
\leave@ idfromc@{@m}%
\DNii@##1##{drop@{##1}}\expandafter\nextii@{Arrowobject@}{}
\drop@{dir}{=>}%
3.4. TWO-CELL FEATURE

These are used to establish the location for the module-map indicator, giving best aesthetic appeal. It finds the location whose parameter value is the average of half-way along the complete curve and the average of parameter values at the the edges.
The end & Log

DOCMODE3

\xyendinput

% $Log: xy2cell.doc,v $
% Revision 3.7  2011/03/14 20:14:00  krisrose
% Preparing for release 3.8.6.
% %
% Revision 3.6  2010/06/10 18:45:49  krisrose
% Reference to GPL by URL.
% %
% Revision 3.5  2010/05/06 17:46:29  krisrose
% Ross Moore's e-mail address updated.
% Many obsolete files degraded to Historic.
% %
% Revision 3.4  2010/04/16 06:06:52  krisrose
% Preparing for a new release...
% %
% Revision 3.3  1996/12/18 14:21:23  ross
% Ross's version
% %
% Revision 3.3.1.1 1996/12/18 08:49:34  ross
% *** empty log message ***
% %
% Revision 3.1  1995/09/05 20:36:33  ross
% Release!
% %
% Revision 3.0  1995/07/07 20:13:19  ross
% Major release w/new User's Guide!
% %
% Revision 2.14  1995/07/05 22:11:25  kris
% Buglets...
% %
% Revision 2.13  1995/07/04 15:04:51  ross
% Ready for release of v3.
% %
% Revision 2.12  1994/10/25 03:01:14  ross
% Final 3beta release [bug fixes & AMS-LaTeX fitting].
% %
% Revision 2.10  1994/07/01 01:19:46  ross
% removed undefined references
% %
3.5 Matrix feature

Vers. 3.14 by Kristoffer H. Rose ⟨krisrose@tug.org⟩

This option implements “Xy-matrices”, i.e., matrices where it is possible to refer to the entry objects by their row/column address. We first describe the general form of Xy-matrices in §3.5.1, then in §3.5.2 we summarise the new ⟨coord⟩inate forms used to refer to entries. In §3.5.3 we explain what parameters can be set to change the spacing and orientation of the matrix, and in §3.5.4 we explain how the appearance of the entries can be changed.

3.5.1 Xy-matrices

The fundamental command of this feature is

\xymatrix ⟨setup⟩ {⟨rows⟩}
that reads a matrix of entries in the generic \TeX row\&column format, \textit{i.e.}, where rows are separated with `\textbackslash\textbackslash' and contain columns separated with `&' (we discuss in the following sections what \textlangle\textit{setup}\textrangle\ can be). Thus a matrix with \textit{maxrow} rows and \textit{maxcol} columns where each entry contains \textit{row}, \textit{col} is entered as

\[
\text{\textbackslash xymatrix}{
  1,1 & 1,2 & \cdots & 1,\text{maxcol} \textbackslash \\
  2,1 & 2,2 & 2,\text{maxcol} \textbackslash \\
  \vdots & \vdots & \ddots & \vdots \\
  \text{maxrow},1 & \text{maxrow},2 & \cdots & \text{maxrow},\text{maxcol} 
}
\]

(\TeXnically the `\&' character represents any `alignment tab', \textit{i.e.}, character with category code 4).

A \textlangle\textit{matrix}\textrangle\ can appear either in an \TeXpicture (as \textlangle\textit{decor}\textrangle\) or “stand-alone”.

The aspects in which \textbackslash xymatrix\ differs from ordinary matrix constructions, such as Plain \TeX's \textbackslash matrix{\ldots} and \LaTeX's \texttt{array} environment, are

- arbitrary \TeXpicture (\textit{decor})ations may be specified in each entry and will be interpreted in a state where \textit{c} is the current entry,

- the entire matrix is an object itself with reference point as the top left entry, and

- a progress message “\textbackslash xymatrix rowsxcols size\textgreater\” is printed for each matrix with \textit{rows} \times \textit{cols} entries and \TeXpicture complexity \textit{size} (the number of primitive operations performed), unless the declaration \textbackslash SilentMatrices\ is issued.

- Entries starting with a `\*' are special (described in §3.5.4)\textsuperscript{2}, so use `{*} to get a `*'.

For example,

\[
\text{\textbackslash xy} \\
\text{\textbackslash xymatrix}{A&B\textbackslash\textbackslash C&D} \\
\text{\textbackslash drop}\texttt{\textbackslash frm{-}} \\
\text{\textbackslash drop}\texttt{\textbackslash cir<8pt>\{} \\
\text{\textbackslash endxy}
\]

will typeset

\[
\begin{array}{cc}
A & B \\
C & D
\end{array}
\]

\textbf{Bug}: Matrix nesting is not safe.

Matrices are often quite slow to typeset so as a convenience all matrices can be set to compile (and not) automatically with the declarations

\begin{tabular}{c}
\textbackslash CompileMatrices \\
\textbackslash NoCompileMatrices
\end{tabular}

Matrices can be compiled or not individually, by using the explicit commands \textbackslash xymatrixcompile and \textbackslash xymatrixnocompil as well as by encapsulating in the usual \textbackslash xyc\texttt{ompileto\{name\}\ldots} (see note 1.5e).

\textbf{Note}: Matrices will only compile correctly if all entries start with a nonexpandable token, \textit{i.e.}, \{ or \texttt{\textbackslash relax} or some non-active character.

\textsuperscript{2}In general it is recommended that entries start with a non-expanding token, \textit{i.e.}, an ordinary (non-active) character, \{, or \texttt{\textbackslash relax}, for compilation to work.
3.5. MATRIX FEATURE

On the following code: The complexity of this option stems from the fact that we cannot build the actual $X\!\times\!Y$-picture until after the matrix has been typeset so that we know the sizes and places of all the entries. The handling described here is thus separated into several subprocedures.

Setup: Not much—only to keep track of the rows and columns. These are not accesible to the user—should they?

\begin{verbatim}
\xynew{count}{Row}
\xynew{count}{Col}
\xydef{maxrow}{0}  \% highest row number used
\xydef{maxcol}{0}  \% highest column number used
\xydef{maxcolrow}{0} \% a row with maximal number of columns
\end{verbatim}

There are some user-defined parameters summarised later.

The following is an experimental set of ‘optimal’ queueing macros using two queues and two counters to keep track of the progress.

\begin{verbatim}
\xynew{toks}{queue@}
\xynew{toks}{queue@@}
\xynew{count}{qcount@}
\xynew{count}{qcount@@}
\xydef{clearq@}{\global{queue@=\{} \global{qcount@=\@ne}
\global{queue@@=\{} \global{qcount@@=\z@}}\xydef{addq@#1}{\global{advance=matrixsize@\@ne}
\ifnum{qcount@@}<qcount@ \let{addq@@=\addq@i} \else \let{addq@@=\addq@ii} \fi
\expandafter{addq@@{\the{queue@@}\addq@@{#1}}}\xylet@{addq@@=\relax}
\xydef{addq@i#1\addq@@#2}{\global{advance=qcount@@\@ne} \global{queue@@=\{} \global{qcount@@=\#1}\#2\}}\xydef{addq@ii#1\addq@@#2}{\global{advance=qcount@@\@ne} \global{queue@@=\{} \global{qcount@@=\#1}\#2\}}\global{queue@=\expandafter{\the{queue@}\the{queue@@}\addq@@{}}}\xydef{finishq@}{\expandafter{addq@i\the{queue@@}\addq@@{}}}\end{verbatim}

Hack: intended to make the addition of tokens to the token list faster: we first add to the ‘small’ $\\{\begin{array}{c} \text{queue}\@\text{as many times as we have added to the ‘big’ queue@ before that. So we know we should} \\
\text{always expand queue@: this is done first (last in the macro text). Then we call addq@@ that will} \\
\text{either add to the small one or add the small one to the big one and then clear the small one!}
\end{array}\}$

Usage: Call \clearq@ first. Then use \addq@ as many times as desired. To get the queue in \queue@ call \finishq@ and use it. Finally clear \queue@ (globally) or continue.

Main procedure: Here is the main code which is concerned with initialisations and output of messages explaining how far the matrix construction has progressed.

\begin{verbatim}
\xynew{count}{matrixsize@}
\xynew{if}{ifnoisymatrices@ \noisymatrices@true}
\xydef{\SilentMatrices}{\noisymatrices@false}
\end{verbatim}
3.5. MATRIX FEATURE

Here is where compilation of matrices is turned on/off.

\xydef@\xymatrixcompile@{\xymatrixcompile@{(#1)}}
\xydef@\xymatrixcompile@{\xymatrixcompile@{(#1)#2}}
\xydef@\CompileMatrices{\let\xymatrix=\xymatrixcompile}
\xydef@\NoCompileMatrices{\let\xymatrix=\xymatrixnocompile}

To Do: Proper nesting that ensures that the matrix state is reestablished after the current one has been typeset.
Each subprocedure is explained below.

Set row-rotation: This sets the general direction of the rows as the current direction indicates. This direction is not known until X\text{-}time so it gets put into the row/column definitions.

\xydef@\xy@cossign@@{+}
\xydef@\xy@sinsign@@{+}
\xydef@\xy@cosabs@@{1}
\xydef@\xy@sinabs@@{0}
\xydef@\xy@tanabs@@{0}
\xydef@\xy@cotabs@@{100}
\xydef@\xymatrix@rotation{
\xy@@{%\edef\xy@cossign@@{\ifdim\cosDirection\p@<\z@-\else+\fi}\%\edef\xy@sinsign@@{\ifdim\sinDirection\p@<\z@-\else+\fi}\%\edef\xy@cosabs@@{%\if\xy@cossign@@\cosDirection\else\xy@cossign@@\cosDirection\fi}\%\edef\xy@sinabs@@{%\if\xy@sinsign@@\sinDirection\else\xy@sinsign@@\sinDirection\fi}%\dimen@=\xy@cosabs@@\p@\dimen@ii=\xy@sinabs@@\p@\ifdim\dimen@<.01\p@\def\xy@cotabs@@{100}\%\else\quotient@\xy@cotabs@@\dimen@\dimen@ii /\fi\dimen@=\xy@cosabs@@\p@\dimen@ii=\xy@sinabs@@\p@\ifdim\dimen@<.01\p@\def\xy@tanabs@@{100}\%\else\quotient@\xy@tanabs@@\dimen@ii\dimen@ /\fi}}

Pretypeset: The purpose of this is to

- set box8 to an halign with all the entries properly typeset to be measured in the next step, and
- store in a queue the operations that will define each entry as an object once the measurement is done and the distances between rows and columns is known.

This is achieved as follows: Row/column counters, maxima, and queues are reset. The tab- and interlineskip is removed to make taking the constructed box apart easier. And the halign template is used to apply an appropriate action for each entry that updates the counters and queue.

The special \xymatrixsavedCOORD@ is used to store the original \COORD@ except when diagrams are nested.

\xylet@\xymatrixsavedCOORD@=\relax
\xydef@\xymatrix@pretypeset@{1}\%
\global\Row=\z@ \xdef\maxcol@{0}\clearq@ \setbox8=\vtop{%
The first entry in each row is special; in particular it moves the entries from the columns of the previous row (if any) into the second-level queue:

While typesetting the entry, \texttt{Xy}-commands are \textit{disabled} by redefining \texttt{xy@} to just ‘eat’ its arguments, so they don’t interfere...
Finally the following ensures that the last row in the diagram is treated correctly: if the last entry was an empty row then we ignore the row entirely if the `/ifoldxymatrix@ switch is set just as we ignore \*-entries above.

```latex
\xydef\preentry@\relax \let\next@=\preentry@i
\ifoldxymatrix@ \global\Col=\@ne \ifx\lastprentry@@\empty
  \global\let\lastprentry@@=\relax \fi
\global\let\lastprentry@@=\relax \next@
\global\let\lastprentry@@=\relax \fi
```
The following procedure represents the similarity of the four lines for the coordinates in the computation above... 

**To Do:** describe this properly...

```latex
\xydef@\pre@emit@#1#2#3#4{#1\A@ #2\B@ #3\R@ #4\dimen@}
\edef\next@{{\the\A@}{\the\B@}{\the\R@}{\the\dimen@}}%
\expandafter\pre@emit@i\next@
\xydef@\pre@emit@i#1#2#3#4#5{\xy@@{%
% Emit R := min(#1,#5#2) + min(#3,#5#4)
\dimen@=#2\R@=#5\dimen@ \dimen@=#1\relax \ifdim\R@>\dimen@ \R@=\dimen@ \fi
\dimen@=#4\B@=#5\dimen@ \dimen@=#3\relax \ifdim\B@>\dimen@ \B@=\dimen@ \fi
\advance\R@\B@}%
\pre@emit@ii}
\xydef@\pre@emit@ii#1#2#3#4{\xy@@{%
% Emit #1 := #1 + #2*[ R + #3#4 ]
\dimen@=#4\dimen@=#3\dimen@ \advance\R@\dimen@ \advance#1#2\R@}}
\xydef@\Wcol@in#1{#1=\csname Wcol@the\Col\endcsname #1=.5#1}
\xydef@\Wcol@out#1{#1=\csname Wcol@the\Col\endcsname #1=.5#1}
\xydef@\Wcol@c@@#1{#1=\csname Wcol@the\count@@\endcsname #1=.5#1}
\xydef@\W@maxin#1{#1=\W@max #1=.5#1}
\xydef@\W@maxout#1{#1=\W@max #1=.5#1}
\xydef@\Hrow@in#1{#1=\csname Hrow@the\Row\endcsname #1=.5#1}
\xydef@\Hrow@out#1{#1=\csname Hrow@the\Row\endcsname #1=.5#1}
\xydef@\Hrow@c@@#1{#1=\csname Hrow@the\count@@\endcsname #1=.5#1}
\xydef@\H@maxin#1{#1=\H@max #1=.5#1}
\xydef@\H@maxout#1{#1=\H@max #1=.5#1}

**Bug:** It should be possible to change the adjustment more profoundly for individual entries, rows, and columns.

**Measure:** Take the created box apart to define the macros \Wcol@col and \Hrow@row containing the width and height of the bounding box of the entries in the row/column. Finally the commands collected in the queue are executed (hopefully they’ll invoke \pre@emit a number of times). This is a hook because some special cases won’t need it...

```
To Do: Allow non-default spacing between particular rows/columns as well as different justification of the entire matrix...

Typeset: Retypeset the entire diagram using a hacked halign: its template typesets each entry as

\POS"row, col" \* object \relax \textsc{xy-pic} commands \relax

where \textit{object} is the contents of the entry translated into \textsc{xy-pic} \langle object \rangle form and \textsc{xycommands} entails the commands used in this particular entry.
Here are the template execution macros: We reset the \texttt{\queue} for each entry, \texttt{\drop} it the usual way except first all \texttt{Xy}-commands are ‘redirected’ to the queue, and then typeset them.

\texttt{\xydef@\entry@{\relax \global\advance\Row\@one \global\Col=z@ \entry@@}}

Again some parsing determines whether this is a *-object or a normal object:

\texttt{\xydef@\entry@@{%}

\texttt{\DN@#1,#2,#3@{\xy@{ENTRY "##3##1,##2"}{\Row=##1 \Col=##2}\relax}

\cfromid@{##3##1,##2}\pfromc@}}%

\texttt{\edef\nextii@{\the\Row,\the\Col,\xymatrixprefix@@ 0} \%}

\texttt{\expandafter\next@\nextii@}

\texttt{\xyFN@\entry@@i}}

\texttt{\xydef@\entry@@i{\let\next@=\entry@@norm}

\ifoldxymatrix@\else \ifx *\next \DN@*{\xyFN@\entry@@star}\fi \fi

\next@}}

\texttt{\xydef@\entry@@star{%}

\ifx *\next \DN@*{\xyFN@\entry@@star@ii} \else \DN@##1\{\xy@@star@i{##1}\} \fi \next@}}

\texttt{\xydef@\entry@@star@i#1#2#3!{%}

\xy@@ix@{{#1}{#2}}\xy@@{\expandafter\dropentry@\the\toks9}\

\setboxz@h{\xyqall@}

\ifx\xymatrixsavedCOORD@\relax \let\xymatrixsavedCOORD@=\COORD@ \fi

\let\COORD@=\xymatrixCOORD@ \everyentry@ #3}\

\setbox\z@=\copy\voidb@x}

\texttt{\xydef@\entry@@star@ii{%}

\ifx \[\next

\DN@[##1]{\DN@####1{\def\entrymodifiers@{[##1]####1}}}

\expandafter\next@\expandafter{\entrymodifiers@\entry@@norm}}%

\else

\DN@##1{\DN@####1{\def\entrymodifiers@{##1####1}}}

\expandafter\next@\expandafter{\entrymodifiers@\entry@@norm}}%

\fi \next@}}

\texttt{\xydef@\entry@@norm{%}

\ifx \next

\DN@##1{\DN@####1{\def\entrymodifiers@{##1####1}}}

\expandafter\next@\expandafter{\entrymodifiers@\entry@@norm}}%

\fi \next@}}

\texttt{\xydef@\entry@@norm#1!{%}

\DN@##1{\DN@####1{\def\entrymodifiers@{##1####1}}}

\expandafter\next@\expandafter{\entrymodifiers@\entry@@norm}}%

\fi \next@}}

\texttt{\xydef@\entry@@norm#1!{%}

\DN@##1{\DN@####1{\def\entrymodifiers@{##1####1}}}

\expandafter\next@\expandafter{\entrymodifiers@\entry@@norm}}%

\fi \next@}}

\texttt{\dropentry@{\relax \global\advance\Row\@one \global\Col=z@ \entry@@}}

\texttt{\ifx xymatrixsavedCOORD@\relax \let xymatrixsavedCOORD@=COORD@ \fi}

\texttt{\let COORD@=xymatrixCOORD@ \everyentry@ #3}\

\setbox\z@=\copy\voidb@x}

Dropping the entry is just retypesetting it at the right location again ignoring the \texttt{Xy}-stuff:

\texttt{\dropentry@{\drop@{\relax} \xymatrixsavedCOORD@=COORD@ \let xymatrixsavedCOORD@=COORD@ \fi}

\texttt{\let COORD@=xymatrixCOORD@ \everyentry@ #1}\

\setbox\z@=\copy\voidb@x}

The macros used for queuing are these; only the \texttt{toks9} replacement is tricky; furthermore queueing is not done if we are saving because the effect will give the same result without risk of extremely long lines in the compiled file!
3.5. MATRIX FEATURE

Insert: Finally we insert the matrix in the \(X\) \(Y\)-picture logically by ensuring that it has the right size

To Do: Should handle other forms than reference point at center of "1,1" as hardcoded here. In particular reference point at baseline of "1,1" would be useful...

3.5.2 New coordinate formats

It is possible within entries to refer to all the entries of the \(X\) \(Y\)-matrix using the following special \(\langle\text{coord}\rangle\)inate forms:

<table>
<thead>
<tr>
<th>(\langle\text{coord}\rangle)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(&quot;r,c&quot;)</td>
<td>Position and extents of entry in row (r), column (c) (top left is &quot;1,1&quot;)</td>
</tr>
<tr>
<td>([\Delta r, \Delta c])</td>
<td>(\Delta r) rows below, (\Delta c) columns right of current entry</td>
</tr>
<tr>
<td>([\langle\text{hop}\rangle^*])</td>
<td>entry reached by (\langle\text{hop}\rangle)s; each (\langle\text{hop}\rangle) is one of (\text{dulr}) describing one ‘move’ to a neighbor entry</td>
</tr>
<tr>
<td>([\langle\text{hop}\rangle^+ \langle\text{place}\rangle])</td>
<td>(\langle\text{place}\rangle) on straight line to non-empty (\langle\text{hop}\rangle^*)</td>
</tr>
</tbody>
</table>

So the current entry has the synonyms \([0,0]\), \([\cdot]\), \([r\cdot]\), \([u\cdot]\), \([d\cdot]\), \([d\cdot]\), etc., as well as its ‘absolute’ name \("r,c"\).

These forms are useful for defining diagrams where the entries are related, \(e.g.,\)

\[
\begin{array}{ccc}
A & \Downarrow & \\
\mid & & \\
B & \longrightarrow & C
\end{array}
\]

was typeset by

\[
\text{\$\$\$xy}\]
If an entry outside the \texttt{Xy}-matrix is referenced then an error is reported.

In case several matrices are used in the same diagram, and they refer to each other, then it is useful to give the matrices different "\langle prefix\rangle" (setup) such that they can refer to each other using the following special coordinate forms that all have the same meaning except the target entry is picked from a particular matrix:

\begin{itemize}
\item \texttt{\langle prefix\rangle} \texttt{r,c}
\item \texttt{\langle prefix\rangle} \Delta r, \Delta c
\item \texttt{\langle prefix\rangle} \texttt{(hop)*}
\item \texttt{\langle prefix\rangle} \texttt{(hop)+ (place)}
\end{itemize}

In fact absolute references \textit{must} always be given using "\langle prefix\rangle\langle row\rangle,\langle col\rangle", \textit{even inside the matrix itself}.

Here is an example using this:

\begin{center}
\begin{tabular}{c|c|c|c}
A & B' & C & D' \\
\hline
A' & B & C' & D
\end{tabular}
\end{center}

was typeset (using the ‘frame’ extension and ‘arrow’ feature) by

\begin{verbatim}
\xymatrix{
A & B \\
C & D }
\POS\frm{-}
\xymatrix{ \\
A' & B' \\
C' & D' }
\POS\frm{-}
\end{verbatim}

These handle the parsing and interpretation of bracketed (coord)inates; the number is computed already at parse time:
The next does the actual lookup at X\text{-}time where we can still print an error message using the saved original form.

\begin{verbatim}
\xydef@\xymatrixCOORD@x{% \\
DN@#1{\expandafter\notrelaxorelse@\csname xymatrixCOORD@i##1\endcsname \\
\xymatrixCOORD@other}[[\\ \\
else\ifx ]\next \DN@]{\xymatrixCOORD@x}\\ \\
else\ifx "\next \DN@"##1"{\DN@{##1}}edef\thematrixprefix@@{\codeof\next@}\\ \\
xyFN@\xymatrixCOORD@ii}\\ \\
else \let\next@=\xymatrixCOORD@other \fi\fi\fi \next@} \\
\xydef@\xymatrixCOORD@i@u{\advance\count@ \m@ne \xyFN@\xymatrixCOORD@ii} \\
\xydef@\xymatrixCOORD@i@d{\advance\count@ \@ne \xyFN@\xymatrixCOORD@ii} \\
\xydef@\xymatrixCOORD@i@l{\advance\count@@ \m@ne \xyFN@\xymatrixCOORD@ii} \\
\xydef@\xymatrixCOORD@i@r{\advance\count@@ \@ne \xyFN@\xymatrixCOORD@ii} \\
\xydef@\xymatrixCOORD@ii{% \\
\ifcat A\noexpand\next \\
DN@##1{\expandafter\notrelaxorelse@\csname xymatrixCOORD@i##1\endcsname \\
\xymatrixCOORD@error}\\ \\
\else\ifx ]\next \DN@]{\xymatrixCOORD@x}\\ \\
else \let\next@=\xymatrixCOORD@xx \fi\fi\fi \next@} \\
\xydef@\xymatrixCOORD@error#1{% \\
\DN@{#1} \xyerror@{illegal <coord> (\codeof\next@): illegal [...] target form}{}\\ \\
\xyFN@\xymatrixCOORD@i} \\
\xydef@\xymatrixCOORD@other#1,#2{% \\
\advance\count@##1\advance\count@@##2 \xymatrixCOORD@x} \\
\xydef@\xymatrixCOORD@other\fi\fi\next@} \\
\xydef@\xymatrixCOORD@xx{% \\
\DN@##1{\xy@@{\expandafter\let\expandafter\next@\csname Q@##1\endcsname \\
\ifx\next@\relax \xyerror@RC{##1}\else \next@ \fi}\\ \\
\afterCOORD@% \\
edef\nextii@{\{\thematrixprefix@@\the\count@,\the\count@@\}}\\ \\
\expandafter\next@\nextii@} \\
\xydef@\xymatrixCOORD@after{% \\
\ifx ]\next \DN@{\afterCOORD@}\\ \\
\let\next@=\xymatrixCOORD@after \fi\fi\fi \next@} \\
\xydef@\xyerror@RC#1{% \\
\xyerror@{in entry \string\the\Row,\the\Col\string: [\codeof\next@] No \codeof\thematrixCOORD@@\space(is \string#1\string) from here}{}\\
\end{verbatim}
3.5.3 Spacing and rotation

Any matrix can have its spacing and orientation changed by adding (setup) ‘switches' between \texttt{\xymatrix} and the opening \texttt{\}. 

The default spacing between entries of matrix is changed with the switches

\begin{verbatim}
\@R\langle add op \rangle \langle \text{dimen} \rangle
\@C\langle add op \rangle \langle \text{dimen} \rangle
\@\langle add op \rangle \langle \text{dimen} \rangle
\end{verbatim}

that change row spacing, column spacing, and both, respectively, as indicated by the \langle add op \rangle and \langle \text{dimen} \rangle, where the \langle \text{dimen} \rangle may be omitted and can be given as one of \texttt{R} and \texttt{C} to indicate the current value of the parameter in question. \textbf{Note:} there is no default.

In addition, \texttt{Xy-pic} can be instructed to use a ‘fixed grid’ for the matrix with the switches

\begin{verbatim}
\@!\langle \text{dimen} \rangle
\@!\langle \text{dimen} \rangle
\@!
\end{verbatim}

that ensure that the row spacing, column spacing, and both, respectively, pretending that all entries have the size of the largest entry (without modifying the real size of the entries, of course, only the spacing – to get the entries to really have the same size use a \@*... (setup) described in §3.5.4 below). The special variants

\begin{verbatim}
\@!0
\@!=\langle \text{dimen} \rangle
\end{verbatim}

pretend that entries have zero or \langle \text{dimen} \rangle height and width for computing row and column spacing; as above inserting \texttt{R} or \texttt{C} just after the \texttt{!} makes this affect only the row or column spacing, e.g., \@!R0 means that the row spacing only is between the centers of the rows.

Finally, the spacing of things that are typeset can be adjusted separately:

\begin{verbatim}
\@M\langle add op \rangle \langle \text{dimen} \rangle
\@W\langle add op \rangle \langle \text{dimen} \rangle
\@H\langle add op \rangle \langle \text{dimen} \rangle
\@L\langle add op \rangle \langle \text{dimen} \rangle
\end{verbatim}

will adjust the entry margin, entry width, entry height, and label separation used (the latter is actually passed to the arrow feature).
The spacing can also be changed for an entire \TeX group by the declarations

\begin{verbatim}
\xymatrixrowsep {add op} {(dimen)}
\xymatrixcolsep {add op} {(dimen)}
\end{verbatim}

The default spacing for both is 2pc.
To Do: optimize processing of matrices with constant grid.
An entire matrix can be rotated by adding a rotation \( \langle \text{setup} \rangle \) of the form

\[ @\langle \text{direction} \rangle \]

This will set the orientation of the rows to \( \langle \text{direction} \rangle \) (the default corresponds to \( \mathsf{r} \), \( i.e. \), rows are oriented left to right).

The code is the first procedure of the main matrix code above (l.304); factors are used by \texttt{\pre@emit}.

### 3.5.4 Entries

The appearance of a single entry can be modified by entering it as

\[ * \langle \text{object} \rangle \langle \text{pos} \rangle \langle \text{decor} \rangle \]

This makes the particular entry ignore the entry modifiers and typeset as a kernel object with the same reference point as the (center of) the default object would have had.

Additional object \( \langle \text{modifier} \rangle \)'s may be added to an otherwise ordinary entry by using the forms

\[ **\langle \text{shape} \rangle \langle \text{entry} \rangle \]
\[ **\langle \text{modifier}\* \rangle \langle \text{entry} \rangle \]

The first sets the default \( \langle \text{shape} \rangle \) for objects (\textit{cf.} note 1.4j), the second a default size (change, \textit{cf.} note 1.4g), and the last makes it possible to add any \( \langle \text{object} \rangle \) modifier of §1.4, \textit{e.g.}, for recentering entries after the default entry form which is equivalent to \texttt{`!C +<2 \times objectmargin>'} (with the effect of centering the object and add the \textit{objectmargin}) to all sides.

**Exercise 3.9:** Typeset the following diagram:

\[
\begin{array}{c}
A \times B \xrightarrow{\mathsf{A}} B \\
/_{\mathsf{B}} \downarrow \downarrow \times_{\mathsf{A}} \\
A \xrightarrow{\mathsf{B} \times} B \times A
\end{array}
\]

(p.581)

It is also possible to use these \( @\langle \text{setup} \rangle \)'s (as usual between \texttt{\xymatrix} and the leading \{):

\[ @*\langle \text{shape} \rangle \]
\[ @* \langle \text{add op} \rangle \langle \text{size} \rangle \]

which are equivalent to changing all entries to behave as if they had started with the similar \( **\)-form.

**To Do:** Allow \( **\langle \text{add op} \rangle \langle \text{size} \rangle \langle \text{entry} \rangle \) for entries.

If the default set of entry modifiers should be changed then the following declaration must be issued before the \texttt{\xymatrix} command; this is the only way to actually switch the initial default centering and spacing off:

\texttt{\entrymodifiers={\langle \text{modifier}\* \}}}
3.5. MATRIX FEATURE

Be warned, however, that changing the entry modifiers in this way cancels any spacing setup commands discussed in §3.5.3 above – indeed the default modifiers combine two things: (1) align entry as if given the modifiers *!A, and (2) ensure that the entry has at least the size requested by any spacing setup. The default entry modifiers can be reestablished with

\entrymodifiers={!V\entrybox}

The default alignment was changed for version 3.8 following the analysis of Alex Perlis [11]; to use the entry alignment used prior to version 3.8 you can use

\entrymodifiers={!C\entrybox}

Exercise 3.10: How did the author typeset the following matrix?

\[
\begin{array}{ccc}
A & B \\
C & D \\
\end{array}
\]

Bug: The four constructions @*[...], **[...], @* (add op) (size), and, **{...}, accumulate in reverse order. Only entries starting with a single * completely override the modifiers (setup) with a @* construction.

\input{example-165}

\def\ENTRYMODIFIERS{!V\entrybox}

\def\ENTRYMODIFIERS{!C\entrybox}

(p.581)
The default encoding in the \entrybox macro and is a mix of the original setting of the size and margins with the +!!A modifier suggested by Alex Perlis [11] (introduced in version 3.8).

% Old code: 
% Horizontal: Lp will be desired width, Rp is "right offset" (from left edge). 
% Make sure width is at least @W (and then shift actual box by half). 
% Add @M margin. 
% Vertical: Dp will be desired height+depth, Up is "up offset" (from baseline). 
% Make sure height is at least @H. 
% Add @M margin. 
% Build the shifted box.
Finally, @1 is short for @M=1pt, i.e., setting the object margin to 1pt.

The individual entries can also be augmented using the following declaration, which will setup ⟨decor⟩ that should be inserted before everything else in each entry. Initially it is empty but

\everyentry={} \everyentry={⟨decor⟩}

will insert ⟨decor⟩ first in each entry; inside the counter registers \Row and \Col are set to the current entry’s row and column, respectively. For example,

\everyentry={⟨\the\Row,\the\Col⟩}
\xymatrix @*[F]@*[o] {} \POS[];[r]**\dir{..} & \POS[];[ur]**\dir{--}

will typeset

\begin{tikzpicture}
\node (1,1) at (0,0) {$1,1$};
\node (1,2) at (0,1) {$1,2$};
\node (2,1) at (0,-1) {$2,1$};
\draw [dashed] (1,1) -- (1,2);
\end{tikzpicture}

Note: When using compilation, changes to \everyentry and \entrymodifiers will not result in recompilation even when the constructed matrix changes – you may have to remove the .xyc file manually.

Exercise 3.11: How did the author typeset the following diagram?

Hints: The arrow feature was used to make the bending arrows and the frame extension for the frames around each cell.
CHAPTER 3. FEATURES

End & log

\endinput

% $Log: xymatrix.doc,v $ % Revision 3.14  2011/03/31 06:10:57  krisrose % !B changed to !V (and !H added).
% % Revision 3.13  2011/03/31 06:04:02  krisrose % Introduced !B vector that extracts original baseline offset.
% % Revision 3.12  2011/03/24 02:26:56  krisrose % Documentation fixes.
% % Revision 3.11  2011/03/14 20:14:00  krisrose % Preparing for release 3.8.6.
% % Revision 3.10  2011/02/19 21:53:21  krisrose % Include workaround to 3.8 spacing change.
% % Revision 3.9  2010/07/27 09:49:34  krisrose % Started xyling (and address updates).
% % Revision 3.8  2010/06/10 18:45:50  krisrose % Reference to GPL by URL.
% % Revision 3.7  2010/05/21 04:36:29  krisrose % Experimental version with Alex’s xymatrix adjustment modifiers as default.
% % Revision 3.6  2010/05/06 19:03:29  krisrose % Strawman.
% % Revision 3.5  2010/04/16 06:06:52  krisrose % Preparing for a new release...
% % Revision 3.4  1997/05/18 01:14:25  krisrose % Essential bugfixes.
% % Revision 3.3  1996/12/19 03:31:56  krisrose % Maintenance release
% % Revision 3.1  1995/09/05 20:31:32  kris % Releasing!
% % Revision 3.0  1995/07/07 20:14:21  kris % Major release w/new User’s Guide!
% % Revision 2.14  1995/07/06 02:56:02  kris % Buglets...
3.6 Graph feature

Vers. 3.11 by Kristoffer H. Rose (krisrose@tug.org)

This option implements ‘XY-graph’, a special combinatoric drawing language suitable for diagrams like flow charts, directed graphs, and various forms of trees. The base of the language is reminiscent of the PIC [5] language because it uses a notion of the ‘current location’ and is based on ‘moves’. But the central construction is a ‘map’ combinator that is borrowed from functional programming.

Header:
1 \% $Id: xygraph.doc,v 3.11 2011/03/14 20:14:00 krisrose Exp $
2 \%
3 \% Xy-pic ‘‘Graph Combinator feature’’ option.
4 \% Copyright (c) 1994-1997 Kristoffer H. Rose <krisrose@tug.org>
5 \%
6 \% This file is part of the Xy-pic package for graphs and diagrams in TeX.
7 \% See the companion README and INSTALL files for further information.
8 \% Copyright (c) 1991-2011 Kristoffer H. Rose <krisrose@tug.org>
9 \%
10 \% The Xy-pic package is free software; you can redistribute it and/or modify
11 \% it under the terms of the GNU General Public License as published by the
12 \% Free Software Foundation; either version 2 of the License, or (at your
13 \% option) any later version.
14 \%
15 \% The Xy-pic package is distributed in the hope that it will be useful, but
16 \% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
XY-graph makes use of facilities of the ‘arrow’ feature option of \( \S 3.3 \), which is therefore required.

The graph state always has the current value of \( ? \) both as the stacktop and in the special \langle id \rangle "?NODE".

In fact \LaTeX users can use a \texttt{graph} environment.
3.6. GRAPH FEATURE

Notes

3.6a. A **move** is to establish a new current node.

3.6b. To **draw** something is simply to draw a line or the specified ⟨arrow⟩ from the current node to the specified target node. The target then becomes the current node. All the features of arrows as described in §3.3 can be used, in particular arrows can be labelled and segmented, but with the change that ⟨path-pos⟩ means ⟨node⟩ as explained in note §3.3e.

3.6c. To **map over a list** is simply to save the current node and then interpret the ⟨list⟩ with the following convention:

- Start each element of the list with the current node as saved and p as the previous list element, and
- let the ? ⟨node⟩ refer to the saved current node explicitly.

3.6d. The & and \ special moves are included to make it simple to enter ‘matrix-like’ things as graphs – note that they will not be automatically aligned, however, for that you should use the !M escape. & is the same as [r] and \ is the same as [r]!{y+(0,-1)-(0,0)} which uses a kernel escape to moves to the first column in the next row (where the first column is on the y-axis of the current coordinate system).

**Note:** If you use the form *{...}* for nodes then you don’t have to change them if you decide to use an XY-matrix.

3.6e. Typeset ⟨it⟩ and make it the current node. Also saves ⟨it⟩ for later reference using "⟨id⟩": if ⟨it⟩ is a simple letter, or digit, then just as "⟨it⟩"; if ⟨it⟩ is of the form {text} or *...{text} then as "text".
With the = addition it is possible to save explicitly in case several nodes have the same text or a node has a text that it is impractical to use for reference. In fact using the form \langle\text{it}\rangle = \langle\text{id}\rangle will only save the node as "\langle\text{id}\rangle" and not as "\langle\text{it}\rangle"!

Exercise 3.12: How did the author typeset this?

\[
A \xleftarrow{} A \xrightarrow{} A
\]

(p.581)

3.6f. Moving by a series of hops is simply moving in a grid as the sequence of duLR (for down/up/left/right) indicates. The grid is a standard cartesian coordinate system with 3pc unit unless the current base is redefined using \[
\text{!}\{\ldots\}\]
with an appropriate \langle\text{pos}\rangle\text{ition} containing : and :: as described in note 1.3d.

To Do: Describe the use of \langle\text{move}\rangle\text{s} with \langle\text{place}\rangle\text{s} in detail… in particular (1) ‘until perpendicular to …’ and (2) ‘until intercepts with …’ can be coded...

3.6g. This ‘escapes’ into the \text{Xy-pic} kernel language and interprets the \langle\text{pos}\rangle \langle\text{decor}\rangle. The current node is then set to the resulting \text{c} object and the grid from the resulting \text{base}.

The effect of the \langle\text{pos}\rangle \langle\text{decor}\rangle can be completely hidden from \text{Xy-graph} by entering it as \{	ext{save} \langle\text{pos}\rangle \langle\text{decor}\rangle \text{restore}\}.

3.6h. It is possible to insert a \langle\text{matrix}\rangle in a graph provided the ‘matrix’ option described in §3.5 has been loaded: it overwrites the node with the result of \text{\xymatrix}\langle\text{matrix}\rangle. Afterwards the graph grid is set as the top left ‘square’ of the matrix, i.e., with \[\text{d}\] and \[\text{r}\] adjusted as they work in the top left entry.

Bug: \[dr\] immediately after the matrix will work as expected, e.g., make the center of "2,2" the current node, but others might not, e.g., \[rr\] will not necessarily place the current node on top of "1,3".
3.6. GRAPH FEATURE

3.6i. It is possible to insert a ⟨polygon⟩ or an ⟨ellipse⟩ in a graph provided the poly option described in §3.7 or the arc option described in §3.9 has been loaded, respectively: it will have c as the current node, p as the previous one, and the the current base has the ⟨hop⟩s [r] and [u] as base vectors.

Note: lattices, knots, etc., can also be used but no special syntax is useful since the !{...} syntax is adequate.

3.6j. This allows setting of some parameters of the graph: !~⟨setup⟩ should be one of the following:

| !~{⟨arrow⟩} | include with every : arrow |
| !~-{⟨arrow⟩} | include with every - line |
| !~*{⟨modifiers⟩} | include with every non-* node |
| !~⟨letter⟩{⟨graph⟩} | define new graph escape !⟨letter⟩ |

These are destructive: the previous value is lost; the default is established by the sequence !~:{!~{-}{!~*{}} making: create simple arrows, - plain lines, and formatting default nodes in math mode with the default objectmargin.

3.6k. {\xydefcsname@{graph !P}#1#{\NODE@poly{#1}}}
The last possibility is also available as a command

\newgraphescape{⟨letter⟩}{⟨graph⟩}

that makes the specified escape generate the ⟨graph⟩ as a macro; with it it is possible to pass arguments to the ⟨graph⟩ using the standard \TeX \def method: The declaration code

\newgraphescape{i}#1#2{
  []!{+O="o#2"*<10pt>{};p/#1**{},"o#2"
    -/4pt/**\circ"2pt{};
    +0;p-/a(-30)24pt/**\dir{"}="X2"
    ;p-/a(-60)24pt/="X1"**\dir{~}
    ;(?(.5),="i#2",
    p-/-a(-60)24pt/**\dir{~},
    "o#2"."i#2"."X1"."X2"})

is (rather complicated kernel code) that makes the node escape !idn typeset an ‘inverter’ oriented with the d corner as the output with input named "in" and output named "on" such that the graph

\xygraph{ []!iR1 ("i1"[l]x - "i1") - [r]z }

will typeset

\hspace{1cm}
\begin{tikzpicture}
  \node (x) at (0,0) {$x$};
  \node (o) at (2,0) {$z$};
  \draw (x) -- (o);
\end{tikzpicture}

The \newgraphescape{⟨letter⟩} declaration defines a macro with name \graph !⟨letter⟩. It is slightly complicated by the fact that we accept \LaTeX-style [n] arguments (when using \LaTeX, that is \(\otimes\). Thus an internal macro named \G!⟨letter⟩ is defined using the appropriate command definition primitive; this is in turn invoked by \graph !⟨letter⟩ after activating the ⟨graph⟩ parser.

The final exercise illustrates much of the above.
Exercise 3.13: Typeset

![Graph Diagram]

(p.582)
### 3.7 Polygon feature

Vers. 3.11 by Ross Moore (ross.moore@mq.edu.au)

This feature provides a means for specifying the locations of vertices for regular polygons, with any number (≥ 3) of sides. Polygons can be easily drawn and/or the vertex positions used to construct complex graphics within an Xy-picture. Many non-regular polygons can be specified by setting a non-square basis.

**Header:**

```latex
\% $Id: xypoly.doc,v 3.11 2011/03/14 20:14:00 krisrose Exp$
\% Xy-pic `Polygon' feature.
\% Copyright (c) 1994-1997 Ross Moore <ross.moore@mq.edu.au>
\% This file is part of the Xy-pic package for graphs and diagrams in TeX.
\% See the companion README and INSTALL files for further information.
\% Copyright (c) 1991-2011 Kristoffer H. Rose <krisrose@tug.org>
\% The Xy-pic package is free software; you can redistribute it and/or modify
\% it under the terms of the GNU General Public License as published by the
\% Free Software Foundation; either version 2 of the License, or (at your
\% option) any later version.
\%
\% The Xy-pic package is distributed in the hope that it will be useful, but
\% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
\% or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License
\% for more details.
\%
\% You should have received a copy of the GNU General Public License along
\% with this macro package; if not, see http://www.gnu.org/licenses/.
\%
\% \ifx\xyloaded\undefined \input xy \fi
\provide{poly}{Polygon feature}{\stripRCS$Revision: 3.11 $}\%
{Ross Moore}{ross.moore@mq.edu.au}\%
{Mathematics Department, Macquarie University, NSW~2109, Australia}
```

Some of the effects use \texttt{ar} so make sure the `arrow' feature is loaded.

```latex
\require{arrow}\xycatcodes
```
A polygon is most easily specified using . . .

\xypolygon\{\}
\xypolygon\{\tok\}
\xypolygon\{\object\}

with a general \object\ at each vertex;

Here \langle\number\rangle is a sequence of digits, giving the number of sides. If used within an \xy\endxy environment then the polygon will be centred on c, the current \langle\pos\rangle. However an \xypolygon\ can be used outside such an environment, as “stand-alone” polygon; the whole picture must be specified within the \xypolygon\ command.

In either case the shape is obtained by spacing vertices equally around the “unit circle” with respect to the current basis. If this basis is non-square then the vertices will lie on an ellipse. Normally the polygon, with at most 12 vertices, is oriented so as to have a flat base when specified using a standard square basis. With more than 12 vertices the orientation is such that the line from the centre to the first vertex is horizontal, pointing to the right. Any other desired orientation can be obtained, with any number of vertices, by using the \{\dots\} as described below.

The general form for \xypolygon\ is . . .

\xypolygon\langle\number\rangle"\langle\prefix\rangle"\{\langle\switches\rangle\dots\}

where the \langle\prefix\rangle and \langle\switches\rangle are optional. Their uses will be described shortly.

A \xypolygon\ establishes positions for the vertices of a polygon. At the same time various things may be typeset, according to the specified \langle\switches\rangle. An \langle\object\rangle may be dropped at each vertex, “spokes” drawn to the centre and successive vertices may be connected as the polygon’s “sides”. Labels and breaks can be specified along the spokes and sides.

Each vertex is automatically named: \"1", \"2", \ldots, \langle\number\rangle with \"0\" as centre. When a \langle\prefix\rangle has been given, names \"\langle\prefix\rangle0\", \ldots, \"\langle\prefix\rangle\langle\number\rangle\" are used instead. While the polygon is being constructed the macro \xypolynum\ expands to the number of sides, while \xypolynode\ expands to the number of each vertex, spoke and side at the time it is processed. This occurs in the following order: \textit{vertex} 1, \textit{spoke} 1, \textit{vertex} 2, \textit{spoke} 2, \textit{side} 1, \textit{vertex} 3, \textit{spoke} 3, \textit{side} 2, \ldots, \textit{vertex} \textit{n}, \textit{spoke} \textit{n}, \textit{side} \textit{n} – 1, \textit{side} \textit{n} where the final side joins the last vertex to the first.

The macro \xypolyname\ holds the name of the polygon, which is \langle\prefix\rangle if supplied. In this case the value of \xypolynum\ is also stored as \langle\prefix\\numsidestr\}, accessible outside the polygon.

As stated above, a polygon with up to 12 vertices is oriented so as to have a flat base, when drawn using a standard square basis. Its vertices are numbered in anti-clockwise order, commencing with the one at horizontal-right of centre, or the smallest angle above this (see example below). With more than 12 vertices then vertex \"1\" is located on the horizontal, extending to the right from centre (assuming a standard square basis). By providing a switch of the form \{\langle\angle\rangle\} then the vertex \"1\" will be located on the unit circle at \langle\angle\rangle° anti-clockwise from “horizontal” — more correctly, from the X-direction in the basis to be used when setting the polygon, which may be established using a \{\langle\\ldots\rangle\} switch.

Exercise 3.14: Give code to typeset these.
One important use of \texttt{⟨prefix⟩} is to allow the vertices of more than one polygon to be accessed subsequently within the same picture. Here are some examples of this, incorporating the \texttt{~:{...}} switch to perform simple rescalings. Firstly the edges of a dodecahedron as a planar graph:

\begin{xy}
/l1.5pc/::\arpolygon5"A"{~:{(1.875,0):}~>{}}{},
\arpolygon5"B"{~:{(-2.95,0):}~>{}}{},
\arpolygon5"C"{~:{(-3.75,0):}~>{}}{},
\arpolygon5"D"{~:{(1.875,0):}~>{}}{},
"A1"\PATH={**@{-}}"B1""C4""B2","A2"\PATH={**@{-}}"B2""C5""B3","A3"\PATH={**@{-}}"B3""C1""B4","A4"\PATH={**@{-}}"B4""C2""B5","A5"\PATH={**@{-}}"B5""C3""B1","C1""D1"**@{-},"C2""D2"**@{-},"C3""D3"**@{-},"C4""D4"**@{-},"C5""D5"**@{-}
\end{xy}

Next a hexagonal pyramid, a rectangular box and an octahedral crystal specified as a triangular antiprism. Notice how the \texttt{~:{...}} switch is used to create non-square bases, allowing the illusion of 3D-perspective in the resulting diagrams:

\begin{xy}
/r2pc/::"A",+(.2,1.5)="B","A",
\arpolygon6{~:{(1,-.1):(0,.33)::}~<="B"**@{-}}{}
\end{xy}
\begin{xy}
/r2pc/::"A"{~:{(0,.7)::}},+(.7,1.1),
\arpolygon4"B"{~:{(0,.75)::}},"A1";"B1"**@{-},"A2";"B2"**@{-},"A3";"B3"**@{-},"A4";"B4"**@{-}
\end{xy}
\begin{xy}
/r2pc/::"A"{~:{(0,0.7)::}},+(.7,1.1),
\arpolygon3"B"{~:{(-.85,0):(-.15,.8)::}}
\end{xy}

\texttt{\xypolynode} simply reads the value of a counter called \texttt{\xypolynode@}. It is set initially at -1, to indicate no polygon yet. This is used to establish the default naming in case polygons are nested.
3.7. POLYGON FEATURE

To allow recursion the values of \( \texttt{xypolyNUMSIDES} \) and \( \texttt{xypolynode} \) are saved upon entering an \( \texttt{xypolygon} \), to be restored at the end.

Read the number of vertices by parsing digits.

Next we check whether a prefix has been supplied for creating the vertex names. If so it is stored as \( \texttt{xypolyPREFIX} \), which otherwise expands to empty. The number of sides is also stored using the prefix.
Currently any tokens remaining before the opening brace are discarded, with an accompanying message.

Next it is time to analyse the braced information, e.g. to set switches and/or adjust the scale. If this information is empty {} then the default values are used.

Vertex object: Unless the first character is~, signifying a “switch”, then the whole of the braced material is taken as specifying the ⟨object⟩ for each vertex. It will be typeset with a circular edge using \drop[o]..., except when there is just a single token ⟨tok⟩. In this case it is dropped as \drop=0{⟨tok⟩}, having zero size. An object can also be dropped at each vertex using the switch~*{...}, in which case it will be circular, with the current objectmargin applied.

The next example illustrates three different ways of specifying a \circ at the vertices.
Switches

The allowable switches are given in the following table:

<table>
<thead>
<tr>
<th>Switch</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>~:{...}</td>
<td>useful for rescaling.</td>
</tr>
<tr>
<td>~*{(object)}</td>
<td>(object) at each vertex.</td>
</tr>
<tr>
<td>~={angle}</td>
<td>align first vertex.</td>
</tr>
<tr>
<td>~&lt;{...}</td>
<td>directional for “spokes”;</td>
</tr>
<tr>
<td>~&lt;&lt;&lt;&lt;{(arrow)}</td>
<td>use ⟨arrow⟩ for spokes;</td>
</tr>
<tr>
<td>~&gt;{...}</td>
<td>labels &amp; breaks on spokes.</td>
</tr>
<tr>
<td>~&gt;&gt;{(arrow)}</td>
<td>use ⟨arrow⟩ for sides;</td>
</tr>
<tr>
<td>~&gt;&gt;&gt;{...}</td>
<td>labels &amp; breaks on sides.</td>
</tr>
</tbody>
</table>

Using ~<<<<{(arrow)} or ~>>{(arrow)} is most appropriate when arrowheads are required on the sides or spokes, or when labels/breaks are required. Here ⟨arrow⟩ is as in figure 3.2, so it can be used simply
to specify the style of directional to be used. Thus \texttt{~<<{}} sets each spoke as a default arrow, pointing outwards from the centre; \texttt{~<<{(array)}} suppresses the arrowhead, while \texttt{~><{(array)}} uses an empty arrow along the sides. Labels and breaks are specified with \texttt{~<>{(array)}} and \texttt{~>>{(array)}}, where the \{\ldots\} use the notation for a \{label\}, as in figure 3.1.

When no tips or breaks are required then the switches \texttt{~<{}} \ldots \texttt{~>>{}} are somewhat faster, since less processing is needed. Labels can still be specified with \texttt{~<>{(array)}} and \texttt{~>>{(array)}}, but now using the kernel’s \{place\} notation of figure 1.1. In fact any kernel code can be included using these switches. With \texttt{~<>{}} the current \(p\) and \(c\) are the centre and vertex respectively, while for \texttt{~>>{}} they are the current vertex and the previous vertex. (The connection from vertex "\langle(number)\rangle" to vertex "1" is done last.) The pyramid above is an example of how this can be used. Both \texttt{~<<{(arrow)}} and \texttt{~<<{(arrow)}} can be specified together, but only the last will actually be used; similarly for \texttt{~>{}} \ldots \texttt{~<>{}} and \texttt{~>{}} \ldots \texttt{~<>{}}.

Use of the \texttt{~={}} switch was described earlier. When using the \texttt{~:{}} more can be done than just setting the base. In fact any kernel code can be supplied here. It is processed prior to any other part of the polygon. The graphics state has \(c\) at the centre of the polygon, \(p\) at the origin of coordinates within the picture and has basis unchanged from what has previously been established. The current point \(c\) will be reset to the centre following any code interpreted using this switch.

Note that \texttt{~!} was appended by \texttt{xypoly@@}, in order to indicate the end of the braced tokens. The macro \texttt{xypolywhich@} is the main parser, with \texttt{xypoly@SPOKES@} and \texttt{xypoly@SIDES@} handling the subcases for \texttt{~<} and \texttt{~>}.

\def\alphanum{\ifcase\xypolynode\or A \or B\or C\or D\or E\or F\or G\or H\fi}
\xy/r3pc/: {\xypolygon3{~={40}}},
{\xypolygon4{~={40}~>{\{--\}}}},
{\xypolygon5{~={40}}},
{\xypolygon6{~={40}~>{\{--\}}}},
{\xypolygon11{~={40}}},
{\xypolygon50{~={40}~>.}}, +/r8pc/,
{\xypolygon7{~<<<<{0}~><{}}
~<<{|*[@{x}]~^{*\{\alphanum}
~>>{\alpha_{\xypolynode}}}}}
\endxy
A further simplification exists for sides and spokes without ⟨arrow⟩s. If ⟨tok⟩ is a single character then \(~⟩⟨tok⟩, \(~⟩{⟨tok⟩}, \(~⟩{{⟨tok⟩}}⟩ all specify the directional \(\text{dir}{⟨tok⟩}\); similarly with the \(~<\) switch. On the other hand, compound directionals require all the braces, e.g. \(~>\{\langle\sim\rangle\} \text{ and } \overset{\sim}{\overset{\sim}{\sim}}\). This is achieved with the macro \texttt{\checkpolytok@#1#2#3#4@!#5} which handles the bare ⟨tok⟩ case by adding braces around ⟨tok⟩ which has been passed as #3, preceding it with #2. This is then \texttt{\def’d} to the control-sequence name passed as #1. Finally continue with the command passed as #5.

After all switches have been processed, remaining tokens are used to specify the ⟨object⟩ for each vertex. Such tokens will be used directly after a \texttt{\drop}, so can include object ⟨modifier⟩s as in figure 1.5. If an ⟨object⟩ has already been specified, using the \(~*\) switch, then the following message will be written to the \TeX{} log:

\texttt{Xy-pic Warning: vertex already specified, discarding unused tokens:}

with tokens at the end indicating what remains unprocessed. Similarly extra tokens before the \{\ldots\} generate a message:

\texttt{Xy-pic Warning: discarding unused tokens:}
Nested Polygons
When \texttt{\xypolygon} is specified within either a \texttt{~<>\{\ldots\} or \texttt{~>>\{\ldots\}} switch for another polygon, then the inner polygon inherits a name which incorporates also the number of the part on which it occurs, as given by \texttt{\xypolynode}. This name is accessed using \texttt{\xypolyname}. In the following example the inner polygon is placed using \texttt{~<>} in order to easily adjust its orientation to the outward direction of the spokes.

\begin{verbatim}
\xypolygon4{~:{/r5pc/:}
~<>{*rm<8pt>{o}\xypolygon4{~:{/-2pc/:}
~*{\xypolyname\xypolynode}}}
\end{verbatim}

Notice how nested polygons inherit names \texttt{"1,1"}, \texttt{"1,2"}, \ldots, \texttt{"4,1"}, \ldots, \texttt{"4,4"} for their vertices. If a (prefix) is supplied at the outermost level then the names become: \texttt{"(prefix)\textbackslash i,\textbackslash j"}. Specifying a (prefix) for the inner polygon overrides this naming scheme. The same names may then be repeated for each of the inner polygons, allowing access afterwards only to the last—possibly useful as a memory saving feature when the vertices are not required subsequently.

Four levels of nesting gives a quite acceptable “Sierpinski gasket”. The innermost triangle is provided by \texttt{\blacktriangle} from the \texttt{AMS} symbol font \texttt{msam5}, at 5-point size. Further levels can be achieved using the POSTSCRIPT backend, otherwise line segments become too small to be rendered using XY-fonts.

\begin{verbatim}
\font\msamv=msam5 at 5pt
\def\blacktriangle{{\msamv\char'116}}
\def\objectstyle{\scriptscriptstyle}
\xypolygon3{~:{/r5.2pc/:}
~>{}~>{?\xypolygon3"a"{~:{(.5,0):}}
~>{}~>{?\xypolygon3"b"{~:{(.5,0):}}
~>{}~>{?\xypolygon3"c"{~:{(.5,0):}}
~>{}~>{?\xypolygon3"d"{~:{(.5,0):}}
~>{}?!/d.5pt/=0\hbox{\blacktriangle}}
\end{verbatim}
3.7. POLYGON FEATURE

Note the use of naming in this example; when processing this manual it saves 13,000+ words of main memory and 10,000+ string characters as well as 122 strings and 319 multi-letter control sequences.

Coordinates for the vertices are read from trigonometry tables using control sequences
which expand to the values of $\cos \frac{2\pi}{n}$, $\sin \frac{2\pi}{n}$, $\cos \frac{4\pi}{n}$, $\sin \frac{4\pi}{n}$, $\cos \frac{5\pi}{n}$, $\sin \frac{5\pi}{n}$ and are also available for other uses. The parameter $n$ must be a non-negative integer up to 12. The complete table is in figure 3.6.

**Specific Polygons:** All the information from the switches is passed as 8 separate parameters to a macro appropriate to the number of sides. The parameters have the following uses, and are set as shown:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>pre-arrow: set by <code>&lt;&lt;{...}</code> else</td>
</tr>
<tr>
<td>#2</td>
<td>vertex (object), set by <code>*{...}</code></td>
</tr>
<tr>
<td>#3</td>
<td>prefix for naming</td>
</tr>
<tr>
<td>#4</td>
<td>post-vertex: empty with \ars else <code>**{...}</code></td>
</tr>
<tr>
<td></td>
<td>where <code>{...}</code> is set by <code>&lt;&lt;{...}</code></td>
</tr>
<tr>
<td>#5</td>
<td>post-arrow/vertex: set by <code>&lt;&lt;{...}</code></td>
</tr>
<tr>
<td>#6</td>
<td>pre-edge/arrow, set by <code>&gt;&gt;{...}</code></td>
</tr>
<tr>
<td>#7</td>
<td>post-edge: empty with \ars else <code>**{...}</code></td>
</tr>
<tr>
<td></td>
<td>where <code>{...}</code> is set by <code>&gt;&gt;{...}</code></td>
</tr>
<tr>
<td>#8</td>
<td>post-arrow/edge, set by <code>&gt;&gt;{...}</code></td>
</tr>
</tbody>
</table>

The organisation of this information is done by `\xyPolygon` which takes as a single parameter a macro appropriate to the number of sides, supplied by `\xypoly`.

```latex
\def\xypoly@@@#1 {\count@=\xypolyNUMSIDES@ \relax
  \ifx\xypolyALIGN@\empty
    \ifnum\count@<\thr@@ \DN@{\xypolytoosmall@{\the\count@}}%
    \else \advance\count@-\thr@@
      \ifcase\count@
        \DN@{\xyPolygon@@@\xytriangle@@}
      \or \DN@{\xyPolygon@@@\xysquare@@}
      \or \DN@{\xyPolygon@@@\xypentagon@@}
      \or \DN@{\xyPolygon@@@\xyhexagon@@}
      \or \DN@{\xyPolygon@@@\xyheptagon@@}
      \or \DN@{\xyPolygon@@@\xyoctagon@@}
      \or \DN@{\xyPolygon@@@\xynonagon@@}
      \or \DN@{\xyPolygon@@@\xydecagon@@}
      \or \DN@{\xyPolygon@@@\xyundecagon@@}
      \else \DN@{\xylargePolygon@{0}{\xypolyNUMSIDES@}}%
    \fi
  \else \DN@{\xylargePolygon@{\xypolyALIGN@}{\xypolyNUMSIDES@}}%
  \fi\next@
\xypoly@@@x}
```

Here is `\xyPolygon@@@#1`. It constructs a list of tokens to be interpreted, after having closed the grouping that was current while parsing. It starts by setting the value of `\xypolyNUMSIDES@`, then a `\save` before adjusting the scale according to the value of `\xypolySCALE@`. 

\[
\begin{align*}
\cos \frac{2\pi}{n} & \quad \sin \frac{2\pi}{n} \\
\cos \frac{4\pi}{n} & \quad \sin \frac{4\pi}{n} \\
\cos \frac{5\pi}{n} & \quad \sin \frac{5\pi}{n}
\end{align*}
\]
3.7. POLYGON FEATURE

Note how \endgroup closes the grouping. All the information for the layout of the polygon is contained in \xypolyNUMSIDES and the specific polygon macro contained in the final \next.

Here is where the polygons are defined, individually.
\def\xyhexagon{\pos1\#30"+(1,0)\#2="31"\#4\#5\relax
\xy@{\advance\xypolygon\@ne}\pos1\#30"+.5,\halfrootthree\#2="32"\#4\#5\relax
\xy@{\advance\xypolygon\@m}\pos1\#31"\#6"\#7\#8\relax
\xy@{\advance\xypolygon}\pos1\#30"-(1,0)\#2="33"\#4\#5\relax
\xy@{\advance\xypolygon\@ne}\pos1\#33"6"\#6"\#7\#8\relax
\xy@{\advance\xypolygon}\pos1\#30"+(\sinTwoPIon7,\cosTwoPIon7)\#2="31"\#4\#5\relax
\xy@{\advance\xypolygon\@ne}\pos1\#30"+(0,1)\#2="32"\#4\#5\relax
\xy@{\advance\xypolygon\@m}\pos1\#31"6"\#7\#8\relax
\xy@{\advance\xypolygon}\pos1\#30"-(\sinThreePIon7,\cosThreePIon7)\#2="33"\#4\#5\relax
\xy@{\advance\xypolygon\@ne}\pos1\#33"6"\#6"\#7\#8\relax
\xy@{\advance\xypolygon}\pos1\#30"-(\sinPIon7,-\cosPIon7)\#2="34"\#4\#5\relax
\xy@{\advance\xypolygon\@m}\pos1\#35"6"\#6"\#7\#8\relax
\xy@{\advance\xypolygon\@ne}\pos1\#35"6"\#6"\#7\#8\relax
\xy@{\advance\xypolygon}\pos1\#30"+(\sinPIon7,\cosPIon7)\#2="34"\#4\#5\relax
\xy@{\advance\xypolygon\@ne}\pos1\#33"6"\#6"\#7\#8\relax
\xy@{\advance\xypolygon}\pos1\#30"-(\sinThreePIon7,-\cosThreePIon7)\#2="36"\#4\#5\relax
\xy@{\advance\xypolygon\@m}\pos1\#35"6"\#6"\#7\#8\relax
\xy@{\advance\xypolygon} \xyhexagon } 
\def\xyheptagon{\pos1\#30"+(\sinTwoPIon7,\cosTwoPIon7)\#2="31"\#4\#5\relax
\xy@{\advance\xypolygon\@ne}\pos1\#30"+(0,1)\#2="32"\#4\#5\relax
\xy@{\advance\xypolygon\@m}\pos1\#31"6"\#7\#8\relax
\xy@{\advance\xypolygon}\pos1\#30"-(\sinThreePIon7,\cosThreePIon7)\#2="33"\#4\#5\relax
\xy@{\advance\xypolygon\@ne}\pos1\#33"6"\#6"\#7\#8\relax
\xy@{\advance\xypolygon}\pos1\#30"-(\sinPIon7,-\cosPIon7)\#2="34"\#4\#5\relax
\xy@{\advance\xypolygon\@m}\pos1\#35"6"\#6"\#7\#8\relax
\xy@{\advance\xypolygon} \xyheptagon}
CHAPTER 3. FEATURES

\xydef\xyoctagon@@#1#2#3#4#5#6#7#8{% 
\xy@@{\xypolynode@=\@ne}\
\POS(#1)+(#3)+(#5)\relax 
\xy@@{\advance\xypolynode@=\tw@}\
\POS(#1)-(#3)-(#5)\relax } 

\xydef\xynonagon@@#1#2#3#4#5#6#7#8{% 
\xy@@{\xypolynode@=\@ne}\
\POS(#1)+(#3)+(#5)\relax 
\xy@@{\advance\xypolynode@=\tw@}\
\POS(#1)-(#3)-(#5)\relax } 

\xydef\xydodecagon@@#1#2#3#4#5#6#7#8{% 
\xy@@{\xypolynode@=\@ne}\
\POS(#1)+(#3)+(#5)\relax 
\xy@@{\advance\xypolynode@=\tw@}\
\POS(#1)-(#3)-(#5)\relax } 

\xypolynode@=\@ne 
\POS(0)+(#3)+(#5)\relax
\xydef\xyundecagon@@#1#2#3#4#5#6#7#8% \
\POS"#30"#1"#30"-(\cosPIon5,\sinPIon5)#2="#37"#4#5\relax \
\xy@{(\advance\xypolynode@\m@ne)% 
\POS"#36"#6"#37"#7#8\relax 
\xy@{(\advance\xypolynode@\tw@)% 
\POS"#30"#1"#30"-(\sinPIon10,\cosPIon10)#2="#38"#4#5\relax 
\xy@{(\advance\xypolynode@\m@ne)% 
\POS"#37"#6"#38"#7#8\relax 
\xy@{(\advance\xypolynode@\tw@)% 
\POS"#30"#1"#30"+(\sinPIon10,-\cosPIon10)#2="#39"#4#5\relax 
\xy@{(\advance\xypolynode@\m@ne)% 
\POS"#38"#6"#39"#7#8\relax 
\xy@{(\advance\xypolynode@\tw@)% 
\POS"#30"#1"#30"-(\cosPIon5,-\sinPIon5)#2="#310"#4#5\relax 
\xy@{(\advance\xypolynode@\m@ne)% 
\POS"#39"#6"#310"#7#8\relax 
\xy@{(\advance\xypolynode@\m@ne)% 
\POS"#310"#6"#31"#7#8\relax }
\xydef\xyundecagon@@#1#2#3#4#5#6#7#8{% 
\xy@{(\xypolynode@=\@ne)% 
\POS"#30"#1"#30"+(\sinFourPIon11,\cosFourPIon11)#2="#31"#4#5\relax 
\xy@{(\advance\xypolynode@\@ne)% 
\POS"#30"#1"#30"+(\sinTwoPIon11,\cosTwoPIon11)#2="#32"#4#5\relax 
\xy@{(\advance\xypolynode@\m@ne)% 
\POS"#31"#6"#32"#7#8\relax 
\xy@{(\advance\xypolynode@\tw@)% 
\POS"#30"#1"#30"+(0,1)#2="#33"#4#5\relax 
\xy@{(\advance\xypolynode@\m@ne)% 
\POS"#32"#6"#33"#7#8\relax 
\xy@{(\advance\xypolynode@\tw@)% 
\POS"#30"#1"#30"+(-\sinTwoPIon11,\cosTwoPIon11)#2="#34"#4#5\relax 
\xy@{(\advance\xypolynode@\m@ne)% 
\POS"#33"#6"#34"#7#8\relax 
\xy@{(\advance\xypolynode@\tw@)% 
\POS"#30"#1"#30"+(-\sinFourPIon11,\cosFourPIon11)#2="#35"#4#5\relax 
\xy@{(\advance\xypolynode@\m@ne)% 
\POS"#34"#6"#35"#7#8\relax 
\xy@{(\advance\xypolynode@\tw@)% 
\POS"#30"#1"#30"-(\sinFivePIon11,\cosFivePIon11)#2="#36"#4#5\relax 
\xy@{(\advance\xypolynode@\m@ne)% 
\POS"#35"#6"#36"#7#8\relax 
\xy@{(\advance\xypolynode@\tw@)% 
\POS"#30"#1"#30"-(\sinThreePIon11,\cosThreePIon11)#2="#37"#4#5\relax 
\xy@{(\advance\xypolynode@\m@ne)% 
\POS"#35"#6"#37"#7#8\relax 
\xy@{(\advance\xypolynode@\tw@)% 
\POS"#30"#1"#30"-(\sinPIon11,\cosPIon11)#2="#38"#4#5\relax 
\xy@{(\advance\xypolynode@\m@ne)% 
\POS"#37"#6"#38"#7#8\relax
We need a looping construction that will not interfere with others used at a high level.

\xydef\xypolyrepeat\{\fi\}
\xydef\xypoly\{}
\xydef\xypolyiterate{}\xypolyiterate\else\fi\}
\xydef\xypolyloop#1\xypolyrepeat\{\def\xypoly\{}#1\xypolyiterate\}

\xydef\xypolygon#1#2#3#4#5#6#7#8#9{\save
\vfromcartesianangle#9\edef\next@{\the\X@c,\the\Y@c\addGT\empty}\%
\xy@={\POS#1#30"+}\expandafter\addtotoks\expandafter{\next@#2="#31"#4#5\relax}\%
\xy@@\don'tleave\the\toks@\%
\%
\xypolyloop\%
\div\dimen@\xypolyNUMSIDES@\relax\advance\dimen@.5\count@\%
\edef\tmp@{%
\noexpand\vfromcartesianangle\{\expandafter\removePT\the\dimen@\}}%
\tmp@ \edef\next@{\the\X@c,\the\Y@c\addGT\empty}\%
\xy@={\advance\xypolygon\{}\count@=\xypolygon\%
\toks@={\POS#1#30"+}#9\edef\nextii@{\the\count@}\%
\expandafter\expandafter\expandafter\addtotoks\expandafter{\next@2="#31"#4#5\relax}\%
\xy@\don'tleave\the\toks@\%
\%
\xypolyloop\%
\dimen@=360\p@\count@=\xypolyNUMSIDES@\relax\advance\dimen@.5\count@\%
\divide\dimen@\xypolyNUMSIDES@\relax
\multiply\dimen@\xypolygon\relax\advance\dimen@9\p@\%
\edef\tmp@{%
\noexpand\vfromcartesianangle\{\expandafter\removePT\the\dimen@\}}%
\tmp@ \edef\next@{\the\X@c,\the\Y@c\addGT\empty}\%
\xy@={\advance\xypolygon\} \count@=\xypolygon\%
\toks@={\POS#1#30"+}#9\edef\nextii@{\the\count@}\%
\expandafter\expandafter\expandafter\addtotoks\expandafter{\next@2="#31"#4#5\relax}\%
\expandafter\expandafter\expandafter\addtotoks\expandafter{\next@2="#31"#4#5\relax}\%
\expandafter\expandafter\expandafter\addtotoks\expandafter{\next@2="#31"#4#5\relax}\%
\expandafter\expandafter\expandafter\addtotoks\expandafter{\next@2="#31"#4#5\relax}\%

3.7. POLYGON FEATURE

\expandafter\addtotoks@\expandafter{\nextii@"#4#5\relax}%
\the\toks@
\count@=\xypolynode@ \xy@{\advance\xypolynode@\m@ne}%
\toks@={\POS"#3}\expandafter\addtotoks@\expandafter{\the\xypolynode@"#6"#3}%
\expandafter\addtotoks@\expandafter{\the\count@"#7#8\relax}%
\the\toks@
\xy@{\advance\xypolynode@\@ne}\count@=\xypolyNUMSIDES@ \relax
\ifnum\xypolynode@<\count@ \relax \xypolyrepeat@
\the\toks@={\POS"#3}\
\expandafter\addtotoks@\expandafter{\xypolyNUMSIDES@"#6"#31"#7#8\relax}%
\the\toks@ \restore }

The end & Log

DOCMODE3

\xyendinput

% $Log: xypoly.doc,v $
% Revision 3.11 2011/03/14 20:14:00 krisrose
% Preparing for release 3.8.6.
% Revision 3.10 2010/06/10 18:45:50 krisrose
% Reference to GPL by URL.
% Revision 3.9 2010/05/06 17:46:30 krisrose
% Ross Moore’s e-mail address updated.
% Many obsolete files degraded to Historic.
% Revision 3.8 2010/04/16 06:06:52 krisrose
% Preparing for a new release...
% Revision 3.7 1999/02/16 15:12:50 krisrose
% Interim release (Y&Y fonts now free).
% Revision 3.4 1997/05/18 01:13:24 ross
% Essential bugfixes.
% Revision 3.3 1996/12/18 09:28:35 ross
% cosmetic changes in documentation
% Revision 3.1 1995/09/05 20:28:57 ross
% Releasing version 3.1!
% Revision 3.0 1995/07/07 20:13:19 ross
% Major release w/new User’s Guide!
% Revision 2.13 1995/07/04 15:04:51 ross
3.8 Lattice and web feature

Vers. 3.7 by Ross Moore (ross.moore@mq.edu.au)

This feature provides macros to facilitate typesetting of arrangements of points within a 2-dimensional lattice or “web-like” structure.

Header:

```
\%% $Id: xyweb.doc,v 3.7 2011/03/14 20:14:00 krisrose Exp$
\%%
\%% Xy-pic ‘‘Lattice and web’’ feature.
\%% Copyright (c) 1994–1996 Ross Moore <ross.moore@mq.edu.au>
\%%
\%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
\%% See the companion README and INSTALL files for further information.
\%% Copyright (c) 1991–2011 Kristoffer H. Rose <krisrose@tug.org>
\%%
\%% The Xy-pic package is free software; you can redistribute it and/or modify
\%% it under the terms of the GNU General Public License as published by the
\%% Free Software Foundation; either version 2 of the License, or (at your
\%% option) any later version.
\%%
\%% The Xy-pic package is distributed in the hope that it will be useful, but
\%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
\%% or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License
\%% for more details.
\%%
\%% You should have received a copy of the GNU General Public License along
\%% with this package; if not, see http://www.gnu.org/licenses/.
\%%
\texttt{\textbackslash ifx\textbackslash xy\textbackslash loaded\textbackslash undefined \textbackslash input xy \textbackslash fi}
\texttt{\texttt{\textbackslash xyp\textbackslash provide\{web\}\{Lattice and web feature\}\{\texttt{\stripRCS$Revision: 3.7 $}\}\%
\{Ross Moore\}\{ross.moore@mq.edu.au\}\%
\{Mathematics Department, Macquarie University, NSW\textasciitilde2109, Australia\}}}
\message{\texttt{\textbackslash message\{lattices,\}}}
\texttt{\texttt{\textbackslash xynew\{count\}\textbackslash lattice@A}}
\texttt{\texttt{\textbackslash xynew\{count\}\textbackslash lattice@B}}
\texttt{\texttt{\textbackslash xydef\{\textbackslash lattice@A\}\textbackslash lattice@A}}
```

We need two counters and macros to access their values.
Currently the only routines implemented with this feature are some “quick and dirty” macros for dropping objects at the points of an integer lattice. **To Do:** More sophisticated routines will be developed for later versions of XY-pic, as the need arises.

Mathematically speaking, let \( \mathbf{u} \) and \( \mathbf{v} \) be vectors pointing in independent directions within the plane. Then the lattice spanned by \( \mathbf{u} \) and \( \mathbf{v} \) is the infinite set of points \( L \) given by:

\[
L = \{a \mathbf{u} + b \mathbf{v}; \text{ for } a, b \text{ integers}\}.
\]

Within XY-pic the vectors \( \mathbf{u} \) and \( \mathbf{v} \) can be established as the current coordinate basis vectors. The following macros typeset a finite subset of an abstract lattice.

\[
\text{\texttt{xylattice#1#2#3#4}} \quad \text{points in lattice}
\]
\[
\text{\texttt{croplattice#1#2#3#4#5#6#7#8}} \quad \ldots \text{in specific rectangle.}
\]

The parameters \#1 ... \#4 are to be integers \( a_{\min}, a_{\max}, b_{\min} \) and \( b_{\max} \), so that the portion of the lattice to be typeset is that collection of vectors in \( L \) for which \( a_{\min} \leq a \leq a_{\max} \) and \( b_{\min} \leq b \leq b_{\max} \).

In the above code, notice how the basis is first established then the \texttt{xylattice} typeset. Doing this within an \texttt{xybox} allows axes to be sized and placed appropriately. Since lattice points are determined by their (integer) coordinate displacements, they can be revisited to add extra \langle object\rangle s into the overall picture. More generally, the origin for lattice-coordinates is the current \langle pos\rangle \( c \), when the \texttt{xylattice} command is encountered. Easy accessibility is maintained, as seen in the next example.

When the basis vectors \( \mathbf{u} \) and \( \mathbf{v} \) are not perpendicular the collection of points with \( a, b \) in these ranges will fill out a skew parallelogram. Generally it is useful to plot only those points lying within a fixed rectangle. This is the purpose of \texttt{croplattice}, with its extra parameters \#5 ... \#8 determining
the ‘cropping’ rectangle within which lattice points will be typeset. Other points will not be typeset even when \( a \) and \( b \) are within the specified ranges. Explicitly the horizontal range of the cropping rectangle is \( X_{\text{min}} \) to \( X_{\text{max}} \), with \( X_{\text{min}} \) being the \( X \)-coordinate of the vector \( \vec{u} \), where \( \vec{u} \) is a (number) (not necessarily an integer). Similarly \( X_{\text{max}} \) is the \( X \)-coordinate of \( \vec{v} \). The vertical extents are \( Y_{\text{min}} \) and \( Y_{\text{max}} \), given by the \( Y \)-coordinates of \( \vec{v} \) respectively.

\[ \begin{align*} 
\text{The routines } &\text{xylattice and } \text{croplattice simply cycle through the allowable range of values} \\
&\text{for } \latticeA \text{ and } \latticeB, \text{ with } \latticeB \text{ varying within the inner loop.} \\
\end{align*} \]

\[
\begin{align*} 
\text{xylattice@A=1} & \text{ elseif } \text{xylattice@B=-1} \\
\text{xylattice@A=0} & \text{ elseif } \text{xylattice@B=1} \\
\text{\xylattice{#1}{#2}{#3}{#4}} & \text{\innerlatticeloop{#4}} \\
\end{align*} \]
With \texttt{xylattice} getting the picture size correct is simply a matter of passing \texttt{X@min}, \texttt{X@max}, etc. outside of the inner group. However with \texttt{croplattice} these are first set to correspond to the ‘cropping rectangle’ then subsequently merged with their original values.
The \latticebody macro. At each lattice point within the specified range for \(a, b\) (and within the cropping rectangle when \croplattice is used), a macro called \latticebody is expanded. This is meant to be user-definable, so as to be able to adapt to any specific requirement. It has a default expansion given by...

\[
\text{\texttt{def\latticebody{\textbf{\{drop\{\bullet\}}}}.}
\]

The following macros may be useful when specifying what to do at each point of the lattice.

\[
\begin{align*}
\text{\texttt{latticebody}} & \quad \text{expanded at lattice points} \\
\text{\texttt{defaultlatticebody}} & \quad \text{resets to default} \\
\text{\texttt{latticeA}} & \quad a\text{-value of lattice point} \\
\text{\texttt{latticeB}} & \quad b\text{-value of lattice point} \\
\text{\texttt{latticeX}} & \quad X\text{-coord, offset in pts...} \\
\text{\texttt{latticeY}} & \quad Y\text{-coord, ...from lattice origin.}
\end{align*}
\]

As in the examples presented above, the object dropped at the lattice point can be varied according to its location, or omitted altogether.

The default \latticebody macro ...

\[
\begin{align*}
\text{\texttt{\xydef\deflatticebody@{\texttt{\{\def\latticebody{\textbf{\{drop\{\bullet\}}}}}}}} \\
\text{\texttt{\xydef\defaultlatticebody{\deflatticebody@}}}
\end{align*}
\]

In the final example the \latticebody macro performs a calculation to decide which lattice points should be emphasised:

\[
\begin{align*}
\text{\texttt{\def\latticebody{\texttt{\{dimen0=\texttt{latticeX pt}}}}}
\text{\texttt{\{ifdim\dimen0>OOpt divide\dimen0 by 64}}}
\text{\texttt{\{dimen0=\texttt{latticeY dimen0 relax}}}
\text{\texttt{\{ifdim Opt>dimen0 \dimen0=-\dimen0 \fi}}}
\end{align*}
\]
The end & Log

DOCMODE3

\xyendinput

% $Log: xyweb.doc,v $% 
% Revision 3.7 2011/03/14 20:14:00 krisrose
% Preparing for release 3.8.6.
% 
% Revision 3.6 2010/06/10 18:45:50 krisrose
% Reference to GPL by URL.
% 
% Revision 3.5 2010/05/06 17:46:30 krisrose
% Ross Moore's e-mail address updated.
% Many obsolete files degraded to Historic.
% 
% Revision 3.4 2010/04/16 06:06:52 krisrose
% Preparing for a new release...
% 
% Revision 3.3 1996/12/18 09:52:11 ross
% checked in with -k by krisrose at 1996/12/18 14:17:11
% 
% Revision 3.3 1996/12/18 09:52:11 ross
% no changes
% 
% Revision 3.1 1995/09/05 20:36:33 ross
% Release!
% 
% Revision 3.0 1995/07/07 20:13:19 ross
% Major release w/new User's Guide!
% 
% Revision 2.13 1995/07/05 10:36:22 ross
% Ready for v3 release?
% 
% NEW for version 3.
3.9 Circle, Ellipse, Arc feature

Vers. 3.8 by Ross Moore (ross.moore@mq.edu.au)

This feature provides a means to specify circles of arbitrary radius, drawn with a variety of line styles. Similarly ellipses may be specified, having arbitrary major/minor axes aligned in any direction. A circular arc joining two points can be constructed with specified tangent direction at one end.

All the curves described here—circles, ellipses and sectors of these—are constructed using the curves from the \texttt{xycurve} extension. As such any comments given there concerning memory requirements are equally valid here, perhaps even more so. Use of the \texttt{xy} PostScript back-end is highly recommended.

The arcs are not truly circular or elliptical, but are approximations given by cubic Bézier segments. Hence the \texttt{xycurve} feature must be loaded.

The basic \texttt{\textbackslash arc} defined here is \texttt{\textbackslash arc}. So far it is only implemented as a \texttt{\textbackslash decor}ation, called \texttt{\textbackslash ellipse}.

\begin{verbatim}
\%\% $Id: xyarc.doc,v 3.8 2011/03/14 20:14:00 krisrose Exp $
\%
\% Xy-pic "Circles, Ellipses and Arcs" feature.
\% Copyright (c) 1995-1997 Ross Moore <ross.moore@mq.edu.au>
\%
\% This file is part of the Xy-pic package for graphs and diagrams in TeX.
\% See the companion README and INSTALL files for further information.
\% Copyright (c) 1991-2011 Kristoffer H. Rose <krisrose@tug.org>
\%
\% The XY-pic package is free software; you can redistribute it and/or modify
\% it under the terms of the GNU General Public License as published by the
\% Free Software Foundation; either version 2 of the License, or (at your
\% option) any later version.
\%
\% The XY-pic package is distributed in the hope that it will be useful, but
\% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
\% or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License
\% for more details.
\%
\% You should have received a copy of the GNU General Public License along
\% with this package; if not, see http://www.gnu.org/licenses/.
\%
\ifx\xyloaded\undefined \input xy \fi
\xyprovide{arc}{Circle, Ellipse, Arc feature}{\stripRCS$Revision: 3.8 $}\%
{Ross Moore}{ross.moore@mq.edu.au}\%
{Mathematics Department, Macquarie University, NSW~2109, Australia}
\%
\xyrequire{curve}
\xyatcodes
\xydef@\ellipse#1#{\xy@{ELLIPSE}{\hbox{\begin{save@} \let\endarc@=\endellipse@@ \xyFN@\preARC@ #1@}}}
\endverbatim
3.9. CIRCLE, ELLIPSE, ARC FEATURE

We start by recording the curve ⟨style⟩ and initialising some methods and ⟨pos⟩ names, to be able to easily access important ⟨pos⟩ and ⟨direction⟩ information.

When the ... in \arc...{⟨style⟩} is particularly simple, e.g. ⟨empty⟩ or ^ or _, then we can branch immediately to \fullCircle@. Otherwise more parsing is required.

When further parsing is required, using \getarcparams@, then an @ token is used to mark the end of the relevant tokens.

Before any parsing at all the current p and c are recorded as named ⟨pos⟩itons 0@p and 0@c, as well as a point 0@m along the line in the current direction from p. A method \origDirection@ is defined to artificially re-impose this direction independent of the p and c that may be current at the time. \startDirection@ is initialised to this method, though parsing may change its value. Other methods are initialised, in particular the default orientation; their values can be expected to change with subsequent parsing.
3.9.1 Full Circles

The \texttt{xypic} feature allows a much wider range of possibilities for typesetting circles than is available with \texttt{\textcircled{}}. Firstly the radius is no longer restricted to a finite collection of sizes. Secondly fancy line (curve) styles are available, as with curved arrows. Finally there are a variety of ways of specifying the desired radius, relative to other parts of the picture being built, as in the following example.

\begin{verbatim}
\texttt{\xy 0;/r5pc/:/dir{=}="p",**!DR{p};
 p+(.5,-.5)*/dir{=}="c","+++!L{c}**/dir{-}
 ,\ellipse<>{:},\ellipse(.5){}
 ,0;(.5,.5)::,"p";"c",\ellipse(.5){.}}
 ,\ellipse<5pt>{=}\endxy}
\end{verbatim}

The following give circles centred at \(c\).

\begin{verbatim}
\ellipse>(⟨style⟩)\quad \text{radius} = \text{dist}(p, c)
\ellipse<⟨dimen⟩>{..}\quad \text{radius is the} \langle \text{dimen} \rangle
\ellipse(⟨num⟩){⟨style⟩}\quad \text{unit circle scaled} \langle \text{num} \rangle,
\quad \text{in the current basis.}
\end{verbatim}

Note that if the current basis is not square then the latter variant, namely \(\texttt{\ellipse(⟨num⟩)}\), will typeset an ellipse rather than a circle. On the other hand the first two variants always specify true circles. In the 2nd case, i.e. when \(⟨\text{dimen}⟩\) is \(⟨\text{empty}⟩\), the size of the object at \(p\) is taken into account when drawing the circle; if this is not desired then kill the size using a null object, e.g. \(;*{};\).

Currently the \texttt{\ellipse} macro works only as a \(⟨\text{decor}⟩\). In future versions there will be an \(⟨\text{object}⟩\) called \texttt{\textbullet{}} having elliptical shape, via \texttt{\textcircled{}} with possibly unequal extents. Also it will be possible to \texttt{\connect\textbullet{}} which will set the current connection so that any place on the full ellipse, not just the visible sector, will be accessible using an extension to the usual \(⟨\text{place}⟩\) mechanism.

\textbf{To Do:} make this be!!

The simplest cases of full circles go straight to \texttt{\textcircled{}}.

\begin{verbatim}
\xydef@{fullCircle@}{\xy@{fullCIRC}{\let\endDirection@=\relax
 \let\doCircleArc@=\doMergeEndArc@@}\circleArc@ }
\endverbatim

If the first non-space token following \texttt{\textbullet{}} is \(<\) then the current \(c\) is taken to be the centre of the circle or ellipse, or sector thereof. The first parsing is done to interpret information concerning the radius, (or axes of an ellipse) along with an orientation. This can be specified either as absolute
dimensions, enclosed in <...>, or relative to the current basis by (...), or based on the line PC using < or <,(dimen)> or (,(num)).

```
\xydef@\getarcparams@{%
  \ifx\space@\next \expandafter\DN@\space{\xyFN@\getarcparams@}\%gobble spaces
  \else\addLT@\next \xy@{<}{\%}
    \addGT@{\addLT@{\DN@{\DN@##1}{\%}{\%}{\%}}}{\%}{\%}{\%}
  \%\else\ifx\next\xy@{()}{\%}
    \addLQ@\next \xy@{}{\%}
  \else\DN@{\%}
    \fi\fi\next@ }
\xydef@\checkRadOpen@{%
  \ifx\next\xy@{()}{\%}
    \xy@{\%}{\%}
  \fi\next@ }
\xydef@\fixedRadiusArc@{%
  \xy@{\%}{\%}
\xydef@\fixedRadiusArc@i{%
  \ifx\next\xy@{()}{\%}
    \xy@{\%}{\%}
  \else\xy@{}{\%}
    \xy@{}{\%}
  \fi\next@ }
\xydef@\arcScale@@{%
  \xy@{\%}{\%}
\xydef@\arcScale@dim#1{%
  \xy@{\%}{\%}
\xydef@\arcRadius@{%
  \xy@{}{\%}
\xydef@\ellipseRatio@{%
```

The centre is now stored as "000", since the value of "00c" will be changed to agree with "00p". This is so that start/finish directions will agree, ensuring all four quadrants be drawn. The following method is actually executed after all the radius/axes information has been parsed.

```
\xydef@\arcScale@@{%
  \xy@{\%}{\%}
\xydef@\arcScale@dim#1{%
  \xy@{\%}{\%}
```

When a full circle or ellipse is required then there are three methods which must be invoked when establishing the local basis change.

`arcScale@` Initially this has value \`arcScale@@` which is based on \$\overrightarrow{\text{PC}}\$. It is changed only by (⟨num⟩) and (⟨num⟩,⟨num⟩); see the code for \`splitRadius@` below. When executed, this method will establish the required basis, at least partially.

```
\xydef@\arcScale@@{%
  \xy@{\%}{\%}
\xydef@\arcScale@dim#1{%
  \xy@{\%}{\%}
```

`arcRadius@` Initially (empty), this changes with ⟨(dimen),⟨dimen⟩⟩ and ⟨(dimen)⟩ to \`arcScale@dim` which gets the radius by a ⟨slide⟩ of given ⟨dimen⟩ along the line \$\overrightarrow{\text{PC}}\$.

```
\xydef@\arcScale@dim#{%xy@{\%}{\%}
  \xy@{}{\%}
\xydef@\arcScale@dim#{%xy@{}{\%}
  \xy@{}{\%}
```

`ellipseRatio@` Initially (empty), this changes with ⟨(dimen),⟨dimen⟩⟩ to \`ellipseScale@dim` which establishes the (\$X@basey;Y@basey\) by a ⟨slide⟩ perpendicular to the line \$\overrightarrow{\text{PC}}\$, or to \`ellipseScale@` with
(, (num)) to simply set the perpendicular basis direction using (num) as a coordinate.

\[\texttt{\texttt{xdef\@\\ellipseScale@dim#1\{xy@\{ellSCALEdim\}\{\setupDirection@\cfromp@}
}\enter@\cplusthec@ \aboveDirection@\xydashl0 \vfromslide@\{#1\}\leave@}
\setbase@\{X@c,Y@c\}\}}\]

\[\texttt{\texttt{xdef\@\ellipseScale@dim#1\{xy@\{ellSCALEdim\}\%}
\{vfromcartesian@000,\#1\setbase@0X@cY@c\}}\]

The methods \texttt{\splitRadius@dim} and \texttt{\splitRadius@} complete the interpretation of the radius/axes specification. They must determine whether one or two pieces of information have been explicitly provided, and act accordingly.

\[\texttt{\texttt{xdef\@\splitRadius@dim#1,#2,#3\{%
\DN@\{#1\}\ifx\next@empty\dimen@=z@else\dimen@=\#1\relax\%\spline\trace\n\expandafter\def\expandafter\arcRadius@\expandafter{\arcScale@dim{#1}}\fi
\DN@\{#2\}\ifx\next@empty\dimen@ii=\#2\else\dimen@ii=#2\relax \expandafter
\expandafter\def\expandafter\ellipseRatio@\expandafter{\ellipseScale@dim{#2}}\fi
\fixedRadiusArc@ }\]

\[\texttt{\texttt{xdef\@\splitRadius@#1,#2,#3\{%
\DN@\{#1\}\ifx\next@empty\dimen@=z@else\dimen@=\#1p@\relax\fi
\DN@\{#2\}\ifx\next@empty\dimen@ii=\#2\else\dimen@ii=#2p@\relax\fi
\ifdim\dimen@ii=z@\ifdim\dimen@ii=z@\expandafter
\expandafter\fixedRadiusArc@ \expandafter{\expandafter{\expandafter{\ellipseScale@dim\{#2\}}\fi
\else
\ifdim\dimen@ii=z@\expandafter
\expandafter\fixedRadiusArc@ \expandafter{\expandafter{\expandafter{\ellipseScale@dim\{#2\}}\fi
\else
\expandafter
\expandafter\fixedRadiusArc@ \expandafter{\expandafter{\expandafter{\ellipseScale@dim\{#2\}}\fi
\fi
\fi \fixedRadiusArc@ }\]

\[\texttt{\texttt{xdef\@\baseOrient@\{xy@\{arcSCALEdim\}\{\save@\cfromid@000\}}%
\texttt{\texttt{\X@c=\Y@c}}\]

\[\texttt{\texttt{\X@c=\#1\X@c \Y@c=\#1\Y@c}}\]

\[\texttt{\texttt{\X@c=\#2\X@c \Y@c=\#2\Y@c}}\]

\[\texttt{\texttt{\advance\X@c\X@c \advance\Y@c\Y@c \czerobase}}\]

\[\texttt{\texttt{idfromc@00p\idfromc@000}}\%\]

\[\texttt{\else\expandafter
\expandafter\fixedRadiusArc@ \expandafter{\expandafter{\expandafter{\ellipseScale@dim\{#2\}}\fi
\fi \fixedRadiusArc@ }\]

\[\texttt{\texttt{xdef\@\baseOrient@\{\DN@\_\}\ifx\next@arcOrient@@}
\texttt{\X@c=\#1\X@c \Y@c=\#1\Y@c}}\]

\[\texttt{\X@c=\#2\X@c \Y@c=\#2\Y@c}}\]

\[\texttt{\advance\X@c\X@c \advance\Y@c\Y@c \czerobase}}\]

\[\texttt{\idfromc@00p\idfromc@000}}\%\]

\[\texttt{\fi \fixedRadiusArc@ }\]

\[\texttt{\texttt{xdef\@\baseOrient@\{}\DN@\_\}\ifx\next@arcOrient@@}
\texttt{\X@c=\#1\X@c \Y@c=\#1\Y@c}}\]

\[\texttt{\X@c=\#2\X@c \Y@c=\#2\Y@c}}\]

\[\texttt{\advance\X@c\X@c \advance\Y@c\Y@c \czerobase}}\]

\[\texttt{\idfromc@00p\idfromc@000}}\%\]

\[\texttt{\fi \message{ellipses,}}\]
3.9.2 Ellipses

There are several ways to specify an ellipse, apart from the method illustrated above in which the basis must be changed from square. Basically we must specify the lengths of the major and minor axes. Also it is necessary to specify an alignment for one axis.

In the following, the ellipse is centred on $c$ and one axis is aligned along the line $pc$, except with the final variant where it aligns with the current basis. When used $(\text{num})$ is treated as a scale factor, multiplying an appropriate length.

\[
\begin{align*}
\text{\texttt{\textbackslash ellipse}} & \langle \text{dimen} \rangle, \langle \text{dimen} \rangle \{ .. \} \quad \text{given axes lengths} \\
\text{\texttt{\textbackslash ellipse}} & \langle \text{dimen} \rangle \{ \langle \text{style} \rangle \} \quad \text{one axis is } \overline{pc} \\
\text{\texttt{\textbackslash ellipse}} & (\langle \text{num} \rangle) \{ \langle \text{style} \rangle \} \quad \text{...perp. axis scaled} \\
\text{\texttt{\textbackslash ellipse}} & (\langle \text{num} \rangle, \langle \text{num} \rangle) \{ .. \} \quad \text{scaled axes aligned} \\
& \quad \text{with basis.}
\end{align*}
\]

In the latter variant, if the second $(\text{num})$ is (empty) then this is equivalent to both $(\text{num})$s having the same value, which is in turn equivalent to the final variant for circles.

\[
\text{\texttt{\textbackslash xy}} \ 0;/r5pc/*\textbackslash dir\{\ast\},***!\textbackslash DR\{.5\}\{p\}*
\textbackslash *\textbackslash frm\{-\};\textbackslash p+(.5,-.5)*\textbackslash dir\{\ast\}="c",\n**\textbackslash dir\{-\},**!\textbackslash UL\{c\},"c","c",\n,\{\textbackslash ellipse\{1,.4\}\{\\}\},\{\textbackslash ellipse\{,.75\}\}\{\\}\,
,\{\textbackslash ellipse<15pt,10pt>\{=\}\}
;**\};\{\textbackslash ellipse<,10pt>\{\\}\}\textbackslash endxy
\]

As with full circles, full ellipses require little preparation.

\[
\begin{align*}
\text{\texttt{\textbackslash xydef\{\textbackslash fullEllipse\}\{\texttt{\textbackslash xy@\{\textbackslash fullELLIPSE\}\%\}}} \\
& \{\texttt{\textbackslash def\textbackslash startDirection\{\}\{\texttt{\textbackslash let\textbackslash doCircleArcs\textbackslash =\textbackslash doMergeEndArcs\textbackslash \}}}\% \\
& \texttt{\textbackslash ellipseArcs\textbackslash %} \\
& \texttt{\textbackslash xy@\{\textbackslash doCIRC\textbackslash arcs\}\{\texttt{\textbackslash leave\ \textbackslash doCircleArcs\textbackslash \}}} \texttt{\textbackslash endarc\textbackslash } \}
\end{align*}
\]

3.9.3 Drawing arcs

This method is common to full circles, ellipses and circular arcs. It first executes the methods \texttt{\textbackslash arcScale\textbackslash }, \texttt{\textbackslash arcRadius\textbackslash }, \texttt{\textbackslash ellipseRatio\textbackslash } where necessary, then examines \texttt{\textbackslash arcOrient\textbackslash } to decide if the arc is to be typeset clockwise or counter-clockwise.

\[
\begin{align*}
\text{\texttt{\textbackslash xydef\{\textbackslash ellipseArcs\}\{\}}} \\
& \texttt{\textbackslash arcScale\textbackslash \ \texttt{\textbackslash arcRadius\textbackslash \ \texttt{\textbackslash ellipseRatio\textbackslash \}}} \\
& \texttt{\textbackslash xy@\{\textbackslash arcCONTROLS\}\{\texttt{\textbackslash baseOrient\ \textbackslash arcControls\ \textbackslash bstartPLACE\textbackslash =\textbackslash relax \}}} \}
\end{align*}
\]

Here is where the “control points” are defined, for each of the four Bézier cubic segments, using coordinates in the constructed basis. The endpoints of the segments—though, not necessarily of the arc—are stored as $0p$, $1c$, $2c$, $3c$ and the tangential control points are stored as $1m$, $2m$, ..., $8m$. 

DRAWING THE CIRCLE/ELLIPSE/ARC MEANS TRACING ALONG THE REQUISITE CUBIC SEGMENTS. THE METHOD \texttt{\doCircleArc@@} IS USED FOR FULL CIRCLES/ELLIPSES WHEN COMPLETE SEGMENTS ARE TO BE TYPESET. THE METHODS \texttt{\doMergeStartArc@}, \texttt{\doMergeEndArc@@} AND \texttt{\doMergeBothArc@} ARE USED WHEN A SEGMENT IS NOT COMPLETE. THE WORK BY “MERGING” THE INITIAL/FINAL POINT OF THE CURVED SEGMENT WITH THE APPROPRIATE POINT REQUIRED TO BE AT THE EXTREMITY OF THE ARC. TYPESETTING OF THE CURVE SHOULD NOT TAKE PLACE WITHIN THE REGION COVERED BY THIS MERGED OBJECT.

TO OVERTAKE THE DIFFICULTY OF THE CURVE LEAVING THEN RE-ENTERING THIS REGION, A SECOND MERGE IS PERFORMED TO EXTEND IT SUITABLY. THIS IS THE PURPOSE OF \texttt{\cfromcontrols@}. DUE TO THE “CONVEX-HULL PROPERTY” OF THE Bézier CUBIC SEGMENT, A POINT CAN BE CONSTRUCTED FROM THE END-POINTS AND CONTROL POINTS WHICH CAN BE USED TO EXTEND THE RECTANGLE TO COVER A SUITABLE PORTION OF THE CURVE.
3.9. CIRCLE, ELLIPSE, ARC FEATURE

\connect@\crvs{#5}\relax
\sinit@\sleave@}
\xydef@\doMergeStartArc@#1\#2\{\save@\ifcase#2\relax
\doMergeStart@00p\{1@m\}0m\{1@c\}\{#1\}\or
\doMergeStart@1@c\{3@m\}4m\{2@c\}\{#1\}\relax\or
\doMergeStart@2@c\{5@m\}6m\{3@c\}\{#1\}\relax\or
\doMergeStart@3@c\{7@m\}8m\{0@p\}\{#1\}\relax\fi \leave@}
\xydef@\doMergeStart@@#1#2#3#4#5#6{%
\cfromid@{#1}\ifx#6\relax
\L@c=.1\p@ \R@c=\L@c \U@c=\L@c \D@c=\L@c \Edge@c={\rectangleEdge}\fi
\enter@\cmergethec@\cfromid@0@x\leave@
\enter@\cmergethec@\cfromcontrols@#4\{#3\}#2\{#1\}\leave@
pfromc@
\senter@\cfromid@{#2}\spushc@\cfromid@{#3}\spushc@\cfromid@{#4}\%
\connect@\crvs{#5}\relax\sinit@\sleave@}
\xydef@\doMergeBothArc@#1\#2\{\save@\ifcase#2\relax
\doMergeBoth@00p\{1@m\}2m\{1@c\}\{#1\}\or
\doMergeBoth@1@c\{3@m\}4m\{2@c\}\{#1\}\relax\or
\doMergeBoth@2@c\{5@m\}6m\{3@c\}\{#1\}\relax\or
\doMergeBoth@3@c\{7@m\}8m\{0@p\}\{#1\}\relax\fi \leave@}
\xydef@\doMergeBoth@@#1#2#3#4#5#6{%
\cfromid@{#1}\ifx#6\relax
\L@c=.1\p@ \R@c=\L@c \U@c=\L@c \D@c=\L@c \Edge@c={\rectangleEdge}\fi
\enter@\cmergethec@\cfromid@0@x\leave@
\enter@\cmergethec@\cfromcontrols@#4\{#3\}#2\{#1\}\leave@
pfromc@
\senter@\cfromid@{#2}\spushc@\cfromid@{#3}\spushc@\cfromid@{#4}\%
\enter@\cmergethec@\cfromid@0@c\leave@
\enter@\cmergethec@\cfromcontrols@#1\{#2\}#3\{#4\}\leave@
\connect@\crvs{#5}\relax\sinit@\sleave@}
\xydef@\doMergeEndArc@#1\#2\{\count@@=#2\relax
\ifnum\count@@>\thr@@ \advance\count@@-4\fi \leave\fi
\doMergeEndArc@@#1
\xydef@\doStraightArc@@#1{\xy@{STRAITarc}{\save@\cfromid@0@p\pfromc@\cfromid@0@c\%
\senter@\connect@\crvs{#1}\sinit@\sleave@}
\xydef@\cfromcontrols@#1#2#3#4{\bgroup
\cfromid@{#1}\L@p=\X@c \U@p=\Y@c
\cfromid@{#4}\R@p=\X@c \D@p=\Y@c
\cfromid@{#3}\A@=\X@c \B@=\Y@c \cfromid@{#2}\%
\ifdim\R@p<\L@p \ifdim\A@>\R@p \R@p=\A@
\else \ifdim\A@<\R@p \R@p=\A@
\fi\fi
\fi\fi\fi
\fi\fi
\fi\fi
\fi\fi
\fi\fi
\fi\fi
\fi\fi
\fi\fi
The next method controls which segments are typeset. It is rather primitive and could well be improved; e.g. to be more like \doEllipseSectors@ with \forwardSectors@ and \backwardSectors@.

First it calculates the \startDirection@ as a number, using the current base. Next it calculates \quadDirection@ which records the angle of the $y$-axis in the current base. Depending on the required orientation, read from \arcOrient@, this latter number is adjusted to be minimal greater than \startDirection@ for a counter-clockwise arc, or maximal less than it for a clockwise one. Similarly the value of \endDirection@ is made larger or smaller than \startDirection@, according to orientation, by adding $\pm 8192$ if necessary.

Each required segment is typeset using \doCircleArc@ except that the final segment uses the value of \doMergeEndArc@. When the end-point is the original $c$ then this value is \doMergeEndArc@, so that the arc will stop at the edge of this $\langle \text{pos} \rangle$, otherwise \doCircleArc@ is used.

(The following macro can be shortened by making use of the methods \forwardSectors@ and \backwardSectors@, as in \doEllipseSectors@.)
\edef\quadDirection@{\the\count@}\fi
\ifx\endDirection@@\empty
\ifx\endDirection@\relax
\begin{group}
cfromid@{0c}\setupDirection@
\edef\next@{\egroup \count@@=\the\Direction}\next@ \relax
\ifnum\count@@>\startDirection@@ \advance\count@@-8192 \fi
\edef\endDirection@@{\the\count@@}\
\else
\count@@=\startDirection@@
\advance\count@@ by-8192 \edef\endDirection@@{\the\count@@}\fi
\else
\count@@=\endDirection@@ \relax \ifnum\count@@>\startDirection@@
\advance\count@@ by-8192 \edef\endDirection@@{\the\count@@}\fi
\fi %\backwardSectors@
\ifnum\endDirection@@<\quadDirection@
\DN@{\expandafter\doCircleArc@@\expandafter{\arcSTYLE}{0}}\fi
\else \DN@{\expandafter\doCircleArc@@\expandafter{\arcSTYLE}{0}}\fi
\fi
\next@ \relax
\count@=\quadDirection@ \relax \def\nextii@{^}\DN@{}\fi
\ifx\nextii@\arcOrient@@
\count@@=\startDirection@@ \advance\count@@ by4096 \relax
\ifnum \count@<\endDirection@@ \relax
\ifnum\endDirection@@>\count@@
\DN@{\expandafter\doCircleArc@@\expandafter{\arcSTYLE}{1}}\fi
\else \DN@{\expandafter\doCircleArc@@\expandafter{\arcSTYLE}{1}}\fi
\fi
\else
\count@@=\startDirection@@ \advance\count@@ by-4096 \relax
\ifnum \count@<\endDirection@@ \relax
\ifnum\endDirection@@<\count@@
\DN@{\expandafter\doCircleArc@@\expandafter{\arcSTYLE}{1}}\fi
\else \DN@{\expandafter\doCircleArc@@\expandafter{\arcSTYLE}{1}}\fi
\fi
\next@ \relax
\% \ifx\next@\empty \else \relax
\count@=\startDirection@@ \relax \def\nextii@{^}\DN@{}\fi
\ifx\nextii@\arcOrient@@
\count@@=\startDirection@@ \advance\count@@ by 4096 \relax
\ifnum \count@<\endDirection@@ \relax
\ifnum\endDirection@@<\count@@
\DN@{\expandafter\doCircleArc@@\expandafter{\arcSTYLE}{1}}\fi
\else \DN@{\expandafter\doCircleArc@@\expandafter{\arcSTYLE}{1}}\fi
\fi
\\next@ \relax
\% \ifx\next@\empty \else \relax
\count@=\startDirection@@ \relax \def\nextii@{^}\DN@{}\fi
\ifx\nextii@\arcOrient@@
\count@@=\quadDirection@ \advance\count@@ by 4096 \relax
\ifnum \count@<\endDirection@@ \relax
\ifnum\endDirection@@<\count@@
\DN@{\expandafter\doCircleArc@@\expandafter{\arcSTYLE}{2}}\fi
\else \DN@{\expandafter\doCircleArc@@\expandafter{\arcSTYLE}{2}}\fi
\fi
\else \advance\count@ by-4096 \relax
\count@@=\quadDirection@ \advance\count@@ by-4096 \relax
3.9.4 Circular and Elliptical Arcs

The \texttt{xyarc} feature handles arcs to be specified in two essentially different ways, according to what information is provided by the user. We call these the “radius-unknown/end-points known” and the “radius-known/end-points unknown” cases.

\begin{itemize}
\item \texttt{radius unknown, end-points known}
\end{itemize}

The simplest case, though not necessarily the most common, is that of a circular arc from \( p \) to \( c \), with radius and centre unspecified. To uniquely specify the arc, the tangent direction at \( p \) is taken to be along the current direction, given by \texttt{\textbackslash Direction}, as set by the latest \texttt{\langle connect\rangle}ion. If no connection has been used, then the default \texttt{\langle direction\rangle} is “up”.

\begin{itemize}
\item \texttt{\ellipose\{\langle style\rangle\}} clockwise arc from \( p \) to \( c \)
\item \texttt{\ellipose^{-}\{\langle style\rangle\}} counter-clockwise arc
\item \texttt{\ellipose\{\langle style\rangle\}} also counter-clockwise
\end{itemize}

With this information only, a unique circle can be found whose radius and centre need not be specified in advance. For a unique arc it is sufficient to specify the orientation around the circle.

The exception is when the current direction is from \( p \) to \( c \), in which case no circle exists. Instead a straight line is typeset accompanied by the following message:

\texttt{Xy-pic Warning: straight arc encountered}

The following example shows how, given three points \( o, p \) and \( c \), to continue the line \( \overline{op} \) by a circular arc to \( c \) joining smoothly at \( p \).
Note how the remainder of the circle can be specified separately. The example also shows how to specify an arc which leaves a particular point perpendicular to a specific direction.

Slightly more complicated is when the tangent direction at \( p \) is specified, but different from the current direction; a unique circular arc can still be defined. More complicated is when a specific tangent direction is required also at \( c \). In this case the arc produced is a segment of an ellipse. (If the required tangent at \( p \) points to \( c \) then a straight segment is drawn, as in the circular case described above.)

\[
\text{\texttt{\textbackslash ellipse\{dir\}_p,\{orient\}\{..\} circular}}
\]
\[
\text{\texttt{\textbackslash ellipse\{dir\}_p,\{orient\},\{dir\}_c\{..\} elliptical}}
\]
\[
\text{\texttt{\textbackslash ellipse\{dir\}_p,\{orient\}\{dir\}_c\{..\} elliptical}}
\]
\[
\text{\texttt{\textbackslash ellipse'\{coord\}/\{orient\}\{..\} elliptical}}
\]

In these cases \( \{dir\}_p \) and \( \{dir\}_c \) are \( \langle \text{direction} \rangle \) specifications, as in figure 1.5 and note 1.4l, and \( \{orient\} \) must be either \( ^\cdot \) or \( _\cdot \) for anti-/clockwise respectively, defaulting to \( ^\cdot \) if \( \{empty\} \). Beware that the \((\ast\{pos\}\{decor\}\ast)\) form must be used for this \( \langle \text{direction} \rangle \) variant, as if an object modifier.

The second and third cases in the above table generally give identical results. The second ‘,’ is thus optional, except in two specific situations:

1. \( \{orient\} \) is empty and \( \{dir\}_c \) has \( ^\cdot \) or \( _\cdot \) as the first token;

2. \( \{orient\} \) is \( ^\cdot \) and \( \{dir\}_c \) has \( ^\cdot \) as first token. Without the \( , \) then \( ^\cdot ^\cdot \) would be interpreted by \TeX{} as part of a special ligature for a hexadecimal character code.

If both \( \{orient\} \) and \( \{dir\}_c \) are \( \{empty\} \) then even the first ‘,’ can be omitted.
Note that only the slope of ∠dir_p and ∠dir_c is significant; rotations by 180° being immaterial.

\[ \text{The } \text{variant establishes the } \langle \text{direction} \rangle \text{ parsing to begin with the direction resulting from } \langle \text{dir} \rangle_p \text{ instead of the original direction. If } \langle \text{dir} \rangle_c \text{ is required to be the original direction then use :0. It cannot be } \langle \text{empty} \rangle \text{ since this is interpreted as requiring a circular arc with unspecified tangent at } c; \text{ see the example above. However when } \langle \text{dir} \rangle_p \text{ and } \langle \text{dir} \rangle_c \text{ are parallel there is a whole family of possible ellipses with the specified tangents.}

With no further hint available, a choice is made based on the distance between } p \text{ and } c. \text{ If the required direction is perpendicular to } pc \text{ this choice results in a circular arc. The optional factor in } =\langle \text{num} \rangle \text{ is used to alter this choice; the default (1) is assumed when nothing follows the } =. \text{ This factor is used to "stretch" the ellipse along the specified direction. For a negative } \langle \text{num} \rangle \text{ the orientation reverses.}

\[4\text{Indeed this is always so. The algorithm used for the general case tends toward parallel lines—clearly unsuitable.}\]
The final variant uses the directions from \( p \) and \( c \) to the given \( \langle \text{coord} \rangle \). If \( \langle \text{orient} \rangle \) is \( \langle \text{empty} \rangle \) then the orientation is determined to give the shortest path along the ellipse. Specifying an \( \langle \text{orient} \rangle \) of \(^\wedge\) or \(_\wedge\) will force the orientation, even if this means travelling ‘the long way’ around the ellipse. For example, see next figure.

**Alternative curves**  In some cases the circular or elliptic curve can be replaced by a curve with different shape, having the same tangent directions at the end-points. When a full circle/ellipse is specified then one gets instead a closed curve constructed from 4 spline segments. Other variants use a single segment, 2 or 3 segments, or some portion of all 4 segments. Possibilities are given in the following table.

<table>
<thead>
<tr>
<th>Ellipse Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\ellipse</code></td>
<td>Elliptical, as above</td>
</tr>
<tr>
<td><code>\ellipse</code>_{q}</td>
<td>Parabolic segments</td>
</tr>
<tr>
<td><code>\ellipse</code>_{c}</td>
<td>Cubic segments</td>
</tr>
<tr>
<td><code>\ellipse</code>_{i}</td>
<td>Interpolating cubic</td>
</tr>
<tr>
<td><code>\ellipse</code>_{p}</td>
<td>Cuspidal cubic</td>
</tr>
<tr>
<td><code>\ellipse</code>_{c}(\langle num \rangle)</td>
<td>Cubic segments, with “looseness”</td>
</tr>
</tbody>
</table>

In the latter case the \( \langle \text{num} \rangle \), typically between 0 and 1, controls how soon the curve begins to bend away from the tangent direction. Smaller values give tighter curves — 0 for straight lines — with \(^c\) being the same as \(^c\langle 1 \rangle\) and \(^q\) is \(^c\langle .66667 \rangle\), that is \( \langle \text{num} \rangle = \frac{2}{3} \).

The curve produced by the “interpolating” variant \(^i\) actually passes through the control point "\( x \)", with slope parallel to the line \( \overline{pc} \). Since the tangents at \( p \) and \( c \) point toward "\( x \)" the curvature is quite gentle until near "\( x \)" where the curve bends rapidly, yet smoothly. This is obtained also by using \(^c\langle 1.33333 \rangle\), that is \( \langle \text{num} \rangle = \frac{4}{3} \). Since \( \langle \text{num} \rangle > 1 \) the “convex hull property” does not hold; indeed the curve is entirely outside the convex hull of \( p \), \( c \) and "\( x \)”, apart from those points themselves.

The ‘cuspidal’ variant \(^p\) is equivalent to \(^c\langle 2 \rangle\). It exhibits a cusp. For \( \langle \text{num} \rangle > 2 \) the curve is so “loose” that it exhibits loops. (The author offers no guarantees on the usefulness of such curves for any particular purpose; however they do look nice.)
CHAPTER 3. FEATURES

\textbf{Hint:} When exploring to find the best location for the “control-point” (e.g. the "x" in the above example), then use \texttt{\textbackslash xycompile} as shown, changing the location outside of the compilation. This speeds up the reprocessing with the changed value.

Furthermore, the ellipses are not true ellipses, but are constructed from four Bézier cubic curves, one for each quadrant of the circle or ellipse. To do this it is sufficient to establish the control points for each cubic segment. This is straightforward, using the following “magic number”, given a square basis such that the desired circle is the unit circle, or rectangular basis for which the “unit circle” is the desired ellipse. Thus $\sim e$ is equivalent to $\sim c(.5517847)$.

\begin{verbatim}
\xydef@\arcmagic@@{0.5517847}
\xylet@\arcmagic=\arcmagic@@
\end{verbatim}

This is the magic number, exactly given by $\frac{11}{12}(\sqrt{385} - 13)$, that helps construct the Bézier cubic curve that best approximates a quarter-turn arc of a circle. It does so with remarkable accuracy, differing by at most .5% of the radius at any angle; the average deviation along the whole quadrant being less than .13%.

\begin{verbatim}
\xydef@\tildeARC@#1{\if q#1\DN@{\xy@{quadARC}\quadARC@ \arc@} \else\if c#1\DN@{\xyFN@\cubicARC@i}\else\if e#1\DN@{\xy@{ellipticARC}\ellipticARC@ \arc@} \else\if i#1\DN@{\xy@{interpARC}\interpARC@ \arc@} \else\xywarning@{unknown arc type: #1 } \DN@{\xy@{ellipticARC}\ellipticARC@ \arc@} \fi\fi\fi\fi\fi \next@ }
\xydef@\cubicARC@i{\ifx\next\space@ \expandafter\DN@\space{\xy@{cubicARC(1)}{\cubicARC@{1}}\arc@} \else\xy@{cubicARC(##1)}{\cubicARC@{##1}}\arc@ \fi \next@ }
\xydef@\ellipticARC@{\let\arcControls@=\cubicARCcontrols@@ \edef\arcparam@{\arcmagic@@}}
\xydef@\cubicARC@#1{\let\arcControls@=\cubicARCcontrols@@ \dimen@=#1\p@ \edef\arcparam@{\expandafter\removePT@\the\dimen@}}
\xydef@\quadARC@{\let\arcControls@=\cubicARCcontrols@@ \edef\arcparam@{.66667}}
\xydef@\cuspidalARC@{\let\arcControls@=\cubicARCcontrols@@ \edef\arcparam@{2}}
\xydef@\interpARC@{\let\arcControls@=\cubicARCcontrols@@ \edef\arcparam@{1.33333}}
\end{verbatim}

\textbf{Avoiding overflows} If $\langle \text{dir}\rangle_p$ and $\langle \text{dir}\rangle_c$ are intended to be equal then the method of the previous paragraph should be used. However it may happen that “nearly parallel” directions may be specified, perhaps by accident. There is then the possibility of “numerical overflow” or a “division by zero”
error. The latter may be accompanied by a warning message:

Xy-pic Warning: division by 0 in \intersect@, replaced by 50

This indicates that the number 50 has been used as the result of a division by zero. In many contexts this will produce an acceptable result. However it may lead to an “overflow” in other situations, or to drawing beyond the normal page boundary. This can be controlled using a ⟨decor⟩ of type \zeroDivideLimit{(num)}, prior to specifying the \ellipse. The value 50 will be replaced by ⟨num⟩ whenever a “division by zero” would otherwise be encountered in an intersection calculation.

\xydef@ARCturn@\{\xy@\bgroup\afterCOORD{%
    \xy@{\edef\next@{\egroup \A@=\the\X@c \B@=\the\Y@c}\next@}\ARCturn@}%
\xydef@ARCturn@\#1\{\DN@{\#1}\DNii@{-}%)%
\else\DNii@{-}\nextii@ {\xy@{-}{\def\arcOrient@@{-}}}%
\else\xy@{}{\def\arcOrient@@{}}\fi\fi
\xy@\ARCturn@@ \processDirections@@ }
\xydef@ARCturn@@{%
    \bgroup\cfromid@{0@p}\X@p=\X@c \Y@p=\Y@c \X@c=\A@ \Y@c=\B@%
\setupDirection@ \edef\next@{\egroup\Direction}\next@\relax
\edef\startDirection@@{\the\Direction}%
\bgroup\cfromid@{0@c}\X@p=\A@ \Y@p=\B@%
\setupDirection@ \edef\next@{\egroup\Direction}\next@\relax
\edef\endDirection@@{\the\Direction}%
\ifx\arcOrient@@\empty
\bgroup\cfromid@{0@p}\X@p=\X@c \Y@p=\Y@c \cfromid@{0@c}%
\setupDirection@ \edef\next@{\egroup\Direction}\next@\relax
\count@=-\count@@ \advance\count@=\the\Direction\relax \count@=-\count@
\ifnum\count@>4096 \def\arcOrient@@{-}%
\else\ifnum\count@<-4096 \def\arcOrient@@{^-}%
\else\ifnum\count@>\z@ \def\arcOrient@@{^-}%
\else\def\arcOrient@@{-}%
\fi
\fi
\fi
\xydef@\circleArc@{%
    \xy@{\cfromid@{0@p}\pfromc@\cfromp@\startDirection@}
\arcCenter@\idfromc@{0@0}%
\def\startDirection@{\bgroup
    \cfromid@{0@0}\pfromc@\cfromid@{0@p}\setupDirection@%
\edef\next@{\egroup \Direction=\the\Direction}\next@\relax
\relax\imposeDirection@}}%
\ifx\endDirection@@\empty
\bgroup\cfromid@{0@p}\X@p=\X@c \Y@p=\Y@c \cfromid@{0@c}%
\setupDirection@ \edef\next@{\egroup\Direction}\next@\relax
\count@=-\count@@ \advance\count@=\the\Direction\relax \count@=-\count@
\ifnum\count@>4096 \def\arcOrient@@{-}%
\else\ifnum\count@<-4096 \def\arcOrient@@{^-}%
\else\ifnum\count@>\z@ \def\arcOrient@@{^-}%
\else\def\arcOrient@@{-}%
\fi
\fi
\fi
\xydef@\circarcArc@{%
    \xy@{\cfromid@{0@p}\pfromc@\cfromp@\startDirection@}
\arcCenter@\idfromc@{0@0}%
\def\startDirection@{\bgroup
    \cfromid@{0@0}\pfromc@\cfromid@{0@p}\setupDirection@%
\edef\next@{\egroup \Direction=\the\Direction}\next@\relax
\relax\imposeDirection@}}%
\ifx\endDirection@@\empty
\bgroup\cfromid@{0@p}\X@p=\X@c \Y@p=\Y@c \cfromid@{0@c}%
\setupDirection@ \edef\next@{\egroup\Direction}\next@\relax
\count@=-\count@@ \advance\count@=\the\Direction\relax \count@=-\count@
\ifnum\count@>4096 \def\arcOrient@@{-}%
\else\ifnum\count@<-4096 \def\arcOrient@@{^-}%
\else\ifnum\count@>\z@ \def\arcOrient@@{^-}%
\else\def\arcOrient@@{-}%
\fi
\fi
\fi
\iffalse\arcRadius@\empty \iffalse\ellipseRatio@\empty \else
\DN@{\arcScale@}\DN@{\arcScale}\next@ \else
\xy@{\bgroup\cfromid@{0@0}\pfromc@\cfromid@{0@c}\setupDirection@%
\edef\next@{\egroup \Direction=\the\Direction}\next@\relax
\def\endDirection@@{\the\Direction}\fi\fi
\ellipseArc@ \xy@{\leave@ \doCircleArcs@}\endarc@ }
\ellipseArc@ \xy@{\doCircleArcs@}\endarc@ 
The following methods calculate the centre of a circle using the perpendicular to the required angle at $p$ and the perpendicular bisector of the line $pc$.

The language for these is a combination of most of that used above, but the interpretation of the ⟨direction⟩s is different...

```
\ellipse<⟨radius⟩>⟨dir⟩₁,⟨orient⟩,⟨dir⟩₂{..}
\ellipse<⟨radius⟩>⟨dir⟩₁,⟨orient⟩,=⟨dir⟩₂{..}
```

where ⟨radius⟩ is one of the forms used above to describe a circle or ellipse. Not all of the ellipse will be typeset—only that arc starting with ⟨dir⟩₁ as tangent vector, tracing via ⟨orient⟩ until the tangent points in direction ⟨dir⟩₂. This effectively extends the notation used with \cir in 1.6.2. Note that rotating a given ⟨dir⟩ᵢ by $180^\circ$ specifies a different arc on the same ellipse/circle. Reversing the ⟨orient⟩ no longer gives the complementary arc, but this complement rotated $180^\circ$.
3.9. CIRCLE, ELLIPSE, ARC FEATURE

\{\text{ellipse}(., .75), ^, _{-}\}\endxy
3.9. **CIRCLE, ELLIPSE, ARC FEATURE**

\[ d\theta = -\cos\text{Direction}\ Y@xbase \ \text{advance}\ d\theta \sin\text{Direction}\ X@xbase \]
\[ \text{setupDirection}@i \]
\[ d\theta X = -\cos\text{Direction}\ X@ybase \ \text{advance}\ d\theta X \sin\text{Direction}\ X@xbase \]
\[ d\theta Y = -\cos\text{Direction}\ Y@ybase \ \text{advance}\ d\theta Y \sin\text{Direction}\ Y@xbase \]
\[ \text{enter}\ cplusthec@0@c = d\theta X \ Y@c = d\theta Y \leave@ \]
\[ \text{czeroEdge}@i \]
\[ \text{setupDirection}@i \]
\[ \text{edef} \text{next}@i{\{\text{egroup}\ cfrothec@ \text{count}@ = \text{the}\text{Direction}\} \text{next}@i} \]
\[ \text{edef} \text{endDirection}@i{\{\text{the}\text{count}@ \text{idfromc}@0@c}\} \%
\]
\[ \text{count}@ = \text{startDirection}@i \ \text{relax} \ \text{count}@ = \text{endDirection}@i \ \text{relax} \]
\[ \text{DN}@{^}`\text{ifx}\text{arcOrient}@i{\text{next}@i} \]
\[ \text{ifnum}\text{count}@ > \text{count}@ \ \text{else}\text{advance}\text{count}@ 8192 \relax \text{fi} \]
\[ \text{else} \]
\[ \text{ifnum}\text{count}@ < \text{count}@ \ \text{else}\text{advance}\text{count}@ -8192 \ \text{relax}\text{fi} \]
\[ \text{edef} \text{startDirection}@i{\{\text{the}\text{count}@}\} \%
\[ \text{edef} \text{endDirection}@i{\{\text{the}\text{count}@}\} \%
\]
\[ \text{xydef}@{\text{doCircleSectors}@{\%} \}
\[ \let\text{doCircleArc}@{\text{doMergeEndArc}@i \ \leave@ \ \leave@} \]
\[ \text{cfrotheid}@000\text{pfro}\text{cfrotheid}@00p \text{setupDirection}@i \]
\[ \text{DN}@{^}`\text{ifx}\text{arcOrient}@i{\text{next}@i} \]
\[ \text{ifnum}\text{count}@ > \text{startDirection}@i \ \text{advance}\text{count}@ 8192 \relax \text{fi} \]
\[ \text{count}@ = \text{count}@ \ \text{advance}\text{count}@ 8192 \ \text{relax} \]
\[ \text{ifnum}\text{count}@ < \text{endDirection}@i \ \text{advance}\text{count}@ 8192 \relax \text{fi} \]
\[ \text{edef} \text{preDirection}@i{\{\text{the}\text{count}@}\} \text{edef} \text{postDirection}@i{\{\text{the}\text{count}@\}} \%
\]
\[ \text{cfrothide}@10c \text{setupDirection}@i \ \text{count}@ = \text{Direction} \]
\[ \text{def} \text{nextii}@{^}`\text{ifx}\text{nextii}@\text{arcOrient}@i \]
\[ \text{DN}@{^}`\text{forwardSectors}@{\text{doEllipseSectors}@i} \%
\[ \text{ifnum}\text{count}@ < \text{preDirection}@i \ \text{advance}\text{count}@ 8192 \relax \text{fi} \]
\[ \text{else} \ \text{count}@ = \text{count}@ \ \text{advance}\text{count}@ 8192 \relax \text{fi} \]
\[ \text{edef} \text{quadDirection}@i{\{\text{the}\text{count}@}\} \%
\]
\[ \text{next}@i} \]
\[ \text{xydef}@{\text{forwardSectors}@{\%} \}
\[ \text{ifnum}\text{startDirection}@i < \text{quadDirection}@i \ \text{def} \text{startSector}@0 \%
\[ \text{else}\text{count}@ < \text{preDirection}@i \ \text{advance}\text{count}@ 4096 \relax \text{fi} \]
\[ \text{ifnum}\text{startDirection}@i < \text{count}@ \ \text{def} \text{startSector}@1 \%
\[ \text{else}\text{count}@ = \text{quadDirection}@i \ \text{advance}\text{count}@ 4096 \relax \text{fi} \]
\[ \text{ifnum}\text{startDirection}@i < \text{count}@ \ \text{def} \text{startSector}@2 \%
CHAPTER 3. FEATURES

```plaintext
\else \def\startSector@@{3}\fi\fi\fi

% 
\ifnum\quadDirection@ <\endDirection@@ \relax
\count@=\preDirection@@ \advance\count@ 4096 \relax
\ifnum\count@<\endDirection@@ \relax
\count@=\quadDirection@ \advance\count@ 4096 \relax
\fi
\ifnum\count@<\endDirection@@ \relax
\count@=\preDirection@@ \advance\count@ 8192 \relax
\fi
\ifnum\count@<\endDirection@@ \relax
\count@=\quadDirection@ \advance\count@ 8192 \relax
\fi
\ifnum\count@<\endDirection@@ \relax
\count@=\preDirection@@ \advance\count@ 12288 \relax
\fi
\ifnum\count@<\endDirection@@ \relax
\count@=\quadDirection@ \advance\count@ 12288 \relax
\fi
\ifnum\count@<\endDirection@@ \def\endSector@@{7}%
\else \def\endSector@@{6}\fi
\else \def\endSector@@{5}\fi
\else \def\endSector@@{4}\fi
\else \def\endSector@@{3}\fi
\else \def\endSector@@{2}\fi
\else \def\endSector@@{1}\fi
\else \def\endSector@@{0}\fi }
\xydef\backwardSectors@@{% 
\ifnum \startDirection@@>\quadDirection@ \def\startSector@@{0}\%
\else\count@=\preDirection@@ \advance\count@-4096 \relax
\ifnum \count@<\endDirection@@ \relax
\count@=\quadDirection@ \advance\count@-4096 \relax
\fi
\ifnum \count@<\endDirection@@ \relax
\count@=\preDirection@@ \advance\count@-8192 \relax
\fi
\ifnum \count@<\endDirection@@ \relax
\count@=\quadDirection@ \advance\count@-8192 \relax
\fi
\ifnum \count@<\endDirection@@ \def\endSector@@{7}%
\else \def\endSector@@{6}\fi
\else \def\endSector@@{5}\fi
\else \def\endSector@@{4}\fi
\else \def\endSector@@{3}\fi
\else \def\endSector@@{2}\fi
\else \def\endSector@@{1}\fi
```

3.9. CIRCLE, ELLIPSE, ARC FEATURE

```
\else \def\endSector@@{0}\fi }
\xydef\doEllipseSectors{%
\ifx\startSector@@\endSector@@
 \DN\expandafter\doMergeBothArc\expandafter{\arcSTYLE}{\endSector@@}\
\else
 \DN\expandafter\doMergeStartArc\expandafter{\arcSTYLE}{\startSector@@}\
 \edef\countArcs@@{\startSector@@}\
 \loop \count@=\countArcs@@ \relax \advance\count\@one
 \ifnum \count@ < \endSector@@ \edef\countArcs@@{\the\count@}\
 \bgroup\expandafter\doCircleArc@@\expandafter{\arcSTYLE}{\countArcs@@}\
 \egroup
 \DN\expandafter\doMergeEndArc\expandafter{\arcSTYLE}{\endSector@@}\
\fi \next@ }
```

parsing of directions  When the starting direction is along the line $\mathbf{pc}$ then a straight segment is typeset, ignoring any requested ending direction.

```
\xydef\checkstartDirection{%
\bgroup\no@@ \edef\next@{\egroup\count@@=\the\Direction}\next@\relax
 \ifnum\count@@=\startDirection@@ \relax \DN\straightArc\relax
 \else \advance\count@@-\startDirection@@ \relax
 \ifnum\count@@<0 \fi 4096 \relax \DN\straightArc\relax
 \else \DN\processDirections\fi\fi\fi \next@ }
\xydef\straightArc{\xywarning{straight arc encountered}\
\xy@@{\let\bstartPLACE@=}\expandafter\doStraightArc@@\expandafter{\arcSTYLE}}
```

```
\xydef\skewEllipse{% \xy@@{\let\bstartPLACE@=}\expandafter\skewEllipse@\expandafter{\arcSTYLE}}
```

```
\xydef\skewCircle{% \let\arcScale@=\relax \def\arcRadius@{} \def\ellipseRatio@{}}
```

```
\xydef\doskewEllipse{%
\def\afterARCextents{%\xy@@{aftARCext}{\let\doCircleArc@=\doMergeEndArc@@}\
 \ellipseArc@ \xy@@{doCircARCs}{\doCircleArcs@}\endarc@ }\relax
 \let\arcScale@=\relax \def\arcRadius@{} \def\ellipseRatio@{}}
```

```
\xydef\skewCircle{% \xy@@{\let\bstartPLACE@=}\expandafter\skewCircle@\expandafter{\arcSTYLE}}
```

```
\xydef\straightArc{\xywarning{straight arc encountered}\
\xy@@{\let\bstartPLACE@=}\expandafter\doStraightArc@@\expandafter{\arcSTYLE}}
```

```
KNOTS AND LINKS FEATURE

The end & Log

3.10 Knots and Links feature

Vers. 3.9 by Ross Moore (ross.moore@mq.edu.au)

This feature provides a language for specifying knots, links and general arrangements of crossing strings.

Header:

```$Id: xynot.doc,v 3.9 2011/03/14 20:14:00 krisrose Exp $```
To Do: Document this feature!

This knot feature is really a ‘construction kit’, providing pieces which may be placed appropriately
to form knots and links. The types of pieces provided are of two kinds: the “crossings”, representing
one string crossing over or under another; and “joins” which are used to connect what would otherwise
be loose ends. Several types of each are provided, along with a simple way of specifying where to place
arrowheads and labels.

All the pieces ultimately use curves from the curve extension, usually indirectly via the arrow
feature. As such, processing can be memory-intensive and may seem rather slow. All the warnings
and advice given elsewhere on techniques to handle pages and individual diagrams with many curves
are especially applicable when using this feature.

Most constructions use \ar so make sure this feature is loaded.
Crossings
A “crossing” is intended to represent two strings passing close by, but not meeting. The macros provided specify typesetting within a square cell of coordinate values; using a non-square basis alters this shape, but see also note 3.10c below, for the technique that was used in the “cinquefoil” example above.

Notes
3.10a. Several families of crossing are provided. Those having names as \v... and \h... are designed to stack respectively vertically and horizontally. More precisely the current (pos) starts at the top-left corner and finishes at either the bottom-left or top-right. Say that a crossing is either a ‘vertical crossing’ or ‘horizontal crossing’ respectively.

This certainly applies to the \..cross.. and \..twist.. families, see figure 3.8 in which the strings enter and leave the square all with vertical tangents or all with horizontal tangents. Indeed all crossings are either vertical or horizontal, with the final letter indicating which for the \xover.. families.

Furthermore there is a natural orientation for each crossing, as well as along each strand. This corresponds to the order in which ink is applied to the printed page, following the natural parametrization of each strand as a curved connection or arrow. This orientation determines whether a crossing is ‘over’ (mathematically, positive or right-handed) or ‘under’ (mathematically, negative or left-handed). It is used in determining the location of labels and the direction of arrowheads placed along the strings. Note that \..cross.. and \..twist.. crossings may set the same curves, but with different orientation and label-positioning.

Figure 3.8 displays the orientation on all the crossings, grouping them into subfamilies consisting of right-handed, left-handed and non-crossings. Also indicated are the default positions for labels and arrow-tips; each piece uses the same code for tips and labels, e.g. \vover<>|>>><{x}|{y}>{z}.

The \x... crossings do not stack easily since their tangents are at 45° to the coordinate axes. It is the last letter in the name which denotes whether the particular crossing is vertical or horizontal. On the other hand \vover, \vunder etc. stack vertically on top of a \vcross, \vtwist etc.; similarly \hover stacks at the left of \hcross, \htwist etc.

$$\xy 0;/r1pc/,:\ \xykparsecross@{\vunder\vtwist\vtwist\vunder-}\endxy
\xy 0;/r1pc/:+(0,-1.5),\ \xykparsecross@{\hover\hcross\hcross\hover-}\endxy$$

Parsing
\xydef@\xykparsecross@{%
\def\xykSCALE@@{}\edef\xyknotPLACE{\xykmidPLACE@}%
\let\xykparser@=\xykparsecross@@
\def\xykdefaultbreak@{\let\xykbreak@=\xykforetemp@}
these macros are common to most crossings.

\xydef\xykcross{#1}{\expandafter\addtotoks\expandafter{{}\knotTIPS}}
\xydef\xykoverstring{\addtotoks{}}\xyknottips
\xydef\xykunderstring{\xykprebreak\xykpostbreak\xyknottips}

“cross” crossings:

The initialisation...
3.10. KNOTS AND LINKS FEATURE

The interface...

The drawing code...

\begin{verbatim}
\xydef@\xyvcrossv\xykcross@\xykunderstring@\xykhvobject@{(0,-1)}
\xydef@\xyhcross\xykcross@\xykunderstring@\xykhvobject@{(1,0)}
\xydef@\xyvcrossneg\xykcross@\xykunderstring@\xykhvobject@{(0,-1)}
\xydef@\xyhcrossneg\xykcross@\xykunderstring@\xykhvobject@{(1,0)}
\xydef@\xyvuncross\xykcross@\xykunderstring@\xykhvobject@{(0,-1)}
\xydef@\xyhuncross\xykcross@\xykunderstring@\xykhvobject@{(1,0)}

{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7#8#9{\save,
  "\_<"{\xykz\_\ar #1@'\"\_<"**{}?+(.25)@+,?+(.375,0)@+,
  "\>_";"\_>"**{}?-(.25)@+,?+(.375,0)@+}"\>_#3#4#5},
  "\_<",{\xykz\_\ar #2@'\"\_>"**{}?+(0,.375)@+,?+(.375,0)@+,
  "\_>_";"\_>_"**{}?-(0,.375)@+,?+(.375,0)@+}"\_>_#6#7#8} \restore\POS #9}}
\xylet@\xykhvobject@{\xykunderstring@\xykcross@\xykhvobject@{(1,0)}}

{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7#8#9{\save,
  "\_<",{\xykz\_\ar #1@'\"\_<"**{}?+(.25)@+,?+(0,.375)@+,
  "\>_";"\_>_"**{}?-(0,.375)@+,?+(0,.375)@+}"\_>_#3#4#5},
  "\_<",{\xykz\_\ar #2@'\"\_>_"**{}?+(0,.375)@+,?+(0,.375)@+,
  "\_<_";"\_<_"**{}?-(0,.375)@+,?+(.25)@+}"\_<_#6#7#8} \restore\POS #9}}
\xylet@\xykcross@\xykhvobject@{\xykunderstring@\xykcross@\xykhvobject@{(1,0)}}

\end{verbatim}
“over” crossings:

The initialisation...

\xydef@\vover\begingroup\def\afterknot@{\xyvover}\%\xydef@\xykprePLACE@{(.2)}\def\xykpostPLACE@{(.9)}\%\xydef@\xykmidPLACE@{(.8)}\def\xykpostbreak@@{[(.725)\knothole]}\%\xyFN@\xykparsecross@ }\%
\xydef@\hover\begingroup\def\afterknot@{\xyhover}\%\xydef@\xykprePLACE@{(.2)}\def\xykpostPLACE@{(.9)}\%\xydef@\xykmidPLACE@{(.8)}\def\xykpostbreak@@{[(.725)\knothole]}\%\xyFN@\xykparsecross@ }\%
\xydef@\vunder\begingroup\def\afterknot@{\xyvunder}\%\xydef@\xykprePLACE@{(.1)}\def\xykpostPLACE@{(.8)}\%\xydef@\xykmidPLACE@{(.2)}\def\xykpostbreak@@{[(.275)\knothole]}\%\xyFN@\xykparsecross@ }\%
\xydef@\hunder\begingroup\def\afterknot@{\xyhunder}\%\xydef@\xykprePLACE@{(.1)}\def\xykpostPLACE@{(.8)}\%\xydef@\xykmidPLACE@{(.2)}\def\xykpostbreak@@{[(.275)\knothole]}\%\xyFN@\xykparsecross@ }\%
\xydef@\vunover\begingroup\def\afterknot@{\xyvunover}\%\xydef@\xykprePLACE@{(.1)}\def\xykpostPLACE@{(.8)}\%\xydef@\xykmidPLACE@{(.2)}\def\xykpostbreak@@{}\%\xyFN@\xykparsecross@ }\%
\xydef@\hunover\begingroup\def\afterknot@{\xyunover}\%\xydef@\xykprePLACE@{(.1)}\def\xykpostPLACE@{(.8)}\%\xydef@\xykmidPLACE@{(.2)}\def\xykpostbreak@@{}\%\xyFN@\xykparsecross@ }\%
\xydef@\xoverv\begingroup\def\afterknot@{\xyxoverv}\%\xydef@\xykprePLACE@{(.15)}\def\xykpostPLACE@{(.85)}\%\xydef@\xykmidPLACE@{(.15)}\def\xykpostbreak@@{\knothole}\%\xyFN@\xykparsecross@ }\%
\xydef@\xunderv\begingroup\def\afterknot@{\xyunderv}\%\xydef@\xykprePLACE@{(.15)}\def\xykpostPLACE@{(.85)}\%
3.10. KNOTS AND LINKS FEATURE

The interface...

\def\xykmidPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(.15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(.15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(85)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(.15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(85)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(85)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%

\xydef@\xoverh\begin{group}\def\afterknot@{\xoverh}\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(85)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(85)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
\def\xykPLACE@{(15)}\def\xykbreak@@{\knothole%\xyFN@\xykparsecross@ }%
The drawing code...

```
{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7#8#9{\save,
"\"","{\xykz\ar #1\"\"}**{}?(.25)@+,(.625)+(-.125)@+,?
"\"","{\xykz\ar #2\"\"}**{}?(.25)@+,+(.125,0)@+,
"\"","{\xykz\ar #3\"\"}**{}?(.25)@+,+(.125,0)@+,
"\"","{\xykz\ar #4\"\"}**{}?(.25)@+,+(.125,0)@+,
"\"","{\xykz\ar #5\"\"}**{}?(.25)@+,+(.125,0)@+,
"\"","{\xykz\ar #6\"\"}**{}?(.25)@+,+(.125,0)@+,
"\"","{\xykz\ar #7\"\"}**{}?(.25)@+,+(.125,0)@+,
"\"","{\xykz\ar #8\"\"}**{}?(.25)@+,+(.125,0)@+,
\restore\POS#9}}
\xylet@\xykoverv=\next

{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7#8#9{\save,
"\"","{\xykz\ar #1\"\"}
"\"","{\xykz\ar #2\"\"}**{}?(.25)@+,(.625)+(-.125)@+,?
"\"","{\xykz\ar #3\"\"}**{}?(.25)@+,+(.125,0)@+,
"\"","{\xykz\ar #4\"\"}**{}?(.25)@+,+(.125,0)@+,
"\"","{\xykz\ar #5\"\"}**{}?(.25)@+,+(.125,0)@+,
\restore\POS#9}}
\xylet@\xykoverh=\next

{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7#8#9{\save,
"\"","{\xykz\ar #1\"\"}
"\"","{\xykz\ar #2\"\"}**{}?(.25)@+,(.625)+(-.125)@+,?
"\"","{\xykz\ar #3\"\"}**{}?(.25)@+,+(.125,0)@+,
\restore\POS#9}}
\xylet@\xykunoverv=\next

{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7#8#9{\save,
"\"","{\xykz\ar #1\"\"}
"\"","{\xykz\ar #2\"\"}**{}?(.25)@+,(.625)+(-.125)@+,?
\restore\POS#9}}
\xylet@\xykunoverh=\next
```

```
{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7#8#9{\save,
"\"","{\xykz\ar #1\"\"}
"\"","{\xykz\ar #2\"\"}**{}?(.25)@+,(.625)+(-.125)@+,?
\restore\POS#9}}
\xylet@\xykxoverv=\next

{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7#8#9{\save,
"\"","{\xykz\ar #1\"\"}
"\"","{\xykz\ar #2\"\"}**{}?(.25)@+,(.625)+(-.125)@+,?
\restore\POS#9}}
\xylet@\xykxoverh=\next
```

```
{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7#8#9{\save,
"\"","{\xykz\ar #1\"\"}
"\"","{\xykz\ar #2\"\"}**{}?(.25)@+,(.625)+(-.125)@+,?
\restore\POS#9}}
\xylet@\xykxoverv=\next

{\xyuncatcodes \gdef\next#1#2#3#4#5#6#7#8#9{\save,
"\"","{\xykz\ar #1\"\"}
"\"","{\xykz\ar #2\"\"}**{}?(.25)@+,(.625)+(-.125)@+,?
\restore\POS#9}}
\xylet@\xykxoverh=\next
```
```
3.10. KNOTS AND LINKS FEATURE

“twist” crossings:

The initialisation...

The interface...
The drawing code...

3.10b. The above examples also show how to use - to get the mirror-image of a particular crossing. Any numerical scale factor can be used by enclosing it within [..] e.g. [2.3] scaling a single piece without affecting the rest of the picture. The scale-factor must occur before any label or arrow-tip specifiers, see below). Vertical crossings remain vertical under scalings; the current ⟨pos⟩ still moves by 1 coordinate unit in the ‘down’ direction. Similarly horizontal crossings remain horizontal. The single character - is a shorthand version for [-1], effectively giving a half-turn rotation in a rectangular basis.
3.10c. A knot-piece need not be rectangular. By specifying $\langle pos_1 \rangle \langle pos_2 \rangle \langle pos_3 \rangle \langle pos_4 \rangle$ the four corners UL, UR, DL, DR are set to the given $\langle pos \rangle$s respectively. The local basis is established so that
\[
\begin{align*}
    r\text{-hop} & \leftrightarrow \frac{1}{2}(\langle pos_2 \rangle - \langle pos_1 \rangle + \langle pos_4 \rangle - \langle pos_3 \rangle) \\
    u\text{-hop} & \leftrightarrow \frac{1}{2}(\langle pos_1 \rangle - \langle pos_3 \rangle + \langle pos_2 \rangle - \langle pos_1 \rangle).
\end{align*}
\]

3.10d. With a non-rectangularly shaped piece it will usually be necessary to adjust the place where the ‘hole’ occurs in the ‘under’ string. This is done by specifying $\langle \text{num} \rangle$, with $0 \leq \langle \text{num} \rangle \leq 1$ being the parameter value of the new location for the hole.
3.10. KNOTS AND LINKS FEATURE

The knot feature allows for the easy placement of the following objects along the strings of a crossing:

- labels on the strings;
- arrowheads for direction or orientation;
- holes in strings, allowing another string to be drawn passing over.

The characters <, > and | are used to indicate to which string portion the object is associated: with | denoting the string which crosses the other, while < and > denote the initial and final portions of the ‘crossed’ string.

A simple label enclosed in braces, for example \vcross\{x\}, is set in math-mode using the \labelstyle, at a pre-determined place on the string portion, shifted in either the ‘above’ or ‘below’ direction from the curve at this point. (For each crossing depicted in figure 3.8 only default values are used for the place and shift-direction.)
3.10h. If the first character within the braces \{..\} is * e.g. \htwist\{*(object)\}, then a general \langle object \rangle may be placed as a label. Furthermore if the first character is ^ or _ or 1, then the interpretation is, e.g. \vtwist\{~(anchor)\{it\}\}, as in 3.2 to place \langle it \rangle as a label along an \ar of the arrow feature.

3.10i. A second character < or > specifies that an arrowhead should appear at the pre-determined place on the chosen string. Here > denotes an arrowhead pointing with the natural orientation, while < points against. Due to the curvature of the strings, it is usually best to \UseComputerModernTips rather than normal arrow-tips.

3.10j. To generate a ‘hole’ use \knothole, or simply \khole, as following token. This generates a ‘break’, in the sense of 3.3j. Indeed such a ‘hole’ is used to separate the two portions of the ‘crossed’ string. Default size for the hole is 5pt, which is alterable via \knotholesize\{(dimen)\}; normally used to set the size for all holes in a diagram.

3.10k. If the resulting \khole is either too large or perhaps non-existent, this could be due to a technicality in the way breaks in curves are handled. This problem should not occur with the standard crossings, using a rectangular basis, but it may occur with non-rectangular bases. An easy ‘fix’ is to include an extra null-break on the string, using <1, >1 or 11, which should place the zero-sized break at parameter value .5 on the curve. The specification should precede a \khole at a higher parameter value, or come after one at a lower value. The aim of the above ‘fix’ is to position the null-break as close as possible to where the curve is farthest from the line joining its end-points, which is usually at parameter value (1.5).

\xydef\xykparsebreak{%
\ifx\space@\next \expandafter\DN@\space\xyFN@\xykparsebreak@%
\else \addLT@\ifx\next
\expandafter\appendtoholder@\xykbacktemp@\next@
\addLT@\DN@\xyFN@\xykparser@%
\else \addGT@\ifx\next
\expandafter\appendtoholder@\xykforetemp@\next@
\addGT@\DN@\xyFN@\xykparser@%
\else \ifx|\next
\expandafter\appendtoholder@\xyknulletemp@\next@
\DN@\xyFN@\xykparser@%
\else \ifx\next \bgroup\DN@\#1\{\checkgroupbreak@\#1\%
\DN@\xyFN@\xykparsebreak@%
\else \ifx\next \knothole@\DN@\#1\{\expandafter\appendtoholder@%
\xykdefaultbreak@\xybreak@\next@
\xyFN@\xykparser@%
\else
\expandafter\xykdefaultbreak@\xybreak@\next@
\xyFN@\xykparser@%
\fi\fi\fi\fi\fi\fi\fi \next@ }

Templates for arrowheads.
\xydef\xykforetips@\{^>*\dir{>}}
\xydef\xykbacktips@\{^<*\dir{<}}
When the next token is a begin-group character \{ then look at the first token inside the group. If it is \^, \_ or | then the group contents will be an \langle anchor\rangle for user-chosen places along the string. Otherwise the place is taken to be the already specified \xyknotPLACE and the group contains a label, to be set in the usual way for labels, using \_ due to the orientation of the curve. However if the first token is \* then the user is assumed to be dropping his own label, possibly with \langle object\rangle modifiers.
Multiple breaks, arrow-heads and labels may be specified along the two strings of a crossing; simply place their specifications one after another; e.g. <>|>>><{x}|{y}>{z} was used in figure 3.8.

The only proviso is that all ‘breaks’ along a single strand must occur with increasing order of parameter position. On the ‘crossed’ string this includes the automatic ‘hole’ to create space for the other string. Hence it is advisable to use just the (+..) and (-..) variants for small adjustments, and to keep these correctly ordered.

Adjustment of position along the strings can be achieved using a ⟨factor⟩, as in \vover{|(+.1)}. Allowed syntax is (⟨sign⟩⟨num⟩) where ⟨sign⟩ is + or - to increment or decrement from the pre-defined value. Also allowable are = or ⟨empty⟩ to set the parameter position to ⟨num⟩, which must lie between 0 and 1 to have any meaning.

This parses adjustments of the \xyknotPLACE.
Arrowheads can also be placed at either, or both, ends of the strings forming a crossing. This is governed by a pair of booleans, initially \{FF\}. It is changed for all subsequent strings in a diagram by \texttt{\textbackslash knottips\{..\}} where the recognised values are \{FF\}, \{FT\}, \{TF\} and \{TT\}, denoting tips (T) or not (F) at the start and end of each string. To add arrowtips at the start of strings in a particular crossing, append the 2-character combination =<; similarly => adds tips at the ends, if not already requested. The combinations == and != specify both (TT) and none (FF) respectively. These 2-character pairs can be mixed in with any specifications for labels and breaks, etc. Multiple pairs compound their effect; in particular =<> gives the same result as ==, while !==< is needed to change \{FT\} into \{TF\}.

These are best used with single pieces, as in the following equation.

$$\nabla \left[ \chi_+ \right] - \nabla \left[ \chi_- \right] = -z \nabla \left[ \chi_- \right]$$

\UseComputerModernTips \knottips{FT}
\def\Conway#1{\mathord{\nabla \Bigl[ \raise5pt\xybox{0;/r1pc/:#1}\Bigr]}}
$$\Conway\htwist - \Conway\htwistneg \;=\; -z \, \Conway\huntwist$$

\texttt{\textbackslash xykparsetips\{\}}
\texttt{\textbackslash def\textbackslash xyknotTIPS\{\} \textbackslash def\textbackslash xyknotPLACE\{\}}
\texttt{\textbackslash let\textbackslash xykparser\textbackslash =\textbackslash xykparsetips\{\}}
\texttt{\textbackslash xyFN\textbackslash =\textbackslash xykparser\{\} \textbackslash xykparsetips\{\}}
Joins

3.10. The “joins” are used to connect the loose ends of crossing strings. In particular “loops” and “caps” are for placing on ends of horizontal or vertical ‘twist’ and ‘cross’ crossings, leaving the current ⟨pos⟩ fixed. The “bends” join non-adjacent crossings of the same type, either horizontal or vertical.

The \xcap.. pieces are designed to join adjacent \xover.. pieces; they move c either vertically or horizontally, as appropriate. Finally the \xbend.. pieces allow for smooth joins of 45° slopes to horizontal or vertical slopes. For these the actual positioning of the piece, see figure 3.9, is not entirely obvious.

Figure 3.9 displays the orientation on the joins. Also indicated are default positions for labels and
3.10. KNOTS AND LINKS FEATURE

...and arrow-tips; each piece uses the same code, e.g. \texttt{\textbackslash vloop <\textbackslash|\textgreater\textless\{x\}\{y\}\{z\}}. Furthermore the current \texttt{	extlangle pos\textrangle} before the piece is drawn is marked using \texttt{\textdegree}; that afterwards is indicated by \texttt{\times} or \texttt{\plus}. The ability to scale in size and place arrow-tips, breaks, labels etc. apply also to \texttt{\langle join\rangle} pieces. The only difference is...

3.10m. The three places referred to by \texttt{\langle,<,|,\rangle} are all on a single string. In particular \texttt{|} is always at the middle of the \texttt{\langle join\rangle}, whereas \texttt{<} and \texttt{>} are at \textit{earlier} and \textit{later} parameter values respectively. Any adjustments involving breaks should occur in increasing parameter order.

\textbf{Parsing}

```
\xdef\xykparsejoin@{%
  \def\xykSCALE@@{}%
  \edef\xyknotPLACE{\xykmidPLACE@}%
  \let\xykparser@=\xykparsejoin@@
  \def\xykdefaultbreak@{\let\xykbreak@=\xykforetemp@
    \def\xyknotPLACE{(.5)}\appendtoholder@}%
  \% \edef\xykdefaultbreak@{\edef\xyknotPLACE{\xykmidPLACE@}\
    \appendtoholder@}%
  \def\xykdefaultbreak@{\edef\xyknotPLACE{\xykmidPLACE@}\
    \appendtoholder@}\addLT@\DN@{\xyprecross@}\
  \else\addGT@\DN@{\xypostcross@}\
  \else\ifx|\next \let\xykbreak@=\xykholetemp@
    \def\xyknotPLACE{(.5)}\appendtoholder@\DN@{\xymidcross@}\
  \else\ifx\next \addEQ@\DN@{\xyFN@\xykparsetips@}\
    \else\addAT@\DN@{\xyFN@\xykjoincontrol@}\
    \xyadjustJOIN@\#1\DN@{\xyknotPLACE@}\xykparsejoin@@}\
  \ife\DN@{\afterknot@}%
  \fi\fi\fi
  \xylet@\xykparsejoin@@=\next
```

3.10n. A parameter can be altered, using \texttt{@\langle adjust\rangle}, to effect subtle adjustments to the shape of any join. Within a rectangular basis the horizontal or vertical tangents are preserved and overall reflection or rotation symmetry is preserved. Thus this parameter affects the ‘flatness’ of a cap or loop, or the amount of curvature is s-bends and z-bends. For \texttt{\textbackslash xcap..s} and \texttt{\textbackslash xbend..s} the 45° angle is altered; this is especially useful to match the tangents when a knot-piece has been specified using the technique of note 3.10c.
The normal range for these parameters is between 0 and 1. Other values can be used with interesting results—the parameter determines the location of control points for a Bézier cubic curve.

<table>
<thead>
<tr>
<th>piece</th>
<th>value</th>
<th>effect on...</th>
</tr>
</thead>
<tbody>
<tr>
<td>.cap</td>
<td>.25</td>
<td>flatness of cap;</td>
</tr>
<tr>
<td>.loop</td>
<td>.75</td>
<td>flatness of loop;</td>
</tr>
<tr>
<td>\sbend.</td>
<td>.75</td>
<td>curvature in the ‘s’;</td>
</tr>
<tr>
<td>\zbend.</td>
<td>.75</td>
<td>curvature in the ‘z’;</td>
</tr>
<tr>
<td>\rcap.</td>
<td>.5</td>
<td>height of cap, slope at base;</td>
</tr>
<tr>
<td>\rxbend.</td>
<td>.5</td>
<td>curvature, slope at base.</td>
</tr>
</tbody>
</table>

The following example gives three ways of specifying a ‘trefoil’ knot, using the poly feature to establish the location of the vertices for knot-pieces. In each the ⟨crossing⟩s are calculated to fit together smoothly; a different way of creating ⟨join⟩s is used in each. Also the third displays subtle changes of the \texttt{\texttt{\texttt{3.10n}}} join control.

\begin{verbatim}
\def\TrefoilA{\xygraph{{0;/r.75pc/:}
  !P3"a"{>}{!P9"b"{>.{(1.3288,0):}->}}
  !P3"c"{>.{(2.5,0):}->}}
  !{\vover{"b2"}"b1"}{"a1"}{"a3"}
  !{"b4";"b2"**\crv{"c1"}}
  !{\vover{"b5"}"b4"}{"a2"}{"a1"}
  !{"b7";"b5"**\crv{"c2"}}
  !{\vover{"b8"}"b7"}{"a3"}{"a2"}
  !{"b1";"b8"**\crv{"c3"}}}\}
%
\def\TrefoilB{\xygraph{{0;/r.75pc/:}
  !P3"a"{>}{!P9"b"{>.{(1.3288,0):}->}}
  !P3"c"{>.{(2.5,0):}->}}
  !{\vover{"b2"}"b1"}{"a1"}{"a3"}
  !{\vcap{"c1"}"c1"}{"b4"}{"b2"}@(+.1)}
  !{\vover{"b5"}"b4"}{"a2"}{"a1"}
  !{\vcap{"c2"}"c2"}{"b7"}{"b5"}@(+.2)}
  !{\vover{"b8"}"b7"}{"a3"}{"a2"}
  !{\vcap{"c3"}"c3"}{"b1"}{"b8"}}}\}
%
\def\TrefoilC{\xygraph{{0;/r.75pc/:}
  !P3"a"{>}{!P12"b"{>.{(1.414,0):}->}}
  !{\vover{"b2"}"b1"}{"a1"}{"a3"}
  !{\save 0;"b2"="b5":"b5", \rxbph @(+.1)\restore}
  !{\vover{"b6"}"b5"}{"a2"}{"a1"}
  !{\save 0;"b6"="b9":"b9", \rxbph @(+.2)\restore}
  !{\vover{"b10"}"b9"}{"a3"}{"a2"}}\}
\end{verbatim}
3.10. KNOTS AND LINKS FEATURE

$$\text{TrefoilA}\quad\text{TrefoilB}$$

These macros are common to most joins.

```
\xydef@\xykjoincontrol@\{.75\}
\xydef@\xykjoincontrol@#1\{DN@{#1}\}
\if\next@\empty\edef\xykjoincontrol@\{.75\}
\else\dimen@=#1\p@\edef\xykjoincontrol@\{\expandafter\removePT@\the\dimen@\fi
\xylet@\xykjoincontrol\empty %\xyk@joincontrol@
\xydef@\xykjoincontrol@(#1){DN@{#1}}
\if\next@\empty\edef\xykjoincontrol@\{.75\}
\else\dimen@=#1\p@\edef\xykjoincontrol@\{\expandafter\removePT@\the\dimen@\fi
\fi
\xydef@\xykjoincontrol@i#1\{DN@{#1}\}
\if\next@\empty\edef\xykjoincontrol@\{.75\}
\else\dimen@=#1\p@\edef\xykjoincontrol@\{\expandafter\removePT@\the\dimen@\fi
\fi
\xydef@\xykjoincontrol@i#1\{DN@{#1}\}
\if\next@\empty\edef\xykjoincontrol@\{.75\}
\else\dimen@=#1\p@\edef\xykjoincontrol@\{\expandafter\removePT@\the\dimen@\fi
\fi
```

The interface...

```
\xydef@\hcap{\begingroup\def\afterknot@{\xyhcap}\
  \def\xykprePLACE@{(.385)}\def\xykpostPLACE@{(.615)}\edef\xykjoincontrol@{.25}\xyFN@\xykparsejoin@ }%
\xydef@\vcap{\begingroup\def\afterknot@{\xyvcap}\
  \def\xykprePLACE@{(.385)}\def\xykpostPLACE@{(.615)}\edef\xykjoincontrol@{.25}\xyFN@\xykparsejoin@ }%
\xydef@\xcapv{\begingroup\def\afterknot@{\xyxcapv}\
  \def\xykprePLACE@{(.2)}\def\xykpostPLACE@{(.8)}\edef\xykjoincontrol@{.5}\xyFN@\xykparsejoin@ }%
\xydef@\xcaph{\begingroup\def\afterknot@{\xyxcaph}\
  \def\xykprePLACE@{(.2)}\def\xykpostPLACE@{(.8)}\edef\xykjoincontrol@{.5}\xyFN@\xykparsejoin@ }%
```

“cap” joins:

The initialisation...

```
\xydef@\hcap{\begingroup\def\afterknot@{\xyhcap}\
  \def\xykprePLACE@{(.385)}\def\xykpostPLACE@{(.615)}\edef\xykjoincontrol@{.25}\xyFN@\xykparsejoin@ }%
\xydef@\vcap{\begingroup\def\afterknot@{\xyvcap}\
  \def\xykprePLACE@{(.385)}\def\xykpostPLACE@{(.615)}\edef\xykjoincontrol@{.25}\xyFN@\xykparsejoin@ }%
\xydef@\xcapv{\begingroup\def\afterknot@{\xyxcapv}\
  \def\xykprePLACE@{(.2)}\def\xykpostPLACE@{(.8)}\edef\xykjoincontrol@{.5}\xyFN@\xykparsejoin@ }%
\xydef@\xcaph{\begingroup\def\afterknot@{\xyxcaph}\
  \def\xykprePLACE@{(.2)}\def\xykpostPLACE@{(.8)}\edef\xykjoincontrol@{.5}\xyFN@\xykparsejoin@ }%
```

The interface...
CHAPTER 3. FEATURES

The drawing code...

"loop" joins:

The initialisation and interface...

The drawing code...
“bend” joins:

The initialisation...

The interface...

\quad xydef@\xyzbendh{\begingroup\def\afterknot@{\xyzbendh}\%
1643 \def\xykpPLACE@{(.25)}\def\xykpPLACE@{(.75)}\%
1644 \def\xykmPLACE@{(.5)}\edef\xykjoincontrol{\xykjoincontrol@@}\%
1645 \xyFN@\xykparsejoin@ }\%
1646 \xydef@\xysbendv{\begingroup\def\afterknot@{\xysbendv}\%
1647 \def\xykpPLACE@{(.25)}\def\xykpPLACE@{(.75)}\%
1648 \def\xykmPLACE@{(.5)}\edef\xykjoincontrol{\xykjoincontrol@@}\%
1649 \xyFN@\xykparsejoin@ }\%
1650 \xydef@\xysbendh{\begingroup\def\afterknot@{\xysbendh}\%
1651 \def\xykpPLACE@{(.25)}\def\xykpPLACE@{(.75)}\%
1652 \def\xykmPLACE@{(.5)}\edef\xykjoincontrol{\xykjoincontrol@@}\%
1653 \xyFN@\xykparsejoin@ }\%
1654 \xydef@\xyzbendv{\begingroup\def\afterknot@{\xyzbendv}\%
1655 \def\xykpPLACE@{(.25)}\def\xykpPLACE@{(.75)}\%
1656 \def\xykmPLACE@{(.5)}\edef\xykjoincontrol{\xykjoincontrol@@}\%
1657 \xyFN@\xykparsejoin@ }\%
1658 \xydef@\xyzbendh{\begingroup\def\afterknot@{\xyzbendh}\%
1659 \def\xykpPLACE@{(.25)}\def\xykpPLACE@{(.75)}\%
1660 \def\xykmPLACE@{(.5)}\edef\xykjoincontrol{\xykjoincontrol@@}\%
1661 \xyFN@\xykparsejoin@ }\%
1662 \xydef@\xyxbendr{\begingroup\def\afterknot@{\xyxbendr}\%
1663 \def\xykpPLACE@{(.15)}\def\xykpPLACE@{(.75)}\%
1664 \def\xykmPLACE@{(.5)}\edef\xykjoincontrol{\xykjoincontrol@\%
1665 \xyFN@\xykparsejoin@ }\%
1666 \xydef@\xyxbendl{\begingroup\def\afterknot@{\xyxbendl}\%
1667 \def\xykpPLACE@{(.15)}\def\xykpPLACE@{(.75)}\%
1668 \def\xykmPLACE@{(.5)}\edef\xykjoincontrol{\xykjoincontrol@\%
1669 \xyFN@\xykparsejoin@ }\%
1670 \xydef@\xyxbendu{\begingroup\def\afterknot@{\xyxbendu}\%
1671 \def\xykpPLACE@{(.15)}\def\xykpPLACE@{(.75)}\%
1672 \def\xykmPLACE@{(.5)}\edef\xykjoincontrol{\xykjoincontrol@\%
1673 \xyFN@\xykparsejoin@ }\%
1674 \xydef@\xyxbendd{\begingroup\def\afterknot@{\xyxbendd}\%
1675 \def\xykpPLACE@{(.15)}\def\xykpPLACE@{(.75)}\%
1676 \def\xykmPLACE@{(.5)}\edef\xykjoincontrol{\xykjoincontrol@\%
1677 \xyFN@\xykparsejoin@ }%
The implementation...

\begin{verbatim}
\def\xyxbendu\{\xykjoin\xyxbendu \xyksetjoin \xykhobject\{+(1,0)\}\}
\def\xyxbendd\{\xykjoin\xyxbendd \xyksetjoin \xykhobject\{+(1,-1)\}\}
\end{verbatim}
### Changing the string-style

It is not necessary to use solid curves; any style available to curves and arrows can be chosen using...

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\knotstyle{⟨char⟩}</td>
<td>use \dir{⟨char⟩}</td>
</tr>
<tr>
<td>\knotstyles{⟨char⟩}{⟨char⟩}</td>
<td>two styles</td>
</tr>
<tr>
<td>\knotSTYLE{⟨code⟩}</td>
<td>use ⟨code⟩</td>
</tr>
</tbody>
</table>

In each case the new style applies to *all* subsequent knot pieces, except that the two styles apply only to crossings. The latter case allows use of object (modifier)s. The ⟨code⟩ consists of two groups {...}, each containing ⟨arrow⟩ forms, as in 3.2 and notes 3.3m, 3.3r. Only the first ⟨arrow⟩ form is used with ⟨join⟩s whereas the two forms are used respectively with the two strings of a ⟨crossing⟩ in the order that they are drawn.

\begin{verbatim}
\xylet@\xykx bendd=\next
\end{verbatim}

#### Initialisation:

*default values for the methods.*

\begin{verbatim}
\xydef@\knotholesize@#1{\def\xykholesize{#1}}
\xydef@\xykholesize{5pt}
\xylet@\knotholesize=\knotholesize@
\xydef@\holesize=\knotholesize@
\xydef@\knottips#1{\def\xyknotTIPS@{#1}}
\xydef@\xyknotTIPS@{FF}
\xydef@\xyknotTIPS@{FF}
\xydef@\xykSCALE@@{}
\%\xydef@\xyknotbreak@@{%}
\xydef@\xykprebreak@@{%}
\xydef@\xykpostbreak@@{%}
\xydef@\xykmidbreak@@{%}
\xydef@\xykpPLACE@{(.25)}%
\xydef@\xykpostPLACE@{(.75)}%
\xydef@\xykmidPLACE@{%}
\xylet@\xykPLACE@=\xykmidPLACE@
\end{verbatim}

\begin{verbatim}
\{\gdef\next\{\@{-}\}{\@{-}}\}
\xylet@\knotSTYLE=\next
\{\gdef\next\#1{\def\knotSTYLE\{\@{\#1}\}{\@{\#1}\}}\}
\xylet@\knotstyle=\next
\{\gdef\next\#1\#2{\def\knotSTYLE\{\@{\#1}\}{\@{\#2}\}}\}
\xylet@\knotstyles=\next
\{\gdef\next\#1{\def\knotSTYLE{\#1}}\}
\xylet@\KNOTstyle=\next
\end{verbatim}

\begin{verbatim}
\{\gdef\next\#1\#2\#3\#4{%\message{\#1:\#2:\#3:\#4}%
\save(0,0);"\#2"-"\#1"+"\#4"-"\#3";(.5,0):
"\#3"-"\#1"+"\#4"-"\#2";:(0,.5)::}}
\xylet@\xykflexbase@=\next
\end{verbatim}
CHAPTER 3. FEATURES

The end & Log

\endinput

% $Log: xyknot.doc,v $  
% Revision 3.9 2011/03/14 20:14:00 krisrose  
% Preparing for release 3.8.6.  
%  
% Revision 3.8 2010/06/10 18:45:50 krisrose  
% Reference to GPL by URL.  
%  
% Revision 3.7 2010/05/14 01:12:16 krisrose  
% Figure fixes.  
%  
% Revision 3.6 2010/05/06 17:46:30 krisrose  
% Ross Moore’s e-mail address updated.  
%  
% Many obsolete files degraded to Historic.  
%  
% Revision 3.5 2010/04/16 06:06:52 krisrose  
% Preparing for a new release...  
%  
% Revision 3.4 1997/05/18 01:13:24 ross  
% Essential bugfixes.  
%  
% Revision 3.3 1996/12/18 09:20:49 ross  
% no changes  
%
3.11 Smart Path option

Vers. 3.6 by George C. Necula (necula@cs.cmu.edu)

Header:

This extends the ‘arrow’ feature, which is therefore required, with a “smart” ⟨path⟩ between two ⟨pos⟩itions.
The \langle turn\rangle syntax is extended with the construction
\begin{align*}
\langle \text{turn} \rangle \rightarrow \text{s} \quad \langle \text{diag} \rangle \quad \langle \text{diag} \rangle \langle \text{turnradius} \rangle
\end{align*}
\ar \in_{out}/5\text{pt} which draws a connector leaving \( p \) in the \( \text{in} \) \langle diag\rangle onal direction and arrives at \( c \) in the \( \text{out} \) \langle diag\rangle onal direction, using 5\text{pt} turns. The connector contains only horizontal or vertical lines and \( \frac{1}{8} \) sectors of circles of the given (optional) \langle turnradius \rangle.

**Bug:** Any labels are placed at the end of the connection.

The smart connection algorithm is as follows:

A smart connector consists only of horizontal and vertical straight segments connected with 1/4 circles. 1/8 circles might be use at the start and end of the connector to reach a horizontal or a vertical.

All possible connectors can be reduced to a series of equivalence classes, where two connectors are considered equivalent if they are identical after removing from each of them all the horizontal and vertical straight segments. Such a connector is called a representative.

The algorithm first computes 10 representatives that start and end in the given directions and have at most one point where the orientation is changed. 5 of the representatives start counter-clockwise and the other 5 start clockwise. Among the counter-clockwise representatives, 1 does not have any change in orientation (one arc of circle starting and ending in the given directions). The other 4 are obtained by changing the orientation in the four points where the circle is tangent to a horizontal or a vertical.

For each representative the algorithm determines if it can be extended with horizontal and vertical segments to reach the destination point when starting in the source point (the directions are already right by construction). For example, horizontal straight lines could be inserted in all the points where the representative is tangent to a horizontal line. In each such point a straight line segment can be inserted to extend the connector horizontally but only in the same direction of the connector at that point (the direction of the connector in a point is towards the destination). For example, if the horizontal distance between the source and destination points is \( D_x \) and the horizontal distance between the start and the end of the representative connector is \( d_x \) then the distance “\( D_x - d_x \)” must be covered using horizontal straight lines. This distance is distributed equally among all the horizontal extension points of the right direction. If no such points exist, then the representative is discarded.

The cost of a connector is the length of the representative plus the total length of straight extension segments, i.e., \( D_x - d_x + D_y - d_y \).

The algorithm selects among all the representatives that are not discarded, the one leading to the shortest connection.

\begin{verbatim}
\xypicture
\ar \in_{out}/5\text{pt} \let\origPATHturn@=\PATHturn@
\def\PATHturn@{%\ifx \space@\next \expandafter\DN@\space{\xyFN@\PATHturn@}%gobble spaces \else\ifx s\next \let\origPATHturn@i=\PATHturn@i\let\PATHturn@i=\PATHsmartturn@i \let\origPATHturn@cir=\PATHturn@cir\let\PATHturn@cir=\PATHsmartturn@cir \DN@ s{\xyFN@\origPATHturn@}% Let the original routines do the parsing \else \DN@{\origPATHturn@}% \fi\fi\next@}
\end{verbatim}
3.11. SMART PATH OPTION

\xydef\PATHsmartturn@cir{%
\edef\nexto{%\{\CIRin@@\}{\{expandafter\noexpand\CIRorient@@\}{\CIRout@@\}}}%
\expandafter\\PATHturn@i\nexto

% This is the changed function for placing the turn
\xydef\PATHsmartturn@i#1#2#3{%
\DN@@#1{%\def\PATHinit@@{\xy@@%\def\sm@CIRin{#1}\def\sm@CIRout{#3}% The IN and OUT directions. Ignore \% the orientation \ifx\sm@CIRout\empty%Only on direction is given. Use it as OUT and %use the previous direction as IN \let\sm@CIRout=\sm@CIRin \let\sm@CIRin=\PATHlastout@@ \fi %#1\relax\xy@@{\enter@{\basefromthebase@}}%\xy@@{\sm@conn}% Draw the connection \xy@@{\X@p=\X@c \Y@p=\Y@c \czeroEdge@% Save the start of the segment \count@=\sm@CIRout\count@=\the\count@% Move forward a dash to % touch the edge \dimen@=\xydashl@ \ABfromdiag@ \advance\X@c\A@ \advance\Y@c\B@ \edef\PATHinitial@@{\xy@from\xy@@{\X@c=\the\X@c \Y@c=\the\Y@c \noexpand\czeroEdge@ %\noexpand\PATHmitslide@@true}}% %\xy@@{\leas@ % \edef\PATHlastout@@{\sm@CIRout}% Store the last direction \count@=\sm@CIRout \dimen@=\xydashl@ \Directionfromdiag@}}%
\expandafter\PATHturn@i\expandafter{\the\toks@}\toks@={}% \let\PATHextra@@=\empty \def\PATHpost@@{\xy@@\PATHpostpos@@}{\sm@CIRout}% Store the last direction \count@=\sm@CIRout \dimen@=\xydashl@ \Directionfromdiag@}}%
\expandafter\PATHextra@@=\empty \let\PATHlabelsextra@@=\relax \let\PATHturn@i=\origPATHturn@i \let\PATHturn@cir=\origPATHturn@cir \xyFN@\PATHturn@ii}
\xydef@\sm@nil{}
\xydef@\sm@nnil{\sm@nil}
\xydef@\sm@maxcost{1000mm}% A very long connector
\def\PATHinit@@{\xy@@%\def\sm@CIRin{\xy@@{\enter@{\basefromthebase@}}}%\xy@@{\sm@conn}% Draw the connection \xy@@{\X@p=\X@c \Y@p=\Y@c \czeroEdge@% Save the start of the segment \count@=\sm@CIRout\count@=\the\count@% Move forward a dash to % touch the edge \dimen@=\xydashl@ \ABfromdiag@ \advance\X@c\A@ \advance\Y@c\B@ \edef\PATHinitial@@{\xy@from\xy@@{\X@c=\the\X@c \Y@c=\the\Y@c \noexpand\czeroEdge@ %\noexpand\PATHmitslide@@true}}% %\xy@@{\leas@ % \edef\PATHlastout@@{\sm@CIRout}% Store the last direction \count@=\sm@CIRout \dimen@=\xydashl@ \Directionfromdiag@}}%
\expandafter\PATHturn@i\expandafter{\the\toks@}\toks@={}% \let\PATHextra@@=\empty \def\PATHpost@@{\xy@@\PATHpostpos@@}{\sm@CIRout}% Store the last direction \count@=\sm@CIRout \dimen@=\xydashl@ \Directionfromdiag@}}%
% This computes and draws the connection
\xydef@\sm@nil{}
\xydef@\sm@nnil{\sm@nil}
\xydef@\sm@maxcost{1000mm}% A very long connector
% This computes and draws the connection
\xydef@\sm@conn{% \xy@showthe p{Source}\xy@showthe c{Dest}% \W@{Computing connector from p in \sm@CIRin\space to c in \sm@CIRout % Radius is \turnradius@}% %\Adjust the end points to the edge of the objects in the given directions \entero{\cfro@@m the\Direction\cfrom@@ the\Direction@@ \cfrom@@ p% Adjust p \count@=\sm@CIRin\count@=\the\count@ \dimen@=\xydashl@ \ABfromdiag@ \advance\X@c\A@ \advance\Y@c\B@ \setupDirection\the\Edge@c\z@ \czeroEdge@ %\pfrom@@c
\reverseDirection@ % Now leave room for one dash to start with (for placing arrow tails)
\count@=\sm@CIRin\count@=\the\count@
\dimen@=\xydashl@ \ABfromdiag@
% Drop the arrow tail in the right place
edef\tmp@{\noexpand\dir\artail@@}\expandafter\sm@drop@\tmp@
\X@c=\X@p\advance\X@c\A@ \Y@c=\Y@p\advance\Y@c\B@ \czeroEdge@
% Draw a straight connection from p to c
\sm@stri
% Set p to the end of the connection
\X@p=\X@c \Y@p=\Y@c
\leave@
% \xy@showthe p{After adjust}%
\enter@{\pfromthep@ \DirectionfromtheDirection@}\pfromc@% Adjust c
\count@=\sm@CIRout\count@=\the\count@
\dimen@=\xydashl@ \ABfromdiag@ \advance\X@p-\A@ \advance\Y@p-\B@
\setupDirection@ \the\Edge@c\z@
\advance\X@c-\A@ \advance\Y@c-\B@% Leave room for a dash to terminate the seg
\czeroEdge@ % The final segment will be drawn by PATH@segment
% Clear the tail setting for the last segment
\def\PATHlabelsnext@@{}
\leave@
% \xy@showthe c{After adjust}%
% Set the turn radius
\R@=\turnradius@
\cirrestrict@@%Adjust the radius to fit the circles
% Initialize variables
\let\sm@circles=\sm@nil
\edef\sm@bestcost{\sm@maxcost}% Best so far is very bad
\sm@trycircles 1%Try counter clockwise first
\sm@trycircles{-1}%Try clockwise after
% % Now check the best
% % \ifdim\sm@bestcost<\sm@maxcost
% % \W@{\Best: \meaning\sm@bestconn}%
% \cfromp@
% \xy@showthe p{Source}%
% \expandafter\sm@conndraw\sm@bestconn
% \else
% \xyerror@{Cannot draw the smart connector}{}
% \fi
% % Compute and examine 5 representatives, for the orientation given
% as #1 (1 for counter-clockwise and -1 for clockwise).
\xydef@\sm@trycircles#1{% 
\ifnum #1>0% Compute the opposite orientation
\def\sm@CIRorienti{-1}%
\else
\xyerror@{Cannot draw the smart connector}{}
\fi
%
\def\sm@CIRorienti{1}%
\fi
% Generate lists of triples (IN,ORIENT,OUT) so that OUT_1=IN_2 and
% IN=IN_1 and OUT=OUT_2, and such that ORIENT_1=#1m ORIENT_2=-ORIENT_1
% Then try each list in turn.
%Try one circle first (a singleton list)
\edef\tmp@{{{\sm@CIRin}{#1}{\sm@CIRout}}}%
\expandafter\sm@trycirclelist\expandafter{\tmp@\sm@nil}%
%Now try lists of two circles.
%Advance "in" so that it is a multiple of 90 degrees (that is, odd)
\edef\sm@savecount@{\the\count@}%
% See if we need to add a 1/8 starting circle
\ifnum\the\count@=\the\count@ \else
  \edef\tmp@{{{\sm@CIRin}{#1}{\sm@CIRout}}}%
  \expandafter\sm@trycirclelist\expandafter{\tmp@\sm@nil}%
\fi
% Advance the direction in \count@ by #1 (each unit is 45deg) in the
% direction #2, taking care of wrap-around
\def\xydef@\sm@advancecount@#1#2{%
  \W@{Advancing count from \the\count@ space by #1\space}
  \ifnum #2>0
    \edef\tmp@{#1}%
  \else
    \edef\tmp@{-#1}%
  \fi
  \edef\sm@savecount@{\the\count@}%
  \sm@advancecount@ 2{#1}\edef\sm@savecount@{\the\count@}% Advance 90 deg.
  \edef\tmp@{{{\sm@CIRin}{#1}{\sm@CIRout}}}%
  \expandafter\sm@trycirclelist\expandafter{\tmp@\sm@nil}%
% Advance the direction in \count@ by #1 (each unit is 45deg) in the
% direction #2, taking care of wrap-around
\endinput
% Round counter to the next odd element (multiple of 90 deg)
% in the given direction
\xydef@\sm@roundcount@#1{% 
\ifcase\the\count@
   \advance\count@ #1\or
   \or \advance\count@ #1\or
   \or \advance\count@ #1\or
   \or \advance\count@ #1\fi
\count@=\the\count@
\ifnum\the\count@<0 \advance\count@ 8\fi
\ifnum\the\count@>7 \advance\count@ -7\fi
\count@=\the\count@
}
\newif\ifsm@firstseg
\newif\ifsm@acceptable
% Given a representative described as a list of triplets ending with sm@nil,
% verify if is can be extended to the right destination
\xydef@\sm@trycirclelist#1{% 
\R@p=\z@\U@p=\R@p% Clear the deltas
\def\sm@exthp{0}\def\sm@exthm{0}% Clear the extension points
\def\sm@dxp{0pt}\def\sm@dyp{0pt}% Reset the extensions
\def\sm@segs{}% % Clear the segment list
\sm@firstsegtrue % Mark the first segment
\def\sm@connlen{0pt}% % Initialize the connector length
\let\sm@tryclcont=\sm@trycirclelist@i
\sm@trycirclelist@i #1%Strip {} to expose the list elements
% % The connector was processed
% % Add an extra segment at then end, if ending in a multiple of 90 deg
\count@@=\sm@CIRout\count@@=\the\count@\count@=\the\count@
\sm@roundcount@{1}% Direction does not matter
\ifnum\count@=\count@@
\edef\sm@segs{\sm@segs{\{\the\count@\}{1}\{\the\count@\}}}%Accum segment
\fi
% \W@{After trying CL deltax=\the\R@p, deltay=\the\U@p}%
% \sm@showext{\space}%
% \W@{ \space Segs: \meaning\sm@segs}%
%Now see if the connector can be extended enough
\A@=\X@p\advance\A@\R@p\advance\A@ -\X@c\A@=-\A@ % A=Dx
\B@=\Y@p\advance\B@\U@p\advance\B@ -\Y@c\B@=-\B@ % B=Dy
\sm@acceptabletrue
% \W@{ \space Dx=\the\A@, Dy=\the\B@}%
\ifdim\A@>0pt
  \ifnum\sm@exthp>0 \dimen@=\A@ \divide\dimen@\sm@exthp
  \edef\sm@dxp{\the\dimen@}%
  \else \sm@acceptablefalse\fi\fi
\ifdim\A@<0pt
  \ifnum\sm@exthm>0 \dimen@=\A@ \divide\dimen@\sm@exthm
  \edef\sm@dxm{\the\dimen@}%
  \else \sm@acceptablefalse\fi\fi
\A@=-\A%@Make it positive
\fi
\ifdim\B@>0pt
  \ifnum\sm@extvp>0 \dimen@=\B@ \divide\dimen@\sm@extvp
  \edef\sm@dyp{\the\dimen@}%
  \else \sm@acceptablefalse\fi\fi
\ifdim\B@<0pt
  \ifnum\sm@extvm>0 \dimen@=\B@ \divide\dimen@\sm@extvm
  \edef\sm@dym{\the\dimen@}%
  \else \sm@acceptablefalse\fi\fi
\B@=-\B@
\fi
\ifs@acceptable
  %Compute the cost of the connector
  \dimen@=\sm@connlen\advance\dimen@\A@\advance\dimen@\B@
  % \W@{ \space Cost is \the\dimen@}%
  \ifdim\dimen@<\sm@bestcost
  \edef\sm@bestcost{\the\dimen@}%
  \edef\sm@bestconn{{\sm@dxp}{\sm@dyp}{\sm@dym}{\sm@segs}}%
  \fi
\fi
\xydef@\sm@showext#1{%
  \W@{#1\space hp=\sm@exthp,hm=\sm@extthm, vp=\sm@extvp
  ,vm=\sm@extvm,len=\sm@connlen}}
\% Process the next element in the list that describes the representative
\% %
\xydef@\sm@trycirclelist@i#1{%
  \def\@tmp{#1}%
  \ifx\@tmp\sm@nnil\let\sm@tryclcont=\relax\else
    \expandafter\sm@tryclcar\@tmp
  \fi
  \sm@tryclcont}
\xydef@\sm@tryclcar#1#2#3{%
% \W@{Trying one turn in #1\space orient #2\space out #3}%
%Compute deltax in R@p and deltay in U@p. Move them to center first.
\dimen@=\ifnum #2<0 -\fi\R@p
% Move them to exit
\count@=#3\count@=\the\count@ \Abfromdiag@
\advance\R@p -\B@ \advance\U@p \A@
% \xy@showdim{ After center}\
% Now update the extension points
\sm@computeext{#1}{#2}{#3}
%
\% Compute the extensions
\%\xydef@\sm@computeext#1#2#3{%
\% \W@{Computing extents for turn in #1\space orient #2\space out #3}\
\ifsm@firstseg
\sm@accumext#1%
\sm@firstsegfalse
\fi
\count@=#1\count@=\the\count@
\dimen@=\sm@connlen
% \W@{Before rounding count@=\the\count@, count@@=\the\count@@}%
\sm@roundcount@{#2}%
% \W@{After rounding count@=\the\count@, count@@=\the\count@@}%
\ifnum\count@=\count@@
\sm@advancecount@ 1{#2}% Advance by 90 degrees if odd. By 45 first
\advance\dimen@ 0.7854\R@% It was incremented with 1/8 circle
\ifnum\count@=#3% Check if done
  \else
    \sm@advancecount@ 1{#2}% Another 45
  \fi
\else
  \advance\dimen@ 0.7854\R@% It was incremented with 1/8 circle
\fi
\else
  \advance\dimen@ 0.7854\R@% It was incremented with 1/8 circle
\fi
\edef@\sm@connlen{\the\dimen@}% Save the connector length
\edef@\sm@segs{\sm@segs{\{#1\}{#2}{\the\count@}}}%Accum segment
\ifnum\the\count@=#3 \else
  \edef@\tmp@{\{\the\count@}{#2}{#3}}%Recursive call
  \expandafter@\sm@computeext\tmp@
\fi
%
% Accumulate the extensions
\%
\xydef@\sm@accumext#1{%
% \W@{Accumulating extent for #1}%
3.11. SMART PATH OPTION

% \sm@showext{Before accum:}%
\ifcase #1%
\or \count@@=\sm@extvm\advance\count@@ by1%
\edef\sm@extvm{\the\count@@}% 1 is V-
\or\or \count@@=\sm@exthp\advance\count@@ by1%
\edef\sm@exthp{\the\count@@}% 3 is H+
\or\or \count@@=\sm@extvp\advance\count@@ by1%
\edef\sm@extvp{\the\count@@}% 5 is V+
\or\or \count@@=\sm@exthm\advance\count@@ by1%
\edef\sm@exthm{\the\count@@}% 7 in H-
\fi
% \sm@showext{After accum:}%

% Draw a computed connector.
% 1,2,3,4 - dx+, dx-, dv+, dv- (lengths of straight connectors in each
% direction)
% 5 - a list of segments, each enclosed in a pair of braces.
\xydef@\sm@conndraw#1#2#3#4#5{%
\def\sm@contlist{\sm@drawseglist}%
\edef\sm@dxp{#1}%
\edef\sm@dyp{#3}%
\edef\sm@dxm{#2}%
\edef\sm@dym{#4}%Save deltas
\sm@drawseglist #5\sm@nil}
\xydef@\sm@drawseglist#1{% Get the first segment
\ifx #1\sm@nil \def\sm@contlist{}\else
\sm@drawseg #1\fi
\sm@contlist}

% Draw a segment
%
\xydef@\sm@drawseg#1#2#3{% A segment is CIRin+CIRorient+CIRout
% \W@{Drawing segment in=#1\space out=\space #3}\xy@showthe c{Seg start}%
\def\CIRin@@{#1}\def\CIRout@@{#3}%
\sm@straight%See if a straight line is needed here. Insert it and
%reset c if needed
\ifnum\CIRin@@=\CIRout@@ %
% Compute the center of the circle
\count@@=\CIRin@@
\dimen@@=\ifnum #2<0 -\fi\R@
\ABfromdiag\advance\X@c -\B@ \advance\Y@c \A@
% \xy@showthe c{Center of circle}%
% \xy@showdim{Before center}%
\ifnum #2>0 \def\CIRorient@@{\CIRacw@}%
\else \def\CIRorient@@{\CIRcw@}\fi
\drop@\literal@{\hbox\bgroup\cir@i}% Place the circle
% Now move the the end of the arc
\dimen@=\ifnum #2<0 -\fi\R@
\count@=\CIRout@@\count@=\count@%Wierd. If I remove this last assignm
% \count@ is not set
\ABfromdiag@
% \xy@showthe c{Before advance}%
% \xy@showdim{Before advance}%
\advance\X@c\B@ \advance\Y@c\A@
% \xy@showthe c{After circle}%
% \xy@showdim{After circle}%
\fi
\xydef@\sm@straight{%
\U@c=\z@\D@c=\U@c\L@c=\U@c\R@c=\U@c
\pfromc@ %Set p at the beginning of the segment
\ifcase \CIRin@@ \or% 1 is V-
  \A@=\sm@dym\advance \Y@c\A@ %Wierd. I have to use the temporary A@!!
\or\or % 3 is H+
  \A@=\sm@dxp\advance \X@c\A@
\or\or % 5 is V+
  \A@=\sm@dyp\advance \Y@c\A@
\or\or % 7 is H-
  \A@=\sm@dxm\advance \X@c\A@ \fi
\ifdim\X@c=\X@p\ifdim\Y@c=\Y@p\else \sm@stri \fi\else \sm@stri \fi
}
\xydef@\sm@connect@#1#{
\DN@##1{\connect@{#1}{##1}}\next@}
\xydef@\sm@drop@#1#{
\DN@##1{\drop@{#1}{##1}}\next@}
\xydef@\smconn#1#2#3{%Memorize the directions
\edef\sm@CIRin{#1}\edef\sm@CIRout{#2}\R@=#3\R@=\the\R@
\sm@conn}
\xyendinput

Bug: This code should probably be merged with the ‘arrow’ feature.

Finally the log.

% $Log: xysmart.doc,v $
% Revision 3.6  2011/03/14 20:14:00  krisrose
% Preparing for release 3.8.6.
3.11. SMART PATH OPTION

% Revision 3.5 2010/07/27 10:04:17 krisrose
% Lifted some documentation.
%
% Revision 3.4 2010/06/10 18:45:50 krisrose
% Reference to GPL by URL.
%
% Revision 3.3 2010/04/26 22:01:48 krisrose
% Documentation fixes (hyperref and other things).
%
% Revision 3.2 2010/04/16 06:06:52 krisrose
% Preparing for a new release...
%
% Revision 3.1 2010/04/13 08:10:26 krisrose
% Up to date with Kris’ development directory.
%
% Received from George.
## Syntax

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>\ar \langle arrow \rangle \langle path \rangle</td>
<td>make \langle arrow \rangle along \langle path \rangle</td>
</tr>
<tr>
<td>\langle arrow \rangle</td>
<td>\rightarrow \langle form \rangle* \langle variant \rangle \langle tip \rangle \langle conn \rangle \langle tip \rangle \langle conn \rangle \langle shape \rangle \langle dir \rangle \langle dimen \rangle \langle anchor \rangle \langle it \rangle \langle letter \rangle \langle space \rangle \langle connchar \rangle \langle dir \rangle \langle control point list \rangle \langle modifier \rangle \langle empty \rangle \langle direction \rangle \langle dist \rangle \langle control point list \rangle \langle control point list \rangle \langle control point list \rangle \langle control point list \rangle \langle control point list \rangle \langle control point list \rangle</td>
</tr>
<tr>
<td>\langle form \rangle</td>
<td>\rightarrow \langle variant \rangle \langle tip \rangle \langle conn \rangle \langle tip \rangle \langle conn \rangle</td>
</tr>
</tbody>
</table>

- \langle arrow \rangle has the \langle form \rangle's
- \langle variant \rangle of arrow
- build arrow^3.3m using \langle variant \rangle of a standard stem and \langle tip \rangle for the head
- build arrow^3.3m using \langle variant \rangle of \langle tip \rangle, \langle conn \rangle, \langle tip \rangle as arrow tail, stem, and head (in that order)
- change stem to the indicated \langle connchar \rangle
- change stem to the indicated \langle connchar \rangle
- add \langle (shape) \rangle to object \langle (modifier) \rangle^3.3r for all objects
- \langle (modifier) \rangle^3.3r for all objects
- slide arrow^3.3s the \langle (dimen) \rangle
- break each segment at \langle (anchor) \rangle with \langle (it) \rangle
- label each segment at \langle (anchor) \rangle with \langle (it) \rangle
- reverse meaning of above and below^3.3t
- \langle (variant) \rangle: plain, above, below, double, or triple
- \langle (tip) \rangle directional named as the sequence of \langle (tipchar) \rangle's
- \langle (tipchar) \rangle any \langle (dir) \ranglectional^3.3n
- \langle (connchar) \rangle directional named as the sequence of \langle (connchar) \rangle's
- \langle (connchar) \rangle any \langle (dir) \ranglectional^3.3n
- \langle (conn) \rangle directional named as the sequence of \langle (tipchar) \rangle's
- \langle (conn) \rangle any \langle (dir) \ranglectional^3.3n

### Figure 3.2: \langle arrow \rangle's.

![Arrow Diagram](image)

### Figure 3.3: Pasting diagram.

![Pasting Diagram](image)
### Syntax and Action

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>⟨twocell⟩</td>
<td>typeset ⟨2-cell⟩ with the ⟨switches⟩ and ⟨Arrow⟩</td>
</tr>
<tr>
<td>⟨2-cell⟩ → ..twocell</td>
<td>typeset two curved arrows</td>
</tr>
<tr>
<td>..uppertwocell</td>
<td>typeset upper curved arrow only</td>
</tr>
<tr>
<td>..lowertwocell</td>
<td>typeset lower curved arrow only</td>
</tr>
<tr>
<td>..compositemap</td>
<td>use consecutive straight arrows</td>
</tr>
<tr>
<td>⟨Arrow⟩ → {⟨tok⟩⟨text⟩}</td>
<td>specifies orientation and label</td>
</tr>
<tr>
<td>{⟨nudge⟩⟨text⟩}</td>
<td>adjust position, use default orientation</td>
</tr>
<tr>
<td>{⟨text⟩}</td>
<td>use default position and orientation</td>
</tr>
<tr>
<td>{⟨tok⟩*⟨object⟩}</td>
<td>use ⟨object⟩ as the label</td>
</tr>
<tr>
<td>{⟨nudge⟩*⟨object⟩}</td>
<td>use ⟨object⟩ in place specified by ⟨what⟩</td>
</tr>
<tr>
<td>⟨tok⟩ → ^</td>
<td>_</td>
</tr>
<tr>
<td>{⟨nudge⟩⟨text⟩}</td>
<td>no Arrow, default is clockwise</td>
</tr>
<tr>
<td>^</td>
<td>_</td>
</tr>
<tr>
<td>{⟨nudge⟩*⟨object⟩}</td>
<td>anti-/clockwise/double-headed/none</td>
</tr>
<tr>
<td>(switches) → (switch)⟨switches⟩</td>
<td>list of optional modifications</td>
</tr>
<tr>
<td>⟨switch⟩ → \empty</td>
<td>use defaults</td>
</tr>
<tr>
<td>^</td>
<td>⟨label⟩</td>
</tr>
<tr>
<td>_</td>
<td>⟨label⟩</td>
</tr>
<tr>
<td>{⟨nudge⟩}</td>
<td>set the curvature, based on ⟨nudge⟩ value</td>
</tr>
<tr>
<td>\empty</td>
<td>do not set the curved arrows</td>
</tr>
<tr>
<td>_</td>
<td>!</td>
</tr>
<tr>
<td>^ ⟨what⟩ { ⟨object⟩ }</td>
<td>use ⟨object⟩ in place specified by ⟨what⟩</td>
</tr>
<tr>
<td>⟨what⟩ → \empty</td>
<td>set curves using the specified ⟨object⟩</td>
</tr>
<tr>
<td>^</td>
<td>_</td>
</tr>
<tr>
<td>_</td>
<td>^</td>
</tr>
<tr>
<td>⟨label⟩ → ⟨text⟩ { ⟨nudge⟩ ⟨text⟩</td>
<td>set ⟨text⟩, displaced by ⟨nudge⟩</td>
</tr>
<tr>
<td>*⟨object⟩</td>
<td>set ⟨object⟩, displaced by ⟨nudge⟩</td>
</tr>
<tr>
<td>⟨nudge⟩ → &lt;⟨number⟩&gt;</td>
<td>use ⟨number⟩ in an appropriate way, e.g., to position object or label along a fixed axis</td>
</tr>
<tr>
<td></td>
<td>\empty</td>
</tr>
</tbody>
</table>

---

**Figure 3.4**: ⟨twocell⟩s
### Syntax

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>\xygraph{⟨graph⟩}</td>
<td>typeset ⟨graph⟩</td>
</tr>
<tr>
<td>⟨graph⟩ → ⟨step⟩*</td>
<td>interpret ⟨step⟩s in sequence</td>
</tr>
<tr>
<td>⟨step⟩ → ⟨node⟩</td>
<td>move 3.6a to the ⟨node⟩</td>
</tr>
<tr>
<td>⟨node⟩</td>
<td>new node ⟨move⟩d relative to current</td>
</tr>
<tr>
<td>&amp;</td>
<td>\ \</td>
</tr>
<tr>
<td>&quot;(id)&quot;</td>
<td>previously saved 3.6e node</td>
</tr>
<tr>
<td>?</td>
<td>currently mapped 3.6c node</td>
</tr>
<tr>
<td>⟨node⟩ (it)</td>
<td>⟨node⟩ with ⟨it⟩ typeset and saved 3.6e there</td>
</tr>
<tr>
<td>⟨node⟩ = &quot;(id)&quot;</td>
<td>⟨node⟩ saved 3.6e as &quot;(id)&quot;</td>
</tr>
<tr>
<td>⟨node⟩ ! ⟨escape⟩</td>
<td>augment node with material in another mode</td>
</tr>
<tr>
<td>⟨move⟩ → ⟨hop⟩*</td>
<td>⟨hop⟩s 3.6f (dulr) from current node</td>
</tr>
<tr>
<td>⟨hop⟩* (place) ⟨move⟩</td>
<td>do ⟨hop⟩s 3.6f but use its ⟨place⟩ and ⟨move⟩ again</td>
</tr>
<tr>
<td>⟨list⟩ → ⟨graph⟩, ⟨list⟩</td>
<td>list of subgraphs 3.6c</td>
</tr>
<tr>
<td>⟨escape⟩ → { ⟨pos⟩ ⟨decor⟩ }</td>
<td>perform ⟨pos⟩ ⟨decor⟩ 3.6g</td>
</tr>
<tr>
<td>M ⟨matrix⟩</td>
<td>insert ⟨matrix⟩ 3.6h</td>
</tr>
<tr>
<td>P ⟨polygon⟩</td>
<td>insert ⟨polygon⟩ 3.6i</td>
</tr>
<tr>
<td>E ⟨ellipse⟩</td>
<td>insert ⟨ellipse⟩ 3.6i</td>
</tr>
<tr>
<td>~ ⟨setup⟩</td>
<td>setup parameters 3.6j</td>
</tr>
</tbody>
</table>

### Figure 3.5: ⟨graph⟩s

| n | \(\sin \frac{n\pi}{n}\) | \(\cos \frac{n\pi}{n}\) | \(\sin \frac{2\pi}{n}\) | \(\cos \frac{2\pi}{n}\) | \(\sin \frac{3\pi}{n}\) | \(\cos \frac{3\pi}{n}\) | \(\sin \frac{4\pi}{n}\) | \(\cos \frac{4\pi}{n}\) | \(\sin \frac{5\pi}{n}\) | \(\cos \frac{5\pi}{n}\) |
|---|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| 0 | 0               | 1               | 0               | 1               | 0               | 1               | 0               | 1               | 0               | 1               | 0               |
| 1 | 0               | -1              | 0               | 1               | 0               | -1              | 0               | 1               | 0               | 1               | -1              |
| 2 | 1               | 0               | 0               | -1              | 0               | 1               | 1               | 0               | 1               | 0               | 1               |
| 3 | .8660254        | .5              | -.8660254       | -.5             | 0               | -1              | -.8660254       | -1              | -.8660254       | .5              | -.8660254       |
| 4 | .70710678       | .70710678       | 1               | 0               | .70710678       | -.70710678     | 0               | -.70710678     | -.70710678     | 0               | -.70710678     |
| 5 | .587785         | .809017         | .951057         | .309017         | .951057         | -.809017       | .587785         | -.809017       | -.809017       | 0               | -.809017       |
| 6 | .5              | .8660254        | .8660254        | .5              | 1               | 0               | -.8660254       | -.5             | -.8660254       | .5              | -.8660254       |
| 7 | .433884         | .900969         | .781831         | .62349          | .974928         | .222521        | .974928         | -.222521       | -.222521       | .781831         | .62349          |
| 8 | .382683         | .92388          | .70710678       | .70710678       | .92388          | .382683        | 1               | 0               | .92388          | -.382683        | .92388          |
| 9 | .34202          | .939693         | .642788         | .766044         | .8660254        | .5              | .984808         | .173648        | .984808         | -.173648        | .984808         |
| 10| .309017         | .951057         | .587785         | .809017         | .309017         | .951057        | .951057         | .309017        | .951057         | 1               | .309017         |
| 11| .281733         | .959493         | .540641         | .841254         | .75575          | .654861        | .909632         | .415415        | .909632         | -.415415        | .909632         |
| 12| .258819         | .965926         | .5              | .8660254        | .70710678       | .70710678      | .8660254        | .5              | .8660254        | .258819         | .965926         |

### Figure 3.6: Trigonometry tables for Polygon vertices.
### 3.11. SMART PATH OPTION

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>⟨knot-piece⟩ → ⟨piece⟩⟨scale⟩⟨knot-labels⟩</td>
<td>interpret knot-piece</td>
</tr>
<tr>
<td>⟨piece⟩ → ⟨crossing⟩</td>
<td>piece is a crossing³.¹⁰ᵃ or a join³.¹⁰ˡ</td>
</tr>
<tr>
<td>⟨scale⟩ → ⟨empty⟩</td>
<td>invert or scale the knot piece³.¹⁰ᵇ;</td>
</tr>
<tr>
<td></td>
<td>[ ⟨(num)⟩ ] alter size and shape³.¹⁰ᶜ using the ⟨pos⟩s</td>
</tr>
<tr>
<td>⟨knot-labels⟩ → ⟨empty⟩</td>
<td>arrowtips at ends, aligned with orientation</td>
</tr>
<tr>
<td></td>
<td>⟨(where)⟩⟨(what)⟩⟨knot-labels⟩ list³.¹⁰ᵏ of arrowtips, breaks and labels³.¹⁰ᵉ</td>
</tr>
<tr>
<td></td>
<td>@⟨(adjust)⟩⟨knot-labels⟩ adjust hole³.¹⁰ᵈ position for a ⟨crossing⟩;</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>⟨knot-tips⟩ → ==</td>
<td>==!</td>
</tr>
<tr>
<td></td>
<td>=&lt;</td>
</tr>
<tr>
<td>⟨where⟩ →</td>
<td>←</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>⟨adjust⟩ → ⟨+(num)⟩</td>
<td>⟨-(num)⟩ adjustment³.¹⁰ᵏ from current value of parameter</td>
</tr>
<tr>
<td></td>
<td>⟨(num)⟩</td>
</tr>
<tr>
<td>⟨what⟩ →</td>
<td>&gt;</td>
</tr>
<tr>
<td></td>
<td>\knothole</td>
</tr>
<tr>
<td></td>
<td>⟨(text)⟩ set³.¹⁰حاسبة ⟨text⟩ as label, using \labelstyle</td>
</tr>
<tr>
<td></td>
<td>{⟨object⟩} drop ⟨object⟩³.¹⁰حاسبة</td>
</tr>
<tr>
<td></td>
<td>⟨(anchor)⟩⟨it⟩⟩ ⟨break⟩ or label³.¹⁰حاسبة as on an ⟨arrow⟩</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.7: ⟨knot-piece⟩ construction set.
CHAPTER 3. FEATURES

Figure 3.8: Knot crossings with orientations and label positions.

Figure 3.9: Knot joins, with orientations, labels, and shifts.
Chapter 4

Drivers

This chapter describes the driver options that support customisation of the \texttt{Xy-pic} output to particular output devices. The first section describes the actual drivers, the second a set of modifications specially applicable to \texttt{POSTSCRIPT} printing.

4.1 Support for Specific Drivers

4.2 \texttt{dvidrv} driver

\textbf{Vers. 3.7 by Ross Moore} (ross.moore@mq.edu.au)

This driver provides support for the “\texttt{emtex} \texttt{\special}” commands, when using one of the standard dvi-drivers: \texttt{dvidot}, \texttt{dvihplj}, \texttt{dvimsp}, \texttt{dviscr} or \texttt{dvivik}, that come with Eberhard Mattes’ \texttt{em-TeX} distribution.

Header:

\begin{verbatim}
%% $Id: xydvidrv.doc,v 3.7 2011/03/14 20:14:00 krisrose Exp $
%% Xy-pic ``dvidrv-driver'' option.
%% Copyright (c) 1995-1996 Ross Moore <ross.moore@mq.edu.au>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-2011 Kristoffer H. Rose <krisrose@tug.org>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, see http://www.gnu.org/licenses/.
\end{verbatim}
Driver installation  Supported \special effects are...

- em-T\TeX line-drawing \specials.
- variable line-widths

The end & Log

\xyendinput
4.3 DVIPS driver

Vers. 3.9 by Ross Moore (ross.moore@mq.edu.au)

This driver provides support for all extensions when using the DVIPS driver by Tomas Rokicki [12]. It has been tested with dvips version 5.55a and dvipsk version 5.58f.

Header:

```latex
\ifx\xyloaded\undefined \input xy \fi
\xyprovide{dvips}{DVIPS driver}{\stripRCS$Revision: 3.9$}\
{Ross Moore}{ross.moore@mq.edu.au}\
{Mathematics Department, Macquarie University, NSW~2109, Australia}
```

```latex
\xydef@\dvips@@#1{\dvimessage@{dvips}{#1}}
\newdriver{\xydef@dvips@@1{\dvimessage@{dvips}{\{#1}}}\
\xyaddsupport{color}\dvipsColor@@
\xyaddsupport{crayon}\dvipsCrayola@@
\xyaddsupport{ps}\dvipsPS@@
\xyaddsupport{rotate}\dvipsRotate@@
\xyaddsupport{line}\dvipsLine@@
```
Driver installation  Supported \special effects are...

- colour, using direct color specials and PostScript.
  DVIPS uses the colour-stack method, due to Tomas Rokicki.

If neither support file colordvi.tex nor colordvi.sty can be found, then the normal colour support will not be available. However the colour support for the PostScript back-end can still be used.

- crayon colours.
  The 68 colours that dvips recognizes by name are not loaded unless the crayon option has been requested.
Although any order of loading options: ps, dvips, color and crayon produces the desired result visually, the POSTSCRIPT code can be different with different loading order. The most easily readable is obtained when crayon is requested last.

- **POSTSCRIPT** back-end.

```
\let\newxycolor@=\newxyDVIPScolor@ \dvipsInclude@{color.pro}\
% \xyputorelse@{colordvi.tex}\
% \{\NoDVIPScolordvi@\let\newxycolor@=\prevxycolor@\% 
\installCrayolaColors@
\let\newxycolor@=\prevxycolor@ }%
```

dvips has three kinds of POSTSCRIPT \special which can be used. These may be used with various features other than just the POSTSCRIPT backend.
All the PostScript definitions passed to dvips by reading the dictionary file are stored in a PostScript dictionary which it defines, called SDict. This must be the current dictionary whenever an Xy-ps command is to be executed. This will be the case whenever the \special{ commands} or \special{! commands} forms are used, but not when the \special{ps:: commands} form is used. Thus the \dvipsSpecial@ macro includes code to open SDict and close it when finished; so also does the \dvipsRaw@ type, since this is used with Xy-pic PostScript operators whose definitions have been placed within SDict.

The \texttt{currentpoint} is defined for the “raw” specials.

\begin{verbatim}
\xydef@dvi\psCurrent@{\let\psxyCurrent@@=\dvi\psCurrent@@}
\xydef@dvi\psCurrent@@{\psraw@@{xyp}}
\end{verbatim}

The PostScript operator called \texttt{xyp} is defined in \texttt{xyps-ps} for storing the location read from \texttt{currentpoint}.

This installs the PostScript backend.

\begin{verbatim}
\xydef@dvi\ps\{\dvi\ps\types@
\let\dvi\ps@@=\psspecials@true
\let\psUnload@@=\dvi\ps\unload
\let\install\xyps@@=\relax \install\xyps@ }
\end{verbatim}

• rotated/scaled diagrams and text, using PostScript.

\begin{verbatim}
\xydef@dvi\ps\rotate@@{\dvi\ps\rotate@}
\xydef@dvi\ps\rotate@{\dvi\ps\types@
\expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{
\xyinputorelse@{xyps-r}{\xydrivernoload@{ps-r}}%}
\dvi\ps\Current@ \dvi\ps\RotScale@ \install\psrotscale@ }
\else\DN@{\xydriverfail@{rotations are}Un\load\rotate@\relax}\fi
\next@ }
\xydef@dvi\ps\rotscale@@{\dvi\ps\rotscale@}
\xydef@dvi\ps\rotscale@{\dvi\ps\types@
\expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{
\xyinputorelse@{xyps-l}{\xydrivernoload@{ps-l}}%}
\dvi\ps\Current@ \dvi\ps\rotscale@ \install\psline@ }
\else\DN@{\xydriverfail@{line-widths are}Un\load\line@\relax}\fi
\next@ }
\end{verbatim}

• variable line-widths and poly-lines, using PostScript.

\begin{verbatim}
\xydef@dvi\ps\line@@{\dvi\ps\line@}
\xydef@dvi\ps\line@{\dvi\ps\types@
\expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{
\xyinputorelse@{xyps-f}{\xydrivernoload@{ps-f}}%}
\dvi\ps\Current@ \dvi\ps\line@ \install\psframes@ }
\else\DN@{\xydriverfail@{line-widths are}Un\load\line@\relax}\fi
\next@ }
\end{verbatim}

• extra frames and fills, using PostScript.

\begin{verbatim}
\xydef@dvi\ps\frames@@{\dvi\ps\frames@}
\xydef@dvi\ps\frames@{\dvi\ps\types@
\expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{
\xyinputorelse@{xyps-f}{\xydrivernoload@{ps-f}}%}
\dvi\ps\Current@ \dvi\ps\frames@ \install\psframes@ }
\else\DN@{\xydriverfail@{frames and fills are}Un\load\psframes@\relax}\fi
\next@ }
\end{verbatim}
4.3. **DVIPS DRIVER**

- patterns and tiles, using **PostScript**.

```latex
\def\dvipsTiles@@{\dvipsTiles@}
\def\dvipsTiles@{\dvipsPStypes@
\after\if\csname xyps-psloaded\endcsname\empty\DN@{%\dvipsdrivernoload@{ps-t}!installPSpatterns@ }%\else\DN@{\xydriverfail@{Patterns are}\UnloadPSpatterns@}\fi\next@ }
```

- **TPIC** drawing commands.

```latex
\def\dvipsTPIC@@{\dvipsTPIC@}
\def\dvipsTPIC@{\setupxyTPIC@}
```

- **em-T\TeX** drawing commands.

```latex
\def\dvipsEM@@{\dvipsEM@}
\def\dvipsEM@{\setupxyEMTeX@}
```

- lu tips.

When the **dvips** option is chosen, those fonts aren’t used, instead, the arrowheads are drawn in PostScript; this adds the appropriate definition to the PostScript prologue.

```latex
\def\dvipsTIPS@{\PSdict@@{/XYdict where pop begin XYdict begin}%\PSdict@@{%lu{0.04 0.0 -0.16 0.125 -0.11 0.0}% ie ((0,0) -- (-1/5,1/8) -- (-1/5*3/4,0) -- cycle) shifted 0.04*right 1 1 6{pop xyfsize mul 6 1 roll}for lineto lineto lineto closepath fill 0 0 0 0 0 0}def}%\PSdict@@{end end}%
```

**The end & Log**

```latex
\endinput
```

% $Log: xydvips.doc,v $ % Revision 3.9 2011/03/14 20:14:00 krisrose % Preparing for release 3.8.6. % % % Revision 3.8 2010/06/10 18:45:50 krisrose % Reference to GPL by URL. % % % Revision 3.7 2010/05/06 17:46:30 krisrose % Ross Moore’s e-mail address updated. % Many obsolete files degraded to Historic.
4.4 DVITOPS driver

Vers. 3.7 by Ross Moore (ross.moore@mq.edu.au)

This file provides support for the DVITOPS driver by James Clark. As of September 1995, it has not been fully tested.

Header:

%% $Id: xydvitops.doc,v 3.7 2011/03/14 20:14:00 krisrose Exp $
%% Xy-pic ``DVITOPS-driver'' option.
%% Copyright (c) 1995-1996 Ross Moore <ross.moore@mq.edu.au>
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-2011 Kristoffer H. Rose <krisrose@tug.org>
%% The Xy-pic package is free software; you can redistribute it and/or modify
4.4. DVI TOPS DRIVER

\%\% it under the terms of the GNU General Public License as published by the
\%\% Free Software Foundation; either version 2 of the License, or (at your
\%\% option) any later version.
\%\%
\%\% The Xy-pic package is distributed in the hope that it will be useful, but
\%\% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
\%\% or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License
\%\% for more details.
\%\%
\%\% You should have received a copy of the GNU General Public License along
\%\% with this package; if not, see http://www.gnu.org/licenses/.
\%\%
\%\% 
\ifx\xyloaded\undefined \input xy \fi
\\xyprovide{dvitops}{DVITOPS driver}{\stripRCS$Revision: 3.7 $} %
{Ross Moore}{ross.moore@mq.edu.au} %
{Mathematics Department, Macquarie University, NSW\textasciitilde 2109, Australia}

\xydef@\dvitops@#1\{\dvimessage@{DVITOPS}{#1}\}
\newdriver{%
\xyaddsupport{color}\dvitopsColor@@
\xyaddsupport{crayon}\dvitopsCrayon@@
\xyaddsupport{ps}\dvitopsPS@@
\xyaddsupport{rotate}\dvitopsRotate@@
\xyaddsupport{line}\dvitopsLine@@
\xyaddsupport{frame}\dvitopsFrames@@
\xyaddsupport{tile}\dvitopsTiles@@
% \xyaddsupport{tpic}\dvitopsTPIC@@
\xyaddsupport{cmtip}\dvitopsCMTIP@@
\xyaddsupport{cmtip}\relax
}

\textbf{Supported} \texttt{\special} \texttt{effects are...}
\begin{itemize}
\item colour, using direct color specials for \texttt{gray}, \texttt{rgb} and \texttt{hsb} colour models; and \texttt{POSTSCRIPT} colour
within diagrams;
\item \texttt{DVITOPS} has a method for using colours locally, however it only colours “regions”, which cannot
be nested. To use this within \texttt{Xy-pic} diagrams would be just asking for trouble.
\end{itemize}
\xydef@\dvitopsColor@@\{\dvitopsColor@\}
\xydef@\dvitopsColor@\{\dvitopsPStypes@
\let\xylocalColor@=\xyDVITOPScolor@
\let\xycolor@push\xycolor@push@@
\def\xycolor@\{\xycolor@@\let\xycolorwarning@=.relax
\expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{% 
\xyinputorelse@{xyps-c}{\xydrivernoload@{ps-c}}\xycatcodes 
\installPScolor@ }\else\DN@{}}\fi \next@
\ifx\xylocalColor@=\xyDVITOPScolor@ \let\xycolor@push@xcolor@push@@
\def\xycolor@\{\xycolor@\let\xycolorwarning@=.relax
\expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{% 
\xyinputorelse@{xyps-c}{\xydrivernoload@{ps-c}}\xycatcodes 
\installPScolor@ }\else\DN@{}}\fi \next@
\ifx\xycrayolaalomaded\empty \DN@{\xystandardcolors@\dvitopsCrayon@{}}% 
\else\let\next@=\xystandardcolors@ 
\fi \next@
\let\dvitopsColor@=.relax
\let\UnloadColor@@=\NoDVITOPScolor@
\%
\xydef@\NoDVITOPScolor@{%
\def\xycolor@@={\xyNoColor@}
\let\xycolor@@=\xyNoColor@
\let\xylocalColor@=\xyNoColor@@
\let\xylocalColorwarning@=\xylocalColorwarning@@
\let\dvitopsColor@@=\dvitopsColor@
}
\xydef@\xyDVITOPScolor@{%bgroup \DN@##1##2{\egroup}%
\ifInvisible@\else\ifHidden@\else\DNii@{\no@@}\if\nextii@\Connect@@
\else\DN@{\xyDVITOPScolor@i}\fi\fi\fi \next@ }
\xydef@\xyDVITOPScolor@i#1#2{\checkXyStyle@
\DN@{#2}\ifnext@\empty\def\tmp@##1{\DN@{\xycolor@begin##1\xycolor@end{#1}}}%
\else\DNii@{gray}\ifnext@\nextii@
\def\tmp@##1{\DN@{\xycolor@begin##1\xycolor@end{gray}{#1}}}%
\else\DNii@{rgb}\ifnext@\nextii@
\def\tmp@##1{\DN@{\xycolor@begin##1\xycolor@end{rgbcolor}{#1}}}%
\else\DNii@{hsb}\ifnext@\nextii@
\def\tmp@##1{\DN@{\xycolor@begin##1\xycolor@end{hsbcolor}{#1}}}%
\else \DN@{\xywarning@{DVITOPS cannot support color: #2 #1}}%
\fi\fi\fi\fi
\toks@={\egroup \let\xy@style@=\relax \def\Drop@@}%
\expandafter\tmp@\expandafter{\Drop@@}%
\expandafter\addtotoks@\expandafter{\expandafter{\next@}\def\Connect@@}%
\expandafter\tmp@\expandafter{\expandafter{\next@}}%
\the\toks@ }%
\xydef@\xycolor@begin{\dvitopsBegin@{color}}
\xydef@\xycolor@end{\dvitopsEnd@{color}\dvitopsColor@x}

\list{\item crayon colours.
\item The 68 colours are those that dvips recognizes by name, thanks to Tomas Rokicki.
\item PostScript back-end.
\item DVITOPS has two kinds of \special which can be used; ‘inline’ and ‘prolog’ for including header files. Also there are the ‘begin’ and ‘end’ types for defining regions.}
\%
Rotations and transformations use a different mechanism, so \texttt{\xyPScurrpt@@} can be set to \texttt{\relax}.

This installs the PostScript backend.
• rotated/scaled diagrams and text, using DVITOPS specials; however these may not be nested.

• variable line-widths and poly-lines, using POSTSCRIPT.

• extra frames and fills, using POSTSCRIPT.
patterns and tiles, using PostScript

- TPIC drawing commands.

The end & Log

% $Log: xydvitops.doc,v $%
% Revision 3.7 2011/03/14 20:14:00 krisrose
% Preparing for release 3.8.6.
%
% Revision 3.6 2010/06/10 18:45:50 krisrose
% Reference to GPL by URL.
%
% Revision 3.5 2010/05/06 17:46:30 krisrose
% Ross Moore’s e-mail address updated.
% Many obsolete files degraded to Historic.
%
% Revision 3.4 2010/04/16 06:06:52 krisrose
% Preparing for a new release...
%
% Revision 3.3 1996/12/18 09:14:36 ross
% checked in with -k by krisrose at 1996/12/18 14:17:11
%
% Revision 3.3 1996/12/18 09:14:36 ross
% minor improvements to file-loading commands
%
% Revision 3.2 1995/09/19 18:21:41 ross
% Bug fix release.
%
% Revision 3.1 1995/09/05 20:28:57 ross
% Releasing version 3.1!
%
% NEW for version 3.
CHAPTER 4. DRIVERS

4.5 OzTeX driver

Vers. 3.7 by Ross Moore (ross.moore@mq.edu.au)

This driver provides the necessary interface to support the POSTSCRIPT back-end and other POST-SCRIPT effects when using the DVI driver of versions 1.8+ of OzTeX by Andrew Trevorrow. Earlier versions of OzTeX should instead use the driver option \xyoption{17oztex}.

Effects such as colour, line-thickness and rotated or scaled diagrams are only partially supported in that the effects cannot be applied to any text or symbols placed using fonts. This is due to the nature of OzTeX (driver), whose optimization of the placement of font-characters precludes the applicability of such effects. Furthermore the POSTSCRIPT dictionary must be available in a file called global.ps or appended to the OzTeXdict.pro. However with version 1.8 and later of OzTeX, there is the alternative of using the dvips (driver), which does support all the POSTSCRIPT effects available in XY-pic.

Note: To use XY-pic effectively with OzTeX requires changing several memory parameters. In particular a ‘Big-TEx’ is needed, along with an increase in the pool_size parameter. Explicit instructions are contained in the file INSTALL.OzTeX of the XY-pic distribution.

Header:

```
%% $Id: xyozt2ex.doc,v 3.7 2011/03/14 20:14:00 krisrose Exp $
%%
%% Xy-pic ‘OzTeX’ driver interface option.
%% Copyright (c) 1995-1996 Ross Moore <ross.moore@mq.edu.au>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-2011 Kristoffer H. Rose <krisrose@tug.org>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, see http://www.gnu.org/licenses/.
%%
\ifx\xyloaded\undefined \input xy \fi
\xyprovide{oztex}{OzTeX driver}{\stripRCS$Revision: 3.7 $}\
{Ross Moore}<ross.moore@mq.edu.au>\
{Mathematics Department, Macquarie University, NSW~2109, Australia}
```

---

1OzTeX is a shareware implementation of TeX for Macintosh available from many bulletin boards and ftp sites; v1.5 and earlier versions were freeware. Email contact: (akt@kagi.com).
\newdriver{
\xyaddsupport{color}\oztexColor@@
\xyaddsupport{crayon}\oztexCrayon@@
\xyaddsupport{ps}\oztexPS@@
\xyaddsupport{rotate}\oztexRotate@@
\xyaddsupport{line}\oztexLine@@
\xyaddsupport{frame}\oztexFrames@@
\xyaddsupport{tile}\oztexTiles@@
\xyaddsupport{cmtip}\relax
}

Supported \special effects are...

- colour, using \texttt{POSTSCRIPT}, but not of font-characters.

\xydef@\oztexColor@@{\OzTeXColours@}
\xydef@\OzTeXColours@{\OzTeXPStypes@}
\let\xylocalColor@=\xyNoColor@@ \def\xycolor@{\xycolor@@}\
\expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{%\XYpredict@
\xyinputorelse@{xyps-c}{\xydrivernoload@{ps-c}}%\installPScolor@ }\else\DN@{}\fi \next@
\ifi\xycrayolaloaded\empty \DN@{\ystandardcolors@\OzTeXCrayon@@{}}%\else\let\next@=\ystandardcolors@\fi \next@
\let\xycolorwarning@=\xycolorwarning@@
\let\oztexColor@@=\OzTeXColours@
\let\UnloadColor@@=\UnloadColor@

- crayon colours, similarly restricted.

The 68 colours are those that \texttt{dvips} recognizes by name, thanks to Tomas Rokicki.

\xydef@\OzTeXCrayon@@{\installCrayolaColors@}

- \texttt{POSTSCRIPT} back-end.

\xydef@\oztexPS@@{\oztexPS@}
\xydef@\oztexPSunload@{\UnloadPS@}
\def\xysdictname{}\UsePSdict@false
\let\installxyps@=\relax
\let\UnloadPS@=\UnloadPS@
\let\xysshapes@true
\let\oztexPS@@=\oztexPS@ }
\xydef@\OzTeXSpecial@#1{\special{empty.ps \{#1\}xy}}
\xydef@\OzTeXMacro@#1{\special{empty.ps \{#1\}xy}}
The PostScript operators @bxy and @exy are defined in xyps-ps for bypassing the mechanism used in OzTeX to isolate code passed in \special commands. Although it is possible (albeit difficult) to devise PostScript to deduce the current location for use with the “raw” specials, this information cannot be used to do rotations and scaling as with other ⟨driver⟩s. Hence the value of \xyPScurrpt@@ is taken to be empty.

This avoids dumping the dictionary into the dvi-file, making sure that instead it goes into a file, specified by \xyPSdictname, which defaults to global.ps.

The parameters #1#2#3 of \xyOzTeXheader gobble tokens that are unnecessary with OzTeXX.

- variable line-widths and poly-lines, using PostScript.
4.5. **OTEX DRIVER**

- extra frames and fills, using **POSTSCRIPT**.

\begin{Verbatim}
\def\oztexFrames@@{\OzTeXFrames@}
\def\OzTeXFrames@{\OzTeXPStypes@
\expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{\%
\xyinputorelse@{xyps-f}\{\xydrivernoload@{ps-f}\}installPSframes@ }%
\else\DN@{\xydriverfail@{frames are}\UnloadPSFrames@\relax}fi
\next@ }
\end{Verbatim}

- patterns and tiles, using **POSTSCRIPT**.

\begin{Verbatim}
\def\oztexTiles@@{\OzTeXTiles@}
\def\OzTeXTiles@{\OzTeXPStypes@
\expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{\%
\xyinputorelse@{xyps-t}\{\xydrivernoload@{ps-t}\}installPSpatterns@
\xystandardpatterns@ }%
\else\DN@{\xydriverfail@{Patterns are}\UnloadPSpatterns@\relax}fi
\next@ }
\end{Verbatim}

- rotated/scaled diagrams and text, recognised but not supported.

In fact OzTEX cannot support rotations and scaling. Instead we go through all the motions
of providing **POSTSCRIPT** support, then set a flag that causes a limited number of warning
messages to be generated as requests are made for these effects.

\begin{Verbatim}
\def\oztexRotate@@{\OzTeXRotate@}
\def\OzTeXRotate@{\OzTeXPStypes@
\expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{\%
\xyinputorelse@{xyps-r}\{\xydrivernoload@{ps-r}\}%
\OzTeXRotScale@ \installPSrotscale@ \xyPSshapes@false }%
\else\DN@{\xydriverfail@{rotations are}\UnloadRotate@\relax}fi
\next@ }
\end{Verbatim}

This never worked properly. It requires too much hacking at Trevorrow’s code for little gain,
since text-rotation can never work anyway. Use the **dvips** option instead!

\begin{Verbatim}
\def\ozTeXRotScale@{%
def\xyrot@start(##1){%{xyp xyt ##1\space xyr}{OzXy begin gsave}}%
def\xyscale@start(##1,##2){%{xyp xyt ##1\space##2\space xys}{OzXy begin gsave}}%
def\xyrot@end{grestore end}
def\xyscale@end{grestore end}}%
\let\ozTeXRotScale@=\xyRotScale@@
\end{Verbatim}

The default code will just do nothing as far as the output is concerned...

\begin{Verbatim}
\def\OzTeXRotScale@{%
\def\xyrot@start(#1){%{xyp xyt #1\space xyr}{OzXy begin gsave}}%
\def\xyscale@start(#1,#2){%{xyp xyt #1\space#2\space xys}{OzXy begin gsave}}%
\def\xyrot@end{grestore end}\def\xyscale@end{grestore end}}%
\end{Verbatim}

The end & Log

\begin{Verbatim}
\let\OzTeXRotScale@=\xyRotScale@@
\end{Verbatim}

\begin{Verbatim}
xyendinput
4.6 OzTeX v1.7 driver

Vers. 3.8 by Ross Moore (ross.moore@mq.edu.au)

This option provides the necessary interface to support the PostScript back-end and other PostScript effects when using the DVI driver of version 1.7 of OzTEx by Andrew Trevorrow. Later versions of OzTEx should instead use the driver option \xyoption{oztex}. Upgrading to version 1.9+ of OzTEx is recommended.

Does not support rotations, scaling and coloured text within diagrams and the PostScript dictionary must be available in a file called global.ps.

Note: To use Xy-pic effectively with OzTEx requires changing several memory parameters. In particular a ‘Big-TEx’ is needed, along with an increase in the pool_size parameter. Explicit instructions

\footnote{OzTEx is a shareware implementation of \TeX for Macintosh available from many bulletin boards and ftp sites; v1.5 and earlier versions were freeware. Email contact: (akt@kagi.com).}
are contained in the file INSTALL.OzTeX of the XY-pic distribution.

**Header:**

```latex
\%% $Id: xy17oztexpol.doc,v 3.8 2011/03/14 20:14:00 krisrose Exp $
\%
\%
%% Xy-pic \```
```
%% driver interface option.
%% Copyright (c) 1995-1996 Ross Moore <ross.moore@mq.edu.au>
%%
%% This file is part of the XY-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-2011 Kristoffer H. Rose <krisrose@tug.org>
%%
%% The XY-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The XY-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, see http://www.gnu.org/licenses/.
%%
%\ifx\xyloaded\undefined \input xy \fi
\xyprovide{17oztexpol}{OzTeX v1.7 driver}{\stripRCS$Revision: 3.8 $}%
{Ross Moore}{ross.moore@mq.edu.au}%
{Mathematics Department, Macquarie University, NSW~2109, Australia}

%\def@\OzTeX@@#1{\dvimessage@{OzTeX v1.7}{#1}}
%newdriver{%
%\xyaddsupport{color}\oldoztexpolColor@@
%\xyaddsupport{crayon}\oldoztxCrayon@@
%\xyaddsupport{ps}\oldoztexpolPS@@
%\xyaddsupport{rotate}\oldoztexpolRotate@@
%\xyaddsupport{line}\oldoztexpolLine@@
%\xyaddsupport{frame}\oldoztexpolFrames@@
%\xyaddsupport{tile}\oldoztexpolTiles@@
%\xyaddsupport{cmtip}\relax
%
\Supported \special effects are...

\item colour, using \texttt{POSTSCRIPT}, but not of font-characters.

%\def\oldoztexpolColor@@{\oldoztxColours@}
%\def\oldoztexpolPS@@{\oldoztexpolPStypes@}
%\let\xylocalColor@=\xyNoColor@@ \def\xycolor@{\xycolor@@}%
```
crayon colours, similarly restricted.

The 68 colours are those that dvips recognizes by name, thanks to Tomas Rokicki.

\xydef@\oldOzTeXCrayon@@{\installCrayolaColors@}

• PostScript back-end.

\xydef@\oldoztexPS@@{\oldoztexPS@}
\def\xyPSdictname{}\UsePSdict@false
\let\installxyps@@=\relax
\let\UnloadPS@@=\UnloadPS@
\let\xyPSshapes@true
\let\oldoztexPS@@=\oldoztexPS@

\xydef@\oldOzTeXSpecial@#1{\special{empty.ps \{#1\}xy}}
\xydef@\oldOzTeXMacro@#1{\special{empty.ps \{#1\}xy}}
\xydef@\oldOzTeXDict@#1{\special{empty.ps @obxy\{#1\}stopped pop @eocyx}}
\xydef@\oldOzTeXinclude@#1{\special{empty.ps @obxy #1 @eocyx}}
\xydef@\oldOzTeXraw@#1{\special{empty.ps @boxy #1 @eocyx}}

\let\PSspecialdict@@=\oldOzTeXDict@
\oldOzTeX@@{PostScript}
The PostScript operators @boxy and @e oxy are defined in xyps-ps for bypassing the mechanism used in \texttt{OzTEX} to isolate code passed in \texttt{\special} commands. Although it is possible (albeit difficult) to devise PostScript to deduce the current location for use with the “raw” specials, this information cannot be used to do rotations and scaling as with other (driver)s. Hence the value of \texttt{\xyPScurrpt@@} is taken to be empty.

This avoids dumping the dictionary into the dvi-file, making sure that instead it goes into a file, specified by \texttt{\xyPSdictname}, which defaults to \texttt{global.ps}.

The parameters #1#2#3 of \texttt{\xyOzTeXheader} gobble tokens that are unnecessary with OzTEX.

- variable line-widths and poly-lines, using PostScript.
- extra frames and fills, using PostScript.
patterns and tiles, using PostScript.

\newcommand{\oldOzTeXTiles}{\oldOzTeXTiles}
\newcommand{\oldOzTeXRotate}{\oldOzTeXRotate}
\newcommand{\xyPSTypes}{\xydrivernoload{ps-t}\installPSPatterns\xystandardpatterns}
\newcommand{\xyRotScale}{\xydrivernoload{ps-r}\installPSrotscale\xyPSshapesfalse}
\newcommand{\xydriverfail}{Patterns are\UnloadPSPatternsrelax}
\newcommand{\xydriverfail}{rotations are\UnloadRotaterelax}

\begin{itemize}
  \item rotated/scaled diagrams and text, recognised but not supported.
\end{itemize}

In fact Oz\TeX cannot support rotations and scaling. Instead we go through all the motions of providing PostScript support, then set a flag that causes a limited number of warning messages to be generated as requests are made for these effects.

The end & Log

\xendinput

% $Log: xy17oztex.doc,v $ % Revision 3.8 2011/03/14 20:14:00 krisrose % Preparing for release 3.8.6.
% % Revision 3.7 2010/06/10 18:45:49 krisrose % Reference to GPL by URL.
% % Revision 3.6 2010/05/06 17:46:29 krisrose % Ross Moore’s e-mail address updated.
% % Many obsolete files degraded to Historic.
% % Revision 3.5 2010/05/06 03:48:05 krisrose % Fixed missing references.
% % Revision 3.4 2010/04/16 06:06:52 krisrose % Preparing for a new release…
% % Revision 3.3 1996/12/18 14:21:23 ros % Ross’s version
% % Revision 3.3.1.1 1996/12/18 08:47:45 ros % *** empty log message ***
4.7. TEXTURES DRIVER

This driver provides support for version 1.7+ of Blue Sky Research’s TEXTURES application for Macintosh\(^3\). It incorporates support for colour and all of Xy-pic’s POSTSCRIPT effects. Earlier versions of TEXTURES should instead use the driver option \texttt{\string\xyoption{16textures}}.

Notice that version 1.7 suffers from a printing bug which may cause a POSTSCRIPT error. A fix is kludged by making sure the first page has been shown in the viewer before any pages with diagrams are sent to the printer.

Header:

\begin{verbatim}
\%% $Id: xytextures.doc,v 3.7 2011/03/14 20:14:00 krisrose Exp $
\%%
\%% Xy-pic ``Textures'' driver interface option.
\%% Copyright (c) 1995-1996 Ross Moore <ross.moore@mq.edu.au>
\%%
\%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
\%% See the companion README and INSTALL files for further information.
\%% Copyright (c) 1991-2011 Kristoffer H. Rose <krisrose@tug.org>
\%%
\%% The Xy-pic package is free software; you can redistribute it and/or modify
\%% it under the terms of the GNU General Public License as published by the
\%% Free Software Foundation; either version 2 of the License, or (at your
\%% option) any later version.
\%%
\%% The Xy-pic package is distributed in the hope that it will be useful, but
\%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
\%% or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License
\%% for more details.
\%%
\%% You should have received a copy of the GNU General Public License along
\end{verbatim}

\(^3\)Macintosh is a trademark of Apple Computer Inc.
\% with this package; if not, see http://www.gnu.org/licenses/.
\%
\if\xyloaded\undefined \input xy \fi
\xyprovide{textures}{Textures driver}{\stripRCS$Revision: 3.7$}\
{Ross Moore}\{ross.moore@mq.edu.au\}\
{Mathematics Department, Macquarie University, NSW~2109, Australia}

\Textures@@#1\{\dvmessage@\{Textures\}{#1}\}
\newdriver{\
\xyaddsupport{color}\TexturesColor@@
\xyaddsupport{crayon}\TexturesCrayon@@
\xyaddsupport{ps}\TexturesPS@@
\xyaddsupport{rotate}\TexturesRotate@@
\xyaddsupport{line}\TexturesLine@@
\xyaddsupport{frame}\TexturesFrames@@
\xyaddsupport{tile}\TexturesTiles@@
\xyaddsupport{cmtip}\relax
}

Supported \special effects are...

- colour, both on-screen and with \textsc{PostScript}

  Textures uses the colour-stack method, but it requires a separate \special to set the new colour after having pushed the previous colour onto the stack.

\TexturesColor@push#1\{\special{color push}\special{color #1}\}

\TexturesColours@\{\texturesPStypes@
\let\xylocalColor@=\xystackcolor@
\let\xycolor@push=\TexturesColor@push
\def\xycolor@{\xycolor@@}
\expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{\%
\xynputorelse@\{xyps-c\}{\xydriverunload@\{ps-c\}}%
\installPScolor@ }\else\DN@{}\fi \next@
\if\xycolor@@\xynoColor@ \let\xycolor@@=\xycolor@raw@\fi
\if\xycrayolaloaded\empty \DN@{\xystandardcolors@\TexturesCrayon@@{}\}}%
\else\let\next@=\xystandardcolors@ \fi \next@
\let\xycolorwarning@=\relax \Textures@@{color}\
\let\TexturesColor@=\relax
\let\UnloadColor@@=\NoTexturesColor@
}

\TexturesColor@@\{\TexturesColours@
\TexturesNoTexturesColor@\{\%\n\let\xylocalColor@=\xynoColor@@
\let\xycolor@=\xynoColor@
\let\xycolorwarning@=\xycolorwarning@@
\let\TexturesColor@=\xynoColor@@
\let\UnloadColor@@=\NoTexturesColours@
}\}
4.7. TEXTURES DRIVER

- crayon colours.

The 68 colours are those that \texttt{dvips} recognizes by name, thanks to Tomas Rokicki.

\begin{verbatim}
\xydef@texturesCrayon@@\{\texturesColor@@\texturesCrayon0\%
\xydef@texturesCrayon0\{\installCrayolaColors0\%

- PostScript back-end.

\begin{verbatim}
\xydef@texturesPS@@\{\texturesPS0\%
\xydef@texturesPSunload@\{\UnloadPS@
\let\UnloadPS@@=\UnloadPS@
\let\texturesPS@@=\texturesPS0\%

\end{verbatim}

Versions 1.7 and later of TEXTURES have the following kinds of \texttt{\special}, the first two executing in the environment defined by the dictionary \texttt{userdict}. In particular these versions of TEXTURES now allow the POSTSCRIPT dictionary to be loaded into the main working dictionary.

- \texttt{\special{rawpostscript \#1}} puts code directly into the POSTSCRIPT file.
- \texttt{\special{postscript \#1}} wraps the code within \texttt{sps...eps}, which involves a basis change within a \texttt{save/restore} pair.
- \texttt{\special{prePostScript \#1}} places the POSTSCRIPT code within the header dictionary, called TeXdict.

Each of these has a corresponding version for reading the POSTSCRIPT commands from a file.

\begin{verbatim}
\xydef@texturesPStypes@{
\xyinputorelse@\texttt{xyps-ps}{\xydrivernoload@\{ps-ps\}}\%
\ifx\PSspecial@@\TexturesSpecial@\else\let\PSspecial@@=\TexturesSpecial@\fi
\let\PSmacro@@=\TexturesMacro@
\let\PSdict@@=\TexturesDict@
\let\PSraw@@=\TexturesRaw@
\let\PSinclude@@=\TexturesInclude@
\let\PSspecialdict@@=\TexturesSpecialDict@
\let\PSobeylines@@=\obeyspaces\%
\fi\texturesCurrpt@\XYpredict@\%
\end{verbatim}

The \texttt{currentpoint} is defined for the “raw” specials. For Textures this is the same as \texttt{\xyPScurrpt@@}, in \texttt{xyps-ps.doc}.

\begin{verbatim}
\xydef@\texturesCurrpt0\{\let\xyPScurrpt@@=\texturesCurrpt0\%
\end{verbatim}
The PostScript operator called \texttt{xyx} is defined in \texttt{xyps-ps} for storing the location placed on the stack by \texttt{xyi}.

This installs the PostScript backend.

- rotated/scaled diagrams and text, using PostScript.
- variable line-widths and poly-lines, using PostScript.
- extra frames and fills, using PostScript.
- patterns and tiles, using PostScript.
This driver provides support for versions 1.5b and 1.6 of Blue Sky Research’s TEXTURES application for Macintosh. It incorporates support for PostScript colour and the XY-ps PostScript back-end. This will not work with versions 1.7 and later; these require the \texttt{\textbackslash{}xyoption{textures}} option.

Macintosh is a trademark of Apple Computer Inc.
Supported \special effects are...

- colour, using PostScript

Textures v1.6 cannot manage colours, except within imported graphics. It can put colours into the PostScript output.
4.8. TEXTURES V1.6 DRIVER

- crayon colours.
  The 68 colours are those that dvips recognizes by name, thanks to Tomas Rokicki.

- POSTSCRIPT back-end.

We must return the binding of \shipout to its initial value.

Early versions (1.5–1.6) of TEXTURES have two kinds of \special, for inserting POSTSCRIPT code into the dvi-file and two more for reading such code from a file:

- \special{rawpostscript #1} puts code directly into the POSTSCRIPT file.
- \special{postscript #1} wraps the code within sps...eps, which involves graphics state changes within a gsave/grestore pair.

Each of these has a corresponding version for reading the POSTSCRIPT commands from a file.
The current point is defined for the “raw” specials. For Textures this is the same as \xyPScurrpt@@@, in xyps-ps.doc.

The PostScript operator called \texttt{xyx} is defined in \texttt{xyps-ps} for storing the location placed on the stack by \texttt{xyi}.

Textures v1.6 requires the PostScript dictionary to be shipped-out with every page. To achieve this efficiently we rebind \texttt{\shipout}, as described in xyps-ps.doc. Next set the flag \texttt{\Usedict@true} after having rebound \texttt{\includePSmessage@} to gobble the attempt to load the dictionary directly into the dvi-file.

This installs the PostScript backend.

- rotated/scaled diagrams and text, using PostScript.

- variable line-widths and poly-lines, using PostScript.
4.8. TEXTURES V1.6 DRIVER

- extra frames and fills, using POSTSCRIPT.

\( \text{\LaTeX} \)

\begin{verbatim}
\installPSline@ }\%
\else\DN@{\xydriverfail@{line-styles are}\UnloadLine@\relax}\fi
\next@ 

\end{verbatim}

\begin{itemize}
\item patterns and tiles, using POSTSCRIPT.
\end{itemize}

\begin{verbatim}
\xydef@\oldtexturesFrames@@{\oldtexturesFrames@}
\xydef@\oldtexturesFrames@{\oldtexturesPStypes@
\expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{% 
\xyinputorelse@{xyps-f}{\xydrivernoload@{ps-f}}}%
\installPSframes@ }%
\else\DN@{\xydriverfail@{frames are}\UnloadPSFrames@\relax}\fi
\next@ }

\xydef@\oldtexturesTiles@@{\oldtexturesTiles@}
\xydef@\oldtexturesTiles@{\oldtexturesPStypes@
\expandafter\ifx\csname xyps-psloaded\endcsname\empty\DN@{% 
\xyinputorelse@{xyps-t}{\xydrivernoload@{ps-t}}}%
\installPSpatterns@ \xystandardpatterns@}
\else\DN@{\xydriverfail@{Patterns are}\UnloadPSpatterns@\relax}\fi
\next@ }
\end{verbatim}

The end & Log

\xyendinput

\begin{verbatim}
% $Log: xy16textures.doc,v $
% Revision 3.7 2011/03/14 20:14:00 krisrose
% Preparing for release 3.8.6.
% 
% Revision 3.6 2010/06/10 18:45:49 krisrose
% Reference to GPL by URL.
% 
% Revision 3.5 2010/05/06 17:46:29 krisrose
% Ross Moore’s e-mail address updated.
% Many obsolete files degraded to Historic.
% 
% Revision 3.4 2010/04/16 06:06:52 krisrose
% Preparing for a new release...
% 
% Revision 3.3 1996/12/18 14:21:23 ross
% Ross’s version
% 
% Revision 3.3.1.1 1996/12/18 08:45:28 ross
% *** empty log message ***
% 
% Revision 3.2 1995/09/19 18:20:20 ross
% Bug fix release.
% 
% The end & Log
% 
% $End: xy16textures.doc,v $
4.9 XDVI driver

Vers. 3.7 by Ross Moore (ross.moore@mq.edu.au)

This driver provides support for extensions when using variants of the xdvi driver, by Eric Cooper, Bob Scheifler, Mark Eichin and others. It has been used successfully with xdvi patchlevel 20, by Paul Vojta, and xdvik version 18f, by Karl Berry.

Some of the supported features assume that the implementation of xdvi is linked to a POSTSCRIPT renderer; e.g. Ghostscript or DISPLAY POSTSCRIPT. If such support is not available, then invoking xdvi using the command xdvi -hushspecials will suppress warning messages that might otherwise be produced. One drawback of such a setup is that much of the POSTSCRIPT is not rendered until after all of the font characters, etc. have been placed on the page. Thus text that was meant to be placed on top of a filled or patterned region may appear to be obscured by it. However when printed, using a POSTSCRIPT printer, the correct placement is obtained.

Header:

```latex
%%% $Id: xyxdvi.doc,v 3.7 2011/03/14 20:14:00 krisrose Exp $
%%% Xy-pic ``xdvi-driver'' option.
%%% Copyright (c) 1995-1996 Ross Moore <ross.moore@mq.edu.au>
%%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%%% See the companion README and INSTALL files for further information.
%%% Copyright (c) 1991-2011 Kristoffer H. Rose <krisrose@tug.org>
%%% The Xy-pic package is free software; you can redistribute it and/or modify
%%% it under the terms of the GNU General Public License as published by the
%%% Free Software Foundation; either version 2 of the License, or (at your
%%% option) any later version.
%%% The Xy-pic package is distributed in the hope that it will be useful, but
%%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%%% or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License
%%% for more details.
%%% You should have received a copy of the GNU General Public License along
%%% with this package; if not, see http://www.gnu.org/licenses/.
```
4.9. **XDVI DRIVER**

Driver installation  
Supported special effects are...

- colour, using PostScript.

Not all versions of xdvi support color specials, so there is no direct support for colour. However parts of pictures rendered using PostScript may appear coloured, due to interpretation of colour commands within the PostScript.

Disable the \xylocalColor@ but enable the \xycolor@ for use by PostScript.
If neither support file `colordvi.tex` nor `colordvi.sty` can be found, then the normal colour support will not be available. However the colour support for the POSTSCRIPT back-end can still be used.

```latex
\xydef@\NoXDVIcolordvi@{%\xywarning@{%Neither colordvi.tex nor colordvi.sty could be found.}%\xyEcolorcheck@}%
```

- crayon colours.

  The 68 colours that `xdvi` recognizes by name are not loaded unless the `crayon` option has been requested.

```latex
\xydef@\xdviCrayola@@@{\xdviColor@@\xdviCrayola@}
\xydef@\xdviCrayola@{\let\prevxycolor@=\newxycolor@
def\newxyXDVIcolor@@#1#{\prevxycolor@{#1}}{#1}\
\let\newxycolor@=\newxyXDVIcolor@
\installCrayolaColors@\let\newxycolor@=\prevxycolor@}%
```

Although any order of loading options: `ps`, `xdvi`, `color` and `crayon` produces the desired result visually, the POSTSCRIPT code can be different with different loading order. The most easily readable is obtained when `crayon` is requested last.

- POSTSCRIPT back-end.

```latex
\xydef@\xdviPS@@@{\xdviPS@}
\xydef@\xdviPS@unload{\UnloadPS@\let\UnloadPS@@=\UnloadPS@\let\xdviPS@@=\xdviPS@ }
```

The latest versions of `xdvi` recognise most forms of `\special` command that are recognised by `dvips`; these are used to support POSTSCRIPT effects.

```latex
\xydef@\xdviPStypes@{%\xyinputorelse@{xyps-ps}{{\xydrivernoload@{ps-ps}}%\let\Pspecial@@=\xdviSpecial@\else\let\Pspecial@@=\xdviSpecial@\let\Pmacro@@=\xdviMacro@
\let\Pdict@@=\xdviDict@\let\Pspecialdict@@=\xdviDict@
\let\Praw@@=\xdviRaw@
\let\Pinclude@@=\xdviInclude@
\let\xyPSobeylines@@=\obeyXDVIlines@
\let\xyPScurrpt@@=\xdviCurrpt@\let\xdvi@@{PostScript}%%\fi\XYpredict@}
\xywarnifdefined\xdviSpecial@
\xywarnifdefined\xdviMacro@
\xywarnifdefined\xdviDict@
All the PostScript definitions passed to xdvi by reading the dictionary file are stored in a PostScript dictionary which it defines, called SDict. This must be the current dictionary whenever an X Y-ps command is to be executed. This will be the case whenever the \special{" commands} or \special{! commands} forms are used, but not when the \special{ps:: commands} form is used. Thus the \xdviSpecial@ macro includes code to open SDict and close it when finished; so also does the \xdviRaw@ type, since this is used with Xy-pic PostScript operators whose definitions have been placed within SDict.

The currentpoint is defined for the “raw” specials.

The PostScript operator called xyp is defined in xyps-ps for storing the location read from currentpoint.

This installs the PostScript backend.

• rotated/scaled diagrams and text, using PostScript.

• variable line-widths and poly-lines, using PostScript.
• extra frames and fills, using PostScript.

• patterns and tiles, using PostScript.

• TPIC drawing commands.

The end & Log

\endinput

% $Log: xxyxdvi.doc,v $ 
% Revision 3.7  2011/03/14 20:14:00 krisrose 
% Preparing for release 3.8.6. 
% 
% Revision 3.6  2010/06/10 18:45:50 krisrose 
% Reference to GPL by URL. 
% 
% Revision 3.5  2010/05/06 17:46:30 krisrose 
% Ross Moore’s e-mail address updated. 
% 
% Many obsolete files degraded to Historic. 
% 
% Revision 3.4  2010/04/16 06:06:52 krisrose 
% Preparing for a new release... 
% 
% Revision 3.3  1996/12/18 09:53:22 ross 
% checked in with -k by krisrose at 1996/12/18 14:17:11 
%
4.10 PDF driver

Vers. 1.7 by Daniel Müllner (http://math.stanford.edu/~muellner)

The PDF support is documented separately in the xypdf.pdf document, typeset by running \LaTeX on xypdf.dtx.

4.11 Extra features with \texttt{POSTSCRIPT} support

4.11.1 xyps-ps.doc

The included file xyps-ps.tex (version 3.12) provides support for \texttt{POSTSCRIPT \special} commands used by the \texttt{ps} backend extension as well as \texttt{POSTSCRIPT}-based options, to produce special effects not available directly with \TeX.

Header:

```
%% $Id: xyps-ps.doc,v 3.12 2011/03/14 20:14:00 krisrose Exp $
%%
%% Xy-pic `PS-PS' PostScript support.
%% Copyright (c) 1993-1997 Ross Moore <ross.moore@mq.edu.au>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-2011 Kristoffer H. Rose <krisrose@tug.org>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
```
PostScript \special commands  The PostScript \specials which are used by X\text-y-ps fall into four broad classes:

1. execute a piece of code, e.g. to draw some graphic object;
2. add a new PostScript definition to the existing dictionary of commands;
3. change the value of some parameters, storing them for later use; and
4. read PostScript commands from a pre-existing file.

Since different drivers may provide different syntax for these classes of command, X\text-y-ps has different macros to optimize to interface to the different drivers. Initially these control-sequence names are bound to macros that do nothing. Upon specifying a driver, they will be bound to a macro appropriate for that driver.

Unloading PostScript requires also turning off the PostScript-backend, if it has been loaded.
4.11. EXTRA FEATURES WITH PostScript SUPPORT

Some drivers, in particular versions of TEXTURES earlier than 1.7, require the PostScript dictionary to be included with every page shipped out. We do this by providing a routine which rebinds the \shipout primitive to a macro \xyPSshipout which prepends the XY-ps PostScript dictionary to the box being shipped-out. This routine \@PSshipout is called, if necessary, when the driver has been specified.

The box register \box9 is assumed to be a scratch register, used globally according to \TeX conventions.

When the PostScript backend is loaded then we need to rebind \shipout to include the full dictionary.

These two control sequences are used in the DVI-driver files to control when the PostScript (driver) can be altered.

4.11.2 Installation

To install use of PostScript requires loading the PostScript dictionary. However this can only be done when a (driver) is known to be able to support it. Hence the \installxyps command should be called only from within a (driver)-file.

Furthermore, the (driver)-file should define a value for \installxyps to including anything specific to that (driver); even if only \let\installxyps=\relax. If \dumpPSdict has been specified, then the dictionary must be written first, so it can be used with the subsequent processing (see the subsection below).

\ifx\undefined\dumpPSdict@@ \xylet\dumpPSdict@@=\relax \fi
\xydef\installxyps@@{@
\xylet\installxyps@@=\relax
In case no ⟨driver⟩ was known when a call to the \installxyps@ method was queued, e.g. by \dumpPSdict{} before any \xyoption{⟨driver⟩}, then a check is made to see there is now a known ⟨driver⟩ which supports PostScript. If so then this installation is cancelled since another should follow, otherwise the default ⟨driver⟩ of dvips is assumed. The command \xyPSdriver#1 is provided to allow users to change this default ⟨driver⟩. This may be necessary when multiple ⟨driver⟩s are required and the final one specified does not support PostScript.

\xydef\xydefaultdriver@[dvips]
\xydef\xyPSdriver#1{\def\xydefaultdriver@[#1]}
\xydef\installxyps@orig{\DN@{}}
\ifx\xydriversselected@\empty
  \DN@{\expandafter\defaultinstallps@\expandafter{\xydefaultdriver@[#1]}}
\else
  \def\do##1{%\expandafter\ifx\csname##1@xy@ps\endcsname\relax
    \DN@{\defaultinstallps@[##1]}}
  \xydriversselected@%
\else\DN@[#1]%!fi\fi\next@
\%\xylet\installxyps@@=\installxyps@
\xydef\defaultinstallps@#1#2{%\xywarning@{no driver specified, using #1}%
  \xyrequire{#1}\csname #1@xy@ps\endcsname{}}
\xydef\null@xy@ps{\installxyps@}

PostScript escape

An extra ⟨shape⟩ modifier key allows arbitrary PostScript code to be applied to the current ⟨object⟩.

\begin{verbatim}
[1⟨postscript code⟩] for special effects
[psxy] stores current location.
\end{verbatim}
4.11. EXTRA FEATURES WITH POSTSCRIPT SUPPORT

Normally the ⟨postscript code⟩ will be a simple command to alter the PostScript graphics state: e.g. ![1 0 0 setrgbcolor] changes the colour used to render parts of the ⟨object⟩. Any number of such ⟨shape⟩ modifiers is allowable, however it is more efficient to combine them into a single modifier, whenever possible.

It is very important that braces { and } do not appear explicitly in any ⟨postscript code⟩, as this may upset the XY-pic ⟨object⟩ parsing. However it is acceptable to have a control sequence name here, expanding into more intricate PostScript code. This will not be expanded until a later (safe) time.

Due to differences within the DVI-drivers, such simple PostScript commands need not affect every part of an ⟨object⟩. In particular the lines, curves and arrowheads generated by X Y-pic use a different mechanism, which should give the same result with all drivers. This involves redefining some PostScript procedures which are always read prior to rendering one of these objects. One simple way to specify a red line is as follows; the xycolor extension provides more sophisticated support for colour. The ⟨shape⟩ modifiers described in the previous section also use this mechanism, so should work correctly with all drivers.

\def\colorxy(#1){% /xycolor{#1 setrgbcolor}def} \connect![\colorxy(1 0 0)]\dir{-}

Note how the braces are inserted within the expansion of the control sequence \colorxy, which happens after parsing of the ⟨connection⟩. The following table shows which graphics parameters are treated in this way, their default settings, and the type of PostScript code needed to change them.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>colour</td>
<td>/xycolor{0 setgray}def</td>
</tr>
<tr>
<td>line-width</td>
<td>/xywidth{.4 setlinewidth}def</td>
</tr>
<tr>
<td>dashing</td>
<td>/xydash{[] 0 setdash}def</td>
</tr>
<tr>
<td>line-cap</td>
<td>/ycap{1 setlinecap}def</td>
</tr>
<tr>
<td>line-join</td>
<td>/xyjoin{1 setlinejoin}def</td>
</tr>
</tbody>
</table>

This feature is meant primarily for modifying the rendering of objects specified in TeX and XY-pic, not for drawing new objects within PostScript. No guarantee can be given of the current location, or scale, which may be different with different drivers. However a good PostScript programmer will be able to overcome such difficulties and do much more. To aid in this the special modifier [psxy] is provided to record the location where the reference point of the current ⟨object⟩ will be placed. Its coordinates are stored with keys xyXpos and xyYpos.

When the \special is placed the following registers contain important values: \L@p = horizontal displacement of XY-pic reference point from the TeX reference point (i.e. left-hand end of the box) of the initial object; \D@p = vertical displacement of XY-pic reference point from the TeX reference point (i.e. the baseline) of the initial object; \R@p = horizontal offset, resulting from ⟨shape⟩ modifiers; \U@p = vertical offset, initially −\D@p but alterable by ⟨shape⟩ modifiers.

\xydef@{xyPSpsxy@{\setboxz@h{\setlength{\dimen@}{\L@p} \vbox{\hbox to\z@{}}} \ht@{-\dimen@}}}

\dimen@i=-\U@p \raise{\dimen@i} 

It is necessary to know the current PostScript location. Unfortunately the currentpoint operator frequently has no value. The following code overcomes this difficulty.

Some drivers may need to define this differently...

The PostScript operator called \texttt{xyx} is defined below, for storing the location placed on the stack by \texttt{xyi}.

\textbf{Technical Note}:

The scoping is achieved by using two \texttt{\specials} so that the resulting PostScript file should ultimately look like:

\begin{verbatim}
... {special code before} ⟨object⟩ {special code after} ...
\end{verbatim}

The “code after” is to cancel the effect, returning the graphics state to what it was prior to the “code before”. Not all \texttt{DVI} drivers can achieve this sequencing. In particular \texttt{OzT\TeX} collects all \texttt{\specials} in the \texttt{DVI} file and places their contents at the beginning of the PostScript file: any effect would be cancelled immediately after it has been established.

The user can add code to both parts by expressing the ⟨\texttt{shape}⟩ modifier as follows:

\begin{verbatim}
*...⟨code before⟩⟨code after⟩...*
\end{verbatim}

\textbf{Further Technical Note}:

The “code before” does two things in addition to that code given explicitly by the user. Firstly it issues a \texttt{gsave} then it opens a new dictionary on the dictionary stack. The “code after” contains the matching \texttt{grestore} after closing the new dictionary.

Since objects can be built as \texttt{\composites} and diagrams can be nested, there is the possibility of generating long chains of nested PostScript dictionaries. For this reason the dictionary is kept small, allowing only 8 key-value pairs to be defined within it. If more are required, the user should define a private dictionary to hold the extra key-value pairs, making sure that it is open when its entries need to be accessed.
4.11. EXTRA FEATURES WITH PostScript SUPPORT

The PostScript operators pu, pp, xyg and xyf are defined below, for push/pop of the XYddict stack, preceded/followed by a gsave/grestore of the PostScript graphics state.

Some utility macros for controlling writing to the log-file.

PostScript Header file
This creates a flag to indicate whether the user wishes to get the PostScript dictionary from an external header file.
These macros are also defined in xyps.doc. To avoid warning messages about redefinitions, that mechanism has been disabled.

This is used to write the dictionary to a file, suitable for inclusion as a resource within any PostScript document. It is fully conforming to Adobe’s document structuring guidelines.
Within the expansion of \texttt{\verb|\xydict@|} the end-of-line tokens are still active. The following expansion seems to work on all systems so far tested.

The dictionary of \textsc{PostScript} commands is split into two pieces according to whether they are relevant to general \textsc{PostScript} effects or only to the \textsc{PostScript} backend. These pieces are generated from code within the files \texttt{xyps-pro.tex} and \texttt{xypsdict.tex}. These files have corresponding \texttt{.doc} versions for documentation.

The following commands cause the parts of the \textsc{PostScript} dictionary to be read from the appropriate files. If \texttt{\ifUsePSdict@@} gives \texttt{\iftrue} then there may be no need to do anything, since the definitions will be subsequently loaded from the header file. Normally it is only necessary to read these files once, using \texttt{\verb|\xypredict@@|}. However some \langle driver\rangle s may need it more frequently, so instead call \texttt{\verb|\xypredict@@|} directly.
4.11.3 Extensions

Several included files handle standard PostScript implementations of \texttt{XY-pic} extensions.

A conditional \texttt{ifxyPSshapes@} is used to indicate whether the special shape effects implemented in \texttt{xyps-r.tex} can actually be fully supported by the current driver. When not available, then an attempt to use these effects simply results in a warning message. After two attempts the warning message “turns itself off”; subsequent attempts are simply ignored cleanly.

\begin{verbatim}
\xynew@if\ifxyPSshapes@ \xyPSshapes@true
\xydef@\xyPScharwarning@@\{\xyPScharmessage@
\gdef\xyPScharwarning@@\{\xyPScharmessage@
\xywarning@{...no further PostScript warnings will be given}%
\global\let\xyPScharwarning@@=\relax }
\endverbatim

Allow new PostScript effects to be defined. This section describes how \texttt{newxyPSshape} is used to define a new effect which is available only in PostScript; \textit{i.e.} having no analogue elsewhere within \texttt{XY-pic}.

Hence a control sequence of this form must first be created, if it does not already exist. Its initial expansion is simply \texttt{xyundefinedEffect@(name)} which produces a warning message.
4.11. EXTRA FEATURES WITH PostScript SUPPORT

When an implementation of the effect is available then \csname xyshape@⟨name⟩\endcsname is rebound to \csname xyPSshape@⟨name⟩\endcsname, which expands to the contents of #2 in \newxyPSshape#1#2. If #2 is empty then the expansion is the \xyPSnotimplemented@langle name⟩@@ warning message, which is especially useful during development.

The end & Log

$Log: xyps-ps.doc,v $
% Revision 3.12 2011/03/14 20:14:00 krisrose
% Preparing for release 3.8.6.
% Revision 3.11 2010/06/10 18:45:50 krisrose
% Reference to GPL by URL.
% Revision 3.10 2010/05/06 17:46:30 krisrose
% Ross Moore’s e-mail address updated.
% Many obsolete files degraded to Historic.
% Revision 3.9 2010/05/04 23:02:15 krisrose
% Tiles are back but broken in PDF.
% Revision 3.8 2010/04/16 06:06:52 krisrose
% Preparing for a new release...
% Revision 3.7 1999/02/16 15:12:50 krisrose
Interim release (Y&Y fonts now free).

% Revision 3.6 1998/03/06 01:28:05 krisrose
% Releasing (with Y&Y fonts).
% Revision 3.4 1997/05/18 01:13:24 ross
% Essential bugfixes.
% Revision 3.3 1996/12/19 03:50:08 ross
% Maintenance release.
% Revision 3.3 1996/12/18 09:55:56 ross
% improvements to the file-loading commands
% more robust installation procedures
% shorter tracing messages
% dictionary no longer loads multiply
% Revision 3.2 1995/09/19 18:21:41 ross
% Bug fix release.
% Revision 3.1 1995/09/05 20:36:33 ross
% Release!
% Revision 3.0 1995/07/07 20:13:19 ross
% Major release w/new User's Guide!
% Revision 2.13 1995/07/04 15:04:51 ross
% Ready for release of v3.
% NEW for version 3.1 by by Ross Moore.

4.11.4  xyps-c.doc

Header:
1  %% $Id: xyps-c.doc,v 3.11 2011/03/14 20:14:00 krisrose Exp $
2  %%
3  %% Xy-pic 'Colour extension' PostScript backend support.
4  %% Copyright (c) 1993-1996 Ross Moore <ross.moore@mq.edu.au>
5  %%
6  %% This file is part of the Xy-pic package for graphs and diagrams in TeX.
7  %% See the companion README and INSTALL files for further information.
8  %% Copyright (c) 1991-2011 Kristoffer H. Rose <krisrose@tug.org>
9  %%
10  %% The Xy-pic package is free software; you can redistribute it and/or modify
11  %% it under the terms of the GNU General Public License as published by the
12  %% Free Software Foundation; either version 2 of the License, or (at your
13  %% option) any later version.
14  %%
4.11. EXTRA FEATURES WITH PostScript SUPPORT

%%%% The Xy-pic package is distributed in the hope that it will be useful, but
%%%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%%%% or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License
%%%% for more details.
%%%%
%%%% You should have received a copy of the GNU General Public License along
%%%% with this package; if not, see http://www.gnu.org/licenses/.
%%%%
%%%% As a special exception, you may use this file and all files derived
%%%% from it without restrictions. This special exception was added with
%%%% version 3.7 of Xy-pic.
%%%%
\expandafter\ifx\csname xyps-cloaded\endcsname\empty
\message{not reloaded}\endinput \fi
\expandafter\let\csname xyps-cloaded\endcsname=\empty \xycatcodes

4.11.5 Colour

The included file xyps-c.tex (version 3.11) provides PostScript support for the effects defined in
the color extension in §2.6.

The PostScript style methods are required, so ensure that they are loaded.

\expandafter\ifx \csname xyps-sloaded\endcsname\empty\else
\xyinputorelse@{xyps-s}{\xyerror@{Could not load xyps-s}{}}\%
\xycatcodes\fi

This file is loaded and its effects are activated automatically whenever \xyoption{color} is re-
quested and the current ⟨driver⟩ supports colours using PostScript. Should there be any need to
turn off this support, the following commands are available; they obey usual TEX groupings.

\NoPScolor remove PostScript support
\UsePScolor reinstate PostScript.

Without PostScript support some drivers may still be able to provide some support for colours. These commands are not guaranteed to work adequately with all drivers. They are provided primarily for testing and trouble-shooting; e.g. with ⟨driver⟩ configurations untested by the authors of Xy-pic, who should be notified of any difficulties.

\xydef@\UsePScolor{\installPScolor@}
\xydef@\NoPScolor{\UnloadColor@}

Installation Installation consists of rebinding the macros \xycolor@ and \newxycolor@ and of
reloading the standard colours to include a description for PostScript. This installation should be
delayed till after color and ps themselves have been installed.

\xydef@\installPScolor@{\installPSstyle@}
\let\xymath@=\xyPSmath@ \let\xynomath@=\xyPSnomath@
\let\checkxyPScolor@=\checkxyPScolor@
\let\xycolor@@=\xyPScolor@@
\xystandardcolors@
\xuncatcodes}
All effects defined in the color extension can be implemented using the PostScript dictionary, loaded by xyps-ps. This file provides “generic” PostScript code which is known to work correctly with most drivers. No attempt is made to provide special code for particular drivers; for driver-specific variations, consult the appropriate xy(driver).doc file.

\xydef@\xyPSmath@\{\hbox{\bgroup\dimen@=.55ex \checkxyPScolor@ \xyinside@}\}
\xydef@\xyPSnomath@\{\hbox{\bgroup\dimen@=\z@ \checkxyPScolor@ \xyinside@}\}
\xydef@\checkxyPScolor@@\{\xycolor@raw@@\}\let\checkxyPScolor@=\relax
\xylet@\checkxyPScolor@=\relax
\xydef@\xyPSnewcolor@#1#2#3#4#5{%
\expandafter\DN@\expandafter{\csname shape \#1\endcsname}\
\DNii@{\relax}\ifx\next@\nextii@\newxycolor{\#1}{}\relax\fi
\DNii@{\#4}\ifx\nextii@\empty\expandafter{\csname xyPSshape@#1@\endcsname}\relax\fi
\expandafter{\csname xyPScolor@{\#2 #3}}\else
\expandafter{\csname xyPSshape@#1@\endcsname}\fi
\let\modPSboxz@=\modXYstyle@ \xyPS@color@@{#1}%
\toks@={\egroup\let\xy@style@=\relax \def\Drop@@}
\expandafter\addtotoks@\expandafter{\expandafter{\Drop@@}\def\Connect@@}
\the\toks@ }
%\xydef@\xyPScolor@#1#1\{#1 /xycolor{#1}def \}
\xydef@\xyPScolor@#1#1\{\xyPSsplitPS@{#1}cc{}\}
\xydef@\xycolor@raw@@#1\{\PSraw @@{yc}

The end & Log

\endinput

% $Log: xyps-c.doc,v $
% Revision 3.11 2011/03/14 20:14:00 krisrose
% Preparing for release 3.8.6.
%
% Revision 3.10 2010/06/10 18:45:50 krisrose
% Reference to GPL by URL.
%
% Revision 3.9 2010/05/06 17:46:30 krisrose
% Ross Moore’s e-mail address updated.
% Many obsolete files degraded to Historic.
4.11. EXTRA FEATURES WITH PostScript SUPPORT

% Revision 3.8 2010/04/16 06:06:52 krisrose
% Preparing for a new release...
%
% Revision 3.7 1999/02/16 15:12:50 krisrose
% Interim release (Y&Y fonts now free).
%
% Revision 3.3 1996/12/18 10:05:50 ross
% minor improvements to file-loading commands
%
% Revision 3.1 1995/09/05 20:36:33 ross
% Release!
%
% Revision 3.0 1995/07/07 20:13:19 ross
% Major release w/new User's Guide!
%
% Revision 2.13 1995/07/04 15:04:51 ross
% Ready for release of v3.
%
% Revision 2.9 1994/06/09 14:38:56 ross
% Release 3beta.
% Includes support for special effects: Rotations, Scaling, Line-width, Colour.
% Back-ends are separated into separate files.
% More back-ends are supported, experimentally --- needs testing.
%
% Revision 2.9 1994/06/09 14:38:56 ross
% Release 3beta.
% Includes support for special effects: Rotations, Scaling, Line-width, Colour.
% Back-ends are separated into separate files.
% More back-ends are supported, experimentally --- needs testing.
%
% NEW for version 2.9 by by Ross Moore.

4.11.6  xyps-f.doc

Header:

%% $Id: xyps-f.doc,v 3.11 2011/03/14 20:14:00 krisrose Exp $
%%
%% Xy-pic "Frame extension" PostScript backend support.
%% Copyright (c) 1995-1997 Ross Moore <ross.moore@mq.edu.au>
%%
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-2011 Kristoffer H. Rose <krisrose@tug.org>
%%
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
The `xypic` package is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this package; if not, see http://www.gnu.org/licenses/.

As a special exception, you may use this file and all files derived from it without restrictions. This special exception was added with version 3.7 of Xy-pic.

The included file `xyps-f.tex` (version 3.11) provides POSTSCRIPT support for the effects defined in the frame extension described in §2.2. It implements some effects otherwise unattainable.

This file is loaded and its effects are activated automatically whenever \`\texttt{\textbackslash xyoption{frame}}` is requested and the current ⟨driver⟩ supports POSTSCRIPT effects for frames. Should there be any need to turn off this support, the following commands are available; they obey usual \LaTeX{} groupings.

\begin{verbatim}
\NoPSframes  remove POSTSCRIPT support
\UsePSframes reinstates POSTSCRIPT.
\end{verbatim}

Without POSTSCRIPT support ellipses may be shown as circles and all filled regions may be represented as black rectangles. These commands are provided primarily for testing and trouble-shooting; e.g. with ⟨driver⟩ configurations untested by the authors of Xy-pic, who should be notified of any difficulties.

### Installation

The ⟨driver⟩-file must call the following macro to rebind control sequences defined in `xyframe`, thereby allowing the POSTSCRIPT methods to be used.

\begin{verbatim}
\xydef\installPSframes{%
  \let\framed@body@@=\xyPSframed@body@
  \let\circled@x@@=\xyPScircled@x@
  \let\ellipsed@x@@=\xyPSellipsed@x@
  \installPSfills@
  \xyuncatcodes}
\xydef\UsePSframes{\installPSframes@}
\xydef\NoPSframes{\UnloadFrames@}
\end{verbatim}

### Frames: solid, dotted and dashed

For solid frames use POSTSCRIPT ovals, otherwise adjust the spacing between dashes or dots according to the lengths, both vertical and horizontal.
4.11. EXTRA FEATURES WITH PostScript SUPPORT

In practice the parameters \#1 and \#2 will hold (dimen)s.

These may be used for dashed frames.

ovals and circles  The width, height, corner-radius are available as \dimen@i, \dimen@ii and \R@ respectively; we only need \R@ here, since the extents hold the real information that we need.
CHAPTER 4. DRIVERS

The PostScript operators ov, ox etc. are defined within xyps-pro, having scope limited to XYdict and sub-dictionaries.

Filled regions  Rectangles are treated as a special case of ovals. The \#2 parameter signifies whether the outline of the object should also be stroked (\#2=\relax) after being filled. Such a stroke is always
The end & Log

$Log: xyps-f.doc,v $  
Revision 3.11 2011/03/14 20:14:00 krisrose  
Preparing for release 3.8.6.

Revision 3.10 2010/06/10 18:45:50 krisrose  
Reference to GPL by URL.

Revision 3.9 2010/05/06 17:46:30 krisrose  
Ross Moore’s e-mail address updated.

Revision 3.8 2010/04/16 06:06:52 krisrose  
Preparing for a new release...

Revision 3.7 1999/02/16 15:12:50 krisrose  
Interim release (Y&Y fonts now free).

Revision 3.4 1997/05/18 01:13:24 ross  
Essential bugfixes.

Revision 3.3 1996/12/18 10:07:22 ross  
cosmetic changes to macro-names

Revision 3.1 1995/09/05 20:36:33 ross  
Release!

Revision 3.0 1995/07/07 20:13:19 ross  
Major release w/new User’s Guide!

Revision 2.13 1995/07/04 15:04:51 ross  
Ready for release of v.3.
CHAPTER 4. DRIVERS

% NEW for version 3.0 by Ross Moore.

4.11.8  xyps-l.doc

Header:

%% $Id: xyps-l.doc,v 3.11 2011/03/14 20:14:00 krisrose Exp $
%% Xy-pic ‘Line extension’ PostScript backend support.
%% Copyright (c) 1993-1997 Ross Moore <ross.moore@mq.edu.au>
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-2011 Kristoffer H. Rose <krisrose@tug.org>
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License
%% for more details.
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, see http://www.gnu.org/licenses/.
%% As a special exception, you may use this file and all files derived
%% from it without restrictions. This special exception was added with
%% version 3.7 of Xy-pic.
\expandafter\ifx\csname xyps-lloaded\endcsname\empty
\message{not reloaded}\endinput \fi
\expandafter\let\csname xyps-lloaded\endcsname=\empty \xycatcodes

4.11.9  Line-styles

The included file xyps-l.tex (version 3.11) provides PostScript support for the effects defined in
the line extension described in §2.4.

The PostScript style methods are required, so ensure that they are loaded.
\expandafter\ifx\csname xyps-sloaded\endcsname\empty\else
\xyinputorelse@{xyps-s}{\xyerror@{Could not load xyps-s}{}}\fi
\xycatcodes\fi

This file is loaded and its effects are activated automatically whenever \xyoption{line} is re-
quested and the current ⟨driver⟩ supports PostScript line styles. Should there be any need to turn
4.11. EXTRA FEATURES WITH PostScript SUPPORT

off this support, the following commands are available; they obey usual \TeX groupings.

\NoPSlines remove PostScript support
\UsePSlines reinstate PostScript.

Without PostScript support lines can be expected to be displayed in the default style, having thickness of \emph{.4pt}. These commands are provided primarily for testing and trouble-shooting; \emph{e.g.} with (driver) configurations untested by the authors of \texttt{Xy-pic}, who should be notified of any difficulties.

\textbf{Installation} The (driver)-file must call the following macro to rebind a control sequence defined in \texttt{xyline}, thereby allowing the PostScript method to be used. The purpose of the \texttt{\textbackslash xyPSlinew@} hook is to allow the PostScript style sequencing method to be used with other back-ends, changing just the contents of the \texttt{\specials} actually placed.

\begin{verbatim}
\xydef\installPSline@{
\installPSlinestyles@ \installPSpolylinestyles@}
\xydef\UsePSlines{\installPSline@}
\xydef\NoPSlines{\UninstallLine@\relax}
\end{verbatim}

widths, joins and caps This, and the rebindings below, is all that is needed for the cleaner, more sophisticated way.

\begin{verbatim}
\xydef\xyPSlineSpecial@#1{\addtospecials{ #1 lw}}
\xydef\xyPScapSpecial@#1{\addtospecials{ #1 lc}}
\xydef\xyPSjoinSpecial@#1{\addtospecials{ #1 lj}}
\xydef\xyPSmiterSpecial@#1{\addtospecials{ #1 ml}}
\end{verbatim}

\begin{verbatim}
\xydef\installPSlinestyles@{
\let\xylinewidth@@=\xylinewidth@
\let\xylinewidth@@=\xylinewidth@
\let\transxyline@@=\transxyline@
\let\xylineSpecial@@=\xyPSlineSpecial@
\let\resetxyline@@=\resetxyline@i
% \let\xy@linecap@@=\xy@linecap@
\let\xy@linecap@@=\xy@linecap@
\let\resetxylinecap@@=\resetxylinecap@i
% \let\xy@linejoin@@=\xy@linejoin@
\let\xy@linejoin@@=\xy@linejoin@
\let\xyjoinSpecial@@=\xyPSjoinSpecial@
\let\resetxylinejoin@@=\resetxylinejoin@i
% \let\xy@linemiter@@=\xy@linemiter@
\let\xy@linemiter@@=\xy@linemiter@
\let\xymiterSpecial@@=\xyPSmiterSpecial@
\let\resetxylinemiter@@=\resetxylinemiter@i
}
\end{verbatim}

The PostScript operators \texttt{lw}, \texttt{lc}, \texttt{lj}, \texttt{ml} are defined in \texttt{xyps-ps} to set the \texttt{linewidth}, \texttt{linecap}, \texttt{linejoin}, \texttt{miterlimit} graphics-state parameters, respectively.

\begin{verbatim}
\xydef\xypolylinePS@Special#1{\PSmacro@[#1 pl]}
\xydef\xypolydotPS@Special#1{\PSmacro@[#1 pt]}
\end{verbatim}
\xydef\xypolydashPS@Special#1\PSmacro\#1 \pd
\xydef\xypolyfillPS@Special#1\PSmacro\#1 \pf
\xydef\xypolyeofillPS@Special#1\PSmacro\#1 \pe
\let\xypolyline@Special=\xypolylinePS@Special
\let\xypolydot@Special=\xypolydotPS@Special
\let\xypolydash@Special=\xypolydashPS@Special
\let\xypolyfill@Special=\xypolyfillPS@Special
\let\xypolyeofill@Special=\xypolyeofillPS@Special
\let\xy@polystyle@@=\xy@polystyle@

Poly-lines

The end & Log

\endinput

% $Log: xyps-l.doc,v $
% Revision 3.11 2011/03/14 20:14:00 krisrose
% Preparing for release 3.8.6.
%
% Revision 3.10 2010/06/10 18:45:50 krisrose
% Reference to GPL by URL.
%
% Revision 3.9 2010/05/06 17:46:30 krisrose
% Ross Moore’s e-mail address updated.
% Many obsolete files degraded to Historic.
%
% Revision 3.8 2010/04/16 06:06:52 krisrose
% Preparing for a new release...
%
% Revision 3.7 1999/02/16 15:12:50 krisrose
% Interim release (Y&Y fonts now free).
%
% Revision 3.4 1997/05/18 01:13:24 ross
% Essential bugfixes.
%
% Revision 3.3 1996/12/18 10:05:50 ross
% minor improvements to file-loading commands
%
% Revision 3.1 1995/09/05 20:36:33 ross
% Release!
%
% Revision 3.0 1995/07/07 20:13:19 ross
% Major release w/new User’s Guide!
%
% Revision 2.13 1995/07/04 15:04:51 ross
% Ready for release of v3.
%
% Revision 2.9 1994/06/09 14:38:56 ross
% Release 3beta.
% Includes support for special effects: Rotations, Scaling, Line-width, Colour.
% Back-ends are separated into separate files.
% More back-ends are supported, experimentally --- needs testing.
%
% Revision 2.9 1994/06/09 14:38:56 ross
% Release 3beta.
% Includes support for special effects: Rotations, Scaling, Line-width, Colour.
% Back-ends are separated into separate files.
% More back-ends are supported, experimentally --- needs testing.
%
% NEW for version 2.9 by by Ross Moore.

4.11.10 xyps-r.doc

Header:
%% $Id: xyps-r.doc,v 3.11 2011/03/14 20:14:00 krisrose Exp $
%%
%% Xy-pic `PostScript Rotations and Scaling'.
%% Copyright (c) 1993-1996 Ross Moore <ross.moore@mq.edu.au>
%% This file is part of the Xy-pic package for graphs and diagrams in TeX.
%% See the companion README and INSTALL files for further information.
%% Copyright (c) 1991-2011 Kristoffer H. Rose <krisrose@tug.org>
%% The Xy-pic package is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by the
%% Free Software Foundation; either version 2 of the License, or (at your
%% option) any later version.
%%
%% The Xy-pic package is distributed in the hope that it will be useful, but
%% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
%% or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License
%% for more details.
%%
%% You should have received a copy of the GNU General Public License along
%% with this package; if not, see http://www.gnu.org/licenses/.
%%
%% As a special exception, you may use this file and all files derived
%% from it without restrictions. This special exception was added with
%% version 3.7 of Xy-pic.
%%
\expandafter\ifx\csname xyps-rloaded\endcsname\empty
\message{not reloaded}\endinput \fi
\expandafter\let\csname xyps-rloaded\endcsname=\empty \xycatcodes
4.11.11 Rotations and scaling

The included file \texttt{xyps-r.tex} (version 3.11) provides \textsc{PostScript} support for the effects defined in the \texttt{rotate} extension described in §2.5.

The \textsc{PostScript} style methods are required, so ensure that they are loaded.

\begin{verbatim}
\expandafter\ifx\csname xyps-sloaded\endcsname\empty\else
\xyinputorelse@{xyps-s}\{\xyerror@{Could not load xyps-s}\}\%
\xycatcodes\fi
\end{verbatim}

This file is loaded and its effects are activated automatically whenever \texttt{\xyoption{rotate}} is requested and the current ⟨driver⟩ supports \textsc{PostScript} rotations. Should there be any need to turn off this support, the following commands are available; they obey usual \TeX\ grouping.

\begin{verbatim}
\NoPSrotate remove \textsc{PostScript} support
\UsePSrotate reinstate \textsc{PostScript} support.
\end{verbatim}

Without \textsc{PostScript} support diagrams can be expected to be displayed unrotated and unscaled. These commands are provided primarily for testing and trouble-shooting; e.g. with ⟨driver⟩ configurations untested by the authors of \textsc{Xy-pic}, who should be notified of persistent difficulties.

Installation  Call the following macro to allow the \textsc{PostScript} mechanisms to be used.

\begin{verbatim}
\xydef@\installPSrotscale@{\installPSstyle@}
\let\xyRotate@@=\xyPSrotate@@
% \let\xySpecialRotate@@=\xyPSspecialRotate@ 
\let\doSpecialRotate@@=\doPSspecialRotate@@
% \let\xyRot@named@=\xyPSpsxy@
% \let\xyRotnamed@=\xyPSpsxy@
\let\xyscale@@=\xyPSscale@@ 
\let\xyuncatcodes \}
\xydef@\UsePSrotate{\installPSrotscale@}
\xydef@\NoPSrotate{\UnloadRotate@\relax}
\end{verbatim}

All effects defined in the \texttt{rotate} extension can be implemented using a \textsc{PostScript} ⟨driver⟩. However different ⟨driver⟩s need not handle things in the same way; different \textsc{PostScript} code may be required to match the specific environment used by the ⟨driver⟩.

This file provides generic \textsc{PostScript} code which is known to work correctly with most drivers. It is written so as to indicate where modifications may be made to accommodate specific ⟨driver⟩s. For such driver-specific variations, consult the appropriate ⟨driver⟩-file, called \texttt{xy⟨driver⟩.doc}.

\begin{verbatim}
\xydef@\xyPSpretransform@{\xyPSrawA@{\preXYtransform@@}}
\xydef@\xyPSposttransform@{\xyPSrawZ@{\postXYtransform@@}}
\let\preXYtransform@=\xyPSpretransform@
\let\postXYtransform@=\xyPSposttransform@
\end{verbatim}

Transform lists
4.11. EXTRA FEATURES WITH PostScript SUPPORT

\xydef@\xyPSscale@@#1#2{%
\xyPSsplitPS@\{xyscale@start(#1,#2)\}\{xyscale@end\}%
\xyPSrotSplit@\xyPSpsxy@
}

Rescaling

\xydef@\xyPSrotate@@#1{%
\xyPSsplitPS@\{\xyrot@start(#1 xyd)\}\{\xyrot@end\}%
\xyPSrotSplit@\xyPSpsxy@
}
\xydef@\xyPSspecialRotate@#1{%
\xyPSsplitPS@\{\xyrot@start(#1)\}\{\xyrot@end\}\xyPSrotSplit@\xyPSpsxy@
}
\xydef@\doPSspecialRotate@@#1@@{\xyPSspecialRotate@{#1}}

Rotations  The PostScript operator \texttt{xyr} is defined in \texttt{xyps-ps.doc}, to store the given parameter as a rotation angle.

Shearing  This feature is not implemented yet.

PostScript commands  The PostScript codes to start the rotation or scaling are given as functions, \texttt{xyrot@start} and \texttt{xyscale@start} with arguments to include a specification of the rotation-angle or scaling factors.

\xydef@\xyRotScale@@{%
\def\xyrot@start(#1){\texttt{xyt #1 space xyr}}%
\def\xyscale@start(#1,#2){\texttt{xyt #1 space #2 space xys}}%
\def\xyrot@end{}
\def\xyscale@end{}}

These strings are placed using \texttt{\xyPSsplitPS@} so as to correctly occur before and after the code for the actual \texttt{⟨object⟩} being typeset. This is always followed by \texttt{\xyPSpsxy@}, which results in PostScript code to identify the current position, storing it with keys \texttt{xyXpos} and \texttt{xyYpos} so as to be accessible to the code for the rotation or scaling.

A specific \texttt{⟨driver⟩} must either call \texttt{\xyRotScale@@} to use this default mechanism, or define its own macro to be called at installation time, which sets alternative expansions to \texttt{\xyrot@start(#1)}, \texttt{\xyscale@start(#1,#2)}, \texttt{\xyrot@end} and \texttt{\xyscale@end}.

The end & Log

\endinput

% $Log: xyps-r.doc,v $
% Revision 3.11 2011/03/14 20:14:00 krisrose
% Preparing for release 3.8.6.
% %
% Revision 3.10 2010/06/10 18:45:50 krisrose
% Reference to GPL by URL.
% %
% Revision 3.9 2010/05/06 17:46:30 krisrose
% Ross Moore's e-mail address updated.
% %
% Many obsolete files degraded to Historic.
% %
% Revision 3.8 2010/04/16 06:06:52 krisrose
% Preparing for a new release...
4.11.12  xyps-t.doc

Header:

% $Id: xyps-t.doc,v 3.11 2011/03/14 20:14:00 krisrose Exp $
% Xy-pic ``Pattern and Tile extension'' PostScript support.
% Copyright (c) 1993-1997 Ross Moore <ross.moore@mq.edu.au>
% This file is part of the Xy-pic package for graphs and diagrams in TeX.
% See the companion README and INSTALL files for further information.
% Copyright (c) 1991-2011 Kristoffer H. Rose <krisrose@tug.org>
% The Xy-pic package is free software; you can redistribute it and/or modify
% it under the terms of the GNU General Public License as published by the
% Free Software Foundation; either version 2 of the License, or (at your
% option) any later version.
% The Xy-pic package is distributed in the hope that it will be useful, but
% WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
4.11. EXTRA FEATURES WITH PostScript SUPPORT

4.11.13 Patterns and tiles

The included file \texttt{xyps-t.tex} (version 3.11) provides PostScript support for the effects defined in the tile extension described in §2.7.

The PostScript style methods are required, so ensure that they are loaded.

\texttt{\expandafter\ifx\csname xyps-tloaded\endcsname\empty\\message{not reloaded}\endinput \fi}
\texttt{\expandafter\let\csname xyps-tloaded\endcsname=\empty \xycatcodes}

\texttt{\expandafter\ifx\csname xyps-sloaded\endcsname\empty\else\\xyinputorelse@{xyps-s}{\xyerror@{Could not load xyps-s}{}}\fi}
\texttt{\xycatcodes}

This file is loaded and its effects are activated automatically whenever \texttt{\xyoption{tile}} is requested and the current \langle driver\rangle supports PostScript patterns. Should there be any need to turn off this support, the following commands are available; they obey usual \TeX{} groupings.

\texttt{\NoPStiles \texttt{remove PostScript support}}
\texttt{\UsePStiles \texttt{reinstate PostScript.}}

Without PostScript support tile patterns can be expected to be displayed as solid black. These commands are provided primarily for testing and trouble-shooting; \emph{e.g.}, with \langle driver\rangle configurations untested by the authors of \texttt{Xy-pic}, who should be notified of any difficulties.

\textbf{Installation} \hspace{0.5cm} Installation consists of rebinding the macros \texttt{\xypattern@} and \texttt{\newxypattern@} and of reloading the standard patterns to include a description for PostScript. This installation should be delayed till after \texttt{pattern} and the PostScript \langle driver\rangle have been installed.

\texttt{\xydef@\installPSpatterns@{\installPSstyle@}
\let\xypattern@=\xyPSpattern@@
\let\xypatternwarning@=\relax
\xystandardpatterns@
\xycatcodes}

\texttt{\xylet@\UnloadPSpatterns@=\Unloadpattern@}
\texttt{\xydef@\UsePStiles{\installPSpatterns@}}
\texttt{\xydef@\NoPStiles{\UnloadPSpatterns@\relax}}

All effects defined in the tile extension can be implemented using the PostScript dictionary, loaded by \texttt{xyps-ps}. This file provides “generic” PostScript code which is known to work correctly
with most drivers. No attempt is made to provide special code for particular drivers; for driver-specific variations, consult the appropriate \texttt{xy\langle driver\rangle}.doc file.

\begin{verbatim}
\xydef\xyPSnewpattern@#1#2#3#4#5{% 
\expandafter\DN@\expandafter{%csname shape [#1]\endcsname}% 
\DNii@{%relax}\next@\nextii@\newxypattern{#1}{%relax\fi 
\DNii@{#4}\nextii@\empty 
\expandafter\DNii@\expandafter{%csname xyPSshape@#1\endcsname}% 
\expandafter\def\nextii@{%xyPSpattern@{#1}{#2 #3}\else 
\expandafter\DNii@\expandafter{%csname xyPSshape@#1\endcsname}% 
\expandafter\def\nextii@{%xyPSpattern@{#1}{#4}\fi 
\ifx\nextii@\empty\expandafter\def\next@{%\xyPSnotimplemented@'#1'@}% 
\else 
\edef\tmp@{%noexpand\let\expandafter\tmp@\next@\expandafter\nextii@}
\expandafter\tmp@i }
\xydef\xyPSpattern@{%ifPSspecials@\expandafter\xyPSpattern@ 
\edef\preStyle@@{%expandafter\addtostyletoks@\expandafter{%xyPSpre@}}% 
\edef\postStyle@@{%expandafter\addtostyletoks@\expandafter{%xyPSpost@}}% 
\let\modPSboxz@=\modXYstyle@ \xyPS@pattern@@{#1}% 
\toks@={%egroup\let\xy@style@=relax \def\Drop@@}%
\expandafter\addtotoks@\expandafter{%\Drop@@\def\Connect@@}%
\expandafter\addtotoks@\expandafter{%\Connect@@} 
\the\toks@ } 
\xydef\xyPS@pattern@@#1{%xyPSsplitPS@{{#1}sp}{}% 
\endinput
\end{verbatim}

The end & Log

% $Log: xyps-t.doc,v $ 
% Revision 3.11 2011/03/14 20:14:00 krisrose 
% Preparing for release 3.8.6. 
% 
% Revision 3.10 2010/06/10 18:45:50 krisrose 
% Reference to GPL by URL. 
% 
% Revision 3.9 2010/05/06 17:46:30 krisrose 
% Ross Moore’s e-mail address updated. 
% Many obsolete files degraded to Historic. 
% 
% Revision 3.8 2010/04/16 06:06:52 krisrose 
% Preparing for a new release... 
% 
% Revision 3.7 1999/02/16 15:12:50 krisrose 
% Interim release (Y&Y fonts now free). 
% 
% Revision 3.4 1997/05/18 01:13:24 ross 
% Essential bugfixes.
4.11. EXTRA FEATURES WITH PostScript SUPPORT

% Revision 3.3 1996/12/18 10:05:50 ross
% minor improvements to file-loading commands
%
% Revision 3.1 1995/09/05 20:36:33 ross
% Release!
%
% Revision 3.0 1995/07/07 20:13:19 ross
% Major release w/new User’s Guide!
%
% Revision 2.13 1995/07/04 15:04:51 ross
% Ready for release of v3.
%
% NEW for version 3.0 by by Ross Moore.
Chapter 5

Fonts

This chapter presents the sources of the \texttt{Xy-pic} standard fonts used by the kernel for computations and as the default way to draw lines and frames in the DVI output; in addition we include font tables for key fonts where the characters are shown in double size.

(In the past, Y&Y Inc. gracefully produced high quality PostScript Type1 fonts of most of these, however, these are no longer needed as the FontForge project and MF2PT1 script together solve the task automatically rather well.)

\textbf{To Do:} Rewrite the \texttt{METAFONT} code to \texttt{mft} format with font tables.

5.1 Semidirectional font

These are fonts with symmetric characters in $128 \times 2$ directions created using the driver file \texttt{xyd2.mf}. The standard repertoire includes only one such font: \texttt{xydash10}.

5.1.1 \texttt{xyd2.mf}

\begin{verbatim}
1 \% $Id: xyd2.mf,v 3.10 2010/06/10 18:45:50 krisrose Exp $
2 \%
3 \% XYD2: generate characters of `Xy-pic SemiDirectional' font.
4 \% Copyright (c) 1992 Kristoffer H. Rose <krisrose@tug.org>
5 \%
6 \% This file is part of the Xy-pic macro package.
7 \%
8 \% The Xy-pic macro package is free software; you can redistribute it and/or
9 \% modify it under the terms of the GNU General Public License as published by
10 \% the Free Software Foundation; either version 2 of the License, or (at your
11 \% option) any later version.
12 \%
13 \% The Xy-pic macro package is distributed in the hope that it will be
14 \% useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
15 \% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General
16 \% Public License for more details.
17 \%
18 \% You should have received a copy of the GNU General Public License along
19 \% with this macro package; if not, see http://www.gnu.org/licenses/.
20 \%
21 \% As a special exception, you may use this file and all files derived
22 \% from it without restriction. This special exception was added with
23 \% version 3.7 of Xy-pic.
24 \%
25 \% CONTENTS: Generate `semidirectional' characters from 0..127 by calling
26 \% `chartowards(Code, Dx, Dy)' with
27 \%
28 \% Code:  0  30  31  63  95  127
29 \% Dx,Dy: 31/32,1 1/32,1 0,-1 1,-1 1,0 1,1
\end{verbatim}

543
5.1.2  xydash10

% $Id: xydash10.mf,v 3.9 2010/06/10 18:45:50 krisrose Exp $
% XyDASH10: dashes for Xy-pic at 10 point.
% Copyright (c) 1991,1992  Kristoffer H. Rose <krisrose@tug.org>
% This file is part of the Xy-pic macro package.
% The Xy-pic macro package is free software; you can redistribute it and/or
% modify it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either version 2 of the License, or (at your
% option) any later version.
% The Xy-pic macro package is distributed in the hope that it will be
% useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General
% Public License for more details.
### 5.1. SEMIDIRECTIONAL FONT

<table>
<thead>
<tr>
<th></th>
<th>'0'</th>
<th>'1'</th>
<th>'2'</th>
<th>'3'</th>
<th>'4'</th>
<th>'5'</th>
<th>'6'</th>
<th>'7'</th>
</tr>
</thead>
<tbody>
<tr>
<td>'00x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'02x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'03x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'04x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'05x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'06x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'07x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'10x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'11x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'12x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'13x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'14x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'15x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'16x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'17x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| '8' | '9' | 'A' | 'B' | 'C' | 'D' | 'E' | 'F' |

---

Figure 5.1: Font table for xydash10 scaled 2000.
5.2 Directional font

These are fonts with characters in 128 directions created using the driver file \texttt{xyd.mf}. The standard repertoire includes several such fonts.

5.2.1 \texttt{xyd.mf}

These are fonts with characters in 128 directions created using the driver file \texttt{xyd.mf}. The standard repertoire includes several such fonts.
5.2. **DIRECTIONAL FONT**

5.2.2 **xyatip10**

1. % $Id: xyatip10.mf,v 3.9 2010/06/10 18:45:50 krisrose Exp $
2. %
3. % XYATIP10: upper arrow tips for Xy-pic at 10 point "technical style".
4. % Copyright (c) 1991-1998 Kristoffer H. Rose <krisrose@tug.org>
5. %
6. % This file is part of the Xy-pic macro package.
7. %
8. % The Xy-pic macro package is free software; you can redistribute it and/or
9. % modify it under the terms of the GNU General Public License as published by
10. % the Free Software Foundation; either version 2 of the License, or (at your
11. % option) any later version.
12. %
<table>
<thead>
<tr>
<th>'0'</th>
<th>'1'</th>
<th>'2'</th>
<th>'3'</th>
<th>'4'</th>
<th>'5'</th>
<th>'6'</th>
<th>'7'</th>
</tr>
</thead>
<tbody>
<tr>
<td>'00x'</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>'01x'</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>'02x'</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>'03x'</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>'04x'</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>'05x'</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>'06x'</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>'07x'</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>'08x'</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>'09x'</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>'11x'</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>'12x'</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>'13x'</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>'14x'</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>'15x'</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>'16x'</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>'17x'</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
</tbody>
</table>

Figure 5.2: Font table for xyatip10 scaled 2000.
5.2. DIRECTIONAL FONT

<table>
<thead>
<tr>
<th>'0'</th>
<th>'1'</th>
<th>'2'</th>
<th>'3'</th>
<th>'4'</th>
<th>'5'</th>
<th>'6'</th>
<th>'7'</th>
</tr>
</thead>
<tbody>
<tr>
<td>'00x'</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>'01x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
</tr>
<tr>
<td>'02x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
</tr>
<tr>
<td>'03x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
</tr>
<tr>
<td>'04x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
</tr>
<tr>
<td>'05x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
</tr>
<tr>
<td>'06x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
</tr>
<tr>
<td>'07x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
</tr>
<tr>
<td>'10x'</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>'11x'</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>'12x'</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>'13x'</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>'14x'</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>'15x'</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>'16x'</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>'17x'</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>'8'</td>
<td>'9'</td>
<td>'A'</td>
<td>'B'</td>
<td>'C'</td>
<td>'D'</td>
<td>'E'</td>
<td>'F'</td>
</tr>
</tbody>
</table>

Figure 5.3: Font table for xybtip10 scaled 2000.
5.2.4 xybsql10

% XYBSQL10: lower squiggles/quarter circles for Xy-pic at 10 point.
% Copyright (c) 1992-2011 Kristoffer H. Rose <krisrose@tug.org>
% 2011 Daniel Mullner <http://math.stanford.edu/~muellner>
% This file is part of the Xy-pic macro package.
% The Xy-pic macro package is free software; you can redistribute it and/or
% modify it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either version 2 of the License, or (at your
% option) any later version.
% The Xy-pic macro package is distributed in the hope that it will be
% useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General
% Public License for more details.
% You should have received a copy of the GNU General Public License along
% with this macro package; if not, see http://www.gnu.org/licenses/.
% As a special exception, you may use this file and all files derived
% from it without restriction.  This special exception was added with
% version 3.7 of Xy-pic.
% CONTENTS: Squiggles are quarter circles with secant in all directions.
% Each has zero bounding box around the start point, i.e., from which the
% quarter circle 'turns left' ... this can be illustrated as follows:
% Code:  [-1] 15 31 47 63 79 95 111 127
5.2. DIRECTIONAL FONT

<table>
<thead>
<tr>
<th>'0'</th>
<th>'1'</th>
<th>'2'</th>
<th>'3'</th>
<th>'4'</th>
<th>'5'</th>
<th>'6'</th>
<th>'7'</th>
</tr>
</thead>
<tbody>
<tr>
<td>'00x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'01x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'02x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'03x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'04x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'05x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'06x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'07x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'10x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'11x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'12x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'13x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'14x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'15x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'16x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'17x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'8'</td>
<td></td>
<td>'9'</td>
<td>'A'</td>
<td>'B'</td>
<td>'C'</td>
<td>'D'</td>
<td>'E'</td>
</tr>
</tbody>
</table>

Figure 5.4: Font table for xybsql10 scaled 2000.
5.3 Special fonts

The kernel circle construction uses the following special font with 1/8 circle segments at various sizes (not scaled):

5.3.1 xycirc10

---

% $Id: xycirc10.mf,v 3.13 2011/03/14 20:14:00 krisrose Exp $
% YXCIRC10: 1/8 circles with varying radii for Xy-pic at 10 point.
% Copyright (c) 1992,2011 Kristoffer H. Rose <krisrose@tug.org>
% 2011 Daniel Müller <http://www.math.uni-bonn.de/people/muellner>
% This file is part of the Xy-pic macro package.
% The Xy-pic macro package is free software; you can redistribute it and/or
% modify it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either version 2 of the License, or (at your
% option) any later version.
% The Xy-pic macro package is distributed in the hope that it will be
% useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General
% Public License for more details.
% You should have received a copy of the GNU General Public License along
% with this macro package; if not, see http://www.gnu.org/licenses/.
% As a special exception, you may use this file and all files derived
% from it without restriction. This special exception was added with
% version 3.7 of Xy-pic.
---

CONTENTS: 1/8 circles with radii from 1 to 32pt dissected into the 1/8
circle segments shown below

6 5

........
7 .. | .. 4 |
:
:
::radius

:
5.3. SPECIAL FONTS

<table>
<thead>
<tr>
<th></th>
<th>'0'</th>
<th>'1'</th>
<th>'2'</th>
<th>'3'</th>
<th>'4'</th>
<th>'5'</th>
<th>'6'</th>
<th>'7'</th>
</tr>
</thead>
<tbody>
<tr>
<td>'00x'</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'01x'</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'02x'</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'03x'</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'04x'</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'05x'</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'06x'</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'07x'</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'10x'</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'11x'</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'12x'</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'13x'</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'14x'</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'15x'</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'16x'</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'17x'</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'0x'</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'1x'</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'2x'</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'3x'</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'4x'</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'5x'</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'6x'</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'7x'</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
</tbody>
</table>

Figure 5.5: Font table for xycirc10.
Reference point --> ___:___|___|___|___:_____|
  : | | | : 
  : | | | : 
  0 .! | !. 3
  ...|...
  1 2

such that each group of 8 characters cc+0 to cc+7 constitute a full circle
with the given radius when typeset on the same baseline (as shown).

The bounding box of each segment is the vertical slice of the unit square
around the circle.

The radius is given for each character group g = cc mod 8 by the formula

radius [pt] = g+1, if 0 <= g < 8
2(g-8) + 10, if 8 <= g < 12  [= 2g-6]
4(g-12) + 20, if 12 <= g < 16  [= 4g-28]

font_identifier "Xycirc"; font_size 10pt;
mode_setup;

rulew# = .4pt#; define_whole_blacker_pixels(rulew); % line thickness
TESTING...we redefine openit because the characters extend far to the
left of the bounding box!
def openit = %let echar = endchar; def endchar = echar; stop ". " enddef;
openwindow currentwindow from origin to (1000,800) at (-200,300)
enddef;

% FONT.

Font dimension 8 is the rule thickness (cf. The TeXbook, app.G)
font_coding_scheme:="xycirc.enc";
fontdimen 8: rulew#; % default_rule_thickness
(It's unnecessary to give the exact control points. MetaFont
automatically chooses a good approximation to a circular arc.)
path eighthcircle;
eighthcircle = right{up} .. {left+up}(right+up)/sqrt2;

Use the following macro that generates an entire group from character cc
to cc+7 with radius:
def makeg(expr g,radius) =
  major# := radius/sqrt2;
  minor# := radius - major#;
  full# := radius;
define_pixels(minor,major,full);
  path arc;
  arc = eighthcircle scaled (full + rulew/2)
    & halfcircle scaled rulew shifted (full+right) rotated 45
    & reverse eighthcircle scaled (full - rulew/2)
    & halfcircle scaled rulew rotated 180 shifted (full+right)
    & cycle;
  beginchar(8g ,minor#,full#,full#);
  fill arc rotated 180 shifted (full+right);
  endchar;
beginchar(8g+1,major#,full#,full#);
fill arc rotated 228 shifted (major+right);
endchar;
beginchar(8g+2,major#,full#,full#);
fill arc rotated 270;
endchar;
beginchar(8g+3,minor#,full#,full#);
fill arc rotated 315 shifted (major+left);
5.4. Optional fonts

The “tips” extension uses the following fonts by default (we only show the 11 point variant).
5.4.1 \texttt{xycmat10.mf}

\begin{verbatim}
% $Id: xycmat10.mf,v 3.9 2010/06/10 18:45:50 krisrose Exp $
%
% XYCMAT10: upper arrow tips for Xy-pic at 10 point "Computer Modern style".
% Copyright (c) 1994-1996 Kristoffer H. Rose <krisrose@tug.org>
%
% This file is part of the Xy-pic macro package.
%
% The Xy-pic macro package is free software; you can redistribute it and/or
% modify it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either version 2 of the License, or (at your
% option) any later version.
%
% The Xy-pic macro package is distributed in the hope that it will be
% useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General
% Public License for more details.
%
% You should have received a copy of the GNU General Public License along
% with this macro package; if not, see http://www.gnu.org/licenses/.
%
% As a special exception, you may use this file and all files derived
% from it without restriction. This special exception was added with
% version 3.7 of Xy-pic.
%
font_identifier "XYCMAT"; font_size 10pt#
mode_setup;
input xycm
input xyatip

% $Log: xycmat10.mf,v $
% Revision 3.9 2010/06/10 18:45:50 krisrose
% Reference to GPL by URL.
%
% Revision 3.8 2010/04/16 06:06:52 krisrose
% Preparing for a new release...
%
% Revision 3.7 1999/02/16 15:12:50 krisrose
% Interim release (Y&Y fonts now free).
%
% Revision 3.3 1996/12/19 03:31:56 krisrose
% Maintenance release
%
% Revision 3.0 1995/07/07 20:14:21 kris
% Major release w/new User's Guide!
%
% Revision 2.13 1995/07/04 15:11:17 kris
% Ready to release v3?
%
% Revision 2.12 1994/10/25 11:34:25 kris
% Interim release just before v3 [works with AMS-LaTeX 1.2]...
%
% Recreated from xyatip10.mf (Revision 2.7 1992/12/14 01:41:26 kris).
%
% Revision 2.6 1993/10/21 21:21:24 kris
% NEW for 2.7...
%
% Based on xyatip10.mf [2.6] and Knuth's SYMBOL.MF.
\end{verbatim}

5.4.2 \texttt{xycmbt10.mf}

\begin{verbatim}
% $Id: xycmbt10.mf,v 3.9 2010/06/10 18:45:50 krisrose Exp $
%
% XCYMBT10: lower arrow tips for Xy-pic at 10 point "Computer Modern style".
% Copyright (c) 1994-1996 Kristoffer H. Rose <krisrose@tug.org>
%
% This file is part of the Xy-pic macro package.
%
% The Xy-pic macro package is free software; you can redistribute it and/or
\end{verbatim}
5.4. OPTIONAL FONTS

% modify it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either version 2 of the License, or (at your
% option) any later version.
%
% The Xy-pic macro package is distributed in the hope that it will be
% useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General
% Public License for more details.
%
% You should have received a copy of the GNU General Public License along
% with this macro package; if not, see http://www.gnu.org/licenses/.
%
% As a special exception, you may use this file and all files derived
% from it without restriction. This special exception was added with
% version 3.7 of Xy-pic.
%
font_identifier "XYCMBT"; font_size 10pt#;
mode_setup;
input xycm
input xybtip

% $Log: xycmbt10.mf,v $% Revision 3.9 2010/06/10 18:45:50 krisrose% Reference to GPL by URL.
%
% Revision 3.8 2010/04/16 06:06:52 krisrose% Preparing for a new release...
%
% Revision 3.7 1999/02/16 15:12:50 krisrose% Interim release (V&Y fonts now free).
%
% Revision 3.3 1996/12/19 03:31:56 krisrose% Maintenance release
%
% Revision 3.0 1995/07/07 20:14:21 kris% Major release w/new User's Guide!
%
% Revision 2.13 1995/07/04 15:11:17 kris% Ready to release v3?
%
% Revision 2.12 1994/10/25 11:34:25 kris% Interim release just before v3 [works with AMS-LaTeX 1.2]...
%
% Recreated from xybtip10.mf (Revision 2.7 1992/12/14 01:41:26 kris).
%
% Revision 2.6 1993/10/21 21:21:24 kris% NEW for 2.7...
%
% Based on xyatip10.mf [2.6] and Knuth's SYMBOL.MF.

5.4.3 xycmat11

% $Id: xycmat11.mf,v 3.8 2010/06/10 18:45:50 krisrose Exp $% XCMAT11: upper arrow tips for Xy-pic at 11 point "Computer Modern style".
% Copyright (c) 1995 Kristoffer H. Rose <kris@diku.dk>
%
% This file is part of the Xy-pic macro package.
%
% The Xy-pic macro package is free software; you can redistribute it and/or
% modify it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either version 2 of the License, or (at your
% option) any later version.
%
% The Xy-pic macro package is distributed in the hope that it will be
% useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General
% Public License for more details.
%
% You should have received a copy of the GNU General Public License along
Figure 5.6: Font table for xycmat11 scaled 2000.
5.4. OPTIONAL FONTS

<table>
<thead>
<tr>
<th>'0'</th>
<th>'1'</th>
<th>'2'</th>
<th>'3'</th>
<th>'4'</th>
<th>'5'</th>
<th>'6'</th>
<th>'7'</th>
</tr>
</thead>
<tbody>
<tr>
<td>'00x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'0x'</td>
</tr>
<tr>
<td>'01x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'1x'</td>
</tr>
<tr>
<td>'02x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'2x'</td>
</tr>
<tr>
<td>'03x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'3x'</td>
</tr>
<tr>
<td>'04x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'4x'</td>
</tr>
<tr>
<td>'05x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'5x'</td>
</tr>
<tr>
<td>'06x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'6x'</td>
</tr>
<tr>
<td>'07x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'7x'</td>
</tr>
<tr>
<td>'08x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'8x'</td>
</tr>
<tr>
<td>'09x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'9x'</td>
</tr>
<tr>
<td>'10x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'Ax'</td>
</tr>
<tr>
<td>'11x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'Bx'</td>
</tr>
<tr>
<td>'12x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'Cx'</td>
</tr>
<tr>
<td>'13x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'Dx'</td>
</tr>
<tr>
<td>'14x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'Ex'</td>
</tr>
<tr>
<td>'15x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'Fx'</td>
</tr>
<tr>
<td>'16x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'Gx'</td>
</tr>
<tr>
<td>'17x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'Hx'</td>
</tr>
</tbody>
</table>

Figure 5.7: Font table for xycmbt11 scaled 2000.

5.4.5 xycmat12.mf

<table>
<thead>
<tr>
<th>'0'</th>
<th>'1'</th>
<th>'2'</th>
<th>'3'</th>
<th>'4'</th>
<th>'5'</th>
<th>'6'</th>
<th>'7'</th>
</tr>
</thead>
<tbody>
<tr>
<td>'00x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'0x'</td>
</tr>
<tr>
<td>'01x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'1x'</td>
</tr>
<tr>
<td>'02x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'2x'</td>
</tr>
<tr>
<td>'03x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'3x'</td>
</tr>
<tr>
<td>'04x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'4x'</td>
</tr>
<tr>
<td>'05x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'5x'</td>
</tr>
<tr>
<td>'06x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'6x'</td>
</tr>
<tr>
<td>'07x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'7x'</td>
</tr>
<tr>
<td>'08x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'8x'</td>
</tr>
<tr>
<td>'09x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'9x'</td>
</tr>
<tr>
<td>'10x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'Ax'</td>
</tr>
<tr>
<td>'11x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'Bx'</td>
</tr>
<tr>
<td>'12x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'Cx'</td>
</tr>
<tr>
<td>'13x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'Dx'</td>
</tr>
<tr>
<td>'14x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'Ex'</td>
</tr>
<tr>
<td>'15x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'Fx'</td>
</tr>
<tr>
<td>'16x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'Gx'</td>
</tr>
<tr>
<td>'17x'</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>'Hx'</td>
</tr>
</tbody>
</table>
CHAPTER 5. FONTS

5.4.6  xycmbt12.mf

% $Id: xycmbt12.mf,v 3.8 1995/07/07 20:14:21 kris Exp krisrose$
% XYCMBT12: lower arrow tips for Xy-pic at 12 point "Computer Modern style".
% Copyright (c) 1994-1996 Kristoffer H. Rose <kris@diku.dk>
% This file is part of the Xy-pic macro package.
% The Xy-pic macro package is free software; you can redistribute it and/or
% modify it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either version 2 of the License, or (at your
% option) any later version.
% The Xy-pic macro package is distributed in the hope that it will be
% useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General
% Public License for more details.
% You should have received a copy of the GNU General Public License along
% with this macro package; if not, see http://www.gnu.org/licenses/.
% As a special exception, you may use this file and all files derived
% from it without restriction. This special exception was added with
% version 3.7 of Xy-pic.
% Based on xycmat10.mf,v 3.0 1995/07/07 20:14:21 kris Exp krisrose

% Revision 3.8 1995/07/07 20:14:21 krisrose
% Reference to GPL by URL.
% Revision 3.7 1999/02/16 15:12:50 krisrose
% Interim release (Y&Y fonts now free).
% Revision 3.3 1996/12/19 04:12:13 krisrose
% New for this maintenance release.
% Based on xycmat10.mf,v 3.0 1995/07/07 20:14:21 kris Exp krisrose

% $Log: xycm12.mf,v 3.8 2010/06/10 18:45:50 krisrose$
% Revision 3.8 2010/06/10 18:45:50 krisrose
% Reference to GPL by URL.
% Revision 3.7 2010/06/10 18:45:50 krisrose
% Internim release (Y&Y fonts now free).
% Revision 3.3 1996/12/19 04:12:13 krisrose
% New for this maintenance release.
% Based on xycmat10.mf,v 3.0 1995/07/07 20:14:21 kris Exp krisrose

% $Id: xycm12.mf,v 3.8 2010/06/10 18:45:50 krisrose$
% Revision 3.8 2010/06/10 18:45:50 krisrose
% Reference to GPL by URL.
% Revision 3.7 2010/06/10 18:45:50 krisrose
% Internim release (Y&Y fonts now free).
% Revision 3.3 1996/12/19 04:12:13 krisrose
% New for this maintenance release.
% Based on xycmat10.mf,v 3.0 1995/07/07 20:14:21 kris Exp krisrose

% $Id: xycm12.mf,v 3.8 2010/06/10 18:45:50 krisrose$
% Revision 3.8 2010/06/10 18:45:50 krisrose
% Reference to GPL by URL.
% Revision 3.7 2010/06/10 18:45:50 krisrose
% Internim release (Y&Y fonts now free).
% Revision 3.3 1996/12/19 04:12:13 krisrose
% New for this maintenance release.
% Based on xycmat10.mf,v 3.0 1995/07/07 20:14:21 kris Exp krisrose

% $Id: xycm12.mf,v 3.8 2010/06/10 18:45:50 krisrose$
% Revision 3.8 2010/06/10 18:45:50 krisrose
% Reference to GPL by URL.
% Revision 3.7 2010/06/10 18:45:50 krisrose
% Internim release (Y&Y fonts now free).
% Revision 3.3 1996/12/19 04:12:13 krisrose
% New for this maintenance release.
% Based on xycmat10.mf,v 3.0 1995/07/07 20:14:21 kris Exp krisrose

% $Id: xycm12.mf,v 3.8 2010/06/10 18:45:50 krisrose$
% Revision 3.8 2010/06/10 18:45:50 krisrose
% Reference to GPL by URL.
% Revision 3.7 2010/06/10 18:45:50 krisrose
% Internim release (Y&Y fonts now free).
% Revision 3.3 1996/12/19 04:12:13 krisrose
% New for this maintenance release.
% Based on xycmat10.mf,v 3.0 1995/07/07 20:14:21 kris Exp krisrose

% $Id: xycm12.mf,v 3.8 2010/06/10 18:45:50 krisrose$
% Revision 3.8 2010/06/10 18:45:50 krisrose
% Reference to GPL by URL.
% Revision 3.7 2010/06/10 18:45:50 krisrose
% Internim release (Y&Y fonts now free).
% Revision 3.3 1996/12/19 04:12:13 krisrose
% New for this maintenance release.
% Based on xycmat10.mf,v 3.0 1995/07/07 20:14:21 kris Exp krisrose

% $Id: xycm12.mf,v 3.8 2010/06/10 18:45:50 krisrose$
% Revision 3.8 2010/06/10 18:45:50 krisrose
% Reference to GPL by URL.
% Revision 3.7 2010/06/10 18:45:50 krisrose
% Internim release (Y&Y fonts now free).
% Revision 3.3 1996/12/19 04:12:13 krisrose
% New for this maintenance release.
% Based on xycmat10.mf,v 3.0 1995/07/07 20:14:21 kris Exp krisrose

% $Id: xycm12.mf,v 3.8 2010/06/10 18:45:50 krisrose$
% Revision 3.8 2010/06/10 18:45:50 krisrose
% Reference to GPL by URL.
% Revision 3.7 2010/06/10 18:45:50 krisrose
% Internim release (Y&Y fonts now free).
% Revision 3.3 1996/12/19 04:12:13 krisrose
% New for this maintenance release.
% Based on xycmat10.mf,v 3.0 1995/07/07 20:14:21 kris Exp krisrose

% $Id: xycm12.mf,v 3.8 2010/06/10 18:45:50 krisrose$
% Revision 3.8 2010/06/10 18:45:50 krisrose
% Reference to GPL by URL.
% Revision 3.7 2010/06/10 18:45:50 krisrose
% Internim release (Y&Y fonts now free).
% Revision 3.3 1996/12/19 04:12:13 krisrose
% New for this maintenance release.
% Based on xycmat10.mf,v 3.0 1995/07/07 20:14:21 kris Exp krisrose

% $Id: xycm12.mf,v 3.8 2010/06/10 18:45:50 krisrose$
% Revision 3.8 2010/06/10 18:45:50 krisrose
% Reference to GPL by URL.
% Revision 3.7 2010/06/10 18:45:50 krisrose
% Internim release (Y&Y fonts now free).
% Revision 3.3 1996/12/19 04:12:13 krisrose
% New for this maintenance release.
% Based on xycmat10.mf,v 3.0 1995/07/07 20:14:21 kris Exp krisrose

% $Id: xycm12.mf,v 3.8 2010/06/10 18:45:50 krisrose$
% Revision 3.8 2010/06/10 18:45:50 krisrose
% Reference to GPL by URL.
% Revision 3.7 2010/06/10 18:45:50 krisrose
% Internim release (Y&Y fonts now free).
% Revision 3.3 1996/12/19 04:12:13 krisrose
% New for this maintenance release.
% Based on xycmat10.mf,v 3.0 1995/07/07 20:14:21 kris Exp krisrose
The following fonts are made to match with the Euler math fonts (again we only show the 11 point variant).

### 5.4.7 xyeuat10.mf

```
\font_identifier "XYEUAT"; font_size 10pt#
mode_setup;
```

### 5.4.8 xyeubt10.mf

```
\font_identifier "XYEUBT"; font_size 10pt#
```
Figure 5.8: Font table for xyeuat11 scaled 2000.
5.4. OPTIONAL FONTS

% XYEUAT11: upper arrow tips for Xy-pic at 11 point "Euler style".
% Copyright (c) 1994-1996 Kristoffer H. Rose <kris@diku.dk>
% This file is part of the Xy-pic macro package.
% The Xy-pic macro package is free software; you can redistribute it and/or
% modify it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either version 2 of the License, or (at your
% option) any later version.
% The Xy-pic macro package is distributed in the hope that it will be
% useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General
% Public License for more details.
% You should have received a copy of the GNU General Public License along
% with this macro package; if not, see http://www.gnu.org/licenses/.
%
% As a special exception, you may use this font file and all files derived
% from it without restriction. This special exception was added with
% version 3.7 of Xy-pic.
%
font_identifier "XYEUAT"; font_size 11pt#
mode_setup;
input xyeuler
input xyatip

% $Log: xyeuat11.mf,v $
% Revision 3.8 2010/06/10 18:45:50 krisrose
% Reference to GPL by URL.
% Revision 3.7 1999/02/16 15:12:50 krisrose
% Interim release (Y&Y fonts now free).
% Revision 3.3 1996/12/19 04:12:13 krisrose
% New for this maintenance release.
% Based on xycmat10.mf,v 3.0 1995/07/07 20:14:21 kris

5.4.10  xyeubt11

% $Id: xyeubt11.mf,v 3.8 2010/06/10 18:45:50 krisrose Exp $
% XYEUBT11: lower arrow tips for Xy-pic at 11 point "Euler style".
% Copyright (c) 1994-1996 Kristoffer H. Rose <kris@diku.dk>
% This file is part of the Xy-pic macro package.
% The Xy-pic macro package is free software; you can redistribute it and/or
% modify it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either version 2 of the License, or (at your
% option) any later version.
% The Xy-pic macro package is distributed in the hope that it will be
% useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General
% Public License for more details.
% You should have received a copy of the GNU General Public License along
% with this macro package; if not, see http://www.gnu.org/licenses/.
% As a special exception, you may use this font file and all files derived
% from it without restriction. This special exception was added with
% version 3.7 of Xy-pic.
%
font_identifier "XYEUBT"; font_size 11pt#
mode_setup;
input xyeuler
input xybtip
<table>
<thead>
<tr>
<th></th>
<th>'0'</th>
<th>'1'</th>
<th>'2'</th>
<th>'3'</th>
<th>'4'</th>
<th>'5'</th>
<th>'6'</th>
<th>'7'</th>
</tr>
</thead>
<tbody>
<tr>
<td>'00x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>\textsuperscript{0x}</td>
</tr>
<tr>
<td>'01x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>\textsuperscript{1x}</td>
</tr>
<tr>
<td>'02x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>\textsuperscript{2x}</td>
</tr>
<tr>
<td>'03x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>\textsuperscript{3x}</td>
</tr>
<tr>
<td>'04x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>\textsuperscript{4x}</td>
</tr>
<tr>
<td>'05x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>\textsuperscript{5x}</td>
</tr>
<tr>
<td>'06x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>\textsuperscript{6x}</td>
</tr>
<tr>
<td>'07x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>\textsuperscript{7x}</td>
</tr>
<tr>
<td>'10x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>\textsuperscript{8}</td>
</tr>
<tr>
<td>'11x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>\textsuperscript{9}</td>
</tr>
<tr>
<td>'12x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A</td>
</tr>
<tr>
<td>'13x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B</td>
</tr>
<tr>
<td>'14x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C</td>
</tr>
<tr>
<td>'15x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>D</td>
</tr>
<tr>
<td>'16x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E</td>
</tr>
<tr>
<td>'17x'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>F</td>
</tr>
</tbody>
</table>

Figure 5.9: Font table for xyeubt11 scaled 2000.

5.4.11 xyeuat12.mf

$Log: xyeuat12.mf,v $ % $Log: xyeubt11.mf,v $ % $Log: xyeubt10.mf,v $ % Revision 3.8 2010/06/10 18:45:50 krisrose % Reference to GPL by URL. % Revision 3.7 1999/02/16 15:12:50 krisrose % Interim release (Y&Y fonts now free). % Revision 3.3 1996/12/19 04:12:13 krisrose % New for this maintenance release. % Based on xycmbt10.mf,v 3.0 1995/07/07 20:14:21 kris Exp

% XYEUT12: upper arrow tips for Xy-pic at 12 point "Euler style". % Copyright (c) 1994-1996 Kristoffer H. Rose <kris@diku.dk>
% This file is part of the Xy-pic macro package. % The Xy-pic macro package is free software; you can redistribute it and/or % modify it under the terms of the GNU General Public License as published by % the Free Software Foundation; either version 2 of the License, or (at your % option) any later version. % The Xy-pic macro package is distributed in the hope that it will be % useful, but WITHOUT ANY WARRANTY; without even the implied warranty of % MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General % Public License for more details. % You should have received a copy of the GNU General Public License along % with this macro package; if not, see http://www.gnu.org/licenses/. % As a special exception, you may use this font file and all files derived
Finally, Jeremy Gibbons has contributed the following fonts meant to look nice with the Lucida family of fonts (we show the 11 point variant).

5.4.13 xylu.mf

Finally, Jeremy Gibbons has contributed the following fonts meant to look nice with the Lucida family of fonts (we show the 11 point variant).
5.4.14 xyatri.mf

% $Id: xyatri.mf,v 3.4 2011/03/14 20:14:00 krisrose Exp $
% xyatri, for use with Lucida
% Copyright (c) 2007 Jeremy Gibbons <jeremy.gibbons@comlab.ox.ac.uk>
% This file is part of the Xy-pic package for graphs and diagrams in TeX.
% Copyright (c) 1991-2011 Kristoffer H. Rose <krisrose@tug.org>
% The Xy-pic macro package is free software; you can redistribute it and/or
% modify it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either version 2 of the License, or (at your
% option) any later version.
% The Xy-pic macro package is distributed in the hope that it will be
% useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General
% Public License for more details.
% You should have received a copy of the GNU General Public License along
% with this macro package; if not, see http://www.gnu.org/licenses/.
% As a special exception, you may use this file and all files derived
% from it without restriction. This special exception was added with
% version 3.7 of Xy-pic.

tipo# = 2segw#; define_pixels(tipo);  \% tip overshoot
segw# = .2pt#; define_whole_blackers_pixels(segw);  \% segment thickness
segl# = 1/5 designsize - tipo#; define_pixels(segl);  \% segment length
segx# = 1/4 designsize; define_pixels(segx);  \% segment height
indent = 1/4;  \% 1 = barbs, 0 = triangular tails

% $Log: xyatri.mf,v $  
% Revision 3.4 2011/03/14 20:14:00 krisrose  
% Preparing for release 3.8.6.
% Revision 3.3 2010/06/10 18:45:50 krisrose  
% Reference to GPL by URL.
% Revision 3.2 2010/04/17 14:45:48 krisrose  
% Generate and extract Type1 fonts.
% Revision 3.1 2010/04/17 04:19:41 krisrose  
% Integrated xytri tips by Jeremy Gibbons.
% based on xycm.
def chartowards(expr cc, dx, dy) =
    beginchar(cc, 0, 0, 0);
    a := angle(dx, dy);
    z0 = (tipo, 0) rotated a;  % tip of tip
    z1 = (-segl, 0) rotated a;  % center of tail
    z2 = (-segl, 1/2 segx) rotated a;  % tail ends
    z3 = indent \[z1, z0\];  % back of tip
    fill z2--z0--z3--cycle;
    penlabels(0, 1, 2, 3);
    endchar
enddef;
def openit = openwindow currentwindow
from origin to (screen_cols, screen_rows) at (-200, 300) enddef;
input xyd
bye.

5.4.15 xybtri.mf

% $Log: xybtri.mf,v $ % Revision 3.4 2011/03/14 20:14:00 krisrose
% Preparing for release 3.8.6.
% Revision 3.3 2010/06/10 18:45:50 krisrose
% Reference to GPL by URL.
% Revision 3.2 2010/04/17 14:45:48 krisrose
% Generate and extract Type1 fonts.
% Revision 3.1 2010/04/17 04:19:41 krisrose
% Integrated xylu tips by Jeremy Gibbons.
% Based on xyatip (triangular rather than curvilinear barbs)

def chartowards(expr cc, dx, dy) =
    beginchar(cc, 0, 0, 0);
    a := angle(dx, dy);
    z0 = (tipo, 0) rotated a;  % tip of tip
    z1 = (-segl, 0) rotated a;  % center of tail
    z2 = (-segl, 1/2 segx) rotated a;  % tail ends
    z3 = indent \[z1, z0\];  % back of tip
    fill z2--z0--z3--cycle;
    penlabels(0, 1, 2, 3);
    endchar
enddef;
def openit = openwindow currentwindow
CHAPTER 5. FONTS

from origin to (screen_cols,screen_rows) at (-200,300) enddef;
input xyd
bye.
% $Log: xybtri.mf,v $ % Revision 3.4 2011/03/14 20:14:00 krisrose % Preparing for release 3.8.6.
%
% Revision 3.3 2010/06/10 18:45:50 krisrose % Reference to GPL by URL.
%
% Revision 3.2 2010/04/17 14:45:48 krisrose % Generate and extract Type1 fonts.
%
% Revision 3.1 2010/04/17 04:19:41 krisrose % Integrated xylu tips by Jeremy Gibbons.
%
% based on xytip (triangular rather than curvilinear barbs)

5.4.16 xyluat10.mf

1 % $Id: xyluat10.mf,v 3.4 2011/03/14 20:14:00 krisrose Exp $ % xyluat10, for use with Lucida

% Copyright (c) 2007 Jeremy Gibbons  <jeremy.gibbons@comlab.ox.ac.uk>
% This file is part of the Xy-pic package for graphs and diagrams in TeX.
% Copyright (c) 1991-2011 Kristoffer H. Rose <krisrose@tug.org>
% The Xy-pic macro package is free software; you can redistribute it and/or
% modify it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either version 2 of the License, or (at your
% option) any later version.
% The Xy-pic macro package is distributed in the hope that it will be
% useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General
% Public License for more details.
% You should have received a copy of the GNU General Public License along
% with this macro package; if not, see http://www.gnu.org/licenses/.
%
% As a special exception, you may use this file and all files derived
% from it without restriction. This special exception was added with
% version 3.7 of Xy-pic.

font_identifier "XYLUAT"; font_size 10pt#;
mode_setup;
input xylu
input xyatri
%
% $Log: xyluat10.mf,v $ % Revision 3.4 2011/03/14 20:14:00 krisrose % Preparing for release 3.8.6.
%
% Revision 3.3 2010/06/10 18:45:50 krisrose % Reference to GPL by URL.
%
% Revision 3.2 2010/04/17 14:45:48 krisrose % Generate and extract Type1 fonts.
%
% Revision 3.1 2010/04/17 04:19:41 krisrose % Integrated xylu tips by Jeremy Gibbons.
%
% based on xycmat10

5.4.17 xylubt10.mf

1 % $Id: xylubt10.mf,v 3.4 2011/03/14 20:14:00 krisrose Exp $ %
5.4. OPTIONAL FONTS

% xylubt10, for use with Lucida.
% Copyright (c) 2007 Jeremy Gibbons <jeremy.gibbons@comlab.ox.ac.uk>
% This file is part of the Xy-pic package for graphs and diagrams in TeX.
% Copyright (c) 1991-2011 Kristoffer H. Rose <krisrose@tug.org>
% The Xy-pic macro package is free software; you can redistribute it and/or
% modify it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either version 2 of the License, or (at your
% option) any later version.
% The Xy-pic macro package is distributed in the hope that it will be
% useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General
% Public License for more details.
% You should have received a copy of the GNU General Public License along
% with this macro package; if not, see http://www.gnu.org/licenses/.
% As a special exception, you may use this file and all files derived
% from it without restriction. This special exception was added with
% version 3.7 of Xy-pic.
font_identifier "XYLUBT"; font_size 10pt#;
mode_setup;
input xylu
input xybtri
% $Log: xylubt10.mf,v $ % Revision 3.4 2011/03/14 20:14:00 krisrose % Preparing for release 3.8.6.
% Revision 3.3 2010/06/10 18:45:50 krisrose % Reference to GPL by URL.
% Revision 3.2 2010/04/17 14:45:48 krisrose % Generate and extract Type1 fonts.
% Revision 3.1 2010/04/17 04:19:41 krisrose % Integrated xylu tips by Jeremy Gibbons.
% based on xycmbt10

5.4.18 xyluat11

% $Id: xyluat11.mf,v 3.4 2011/03/14 20:14:00 krisrose Exp $ % xyluat11, for use with Lucida
% Copyright (c) 2007 Jeremy Gibbons <jeremy.gibbons@comlab.ox.ac.uk>
% This file is part of the Xy-pic package for graphs and diagrams in TeX.
% Copyright (c) 1991-2011 Kristoffer H. Rose <krisrose@tug.org>
% The Xy-pic macro package is free software; you can redistribute it and/or
% modify it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either version 2 of the License, or (at your
% option) any later version.
% The Xy-pic macro package is distributed in the hope that it will be
% useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General
% Public License for more details.
% You should have received a copy of the GNU General Public License along
% with this macro package; if not, see http://www.gnu.org/licenses/.
% As a special exception, you may use this file and all files derived
% from it without restriction. This special exception was added with
% version 3.7 of Xy-pic.
font_identifier "XYLUAT"; font_size 11pt#;
mode_setup;
5.4.19 xylubt11

<table>
<thead>
<tr>
<th>'0'</th>
<th>'1'</th>
<th>'2'</th>
<th>'3'</th>
<th>'4'</th>
<th>'5'</th>
<th>'6'</th>
<th>'7'</th>
</tr>
</thead>
<tbody>
<tr>
<td>'0x'</td>
<td>'0x'</td>
<td>'0x'</td>
<td>'0x'</td>
<td>'0x'</td>
<td>'0x'</td>
<td>'0x'</td>
<td>'0x'</td>
</tr>
<tr>
<td>'1x'</td>
<td>'1x'</td>
<td>'1x'</td>
<td>'1x'</td>
<td>'1x'</td>
<td>'1x'</td>
<td>'1x'</td>
<td>'1x'</td>
</tr>
<tr>
<td>'2x'</td>
<td>'2x'</td>
<td>'2x'</td>
<td>'2x'</td>
<td>'2x'</td>
<td>'2x'</td>
<td>'2x'</td>
<td>'2x'</td>
</tr>
<tr>
<td>'3x'</td>
<td>'3x'</td>
<td>'3x'</td>
<td>'3x'</td>
<td>'3x'</td>
<td>'3x'</td>
<td>'3x'</td>
<td>'3x'</td>
</tr>
<tr>
<td>'4x'</td>
<td>'4x'</td>
<td>'4x'</td>
<td>'4x'</td>
<td>'4x'</td>
<td>'4x'</td>
<td>'4x'</td>
<td>'4x'</td>
</tr>
<tr>
<td>'5x'</td>
<td>'5x'</td>
<td>'5x'</td>
<td>'5x'</td>
<td>'5x'</td>
<td>'5x'</td>
<td>'5x'</td>
<td>'5x'</td>
</tr>
<tr>
<td>'6x'</td>
<td>'6x'</td>
<td>'6x'</td>
<td>'6x'</td>
<td>'6x'</td>
<td>'6x'</td>
<td>'6x'</td>
<td>'6x'</td>
</tr>
<tr>
<td>'7x'</td>
<td>'7x'</td>
<td>'7x'</td>
<td>'7x'</td>
<td>'7x'</td>
<td>'7x'</td>
<td>'7x'</td>
<td>'7x'</td>
</tr>
<tr>
<td>'8'</td>
<td>'9'</td>
<td>'A'</td>
<td>'B'</td>
<td>'C'</td>
<td>'D'</td>
<td>'E'</td>
<td>'F'</td>
</tr>
</tbody>
</table>

Figure 5.10: Font table for xyluat11 scaled 2000.
### 5.4. OPTIONAL FONTS

<table>
<thead>
<tr>
<th>'0'</th>
<th>'1'</th>
<th>'2'</th>
<th>'3'</th>
<th>'4'</th>
<th>'5'</th>
<th>'6'</th>
<th>'7'</th>
</tr>
</thead>
<tbody>
<tr>
<td>'00x'</td>
<td>'01x'</td>
<td>'02x'</td>
<td>'03x'</td>
<td>'04x'</td>
<td>'05x'</td>
<td>'06x'</td>
<td>'07x'</td>
</tr>
<tr>
<td>'08x'</td>
<td>'09x'</td>
<td>'A'</td>
<td>'B'</td>
<td>'C'</td>
<td>'D'</td>
<td>'E'</td>
<td>'F'</td>
</tr>
</tbody>
</table>

![Figure 5.11: Font table for xylubt11 scaled 2000.](image)

1. % Public License for more details.
2. % You should have received a copy of the GNU General Public License along
3. % with this macro package; if not, see http://www.gnu.org/licenses/.
4. % As a special exception, you may use this file and all files derived
5. % from it without restriction. This special exception was added with
6. % version 3.7 of Xy-pic.
7. font_identifier "XYLUBT"; font_size 11pt#;
8. mode_setup;
9. input xylu
10. input xybtri
11. % $Log: xylubt11.mf,v $ %
12. % Revision 3.4 2011/03/14 20:14:00 krisrose
13. % Preparing for release 3.8.6.
14. % Reference to GPL by URL.
15. % Reference to GPL by URL.
16. % Revision 3.3 2010/06/10 18:45:50 krisrose
17. % Generate and extract Type1 fonts.
18. % Integrated xylu tips by Jeremy Gibbons.
19. % based on xycmbt11.

#### 5.4.20 xyluat12.mf

1. % $Id: xyluat12.mf,v 3.4 2011/03/14 20:14:00 krisrose Exp $ %
2. % xyluat12, for use with Lucida
3. % Copyright (c) 2007 Jeremy Gibbons <jeremy.gibbons@comlab.ox.ac.uk>
4. %
5.4. OPTIONAL FONTS

% $Log: xylubt12.mf,v $ %
% Revision 3.4 2011/03/14 20:14:00 krisrose %
% Preparing for release 3.8.6. %
%
% Revision 3.3 2010/06/10 18:45:50 krisrose %
% Reference to GPL by URL. %
%
% Revision 3.2 2010/04/17 14:45:48 krisrose %
% Generate and extract Type1 fonts. %
%
% Revision 3.1 2010/04/17 04:19:41 krisrose %
% Integrated xylu tips by Jeremy Gibbons. %
%
% Based on xycmbt12.
Appendix A

Answers to all exercises

Answer to exercise 1.1 (p.23): In the default setup they are all denote the reference point of the \textit{XY}-picture but the cartesian coordinate \langle pos \rangle (0,0) denotes the point \textit{origo} that may be changed to something else using the : operator.

Answer to exercise 1.2 (p.30): Use the (pos)ition \langle X, Y \rangle+"ob".

Answer to exercise 1.3 (p.30): It first sets \( c \) according to “…” Then it changes \( c \) to the point right of \( c \) at the same distance from the right edge of \( c \) as its width, \( w \), \textit{i.e.},

\[
\begin{array}{c}
\text{The} \\
\hline
w \quad w
\end{array}
\]

Answer to exercise 1.4 (p.39): The \langle \text{coord} \rangle {{"A";}"B": "C";"D", x}” returns the cross point. Here is how the author typeset the diagram in the exercise:

\begin{verbatim}
\xy
% set up and mark A, B, C, and D:
(0,0)="A" *\cir<1pt>{}**!DR{A},
(7,10)="B" *\cir<1pt>{}**!DR{B},
(13,8)="C" *\cir<1pt>{}**!DL{C},
(15,4)="D" *\cir<1pt>{}**!DL{D},
%
% goto intersection and name+circle it:
{"A";}"B":"C";"D",x} ="I" *\cir<3pt>{},
%
% make dotted lines:
"I";"A"**{} +/1pc/-/1pc/ **@{..},
"I";"D"**{} +/1pc/-/1pc/ **@{..}
%
\endxy
\end{verbatim}

A ?!... (place) could also have been used.

Answer to exercise 1.5 (p.41): To copy the \( p \) value to \( c \), \textit{i.e.}, equivalent to “p”.

575
Answer to exercise 1.6 (p.41): When using the kernel connections that are all straight there is no difference, e.g., **{}< and **{}<E denote exactly the same position. However, for other connections it is not necessarily the case that the point where the connection enters the current object, denoted by ?<, and the point where the straight line from p enters the object, denoted by +E, coincide.

Answer to exercise 1.7 (p.41): The code typesets the picture

![Box](image)

Answer to exercise 1.8 (p.42): This does the job, saving each point to make the previous point available for the next piece:

\begin{verbatim}
\xy
@={(0,-10),(10,3),(20,-5)},
s0="prev" @@{"prev"};**@{-}="prev"
\endxy
\end{verbatim}

Notice how we close the line by first saving s0, the last point visited, such that the first point will be connected to it.

Answer to exercise 1.9 (p.45): The author used

\begin{verbatim}
\xy =.+{.+DL(2)}.+UR(2)}"dbl",
 \*+<3pc,2pc>{+}*\frm{.}, "dbl"*\frm{--}
\endxy
\end{verbatim}

to typeset the figure in the exercise.

Answer to exercise 1.10 (p.52): The first typesets “a” centered around 0 and then moves c to the lower right corner, the second typesets “a” above the 0 point and does not change c. With a “+” at 0 they look like this: + and ++.

Answer to exercise 1.11 (p.53): They have the outlines

\[ \sum \quad \text{and} \quad \mathbb{H} \]

because the first is enlarged by the positive offset to the upper right corner and the second by the negative offset to the lower left corner.

Answer to exercise 1.12 (p.65): The first has no effect since the direction is set to be that of a vector in the current direction, however, the second reverses the current direction.

Answer to exercise 1.13 (p.66): None in both cases.

Answer to exercise 1.14 (p.81): One way is

\begin{verbatim}
$$\xy
 *{+}; p+(6,3)*{+} **{} *(1)
 *@{-} */-5pt/\dir{-}
 */-5pt/\dir{-}
\endxy$$
\end{verbatim}
Thus we first create the two +s as \( p \) and \( c \) and connect them with the dummy connection \(*(*)\) to setup the direction parameters. Then we move 'on top of \( c \)' with \(*(1)\) and position the four sides of the square using ~ and _ for local direction changes and /\( \text{dimen} / \) for skewing the resulting object by moving its reference point in the opposite direction.

**Answer to exercise 1.15 (p.89):** One way is to add extra half circles skewed such that they create the illusion of a shade:

\[
\text{\begin{verbatim}
\$\$\xy
  *\cir<5pt>{}
  */<-.2pt,.2pt>\cir<5pt>{dr^ul}
  */<-.4pt,.4pt>\cir<5pt>{dr^ul}
  */<-.6pt,.6pt>\cir<5pt>{dr^ul}
\endxy$
\end{verbatim}
\]

**Answer to exercise 2.1 (p.125):** This is the code that was actually used:

\[
\text{\begin{verbatim}
xymatrix @+0.4pc {0,20} [o]+{A};(60,0) *[o]+{B}="B"
  **\crv{} \POS?(.4)*.+!UR{0},"B"
  **\crv{(30,30)} \POS?-*!D{1},"B"
  **\crv{(20,40)&(40,40)} \POS?-*!D{2},"B"
  **\crv{(10,20)&(30,20)&(50,-20)&(60,-10)} \POS?-*!UR{4}
\end{verbatim}
\]

**Answer to exercise 2.2 (p.125):** This is the code that was used to typeset the picture:

\[
\text{\begin{verbatim}
xymatrix @+0.4pc {0,20} [o]+{A};(60,0) *[o]+{B}
  **\crv{} \POS?(.4)*.+!UR{0},"B"
  **\crv{(10,20)&(30,20)&(50,-20)&(60,-10)}
  ?<\dir{<}> ?>*\dir{>}
  ?.65)*{\oplus} \POS!LD!/^-5pt/\{x\}
  ?.65)/12pt/*{\oplus} \POS!LD!/^-5pt/\{x'\}
  ?.28)*=0\{0\text{otimes}^-!/40pt/\{Q="q"
  +/100pt/\{P,"q" **\dir{-}
\end{verbatim}
\]

**Answer to exercise 2.3 (p.126):** Here is the code that was used to typeset the picture:

\[
\text{\begin{verbatim}
def\sz#1{\hbox{$_{^{#1}}$}}
xymatrix @+0.4pc {0,0} [o]+{A};(30,-10) *[o]+{B}="B",**\dir{-},
  "B"**\crv{(5,20)&(20,25)&(35,20)}
  ?<(0)*\dir{<}="a" ?>(1)*\dir{>}="h"
  ?.1)*\dir{<}="b" ?.9)*\dir{>}="i"
  ?.2)*\dir{<}="c" ?.8)*\dir{>}="j"
  ?.3)*\dir{<}="d" ?.7)*\dir{>}="k"
  ?.4)*\dir{<}="e" ?.6)*\dir{>}="l"
  ?.5)*\dir{<}="f",
  "a"!*RC\txt{\sz{\lt}};
  "h"!*LC\txt{\sz{\gt}},**\dir{.},
  "b"!*RD{\sz{.1}};
  "i"!*L{\sz{.9}},**\dir{-},
\end{verbatim}
\]
Answer to exercise 2.4 (p.203): Here is how:

\xy
(0,0) +++={A} */frm{o} ;
(10,7) +++={B} */frm{o} */frm{.}
\endxy

Answer to exercise 2.5 (p.208): The *\cir {} operation changes c to be round whereas *\frm {o} does not change c at all.

Answer to exercise 2.6 (p.211): Here is how:

\xy
(0,0) +++={A} ;
(10,7) +++={B} */frm{.}
*/frm{\}} ; */frm{\_\}
\endxy

The trick in the last line is to ensure that the reference point of the merged object to be braced is the right one in each case.

Answer to exercise 2.7 (p.239): This is how the author specified the diagram:

\UseCrayolaColors
\xy\drop*[1.25]\xybox\POS
(0,0)*{A};(100,40)*{B}**{}
?<<[@_][red][o]=<5pt>{\heartsuit};
?>>*[@[Plum][o]=<3pt>{\clubsuit}
**[1][1.5pt][thicker]\dir{-},
?(.1)*[left]!RD\txt{label 1}*[red]\frm{.}
?(.2)*[!gsave newpath
xyXpos xyYpos moveto 50 dup rlineto
20 setlinewidth 0 0 1 setrgbcolor stroke
grestore][psxy]{.},
?(.2)*[@\txt{label 2}*[red]\frm{.},
?(.2)*[BurntOrange]{*},
?(.3)*[halfsize]\txt{label 3}*[red]\frm{.}
?(.375)*[flip]\txt{label 4}*[red]\frm{.}
?(.5)*[dblsize]\txt{label 5}*[red]\frm{.}
?(.5)*[WildStrawberry]{*},
?(.7)*[hflip]\txt{label 6}*[red]\frm{.}
?(.8)*[vflip]\txt{label 7}*[red]\frm{.}
Answer to exercise 3.1 (p.317): Here is what the author did:
\xy **{A}**\cir<10pt>{}="me"
\PATH `ul^-ur,"me" "me" |>*:(1,-.25)\dir{>}
\endxy

The trick is getting the arrow head right: the \dir modifier to the explicit \dir \langle object \rangle does that.

Answer to exercise 3.2 (p.319): The author did
\xy(0,0)
\ar {}(30,7) "A"="a"
\POS(10,12)*+{\txt{label}} \ar "a"
\endxy

Answer to exercise 3.3 (p.324): Here is the entire \textit{XY}-picture of the exercise:
\xy ;<1pc,0pc>:
\POS(0,0)*+{A}
\ar +(0,3)"A"*\cir{}
\ar 02 +( 0,3)"A"*\cir{}
\ar 03 +( 2,3)"A"*\cir{}
\POS(6,0)"B"
\ar {||.>>} 02+( 0,3)"B"*\cir{}
\ar {||.>>} 03+( 2,3)"B"*\cir{}
\endxy

The first batch use the default \textit{->} specification.

Answer to exercise 3.4 (p.325): The author used
\newdir{ >}{{}*!/-5pt/\dir{>}}

Answer to exercise 3.5 (p.325): The author used
\xy
\ar {}(20,7)\composite{\dir{x}+\dir{+}}<<
\endxy

Answer to exercise 3.6 (p.326): The author used
\xy *\cir"b" \ar@{(ur,ul)} c
\ar@{.}+(dr,ul) (20,0)*\bullet
\endxy
Note that it is essential that the curving specification comes after the arrow style.

**Answer to exercise 3.7 (p.338):** Here is the code used to typeset the pasting diagram in figure 3.3.
\begin{verbatim}
\ymatrixrowsep{1.5pc}
\ymatrixcolsep{3pc}
\begin{diagram}
& & \relax\rtwocell<0>^{f_3^{}\;}{\omit} \\
&\relax\ddtwocell<0>{\omit} & \\
&\relax\drtwocell<0>^{{f_4^{}}}{<3>}
\enddiagram
\end{verbatim}

For the straight arrows, it would have been simpler to use `\..to provided xyarrow has been loaded. Instead `\..twocell<0>...{\omit}` was used to illustrate the versatility of nudging and `\omit`; thus xy2cell can completely handle a wide range of diagrams, without requiring `xyarrow`. Note also the use of `\relax` at the start of each new cell, to avoid premature expansion of a complicated macro, which can upset the compiling mechanism.

**Answer to exercise 3.8 (p.342):** Here is the code used by the author to set the first diagram.
\begin{verbatim}
{\uppercurveobject{{?}}}
{\lowercurveobject{{\circ}}}
\ymatrixcolsep{5pc}
\ymatrixrowsep{2pc}
\begin{diagram}
\relax\txt{ FUN }\rtwocell<8>{!}&
\relax\txt{ gaMES }
\enddiagram
\end{verbatim}

Here is the code used for the second diagram.
\begin{verbatim}
\ymatrixcolsep{2.5pc}
\ymatrixrowsep{4pc}
\begin{diagram}
\relax\txt{<1.5cm}{\bf Ground State}
\rtwocell<12>``{+++///-2.5pt/\dir{>}}
``{+++///-2.5pt/\dir{<}}
\{<1.5}\txt{small continuous power}}
\{<1.5}\txt{small pulsed emission}}
\enddiagram
\end{verbatim}
Answer to exercise 3.9 (p.370): The author did
\xymatrix @!=1pc {
**[l] A\times B \\
& \ar[r]^/-A/ \ar[d]/-B/
B \\
A \ar[r]_{B\times} & **[r] B\times A
}

Answer to exercise 3.10 (p.371): Modifiers are used to make all entries round with a frame – the general form is used to ensure that the sequence is well-defined. Finally the matrix is rotated to make it possible to enter it as a simple square:
\entrymodifiers={=<1pc>[o][F-]}
\xymatrix @ur {
A \save[+<1pc>] ;[r] **\dir{-},
[] ; [dr]**\dir{-},
[] ; [d] **\dir{-}\restore
& B \\\nC & D }

Answer to exercise 3.11 (p.373): Here is how:
\xymatrix @W=3pc @H=1pc @R=0pc @*[F-] {% 
: \save+<-4pc,1pc>** \hbox{\it root} 
\ar[] 
\restore
\\\n{\bullet}
\save*{}
\ar' r[d]+/r4pc/'[dd][dd] 
\restore
\\
{\bullet}
\save*{}
\ar' r[d]+/r3pc/'[d]+/d2pc/
' [uu]+/13pc/'[uu] 
\restore
\\
1 }

Answer to exercise 3.12 (p.380): The first $A$ was named to allow reference from the last:
\xygraph{
Answer to exercise 3.13 (p.382): The author did
\SelectTips{cm}{}
\objectmargin={1pt}
\xygraph{!{0;(.77,-.77):0}
!~:{@{-}|@{>}}
w (:{r(.6)}{x_1}
,:[d]z:[r]y:[u(.6)]{x_2}:'x_1':'z"
:@( {{"w";"z"}, {{"y";"z"}}})'z";'x_2' )
}

It also shows that one can use {}s inside delimited arguments provided one adds a space to avoid the {}s being shaved off!

Answer to exercise 3.14 (p.385): Here is the code actually used to typeset the \xypolygon s, within an \xygraph . It illustrates three different ways to place the numbers. Other ways are also possible.
\def\objectstyle{\scriptscriptstyle}
\xy \xygraph{!{/r2pc/:
[] !P3"A"{\bullet}
"A1"!{+U+++!D{1}} "A2"!{+LD+++!RU{2}}
"A3"!{+RD+++!LU{3}} "A0"
[rrr]*{0}*{\cir<5pt>}{}
!P6"B"{{<-\cir<5pt>}}} !P6"B"{{<-\cir<5pt>}}
"B1"1 "B2"2 "B3"3 "B4"4 "B5"5 "B6"6 "B0"
[rrr]0 !P9"C"{~*{\xypolynode})))\endxy
Appendix B

Backwards Compatibility

Backwards compatibility with \textsc{Xy-pic} version 2 is ensured by the ‘v2’ option and its support files.

B.1 Version 2 Compatibility

\textbf{Vers. 3.8 by Kristoffer H. Rose} (krisrose@tug.org)

This section describes the special backwards compatibility with \textsc{Xy-pic} version 2: diagrams written according to the “Typesetting diagrams with \textsc{Xy-pic}: User’s Manual” [14] should typeset correctly with this loaded. The compatibility is available either as an \textsc{Xy-option} or through the special files \texttt{xypic.sty} and \texttt{xypic.tex} described below.

There are a few exceptions to the compatibility: the features described in §B.1.1 below are not provided because they are not as useful as the author originally thought and thus virtually never used. And one extra command is provided to speed up typesetting of documents with \textsc{Xy-pic} version 2 diagrams by allowing the new compilation functionality with old diagrams.

The remaining sections list all the obsolete commands and suggest ways to achieve the same things using \textsc{Xy-pic} 3.8.9, \textit{i.e.}, without the use of this option. They are grouped as to what part of \textsc{Xy-pic} replaces them; the compilation command is described last.

\textbf{Note:} “version 2” is meant to cover all public releases of \textsc{Xy-pic} in 1991 and 1992, \textit{i.e.}, version 1.40 and versions 2.1 through 2.6. The published manual cited above (for version 2.6) is the reference in case of variations between these versions, and only things documented in that manual will be supported by this option!\footnote{In addition a few of the experimental facilities supported in v2.7–2.12 are also supported.}
APPENDIX B. BACKWARDS COMPATIBILITY

Before we start, the old name macro.

\xylet@XY=\xy

B.1.1 Unsupported incompatibilities

Here is a list of known incompatibilities with version 2 even when the v2 option is loaded.

- Automatic ‘shortening’ of arrow tails using |<< breaks was a bug and has been ‘fixed’ so it does not work any more. Put a |<\hole break before it.

- The version 2.6 * position operator is not available. The version 2.6 construction $t_0; t_1*(x,y)$ should be replaced by the rather long but equivalent construction

\[
\{t_0; p+/r/; t_1\}
== "1"; p+/u/, x;(0,0);::(x,y)
\]

In most cases $t_0; t_1**{}?({x})$, possibly with a trailing +/.../, suffices instead.

- Using $t_0; t_1:(x,y)$ as the target of an arrow command does not work. Enclose it in braces, i.e., write

\[
\{t_0, t_1:(x,y)\}
\]

- The older \pit, \apit, and \bpit commands are not defined. Use \dir{>} (or \tip) with variants and rotation.

- The even older notation where an argument in braces to \rto and the others was automatically taken to be a ‘tail’ is not supported. Use the supported |<... notation.

If you do not use these features then your version 2 (and earlier) diagrams should typeset the same with this option loaded except that sometimes the spacing with version 3 is slightly different from that of version 2.6 which had some spacing bugs.

B.1.2 Obsolete kernel features

The following things are added to the kernel by this option and described here: idioms, obsolete positions, obsolete connections, and obsolete objects. For each we show the suggested way of doing the same thing without this option:
B.1. VERSION 2 COMPATIBILITY

Removed \(\text{AMS}\text{-\TeX} \) idioms

Some idioms from \(\text{AMS}\text{-\TeX} \) are no longer used by \text{Xy-pic}: the definition commands \texttt{\textbackslash define} and \texttt{\textbackslash redefine}, and the size commands \texttt{\textbackslash dsize}, \texttt{\textbackslash tsize}, \texttt{\textbackslash ssize}, and \texttt{\textbackslash sssize}. Please use the commands recommended for your format—for plain \TeX\ these are \texttt{\textbackslash def} for the first two and \texttt{\textbackslash displaystyle}, \texttt{\textbackslash textstyle}, \texttt{\textbackslash scriptstyle}, and \texttt{\textbackslash scriptscriptstyle} for the rest. The \texttt{v2} option ensures that they are available anyway.

\begin{verbatim}
138 \ifx\redefine\undefined \let\redefine=\def \fi
139 \ifx\define\undefined \let\define=\xydef@ \fi
141 \ifx\dsize\undefined \let\dsize=\displaystyle \fi
142 \ifx\tsize\undefined \let\dsize=\textstyle \fi
143 \ifx\ssize\undefined \let\ssize=\scriptstyle \fi
144 \ifx\sssize\undefined \let\sssize=\scriptscriptstyle \fi
\end{verbatim}

Version also 2 used the \(\text{AMS}\text{-\TeX} \) \texttt{\textbackslash text} and a (non-object) box construction \texttt{\Text} which are emulated—\texttt{\text} is only defined if not already defined, however, using the native one (of \(\text{AMS}\text{-\TeX} \) or \(\text{AMS}\text{-\LaTeX} \) or whatever) if possible. Please use the \texttt{\txt} object construction described in §1.6.3 directly since it is more general and much more efficient!

\begin{verbatim}
155 \ifx\text\undefined
156 \def\text{\relax\textC}\
157 \xydef@\textC#1{\relax
158 \ifmmode\mathchoice
159 {\hbox{\the\textfont0\relax#1}}%
160 {\hbox{\the\textfont0\relax#1}}%
161 {\hbox{\the\scriptfont0\relax#1}}%
162 {\hbox{\the\scriptscriptfont0\relax#1}}%
163 \else{\relax#1}\fi}
164 \fi
166 \xydef@\Text{\relax\xyFN@\Text@}
168 \xydef@\TextC{\relax\textC}%
170 \xydef@\Text@{%
171 \addLT@\ifx\next \addGT@{\addLT@\DN@\#1}{\A@=\#1\Text@i}%
172 \else \DN@{\A@=\maxdimen \Text@i}\fi \next@}
174 \xydef@\Text@i{%
175 \ifmmode\mathchoice
176 \{\expandafter\Text@ii\the\textfont0\{#1}\}%
177 \{\expandafter\Text@ii\the\textfont0\{#1}\}%
178 \{\expandafter\Text@ii\the\scriptfont0\{#1}\}%
179 \{\expandafter\Text@ii\the\scriptscriptfont0\{#1}\}%
180 \else
181 \leavevmode \expandafter\Text@ii\the\textfont0{#1}%
182 \fi\ignorespaces}
184 \xydef@\Text@i{\hbox\bgroup \txt@i}
\end{verbatim}

Obsolete state

Upto version 2.6 users could access the state variables \texttt{\textbackslash cL}, \texttt{\textbackslash cR}, \texttt{\textbackslash cH}, and \texttt{\textbackslash cD}, which are defined.
\xylet@cL=\LC\xylet@cR=\RC
\xylet@cD=\DC
\xylet@cH=\HC

From v2.7 to 2.12 users could use the names of the state \texttt{dimen} registers \Xmin, \Xmax, \Ymin, and \Ymax; \Xp, \Yp, \Dp, \Up, \Lp, and \Rp; \Xc, \Yc, \Dc, \Uc, \Lc, and \Rc; \Xorigin, \Yorigin, \Xbase, \Ybase, \Xybase, and \Yybase. Now the same effect can be achieved using \texttt{corner}s but \texttt{v2} defines the aliases.

\xylet@cXc=\Xc\xylet@cYc=\Yc\xylet@cDc=\Dc\xylet@cLc=\Lc\xylet@cRc=\Rc\xylet@cXp=\Xp\xylet@cYp=\Yp\xylet@cUp=\Up\xylet@cDp=\Dp\xylet@cLp=\Lp\xylet@cRp=\Rp\xylet@cXc=\Xc\xylet@cYc=\Yc\xylet@cDc=\Dc\xylet@cLc=\Lc\xylet@cRc=\Rc\xylet@cXp=\Xp\xylet@cYp=\Yp\xylet@cUp=\Up\xylet@cDp=\Dp\xylet@cLp=\Lp\xylet@cRp=\Rp

\xylet@cXorigin=\Xorigin\xylet@cYorigin=\Yorigin\xylet@cXbase=\Xbase\xylet@cYbase=\Ybase\xylet@cXybase=\Xybase\xylet@cYybase=\Yybase\xylet@cXmin=\Xmin\xylet@cYmin=\Ymin\xylet@cXmax=\Xmax\xylet@cYmax=\Ymax

**Obsolete position manipulation**

In version 2 many things were done using individual \texttt{(decor)} control sequences that are now done using \texttt{(pos)} operators.

<table>
<thead>
<tr>
<th>Version 2 positioning</th>
<th>Replacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>\go\texttt{(pos)}</td>
<td>\POS;p,\texttt{(pos)}</td>
</tr>
<tr>
<td>\aftergo\texttt{(decor)}\texttt{(pos)}</td>
<td>\afterPOS\texttt{(decor)}\texttt{p,\texttt{(pos)}}</td>
</tr>
<tr>
<td>\merge</td>
<td>\POS.p\texttt{relax}</td>
</tr>
<tr>
<td>\swap</td>
<td>\POS;\texttt{relax}</td>
</tr>
<tr>
<td>\Drop\texttt{(text)}</td>
<td>\drop+\texttt{(text)}</td>
</tr>
</tbody>
</table>

The code is basically that of xy.doc 2.6...
Version 2 compatibility

Obsolete connections

These connections are now implemented using directionals.

<table>
<thead>
<tr>
<th>Version 2 connection</th>
<th>Replacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>\none</td>
<td>\connect h\dir{}</td>
</tr>
<tr>
<td>\solid</td>
<td>\connect h\dir{-}</td>
</tr>
<tr>
<td>\Solid</td>
<td>\connect h\dir2{-}</td>
</tr>
<tr>
<td>\Ssolid</td>
<td>\connect h\dir3{-}</td>
</tr>
<tr>
<td>\dashed</td>
<td>\connect h\dir{-}</td>
</tr>
<tr>
<td>\Dashed</td>
<td>\connect h\dir2{-}</td>
</tr>
<tr>
<td>\Ddashed</td>
<td>\connect h\dir3{-}</td>
</tr>
<tr>
<td>\dotted</td>
<td>\connect h\dir{.}</td>
</tr>
<tr>
<td>\Dotted</td>
<td>\connect h\dir2{.}</td>
</tr>
<tr>
<td>\Ddotted</td>
<td>\connect h\dir3{.}</td>
</tr>
<tr>
<td>\dottedwith{text}</td>
<td>\connect h{(text)}</td>
</tr>
</tbody>
</table>

Note how the ‘hidden’ specifier h should be used because version 2 connections did not affect the size of diagrams.

Obsolete tips

These objects all have \dir-names now:

<table>
<thead>
<tr>
<th>Version 2 tip</th>
<th>Replacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>\notip</td>
<td>\dir{}</td>
</tr>
<tr>
<td>\stop</td>
<td>\dir{}</td>
</tr>
<tr>
<td>\astop</td>
<td>\dir^{-}{[]}</td>
</tr>
<tr>
<td>\bstop</td>
<td>\dir{-}{[]}</td>
</tr>
</tbody>
</table>
APPENDIX B. BACKWARDS COMPATIBILITY

\tip \dir{>}
\atip \dir^{>}
\btip \dir_{>}
\Tip \dir2{>}
\aTip \object=<5pt>:(32,-1)\dir^{>}
\bTip \object=<5pt>:(32,+1)\dir_{>}
\Ttip \dir3{>}
\ahook \dir^{(}
\bhook \dir_{(}
\aturn \dir^{'}
\bturn \dir_{'}

\xydef@notip{\dir{}}
\xydef@stop{\dir{|}}
\xydef@astop{\dir^{|}}
\xydef@bstop{\dir_{|}}
\xydef@tip{\dir{>}}
\xydef@atip{\dir^{>}}
\xydef@btip{\dir_{>}}
\xydef@Ttip{\dir3{>}}
\xywarnifdefined\aTip
\xywarnifdefined\bTip
{\xyuncatcodes
\gdef\aTip{\object=<5pt>:(32,-1)\dir^{>}}
\gdef\bTip{\object=<5pt>:(32,+1)\dir_{>}}
\xydef@Ttip{\dir3{>}}
\xydef@ahook{\dir^{(}}
\xydef@bhook{\dir_{(}}
\xydef@aturn{\dir^{'}}
\xydef@bturn{\dir_{'}}

The older commands \pit, \apit, and \bpit, are not provided.

Obsolete object constructions

The following object construction macros are made obsolete by the enriched ⟨object⟩ format:

<table>
<thead>
<tr>
<th>Version 2 object</th>
<th>Replacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>\rotate(⟨factor⟩)(tip)</td>
<td>⟨object: ⟨factor⟩, ⟨factor⟩⟩⟨(tip)⟩</td>
</tr>
<tr>
<td>\hole</td>
<td>⟨object+⟩</td>
</tr>
<tr>
<td>\squash(tip)</td>
<td>⟨object=0⟩⟨(tip)⟩</td>
</tr>
<tr>
<td>\grow(tip)</td>
<td>⟨object+=⟩⟨(tip)⟩</td>
</tr>
<tr>
<td>\grow⟨dimen⟩(tip)</td>
<td>⟨object+=⟨dimen⟩⟩⟨(tip)⟩</td>
</tr>
<tr>
<td>\squarify{(text)}</td>
<td>⟨object+=⟨(text)⟩</td>
</tr>
<tr>
<td>\squarify⟨dimen⟩{(text)}</td>
<td>⟨object+=⟨(dimen)⟩⟨(text)⟩</td>
</tr>
</tbody>
</table>
where rotation is done in a slightly different manner in version 3.8.9 (it was never accurate in version 2).

These are mostly just the unmodified version 2.6 code:

\xydef@\rotate{\hbox{bgroup
\DN@\ifsx \next%
 \DN@{\rotate@i\fi\next0}\xyFN@\next0}
\xydef@\rotate@i1\{\reverseDirection@ #1\OBJECT@x
\xydef@\rotate@i12\{\dimen@=#1\p@
\ifdim \dimen@=-\p@ \aboveDirection@\xydash10
\else \ifdim\dimen@<-.5\p@ \belowDirection@\xydash10
\else \ifdim\dimen@<.5\p@ \vDirection@(#1,-1){1pc}
\else \ifdim\dimen@<1.5\p@ \vDirection@(#1,1){1pc}
\else \ifdim\dimen@<-1.5\p@ \vDirection@(-1,-1){1pc}
\else \ifdim\dimen@<-.5\p@ \vDirection@(-1,1){1pc}
\else \ifdim\dimen@<.5\p@ \vDirection@(#1,-1){1pc}
\else \ifdim\dimen@<1.5\p@ \vDirection@(#1,1){1pc}
\else \ifdim\dimen@<-1.5\p@ \vDirection@(-1,-1){1pc}
\fi\fi\fi\fi\fi\fi
\fi\fi\fi\fi\fi\fi
\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi
\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi}
\xydef@\squash#1{\ifmmode\setboxz@h{\m@th${\mathstrut}#1{\mathstrut}$}\else\setboxz@h{#1}\fi
\setboxz@h{\kern-.5\wdz@ \dimen@=.5\ht\z@ \advance\dimen@-.5\dp\z@
\lower\dimen@ \boxz@}
\wdz@=\z@ \ht\z@=\z@ \dp\z@=\z@ \boxz@}
\xydef@\grow{\DN@\ifsx <\next \DN@<###1>{\dimen@=#1\empty}\else \DN@\dimen@=\empty\fi\next0\xyFN@\next0}
\xydef@\grow@#1\{\relax
\setboxz@h{\kern\dimen0 \m@th$#1\kern\dimen0$}\else \setboxz@h{\kern\dimen0 \kern\dimen0\{}\fi
\dimen@=\ht\z@ \advance\dimen@\dimen0i\i
\dimen@=\dp\z@ \advance\dimen@\dimen0i\i\dimen0i\i\i \boxz@}
\xydef@\squarify{\addEQ@\addPLUS@\object}
\xydef@\qc{\corn@}

We also reimplement the version 2 undocumented \qc quarter circle which is now called \corn of the frame extension.

B.1.3 Obsolete extensions & features

Version 2 had commutative diagram functionality corresponding to the frames extension and parts of the matrix and arrow features. These are therefore loaded and some extra definitions added to emulate commands that have disappeared.
Frames

The version 2 frame commands are emulated using the frame extension (as well as the \dotframed, \dashframed, \rounddashframed commands communicated to some users by electronic mail):

<table>
<thead>
<tr>
<th>Version 2 object</th>
<th>Replacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>\framed</td>
<td>\drop\frm{}</td>
</tr>
<tr>
<td>\framed&lt;(dimen)&gt;</td>
<td>\drop\frm&lt;(dimen)&gt;{}</td>
</tr>
<tr>
<td>\Framed</td>
<td>\drop\frm={}</td>
</tr>
<tr>
<td>\Framed&lt;(dimen)&gt;</td>
<td>\drop\frm&lt;(dimen)&gt;{}</td>
</tr>
<tr>
<td>\dotframed</td>
<td>\drop\frm{.}</td>
</tr>
<tr>
<td>\dashframed</td>
<td>\drop\frm{--}</td>
</tr>
<tr>
<td>\rounddashframed</td>
<td>\drop\frm{o-}</td>
</tr>
<tr>
<td>\circled</td>
<td>\drop\frm{o}</td>
</tr>
<tr>
<td>\Circled</td>
<td>\drop\frm{oo}</td>
</tr>
</tbody>
</table>

For each of the above commands (and a few hidden ones) we parse the above and convert to the proper format.

Matrices

The \texttt{\textbackslash{diagram} (rows)} \texttt{\textbackslash{enddiagram}} command is provided as an alias for \texttt{\textbackslash{ymatrix} (rows)} centered in math mode and \LaTeX\texttt{\textbackslash{diagrams}} changes it to use \texttt{\textbackslash{begin} \ldots \textbackslash{end}} syntax. v2 sets a special internal ‘old matrix’ flag such that trailing `\' are ignored and entries starting with `* are safe.
B.1. VERSION 2 COMPATIBILITY

\xyrequire{matrix}\xycatcodes
\xydef\diagram\enddiagram{\relax\diagram\enddiagram}{#1}
\xydef\diagram{\relax\ifmmode\DN@##1##2{\vcenter{\oldxymatrix@true \xy##1{\xymatrix{##2}}\endxy}}\else\DN@##1##2{{\oldxymatrix@true \xy/u.8ex/##1{\xymatrix{##2}}}\endxy}\fi\next@}
\xydef\LaTeXdiagrams{\def\diagram{\ltxdiagram\diagram@@}\def\diagramnocompile{\ltxdiagram\literal}\def\diagramcompileto##1{\ltxdiagram{\NoCompileMatrices\xycompileto{##1}}}}
\xydef\ltxdiagram@@#1#2\end#3{\relax\DN@{#3}\DNii@{diagram}\ifx\next@\nextii@\diagram@{#1}{#2}\enddiagram\else\xyerr{Put \string{\string} around \string{\begin...\string{\end} space within diagrams}\fi}
\xylet\diagram@@=\literal
\xydef\NoisyDiagrams{}
\NoisyDiagrams is ignored because the matrix feature always outputs progress messages.

Finally the version 2 \spreaddiagramrows, \spreaddiagramcolumns spacing commands are emulated using \xymatrixrowsep and \xymatrixcolsep:
\xydef\spreaddiagramrows#1{\addPLUS\xymatrixrowsep{#1}}
\xydef\spreaddiagramcolumns#1{\addPLUS\xymatrixcolsep{#1}}

Arrows

The main arrow commands of version 2 were the \morphism and \definemorphism commands which now have been replaced by the \ar command.

v2 provides them as well as uses them to define the version 2 commands \xto, \xline, \xdashed, \xdotted, \xdouble, and all the derived commands \dto, \urto, ...; the \arrow commands of the β-releases of v3 is also provided.

Instead of commands like \rrto and \uldouble you should use the arrow feature replacements \ar[rr] and \ar[=][ul].

The predefined turning solid arrows \lltou, ..., \tord are defined as well; these are now easy to do with (turn) s.

\xyrequire{arrow}\xycatcodes
\xylet\arrow=\ar
\morphism is a reduced version of the \ar command.

\xyrequire{arrow}\xycatcodes
\%\xyucatcodes \catcode35=6
\% \def\next#1#2#3{\relax \PATH \PATH ~={#1} ~>{\POS?>*{#2}\relax} ~<{\POS;?<*{#3};\relax}}
\%\xyucatcodes
\%\xylet\morphism=\next
\{\xyucatcodes
\gdef\morPHISM#1#2#3{\PATH={\oldconnect{#1}}\PATH={\oldconnect{#1}}\PATH={\oldconnect{#1}}}~<{\POS;>?*{#2};\relax}}
\xydef\morphism{\relax\morPHISM}
\xylet\connect@iii=\connect
\xydef\oldconnect#1{\%
To Do: Unpack this to what \PATH does...
\definemorphism is essentially unchanged except we now use the privacy checking commands.

\xydef\definemorphism{to}\solid\tip\notip
\definemorphism{line}\solid\notip\notip
\definemorphism{dashed}\dashed\notip\notip
\definemorphism{dotted}\dotted\notip\notip
\definemorphism{double}\Solid\notip\notip

To Do: These could be hugely optimised by using the arrow feature directly...

Next the predefined bent morphisms of version 2. These required some hacking to work; I hope this doesn’t mean that the ⟨path⟩ semantics has changed to much...
Finally the version 2 arrow radius default command that actually never worked but some people might have hoped...

B.1.4 Obsolete loading

The v2 User’s Manual says that you can load Xy-pic with the command \input xypic and as a \TEX 2.09 ‘style option’ [xypic]. This is made synonymous with loading this option by the files xypic.tex and xypic.sty distributed with the v2 option.

xypic.doc

This file (version 3.6) just loads the v2 feature.
Here it is...
\input xyv2
\endinput

The log reveals that this was originally the main file.
xypic.sty

\input{xy.sty}
\xyoption{v2}
\catcode`\@=\xystycatcode
\endinput

\xydef\diagramcompileto#1#2\enddiagram{elax
\diagram{\NoCompileMatrices\xycompileto{#1}}{#2}}

B.1.5 Compiling v2-diagrams

In order to make it possible to use the new compilation features even on documents written with \texttt{XY-pic v2}, the following command was added in v2.12:

\begin{verbatim}
\diagramcompileto{ (name) } ... \enddiagram
\end{verbatim}

which is like the ordinary diagram command except the result is compiled (see note 1.5e). Note that compilation is not quite safe in all cases!

\begin{verbatim}
\xydef@\diagramcompileto#1#2\enddiagram{\relax
\end{verbatim}
There is also the following command that switches on automatic compilation of all diagrams created with the v2 \texttt{\textbackslash diagram ... \textbackslash enddiagram} command:

\begin{verbatim}
CompileAllDiagrams \{\langle \text{prefix} \rangle \} \\
NoCompileAllDiagrams \\
ReCompileAllDiagrams
\end{verbatim}

will apply \texttt{\textbackslash xycompile\{}\langle \text{prefix}\rangle n\}\{\ldots\} to each diagram with $n$ a sequence number starting from 1. Use \texttt{\textbackslash CompileMatrices} and \texttt{\textbackslash CompilePrefix} instead!

If for some reason a diagram does not work when compiled then replace the \texttt{\textbackslash diagram} command with \texttt{\textbackslash diagramnocompile} (or in case you are using the \texttt{\textbackslash LaTeX} form, \texttt{\textbackslash begin\{}\texttt{\textbackslash diagramnocompile}\}).

791 \texttt{\textbackslash xydef@\textbackslash diagramnocompile#1\textbackslash enddiagram\{}\texttt{\textbackslash relax\textbackslash diagram@\textbackslash literal@\{\#1\}}\texttt{\textbackslash enddiagram\}\

795 \texttt{\textbackslash xydef@\textbackslash CompileAllDiagrams#1\{\textbackslash CompilePrefix\{\#1\}\textbackslash CompileMatrices\}\

796 \texttt{\textbackslash xydef@\NoCompileAllDiagrams{\textbackslash NoCompileMatrices\}}\

797 \texttt{\textbackslash xydef@\ReCompileAllDiagrams{\textbackslash CompileMatrices\}}

\textbf{End \\& log}

\begin{verbatim}
\xyendinput
\end{verbatim}

\texttt{\% $\!\text{Log: xyv2.doc,v}$} \\
\texttt{\% Revision 3.8 2011/03/14 20:14:00 krisrose} \\
\texttt{\% Preparing for release 3.8.6.} \\
\texttt{\%} \\
\texttt{\% Revision 3.7 2010/07/27 09:49:34 krisrose} \\
\texttt{\% Started xyling (and address updates).} \\
\texttt{\%} \\
\texttt{\% Revision 3.6 2010/06/10 18:45:50 krisrose} \\
\texttt{\% Reference to GPL by URL.} \\
\texttt{\%} \\
\texttt{\% Revision 3.5 2010/04/16 06:06:52 krisrose} \\
\texttt{\% Preparing for a new release...} \\
\texttt{\%} \\
\texttt{\% Revision 3.4 1997/05/18 01:14:25 krisrose} \\
\texttt{\% Essential bugfixes.} \\
\texttt{\%} \\
\texttt{\% Revision 3.3 1996/12/19 14:43:55 krisrose} \\
\texttt{\% Maintenance release.} \\
\texttt{\%} \\
\texttt{\% Revision 3.2 1995/09/19 18:22:27 kris} \\
\texttt{\% Bug fix release.} \\
\texttt{\%} \\
\texttt{\% Revision 3.1 1995/09/05 20:31:32 kris} \\
\texttt{\% Releasing!} \\
\texttt{\%} \\
\texttt{\% Revision 3.0 1995/07/07 20:14:21 kris} \\
\texttt{\% Major release w/new User’s Guide!} \\
\texttt{\%} \\
\texttt{\% Revision 2.14 1995/07/05 22:10:12 kris}
B.2. OBSOLETE FONTS

These fonts are obsolete since v2.7 but remain part of the distribution to make it possible to print DVI files created with version 2.6 and earlier versions. We show the “misc” font in reduced size so it fits.

B.2.1 xyline10

These fonts are obsolete since v2.7 but remain part of the distribution to make it possible to print DVI files created with version 2.6 and earlier versions. We show the “misc” font in reduced size so it fits.

% $Id: xyline10.mf,v 3.5 2010/06/10 18:45:50 krisrose Exp $ **tex**
1 %
2 % XYLINE10: line segments for XY mode at 10 point.
3 % Copyright (c) 1991,1992 Kristoffer H. Rose <krisrose@tug.org>
4 %
5 % This file is part of the XY-pic macro package.
6 %
7 % The XY-pic macro package is free software; you can redistribute it and/or
8 % modify it under the terms of the GNU General Public License as published by
9 % the Free Software Foundation; either version 2 of the License, or (at your
10 % option) any later version.
11 %
12 % The XY-pic macro package is distributed in the hope that it will be
13 % useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
14 % MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General
15 % License for more details.
16 %
17 % You should have received a copy of the GNU General Public License along
Table 1: Font table for `xyline10 scaled 2000`.

<table>
<thead>
<tr>
<th>00x</th>
<th>01x</th>
<th>02x</th>
<th>03x</th>
<th>04x</th>
<th>05x</th>
<th>06x</th>
<th>07x</th>
<th>08x</th>
<th>09x</th>
<th>0Ax</th>
<th>0Bx</th>
<th>0Cx</th>
<th>0Dx</th>
<th>0Ex</th>
<th>0Fx</th>
</tr>
</thead>
<tbody>
<tr>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
</tbody>
</table>

Figure B.1: Font table for `xyline10 scaled 2000`.
B.2. OBSOLETE FONTS

% with this macro package; if not, see http://www.gnu.org/licenses/.
% CONTENTS: Line segments going either segl# down or to the right. Codes
% range from 0 to 127, distributed evenly in each of the
% intervals between the 'purely' horizontal/
% vertical/diagonal characters with code 31, ..../
% 63, 95, and 127 (code -1 would be pure as : /:
% well if it existed):
% ..../ ..../ | o o.... o.... o---- o...:
% : / : / : | | :\ : :\ :
% :/ : :/ : | | : \ : : \
% o...: o...: o | :...
% Code: [-1] 0 30 31 63 64 95 127

font_identifier "XYLINE"; font_size 10pt#;
font_coding_scheme:="XY line segments";
mode_setup;

% METANESS...

segl# = 1/2 designsize; define_pixels(segl); % line segment length
rulew# = .4pt#; define_whole_blacker_pixels(rulew); % line thickness

% drawsegment draws a line from (0,0) to the argument point...
def drawsegment expr endpoint =
z0 = (0,0); z1 = endpoint;
pickup pencircle scaled rulew rotated (angle (z1-z0) - 90);
draw z0--z1; penlabels(0,1) enddef;

% TESTING...we redefine openit because the characters extend far to the
% left of the bounding box!
%
def openit = openwindow currentwindow
from origin to (screen_cols,screen_rows) at (-200,300) enddef;

% FONT.

% Font dimension 8 is the rule thickness (cf. The TeXbook, app.G)
fontdimen 8: rulew#;

% The characters follow...

for cc = 0 step 1 until 30:
beginchar(cc,(31-cc)/32*segl#,segl#,0); drawsegment (w,h); endchar;
endfor;

for cc = 31 step 1 until 63:
beginchar(cc,(cc-31)/32*segl#,0,segl#); drawsegment (w,-d); endchar;
endfor;

for cc = 64 step 1 until 94:
beginchar(cc,segl#,0,(95-cc)/32*segl#); drawsegment (w,-d); endchar;
endfor;

for cc = 95 step 1 until 126:
beginchar(cc,segl#,cc-95)/32*segl#); drawsegment (w,h); endchar;
endfor;

beginchar(127,segl#,segl#,0); drawsegment (w,h); endchar;
bye.

% $Log: xyline10.mf,v $% Revision 3.5 2010/06/10 18:45:50 krisrose
% Reference to GPL by URL.
% Revision 3.4 2010/04/16 06:06:52 krisrose
% Preparing for a new release...
% Revision 3.3 1996/12/19 03:31:56 krisrose
Figure B.2: Font table for xyqc10 scaled 2000.
This file is part of the XY-pic macro package.

The XY-pic macro package is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

The XY-pic macro package is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this macro package; if not, see http://www.gnu.org/licenses/.

CONTENTS: Quarter circles in all directions:

Each character has a cosinoid bounding box :-) with wd = cos I = sin F, where I,F are the angles from direction --> to the initial,final direction.

font_identifier "XYQC"; font_size 10pt#

font_coding_scheme:="XY quarter circles"

mode_setup;

% METANESS...

qcd# = 1/2 designsize; % quarter circle diameter
rulew# = .4pt#; % quarter circle line width
define_pixels(qcd,rulew);

% beginqc draws the quarter circle starting in the direction towards endpoint and turning left...
def beginqc(expr cc,ex,ey) =
  alpha := angle (ex,ey);
  cv := sqrt(1/2) * abs(sind(alpha + 135));
  beginchar(cc,cv+qcd#,0,0);
  pickup pencircle scaled rulew rotated (alpha - 90);
  if (cc<64): draw quartercircle shifted (-.5,0) rotated (alpha - 90) scaled qcd
  else: draw quartercircle shifted (-.5,0) rotated (alpha - 90) shifted (cv,0) scaled qcd
  f1
  endif;
enddef;

% TESTING...we redefine openit because the characters extend far to the left of the bounding box!

def openit = openwindow currentwindow from origin to (screen_cols,screen_rows) at (-200,200) enddef;

% FONT.

% Font dimension 8 is the rule thickness (cf. The TeXbook, app.G).
fontdimen 8: rulew#;

% Here are the characters...

for cc = 0 step 1 until 31:
  beginqc(cc,cc-15,-16); endchar;
endfor;

for cc = 32 step 1 until 63:
  beginqc(cc,16,cc-47); endchar;
endfor;
APPENDIX B. BACKWARDS COMPATIBILITY

for cc = 64 step 1 until 95:
    beginqc(cc,79-cc,16); endchar;
endfor;

for cc = 96 step 1 until 127:
    beginqc(cc,-16,111-cc); endchar;
endfor;

B.2.3  xymisc10

% CONTENTS:
% Metaness.
% Testing...
% Font.
% Dot fitting the rule width.
% Quarter circles with radii from 1 to 10 and 12 to 30 pt; all have
% height = depth = .5width = radius and are centered such that they
% enter and leave the box in the middle of the sides.

font_identifier "XYMISC"; font_size 10pt#;
font_coding_scheme:="XY miscellaneous";
mode_setup;

% METANESS...

rulew# = .4pt#; define_whole_blacker_pixels(rulew); % line thickness

def pickuppen = pickup pencircle scaled rulew enddef;

rulew = .4pt; define_white_blacker_pixels(rulew); % line thickness

def pickuppen = pickup pencircle scaled rulew enddef;

def openit = \let echar = endchar; def endchar = echar; stop "." enddef;

font_dimension 8 is the rule thickness (cf. The TeXbook, app.G)
### B.2. OBSOLETE FONTS

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>'00x</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'01x</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'02x</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'03x</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'04x</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'05x</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'06x</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'07x</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'10x</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'11x</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'12x</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'13x</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'14x</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>'15x</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
</tbody>
</table>

Figure B.3: Font table for `xymisc10` scaled 578.
\texttt{fontdimen 8: rule#;} \\
\% DOT FOR DOTTED LINES AND DUMMY CIRCLE... \\
\% \\
\% beginchar(0,0,0,0); pickuppen; drawdot (0,0); endchar; \\
\% beginchar(1,0,0,0); pickuppen; drawdot (0,0); endchar; \\
\% beginchar(2,0,0,0); pickuppen; drawdot (0,0); endchar; \\
\% beginchar(3,0,0,0); pickuppen; drawdot (0,0); endchar; \\
\% QUARTER CIRCLES... The four circles with radius $N$ pt, $0 < N \leq 10$, are \\
\% located at $N*4 + \text{QUADRANT}$, where $1 \leq \text{QUADRANT} \leq 4$. All have width as \\
\% their radius with reference point at the level of the horizontal end of the \\
\% arc and height, depth as the extent of the arc. \\
\% \\
\% for radius = 1 step 1 until 10: \\
\% beginchar(radius*4,2*radius*pt#,radius*pt#,radius*pt#); \\
\% pickuppen; \\
\% draw quartercircle rotated 180 shifted (1,.5) scaled (radius*2pt); \\
\% endchar; \\
\% beginchar(radius*4+1,2*radius*pt#,radius*pt#,radius*pt#); \\
\% pickuppen; \\
\% draw quartercircle rotated -90 shifted (0,.5) scaled (radius*2pt); \\
\% endchar; \\
\% beginchar(radius*4+2,2*radius*pt#,radius*pt#,radius*pt#); \\
\% pickuppen; \\
\% draw quartercircle shifted (0,-.5) scaled (radius*2pt); \\
\% endchar; \\
\% beginchar(radius*4+3,2*radius*pt#,radius*pt#,radius*pt#); \\
\% pickuppen; \\
\% draw quartercircle rotated 90 shifted (1,-.5) scaled (radius*2pt); \\
\% endchar; \\
\% endfor; \\
\% \\
\% For 12 <= $N$ <= 40 we step in 2pt increments... so for 10 <= $N$ <= 40 the \\
\% quarter circle with radius $N$ is located at $(N \div 2 + 20 + \text{QUADRANT})$. \\
\% \\
\% for radius = 12 step 2 until 40: \\
\% beginchar(radius*2+20,2*radius*pt#,radius*pt#,radius*pt#); \\
\% pickuppen; \\
\% draw quartercircle rotated 180 shifted (1,.5) scaled (radius*2pt); \\
\% endchar; \\
\% beginchar(radius*2+21,2*radius*pt#,radius*pt#,radius*pt#); \\
\% pickuppen; \\
\% draw quartercircle rotated -90 shifted (0,.5) scaled (radius*2pt); \\
\% endchar; \\
\% beginchar(radius*2+22,2*radius*pt#,radius*pt#,radius*pt#); \\
\% pickuppen; \\
\% draw quartercircle shifted (0,-.5) scaled (radius*2pt); \\
\% endchar; \\
\% beginchar(radius*2+23,2*radius*pt#,radius*pt#,radius*pt#); \\
\% pickuppen; \\
\% draw quartercircle rotated 90 shifted (1,-.5) scaled (radius*2pt); \\
\% endchar; \\
\% endfor; \\
\% \\
\% bye. \\
\% \\
\% $Log: xymisc10.mf,v$ \\
\% Revision 3.5 2010/06/10 18:45:50 krisrose \\
\% Reference to GPL by URL. \\
\% \\
\% Revision 3.4 2010/04/16 06:06:52 krisrose \\
\% Preparing for a new release... \\
\% \\
\% Revision 3.3 1996/12/19 03:31:56 krisrose \\
\% Maintenance release \\
\% \\
\% Revision 3.0 1995/07/07 20:14:21 kris \\
\% Major release w/new User's Guide! \\
\% \\
\% Revision 2.6 1992/06/24 01:23:34 kris \\
\% Cleaned up.
B.2. OBSOLETE FONTS

% Revision 2.1 1992/01/02 14:54:07 kris
% Release version.
%
% Revision 1.6 1991/11/27 06:54:21 kris
% \beta-test on DIKU.
%
% Revision 1.5 1991/10/21 23:19:08 kris
% Version described in DIKU student report 91-7-10.
%
% Revision 1.4 1991/08/22 01:07:46 kris
% Now has quarter circles to radius 40pt as required for xy.doc[1.28].
%
% Revision 1.3 1991/07/21 21:04:35 kris
% Tested with xy.doc[1.19].
%
% Revision 1.2 1991/07/19 14:52:26 kris
% Changed quarter circles to fit xy.doc[1.19].
%
% Revision 1.1 1991/06/24 20:57:37 kris
% Works with xy.doc [1.6] :-}

605
Appendix C

Licenses

C.1 GNU General Public License

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software— to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation’s software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author’s protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors’ reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone’s free use or not licensed at all.
The precise terms and conditions for copying, distribution and modification follow.

Terms and conditions for copying, distribution and modification

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The “Program”, below, refers to any such program or work, and a “work based on the Program” means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term “modification”.) Each licensee is addressed as “you”.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program’s source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.
Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the
Program subject to these terms and conditions. You may not impose any further restrictions on the recipients’ exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY
11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WAR-
RANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.
EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR
OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY
KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE
PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU AS-
SUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRIT-
ING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY
AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO
YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CON-
SEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE
PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING
RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR
A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN
IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF
SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the
best way to achieve this is to make it free software which everyone can redistribute and change under
these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of
each source file to most effectively convey the exclusion of warranty; and each file should have at least
the “copyright” line and a pointer to where the full notice is found.

<one line to give the program’s name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License along
with this program; if not, write to the Free Software Foundation, Inc.,
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

Also add information on how to contact you by electronic and paper mail.
If the program is interactive, make it output a short notice like this when it starts in an interactive
mode:
Gnomovision version 69, Copyright (C) 19yy name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
This is free software, and you are welcome to redistribute it
under certain conditions; type ‘show c’ for details.

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate parts of the
General Public License. Of course, the commands you use may be called something other than ‘show
w’ and ‘show c’; they could even be mouse-clicks or menu items—whatever suits your program.
You should also get your employer (if you work as a programmer) or your school, if any, to sign a
“copyright disclaimer” for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program
‘Gnomovision’ (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary pro-
grams. If your program is a subroutine library, you may consider it more useful to permit linking
proprietary applications with the library. If this is what you want to do, use the GNU Library
General Public License instead of this License.

C.2 Font General Public License Exceptions

Version 1, October 1998

Copyright (C) 1998 Kristoffer H. Rose (krisrose@tug.org)

Everyone is permitted to copy and distribute verbatim copies of this license document, but
changing it is not allowed.

Preamble

Software for typesetting frequently includes fonts of characters. If this software is covered by the
GNU General Public License, then a document, created as a result of running the software, which also
contains the fonts (for example for presentation purposes), is a “derived” work. This license grants
specific rights to separately distribute fonts in order to make it clear that documents containing the
fonts (with the purpose of printing and viewing the characters and images depicted in the document)
are permitted.

Exception Terms and Conditions

0. This License applies to any font or other work which contains a notice placed by the copyright
holder saying it may be distributed under the terms of the GNU General Public License, either
version 2 or a later version, below the “License”, with the explicit mention that the exceptions
of this Font General Public License Exceptions, below the “Exceptions”, apply.

The “Font”, below, refers to just the single file that has the explicit mention that these Exceptions apply.

All other terms used are those of the License.
1. You may distribute any unmodified version of the Font without the requirement that the full remaining source of the Program is enclosed.

   An intact copy of the Font file with the same name, insofar as the environment where the file is stored permits this, is considered as an unmodified version.

2. You may distribute files with the Font translated to another format provided that the following two conditions are met:

   a) The name of the Font, if it appears, must remain the same or be as close to the original name as the format permits.

   b) You accompany the file by a prominent notice about the lack of warranty for the Font as detailed in the GPL.

3. You may include fragments of the Font, or of the Font translated to another format, in documents or other work with the purpose of making it possible to view or print the document, provided that you accompany the document with a prominent notice about the lack of warranty of any kind for the Font, as detailed in the GPL.

   End of Exception Terms and Conditions

Appendix: How to Apply These Terms to Your New Fonts

If you develop a new font or an icon as part of a program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms. However, in order to avoid encumbering the use of your font in printable and viewable formats, you may want to include this exception notice with it.

To do so, attach the following notices to the font source file. It is safest to attach them to the start of the source file to most effectively convey the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

   <one line to give the font’s name and a brief idea of what it does.>
   Copyright (C) yyyy <name of author>

   This program is free software; you can redistribute it and/or modify
   it under the terms of the GNU General Public License as published by
   the Free Software Foundation; either version 2 of the License, or
   (at your option) any later version.

   This program is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
   GNU General Public License for more details.

   You should have received a copy of the GNU General Public License
   along with this program; if not, see http://www.gnu.org/licenses/.

   Special permission is granted to distribute this font separately
   under the terms of the Font General License Exceptions enclosed
   with this program.

   Otherwise follow the instructions in the appendix of the GNU General Public License.
Appendix D

Distribution support files

This appendix presents the sources of various support files that are part of the Xy-pic source distribution (except the file COPYING that contains the text reproduced in appendix A): the files with ‘hype’ for the package, the installation instructions, and the generation instructions (to make Xy-pic from the sources).

D.1 Hype

These files explain what Xy-pic is, including as a CTAN ‘LaTeX2ε bundle’ standard.

D.1.1 CATALOG

1 Name: xypic
2 Version: 3.8.9 <2013/10/06>
4 Description: Xy-pic is a package for typesetting graphs and diagrams.
5 It is structured as several modules, each defining a custom notation for
6 a particular kind of graphical object or structure. Example objects are
7 arrows, curves, frames, and colouring/rotation on drivers that support
8 it; these can be organised in matrix, directed graph, path, polygon, knot,
9 and 2-cell structure. Xy-pic works with most formats, including LaTeX,
10 AMS-LaTeX, AMS-TeX, and plain TeX, and has been used to typeset
11 complicated diagrams from many application areas including category
12 theory, automata theory, algebra, neural networks, and database theory.
13 Keywords: diagram, picture, graph, arrow, curve, commutative diagram,
15 knot, braid, two-cell, cartesian coordinates, tree.
17 Author: Kristoffer H. Rose
18 Problems-To: xy-pic@tug.org
19 Address: IBM T.J.Watson Research Center, PD Box 704, Yorktown Heights, NY 10598, USA
21 <URL: http://xy-pic.sourceforge.net>

D.1.2 README

1 -*-text-*- $Id: README,v 3.11 2010/05/06 17:46:29 krisrose Exp $
3 ===================================================================
4 THE Xy-pic PACKAGE FOR TYPESETTING GRAPHS AND DIAGRAMS
5 ===================================================================
7 Xy-pic is a package for typesetting graphs and diagrams with TeX.
9 It works with most formats, including LaTeX (both 2.09 and 2e),
10 AMS-LaTeX, AMS-TeX, and plain TeX.
12 It has been used to typeset complicated diagrams from many application
13 areas including category theory, automata theory, algebra, neural
14 networks, and database theory.
16 - o -
18 See below in this file for an overview of the structure and features!
Xy-pic is structured as a ‘kernel’ and several orthogonal modules called
‘options’, each defining a custom notation for a particular kind of
graphical object or structure.

These (combinable) structures are available:

* A graph combinator mode where diagrams are specified the way they are
  composed as graphs.
* A matrix-like mode where the dimensions of the drawing are computed by
  aligning diagram entries in rows and columns (this is the ‘diagram’
  mode Xy-pic version 2 users are used to).
* A polygon mode where diagrams shaped as regular polygons are entered
  in a simple way.
* A mode for typesetting beautiful knots and links.
* A general object-oriented ‘turtle graphic’ drawing language for
  specifying graphs with objects and connections between them in a
  manner independent of orientation.
* An experimental facility for generating movies where diagrams wiggle
  and bend into place...

The following objects may be used:

+ Positions can be given in variety of formats (extendable) including
  user defined coordinates (x,y) and relative to previous positions,
  objects, object edges, and points on connections.
+ Objects may be circular, elliptic, or rectangular (more shapes can be
  added) and adjusted in several ways.
+ Large library of objects with mnemonic names.
+ Objects that orient themself along a connection when placed relative
  to it, e.g., ‘\dir{|-}’ is like \vdash but thus oriented; new such
  objects can be defined in a convenient way.
+ TeX ‘boxes’, i.e., text and mathematical formulae.
+ Includes circle segments and optionally arbitrary elliptical,
  quadratic, and cubic arcs.
+ Connections are aligned between the reference points of objects but
  start and end on the edges.
+ Any object can be used to build a connection (using ‘diagonal
  filling’); Library objects provide common line types.
+ Flexible notation for drawing arrows and general paths with tail,
  stem, and head built from any object(s). Special support for arrows
  that cross each other, arrows that ‘go by’ other entries, paired
  arrows (including support for 2-cells), curved arrows, and arrows with
  bends.
+ Library of frames and braces.
+ Special notation for rotation, scaling, colour, and line thickness.
  The correct typesetting of these features requires a backend that
  supports it (i.e., PostScript) but even when this is not available
  Xy-pic tries to approximate what is requested such that at least the
  picture size is stable (and thus page breaks).
+ Output can use PDF or PostScript for drawing (several \special
  formats are supported, including pdftex, dvips, andTextures).
  Notation for inclusion of literal PostScript is available.
+ Other graphics can be imported and then ‘decorated’ using Xy-pic.

All object forms may be used with all structures.

Finally, enjoy Xy-pic!

Sincerely,

Kristoffer H. Rose  <krisrose@tug.org>
This is to announce a release of the diagram typesetting package Xy-pic. It has been more than a decade, so we found that a release was in order.

GENERAL

Xy-pic is a package for typesetting a variety of graphs and diagrams with TeX. Xy-pic works with most formats (including LaTeX, AMS-LaTeX, AMS-TeX, and plain TeX).

Further specifics of the package are in the distribution README file and other documentation.

LATEST NEWS

Release 3.8.9 includes Michael Barr's "diagxy" macros as the "barr" feature.

Release 3.8.8 corrects a typo in xyframes (thanks to Norbert Preining).

Release 3.8.7 includes a fix to the squiggly fonts (thanks, Daniel) to avoid PDF font generation errors.

Release 3.8.6 includes a fix to xypdf of colour allocation (thank you to Leslie Saper for the report).

Release 3.8.5 fixes some problems with xypdf, notably when used together with the beamer package.

Release 3.8.4 fixes a scoping bug in xypdf that made color definitions "bleed" to surrounding text (thanks to John G. Bullock for the report).

Release 3.8.3 fixes the xycirc10 font to work around a limitation of the mf2pt1 tool (thanks to Daniel M"ullner's fix).

Release 3.8.2 fixes the xypic.map file to load the bitmap fonts correctly (thanks to Alexander Perlis for the analysis).

Release 3.8.1 includes the missing ".enc" files in the distribution.

The primary purpose of release 3.8 is to incorporate support for proper PDF output by including Daniel M"ullner's "xypdf" package using native PDF for drawing. The driver is integrated with most of Xy-pic's extensions, however, there may still be gray areas where support is sketchy, so please try your favorite diagrams with pdfTeX and tell us if it works for you. We're grateful to Daniel for having contributed such a key piece of infrastructure that Xy-pic has been lacking for many years.

This version also marks the first version where xymatrix entries are aligned by their math axis by default as suggested by Alexander Perlis (in TUGboat 22 (2001), 330–334); thanks to Alexander for the thorough analysis. Please check whether your matrices still work as you expect!

Also thanks to Jeremy Gibbons for contributing the new "lu" style of arrow tips suitable for use with Lucida, and to Scott Pakin and the FontForge team for the great programs now used to generate all Type1 fonts directly from their METAFONT sources.

Release 3.8 furthermore fixes a few bugs in release 3.7; do tell if you believe that any of the "fixes" are really new bugs.

Finally, thanks to our loyal users; indeed if you enjoy Xy-pic then please rate us on http://sourceforge.net/projects/xy-pic where Xy-pic development is now completely in the open!
Xy-pic can be retrieved through the World Wide Web Xy-pic ‘home page’:
http://xy-pic.sourceforge.net

Make sure to check that you have reached a version 3.8 copy (some archives take a while to mirror the latest files)!

The first public release (version 1.40) of Xy-pic was created by Kristoffer H. Rose, then at DIKU, U of Copenhagen, and distributed via Usenet on December 19, 1991. This quickly became version 2 of which version 2.6 was stable for a few years.
The thorough rewrite that became version 3 is a continued collaboration with Ross Moore, Macquarie U, Sydney, initiated through a visit to Macquarie (Jan-May 1994 supported by the Australian Research Council, Macquarie University, and using donated DEC equipment). However, full backwards compatibility is maintained (except for the unavoidable but fully documented obscure cases).

Xy-pic is principally Copyright (c) 1991-2011 by Kristoffer H. Rose and 1994-2011 by Ross Moore, with contributions by several others, all under GNU COPYLEFT (GPL) which means that you can use the package for any purpose but if you provide the macros or any code derived from them to a third party then you are obliged to include the entire Xy-pic package (full details in the file COPYING). The FONTCOPYING file details the special permissions to distribute fonts without the full requirements of the GPL.

This is the end of the announcement. Enjoy Xy-pic!

---

D.2 Installation instructions

This file explains how Xy-pic is downloaded and installed, i.e., where the various files of the distribution should reside in standard TDS system.

D.2.1 INSTALL

---

This file explains how you install the runtime files in the distribution of Xy-pic. See ‘README’ for general information.

Contents:

-1- Copyleft
-2- Overview of files and where they should be installed
-3- Bugs & comments & where to get help

Note that the entire Xy-pic package is distributed with copyleft:

Copyright (c) 1991-2011 Kristoffer H. Rose <krisrose@tug.org>
Copyright (c) 1994-2011 Ross Moore <ross.moore@mq.edu.au>

The Xy-pic package is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
The Xy-pic package is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
You should have received a copy of the GNU General Public License along with this package; if not, see http://www.gnu.org/licenses/.

What this means is that if you install Xy-pic then you should (1) make sure that the file ‘COPYING’ is accessible to the users, and (2) be prepared to provide Xy-pic (including sources including of your modifications) to any of your users that request it (you may charge a fee for this if you are so inclined).

Note: Some option files may be copyrighted by other people and use a different (more relaxed) license but the conditions remain the same for the entirety of Xy-pic.

--2- OVERVIEW OF FILES AND WHERE THEY SHOULD BE INSTALLED

In this section we explain in 10 steps where the different groups of Xy-pic files in the ‘run’ and ‘doc’ distribution should be installed. For each file category we indicate the location recommended by TDS, the TeX Directory Standard (as formed by a TUG working group of that name).

IMPORTANT

---

If you have an older system (more than 10 years) then these instructions may not be suitable. Go to the source directory http://xy-pic.cvs.sourceforge.net/viewvc/xy-pic/src/Historic/ where we keep some older instructions, or ask on the mailing list.

Step 1. Availability & License

These eight files contain general information and should be installed in a publicly readable place:

- README: generic information on Xy-pic
- CATALOG: version information and summary
- TRAILER: announcement and availability
- INSTALL: how to install (this file)
- MANIFEST: list of all files
- VERSIONS: the RCS version ids of all source files
- COPYING: GNU General Public License
- FONTCOPYING: Special license for fonts

TDS directory: texmf/doc/generic/xypic

Step 2. Macros

Files ‘texinputs/*’ are TeX and LaTeX macro files that must be installed in a place which TeX and LaTeX will search (e.g., with un*x TeX, a directory users include in the TEXINPUTS environment variable).

TDS directory: texmf/tex/generic/xypic

Step 3. TeX Font Metric

Files ‘texfonts/*.tfm’ should be installed in a directory that TeX will search for fonts (e.g., with un*x TeX, a directory users include in the TEXFONTS environment variable).

TDS directory: texmf/fonts/tfm/public/xypic

Step 4. METAFONT Sources

Files ‘mfinputs/*.mf’ should be installed in some directory where the METAFONT program will search for them (e.g., with un*x TeX, a directory included in the MFINPUTS environment variable).

NOTE: The METAFONT source files are only useful on installations where the printer driver and previewer can utilise them to generate the required bitmap fonts ‘on-line’. If this is possible on your installation then please use it: it saves substantial amounts of space and means that you will automatically use fonts optimised for each printer (provided everything is correctly set up, of course).

TDS directory: texmf/fonts/source/public/xypic
Step 5. PostScript Type1 Fonts
-------------------------------
Files `type1/*.pfb' contain versions of the Xy-pic fonts in special
PostScript form that may be useful on some installations. They should
be installed where the applications using them look for such fonts (if
used with DVIPS then the `ps/xypic.map' file from the next group
should also be installed).

NOTE: Some installations, notably teTeX 0.4, will PREFER using these
fonts over the METAFONT-generated equivalents (IMHO this is a bug in
the MakeTeXPK script). On those systems we recommend that you ALSO
install the PK fonts to circumvent the problem.

IMPORTANT: Make sure that you do not have any old Xy-pic PostScript
`xy*.pfa' fonts floating around in your system - search for and remove
all files named `xy*.pfa' when you have installed the new fonts. One
common way old (and wrong) fonts is manifest is when some arrowheads
appear skewed or too thin.

TDS directory: texmf/fonts/type1/public/xypic

Step 6. PostScript Font Metric Files
------------------------------------
Files `pfm/*.pfm' are special files for some font handling software.
Install them if you need them.

TDS directory (usually): texmf/fonts/pfm/public/xypic

Step 7. Adobe Font Metric Files
-------------------------------
Files `afm/*.afm' are special files for the Adobe Type Manager.
Install them if you need them.

TDS directory (usually): texmf/fonts/afm/public/xypic

Step 8. PostScript Header Files
-------------------------------
Files `ps/*' contain PostScript header files. If you plan to use
the PostScript extension option then you should copy these files to a
place where your PostScript DVI driver will find them.

TDS directory (usually): texmf/dvips/xypic

Step 9. PostScript Font Map Files
---------------------------------
Files `map/*' contain PostScript and PDF font mappings that link each
font to the used encoding.

TDS directory (usually): texmf/fonts/map/dvips/xypic

Step 10. PDF Font Encoding Files
---------------------------------
Files `enc/*' contain PDF encoding files (that name each character in
the special Xy-pic fonts). If you plan to use the PDF backend then
you should copy these files to a place where PDFTeX will find them.

TDS directory: texmf/fonts/enc/dvips/xypic

Step 11. Documentation
----------------------
The documentation is provided in Adobe PDF files.

doc/xyrefer.pdf Reference Manual for experienced users

The source code documentation is also included.
doc/xysource.pdf TeXnical documentation for TeXperts
doc/xypdf.pdf TeXnical details about PDF generation
doc/xysrc.tar.gz the actual source files

TDS directory: texmf/doc/generic/xypic

That's all.

---------------------------------------------------------------------------
-3- BUGS & COMMENTS & WHERE TO GET HELP
---------------------------------------------------------------------------

We will appreciate reports on any problems you may encounter and
opinions you have on how the usefulness of Xy-pic can be improved.

Please report using either

Electronic mail (Internet): xy-pic@tug.org [preferred]

Paper mail: Kristoffer H. Rose
IBM T.J.Watson Research Center
P.O.Box 704
Yorktown Heights, NY 10598
USA

W3 <URL: http://sourceforge.net/projects/xy-pic/>

This is the end of INSTALL. We hope you'll enjoy Xy-pic!

-- Ross Moore and Kristoffer Rose

D.3 Generation

These files are relevant when Xy-pic is generated from sources on a un*ix system.

D.3.1 MAKE

Welcome to the source distribution of the Xy-pic macros for typesetting
diagrams! Below you will find the following sections:

-1- License information
-2- Making the package and manual from sources
-3- Bugs & comments
-4- Distribution log

Please see the file ‘README’ for a summary of features ‘TRAILER’ for how
you can obtain the newest Xy-pic, and the various ‘INSTALL’ files if you
have already retrieved pregenerated macros and fonts.

The Xy-pic package is licensed under ‘GNU CopyLeft’:

Xy-pic: Typesetting graphs and diagrams with TeX.
Copyright (c) 1991-2011 Kristoffer H. Rose <krisrose@tug.org>
The Xy-pic macro package is free software; you can redistribute it
and/or modify it under the terms of the GNU General Public License as
published by the Free Software Foundation; either version 2 of the
License, or (at your option) any later version.
The Xy-pic macro package is distributed in the hope that it will be
useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
General Public License for more details.
You should have received a copy of the GNU General Public License
along with this package; if not, see http://www.gnu.org/licenses/.
(Some options may be copyright by other people but the conditions remain
the same).
The full text of the GNU General Public License is supplied with Xy-pic
in the file ‘COPYING’ and reproduced in the source documentation.
As a special provision, the FUNTS provided with Xy-pic are provided
under a much more relaxed license to avoid any risk that documents
where the fonts are embedded are themselves covered by this copyright.

-Making the package and manual from sources
NOTE FOR OTHER NON-UN*X USERS: If you do not have a un*x system then you are best advised to install as described in `INSTALL'.

With a "TeX Live" installation - the common kind these days - the Makefile should work without any modifications.*

Go through the Makefile and check that everything is to your liking, then you can do the usual ‘make’ and ‘make install’ to install the package.

That is all.

* The source documentation, however, does requires a more complete CTAN installation.

-----------------------------------------------------------------------
-3- BUGS & COMMENTS
-----------------------------------------------------------------------
I will appreciate reports on any problems you may encounter and opinions you have on how the usefulness of Xy-pic can be improved.

Please report using either

Electronic mail (Internet): xy-pic@tug.org [preferred]

Paper mail: Kristoffer H. Rose
IBM T.J.Watson Research Center
P.O.Box 704
Yorktown Heights, NY 10598
USA

W3 <URL: http://www.krisrose.net>

-----------------------------------------------------------------------
-4- DISTRIBUTION LOG
-----------------------------------------------------------------------
The following versions have been distributed publicly:

$Log: MAKE,v $
Revision 3.12 2011/03/14 20:14:00 krisrose
Preparing for release 3.8.6.

Revision 3.11 2010/06/10 18:45:49 krisrose
Reference to GPL by URL.

Revision 3.10 2010/05/13 23:26:49 krisrose
TeXnical documentation not included in standard package.

Revision 3.9 2010/05/06 17:46:29 krisrose
Ross Moore’s e-mail address updated.

Revision 3.8 2010/04/25 21:48:05 krisrose
First proper integration of xypdf into Xy-pic "make dist".

Revision 3.7 2010/04/21 23:30:18 krisrose
Stable release. Font license exception added.

Revision 3.6 2010/04/16 06:06:51 krisrose
Preparing for a new release...

Revision 3.5 1997/05/28 22:40:26 krisrose
Fixed missing breaks bug.

Revision 3.3 1996/12/19 03:31:56 krisrose
Maintenance release

Revision 3.1 1995/09/05 20:31:32 kris
Releasing!

Revision 3.0 1995/07/07 20:14:21 kris
Major release w/new User’s Guide!

Renamed to MAKE for version 3 source description.

Revision 2.12 1994/10/25 11:34:25 kris
Interim release just before v3 [works with AMS-LaTeX 1.2]...

Revision 2.11 1994/07/05 10:37:32 kris
Third 3beta release [bug fixes].

Experimental graph feature included (for ECCT-94 presentation).

Revision 2.10 1994/06/15 12:59:44 kris
Second 3beta release [bug fixes].

Revision 2.8 1994/04/11 09:31:09 kris
D.3. GENERATION

Second (bug fix) 3alpha release [corrected].
Revision 2.7 1994/03/08 02:06:01 kris
Release 3alpha.
MAJOR REWRITE of distribution to prepare for Xy-pic 3 functionality.
Curve extension code contributed by Ross Moore <ross.moore@mq.edu.au>;
support funding from MURG, ARC, and equipment from DEC.
Revision 2.6 1992/06/24 01:23:34 kris
Added 'hook' tips using new font xyqc10.
Added new POSTions: * and !.
Added triple lines \Ssolid, \Ddashed, \Ddotted, and companion \Ttip.
Reorganised manual.
Revision 2.5 1992/02/24 03:30:54 kris
Fixed bugs in \Direction calculation logic...
Added \Ddashed to \rotate to allow arbitrary rotation.
' intermediate points now accept an optional /RADIUS argument.
Added \Ttip with wide tip.
Revision 2.4 1992/01/22 02:15:10 kris
Fixed bugs [with thanks]:
\ works with AMS-LaTeX; don't use \Lett [Werner Struckmann/Darrel Hankerson]
No spurious arrow heads with LaTeX; \pit now undefined [Werner Struckmann]
\Solid works: sets \Density [Dave Bowen]
Short diagonal lines work...major rewrite of \connectv [Eric Domenjoud]
Revision 2.3 1992/01/10 21:38:27 kris
Several bugs fixed [with thanks]:
The \ddtoX and \utoX arrows were interchanged [Nico Verwer].
Some diagonal lines were wrong [Eric Domenjoud].
AMS-LaTeX users had name clash problems [Werner Struckmann].
The installation instructions in the Makefile were buggy [Leen Torenvliet].
Revision 2.2 1992/01/09 04:05:40 kris
Patched to fix problem with \text and support AMS-LaTeX.
Revision 2.1 1992/01/02 14:54:07 kris
Release version.
Revision 1.40 1991/12/17 04:53:23 kris
Version distributed as 'final draft' on Usenet.

This is the end of the source overview. I hope you will enjoy Xy-pic!

D.3.2 Makefile

# $Id: Makefile,v 3.74 2013/10/06 01:12:08 krisrose Exp$
#
# This file is part of the Xy-pic macro package.
# See the README and INSTALL files for further information.
# The Xy-pic package is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by the
# Free Software Foundation; either version 2 of the License, or (at your
# option) any later version.
# You should have received a copy of the GNU General Public License
# for more details.
# Update these together with xy.doc's xyversion!

# Copyright (c) 1991-2011 Kristoffer H. Rose <krisrose@tug.org>
APPENDIX D. DISTRIBUTION SUPPORT FILES

24 VERSION = 3.8.9
25 SHORTVERSION = 389
26
27 # CONTENTS:
28 #
29 # Configuration section:
30 #
31 # Directories.
32 # Fonts.
33 # Programs.
34 #
35 # Generic targets.
36 # Macros.
37 # Fonts.
38 # PostScript and PDF font support.
39 # Documentation.
40 # Home page.
41 # Distribution.
42 # (Xy-pic, the movie.)
43 # Tags.
44 #
45 # Log.

47 #### CONFIGURATION SECTION ##################################################################
48 # Please configure as instructed below for each of the DIRECTORIES, FONTS, 
49 # and PROGRAMS parts by editing the definitions to fit your system.
50 #### DIRECTORIES.
51 #
52 # The following variables define the directories where the various parts of 
53 # Xy-pic should be installed.
54 #
55 # You should *EITHER* set TDSROOT to the root of a TDS-compliant directory 
56 # tree, usually named .../texmf (if you have a TDS [TeX Directory Structure] 
57 # compliant system then that is in fact all you need to do): 
58 #
59 # USR = /usr/local
60 # TDSROOT = $(USR)/share/texmf
61 #
62 # *OR* you should set each of the following variables right for you system 
63 # (do this if your setup does not match TDS exactly):
64 #
65 # TEXINPUTDIR = $(TDSROOT)/tex/generic/xypic
66 # TEXFONTDIR = $(TDSROOT)/fonts/tfm/public/xypic
67 # TEXDOCDIR = $(TDSROOT)/doc/generic/xypic
68 # MFINPUTDIR = $(TDSROOT)/fonts/source/public/xypic
69 # TYPE1DIR = $(TDSROOT)/fonts/type1/public/xypic
70 # PFMDIR = $(TDSROOT)/fonts/pfm/public/xypic
71 # AFMDIR = $(TDSROOT)/fonts/afm/public/xypic
72 # PSHEADERDIR = $(TDSROOT)/ps/xypic
73 #
74 # FONTDIR is the directory where bitmap fonts are stored and is special 
75 # (If you do not plan to install bitmap fonts because they are automatically 
76 # generated and remember to set the MAGS variable below to ' ' then you may 
77 # ignore this step.)
78 #
79 # Since the location of individual fonts sometimes depend on properties of 
80 # the font, you can insert 
81 #
82 # $${dpi} for the font resolution in ‘Dots Per Inch’ (NOT the 
83 # device resolution---that should be inserted directly)
84 #
85 # $${mode} for the METAFONT ‘mode’ used to generate the font
86 #
87 # ---what these get set to depends on the next section.
88 #
89 # For example, if you set FONTDIR=/usr/lib/tex/pk$${dpi} and decide in the 
90 # next section to generate PK fonts at 300 dpi using a generic mode 
91 # (localfont) then you get files named /usr/lib/tex/pk300/xy*.300pk.
92 #
93 # If your installation still uses the (inefficient) principle of having all 
94 # PK files in the same directory as the TFM files then you should just set 
95 # FONTDIR=$(TEXFONTDIR).
D.3. GENERATION

98 #
99 FONTDIR = $(TDSROOT)/fonts/pk/\{mode\}/public/xypic/dpi\{dpi\}
100 ##### FONTS.
101 #
102 # The META FONT program needs to know the 'mode' of your printer in order to
103 # be able to generate correct fonts for it. On most installations the mode
104 # 'localfont' is made synonymous with the mode required for the default
105 # printer; in that case you can just pick the 'MODES = localfont' line. If
106 # this is not the case -- or you need fonts for several output devices---then
107 # you will have to consult the local META FONT guide/guru to inquire what
108 # modes are used at your site. A good place to look for the modes used is in
109 # the MakeTeXPK script found on some sites.
110 #
111 # Set MODES to the list of printers you wish to use:
112 #
113 MODES = localfont
114 #MODES = cx ljfour
115 #
116 # You should also decide which TeX \magsteps you wish the font to be
117 # generated at.
118 #
119 # The default is . (dot) which prevents installation of any bitmaps because
120 # most modern installations generate them automatically.
121 #
122 MADS = .
123 #MADS = 0
124 #MADS = 0 0.5 1 2
125 ##### PROGRAMS.
126 #
127 SHELL = /bin/sh
128 #
129 TEX = TEXFONTS=.:$(TEXFONTDIR):$$TEXFONTS tex
130 LATEX = TEXFONTS=.:$(TEXFONTDIR):$$TEXFONTS latex
131 PDFLATEX = TEXFONTS=.:$(TEXFONTDIR):$$TEXFONTS pdflatex
132 BIBTEX = BIBINPUTS=. bibtex
133 MAKEINDEX = makeindex
134 #
135 # The standard post-DVI processing commands...
136 #
137 DVIPS = dvips -D1200 -t $(PAPER) -f
138 DVIPDFMX = dvipdfmx -p $(PAPER) -V 5 -g 2pt
139 PS2PDF = ps2pdf -sPAPERSIZE=$(PAPER) -dCompatibilityLevel=1.5 -dPDFSETTINGS=/prepress
140 #
141 # Set MF to a command that will run plain META FONT on your system.
142 MF = MFINPUTS=.:$(MFINPUTDIR):$$MFINPUTS mf
143 #
144 # Set MPOST to a command that will run METAPOST on your system.
145 MPOST = MFINPUTS=.:$(MFINPUTDIR):$$MFINPUTS mpost
146 #
147 # Used to generate PostScript font metric files (NOT outlines).
148 MFTRACE = mtrace --formats=afm,pmf
149 #
150 # Set GFTOPK to the command "false" and PK to "gf" if you use GF rather than PK
151 # bitmap font files. There is no support for PXL-files [are they still used?].
152 # (Note: the default setting of MADS above doesn't use bitmaps at all.)
153 #
154 GFTOPK = gftopk
155 PK=gf
156 #
157 RM = rm -fr
158 CP = cp -f
APPENDIX D. DISTRIBUTION SUPPORT FILES

MV = mv -f
#
Set the INSTALLs to cp -f if your system does not have the install command.
#
INSTALL = install -c -m 444
INSTALLW = install -c -m 644
INSTALLX = install -c -m 555
#
MKDIR = mkdir -p -m 755
#
# Set MKDIR to a mkdir command that creates intermediate directories.
#
MD5SUM = md5sum
SIGN = (cd;gpg --clearsign)
#
DOC2TEX <file>: output efficient .tex file from inefficient .doc <file> with
# inlined documentation [my favourite hack :-]. Removes all DOCMODE lines,
# leaving only things before the first and in DOCMODE(...DOCMODE) brackets;
# also removes all blank lines and comment lines.
# If it breaks your sed you can just set DOC2TEX to cat although that will
# make the installed macro files three times larger and the reading of them
# somewhat slower...so in that case you are probably better of getting the
# 'normal' distribution where this has already been done for you!
# If you are on a Windows system or otherwise adventurous then you can also
# try the included "doc2tex.com" script contributed by R.Gaertner.
DOC2TEX = sed \
-e '/^\*.DOCMODE/ b Yes' \
-e '/^\*.DOCMODE/b No' \
-e 'x' \ 
-e '/XXDONTCOPY%b Ignore' \ 
-e 'x' \ 
-e 's/\[%.*$$/\[%/' -e 's/}%.*$$/}%/' -e 's/^%% \$$Id/%% $@ from $$Id/' \
-e 's/REPLACEWITHVERSION/$\{VERSION\}/' \
-e 's/REPLACEWITHSHORTVERSION/$\{SHORTVERSION\}/\ ' \
-e 'b End' \
-e 's/No\ -e s/\%/XXDONTCOPY%\ -e \m -e \d' \ 
-e 'Yes\ -e s/\%/\ -e \m -e \d' \ 
-e 'Ignore\ -e \m -e \d' \ 
-e 'End' \
#
MF2TFM script: make .tfm file and all requested bitmap files from .mf
# file by executing METAFONT with $$\{mode\} and $$\{dpi\} as described above.
# NOTE: for use as implicit make rule; do not modify unless you are sure you
# need to!
MF2TFM = set -x; for mode in $(MODES); do \
  if [ "$(MAGS)" = "." ]; then \ 
    $(MF) \"\mode=\$\{mode\}; input \$<\"; \
    $(RM) \$*.gf; \ 
  else \ 
    for mag in $(MAGS); do \ 
      $(MF) \"\mode=\$\{mode\}; mag=magstep($$mag); input \$<\"; done; \ 
      for f in \$*.gf; do if test -f "$\{f\}"; then \ 
        if $(GFTOPK) ./$$f; then $(RM) $$f; fi else ; fi; done; \ 
      fi; done \ 
  fi; done \
#
Compression.
#
TAR <files> output tar archive with <files>
D.3. GENERATION

# GZIP compression filter
# GUNZIP uncompression filter
# GZ filename extension appropriate for compressed files

# TODO: change compression to proper GNU gzip once everyone supports it...

TAR = tar cf -
GZIP = gzip -fv9
GUNZIP = gunzip
GZ = .gz
ZIP = zip
UNZIP = unzip -qaoLDD

# GNU AWK (gawk) or (in a pinch) NAWK.
GAWK = gawk

# Finally the paper size to use!
PAPER = letter

### END OF CONFIGURATION SECTION ###################################################################

# GENERIC TARGETS.

all: macros fonts manual
.PHONY: all install clean realclean sterile

install: all install.macros install.trailer 
install.fonts install.ps

realclean:: clean
$(RM) `cat .cvsignore`
touch xydoc.front xydoc.back

# MACROS.

TRAILERSOURCES = TRAILER COPYING CATALOG.doc FONTCOPYING FONTCOPYING.patch 
MAKE README Makefile INSTALL
KERNELSOURCES = xy.doc xyidioms.doc xyrecat.doc
OPTIONSOURCES = 

MACROSOURCES = $(TRAILERSOURCES) $(KERNELSOURCES) $(LATEXSTYLES) 

LATEXSTYLES = xy.sty xypic.sty movie.cls
OPTIONDATA = xymacpat.xyp

NOTES = TRAILER COPYING FONTCOPYING CATALOG README 
INSTALL VERSIONS Xy-pic.html Xy-logo.png
KERNEL = xy.tex xyidioms.tex xyrecat.tex
OPTION = 

xyframe.tex xyctip.tex xyttex.tex xycurve.tex 
xyline.tex xyrotate.tex xycolor.tex xycrayon.tex xytile.tex 
xyimport.tex xypic.tex xytp-f.tex 
xyzips.doc xyzips-ps.doc xyzips-pro.doc xyzips-col.doc xyzips-c.doc xyzips-f.doc 
xyzips-l.doc xyzips-r.doc xyzips-s.doc xyzips-t.doc 
xypredict.tex xypatt.tex xydocps.tex 
xydummy.doc xyall.doc xymatrix.tex xyarrow.doc xygraph.tex 
xy2cell.doc xypoly.doc xyarc.doc xyknot.doc xyweb.doc xyling.doc 
xypics.doc xypic.tex xytextures.tex xytexstyles.tex xytexstyles.tex xytexstyles.tex xytexstyles.tex xytexstyles.tex 
xymatrix.doc xyvpt.tex xymatrix.tex xymatrix.tex xymatrix.tex xymatrix.tex 
xymovie.doc xyv2.doc xyimg.tex xypicture.tex 
xyecula.doc xybarr.doc xyzzdoc.tex 
xypdf.doc xypdf.tex

LATEXSTYLES = xy.sty xypic.sty movie.cls
OPTIONDATA = xymacpat.xyp

# $(TRAILERSOURCES) $(KERNELSOURCES) $(LATEXSTYLES) 
# $(OPTIONSOURCES) $(OPTIONDATA)

KERNEL = xy.tex xyidioms.tex xyrecat.tex
OPTION = 

xyframe.tex xyctip.tex xyttex.tex xycurve.tex 
xyline.tex xyrotate.tex xycolor.tex xycrayon.tex xytile.tex
MACROS = $(KERNEL) $(OPTION) \\
$(LATEXSTYLES) \\
PSPRO = xy$(SHORTVERSION)dict.pro \\
PSMAP = xypic.map \\
SUPPORT = install-tds \\
dvito.gif9a pmrawttocropwhite.c

# Macros have .tex suffix and are generated from the .doc files.

#.SUFFIXES: .tex .doc
#.doc.tex:; $(DOC2TEX) $< > $@

# .PHONY: macros install.macros
# macros: $(MACROS)
# install.macros: $(MACROS)
# -test -d $(TEXINPUTDIR) || $(MKDIR) $(TEXINPUTDIR)
# for f in $(MACROS); do $(INSTALL) $$f $(TEXINPUTDIR)/$$f; done

# Kernel dependencies:

# Extension option dependencies:

# Feature option dependencies:
D.3. GENERATION

391 xyall.tex: xyall.doc
392 #
393 xymatrix.tex: xymatrix.doc
394 xyarrows.tex: xyarrows.doc
395 xygraphs.tex: xygraphs.doc
396 xy2cell.tex: xy2cell.doc
397 xypoly.tex: xypoly.doc
398 xyarc.tex: xyarc.doc
399 xyknot.tex: xyknot.doc
400 xyweb.tex: xyweb.doc
401 #
402 # Driver option dependencies:
403 #
404 xycmactex.tex: xycmactex.doc
405 xydvips.tex: xydvips.doc
406 xydvitops.tex: xydvitops.doc
407 xydvidrv.tex: xydvidrv.doc
408 xydocps.tex: xydocps.doc
409 yxtexstyles.tex: yxtexstyles.doc
410 xy16texstyles.tex: xy16texstyles.doc
411 xyoxtex.tex: xyoxtex.doc
412 xymovie.tex: xymovie.doc
413 xy17oxtex.tex: xy17oxtex.doc
414 xyxdvi.tex: xyxdvi.doc
415 #
416 # PDF option is special.
417 #
418 xypdf.tex xypdf-co.tex xypdf-cu.tex xypdf-fr.tex xypdf-li.tex xypdf-ro.tex: xypdf.dtx xypdf.ins
419 $(LATEX) xypdf.ins
420 #
421 # Backwards compatibility dependencies:
422 #
423 xypic.tex: xypic.doc
424 #
425 # Required documentation files
426 #
427 install.trailer: $(NOTES)
428 -test -d $(TEXDOCDIR) || $(MKDIR) $(TEXDOCDIR)
429 for f in $(NOTES); do $(INSTALL) $$f $(TEXDOCDIR)/$$f; done
430 #
431 # CATALOG for CTAN...are these still used?
432 CATALOG: CATALOG.doc xy.tex
433 $(LATEX) CATALOG.doc
434 $(MV) CATALOG.out CATALOG
435 #
436 # The VERSIONS is just a list of the CVS versions.
437 VERSIONS: $(SOURCES)
438 sed -n 's/^.*\[\$\$\Id\[:\]\[^$$\]*\]\.$$/\1/p' $(SOURCES) | sort -u >VERSIONS
439 #
440 # PHONY.
441 #
442 fonts: texfonts $(PK)fonts fonts psfonts 
443 install.fonts
444 #
445 #
446 #
447 fonts: texfonts $(PK)fonts psfonts
448 #
457 #
FONTSOURCEs = $(METAFONTS) xytest.mf
# Note: we only depend on the tfm-files, so you must remove them to
# regenerate the fonts at all desired magnifications.
#Suffixes: .mf .tfm .pfb .afm .pfm .pfa
#FONTSOURCES = $(METAFONTS) xyqc10.mf
# FONTSOURCES = $(METAFONTS) xyline10.mf
# FONTSOURCES = $(METAFONTS) xymisc10.mf

$FONTDIR$ = $(TEXFONTDIR) $TEXFONTS$; $(METAFONTS)$

xydash10.mf: xyd.mf xyd.mf xytest.mf
xybsql10.mf: xyd.mf xyd.mf xytest.mf
xycirc10.mf: xycirc.mf xytest.mf
xyatip10.mf: xyatip.mf xyatip.mf xytest.mf
xybtip10.mf: xyatip.mf xyatip.mf xytest.mf
xycmat10.mf: xycmat.mf xytest.mf
xycmbt10.mf: xycmbt.mf xytest.mf
xycmat11.mf: xycmat.mf xytest.mf
xycmbt11.mf: xycmbt.mf xytest.mf
xycmat12.mf: xycmat.mf xytest.mf
xycmbt12.mf: xycmbt.mf xytest.mf
xyeuat10.mf: xyeuat.mf xytest.mf
xyeubt10.mf: xyeubt.mf xytest.mf
xyeuat11.mf: xyeuat.mf xytest.mf
xyeubt11.mf: xyeubt.mf xytest.mf
xyeuat12.mf: xyeuat.mf xytest.mf
xyeubt12.mf: xyeubt.mf xytest.mf
xyluat10.mf: xyluat.mf xytest.mf
xylubt10.mf: xylubt.mf xytest.mf
xyluat11.mf: xyluat.mf xytest.mf
xylubt11.mf: xylubt.mf xytest.mf
xyluat12.mf: xyluat.mf xytest.mf
xylubt12.mf: xylubt.mf xytest.mf

# OBSOLETE ONES...
xyline10.mf: xyline10.mf xytest.mf
xyqc10.mf: xyqc10.mf xytest.mf
xymisc10.mf: xymisc10.mf xytest.mf

# POSTSCRIPT AND PDF FONT SUPPORT
# Generated PostScript Type1 fonts.

TYPE1FONTS = \\
xyatip10.pfb xyeuat10.pfb xybtip10.pfb xycmat10.pfb xyd2.mf xydash10.pfb \\
D.3. GENERATION


PSFONTS = $(TYPE1FONTS) $(AFMFONTS) $(PFMFONTS)

PDFENC = xyd.enc xyd2.enc xycirc.enc

# These use the mf2pt1.zip package (included unmodified in the sources as required by the rules).

Daniel: --bpppix=.04 decreases the default precision (bpppix=.02) by 1 bit (which # is not significant) and prevents an error message

%.pfb: %.mf mf2pt1.mem
  ./mf2pt1/mf2pt1.pl --encoding=xyd.enc --fontversion=001.$(SHORTVERSION) --rounding=.0001 --bpppix=.04 $<

# Special cases for the two non-xyd-encoded fonts.

xydash10.pfb: xydash10.mf mf2pt1.mem xyd2.enc
  ./mf2pt1/mf2pt1.pl --encoding=xyd2.enc --fontversion=001.$(SHORTVERSION) --rounding=.0001 xydash10.mf

xycirc10.pfb: xycirc10.mf mf2pt1.mem xycirc.enc
  ./mf2pt1/mf2pt1.pl --encoding=xycirc.enc --fontversion=001.$(SHORTVERSION) --rounding=.0001 xycirc10.mf

# The helper script.

  $(MPOST) -progname=mpost -ini mf2pt1/mf2pt1.mp \dump

# Generating pfm/afm Type1 font metric files from the base name of a METAFONT file.

%.afm %.pfm: %.tfm
  $(MFTRACE) --encoding=xyd.enc $*

# Special cases for non-xyd-encoded fonts.

xydash10.afm xydash10.pfm: xydash10.tfm
  $(MFTRACE) --encoding=xyd2.enc xydash10

xycirc10.afm xycirc10.pfm: xycirc10.tfm
  $(MFTRACE) --encoding=xycirc.enc xycirc10

# PostScript headers.

xy$(SHORTVERSION)dict.pro: xydocps.tex xypsdict.tex xypspatt.tex xyps-pro.tex
  $($(RM) xy$(SHORTVERSION)dict.pro)
  $(TEX) xydocps.tex

# DOCUMENTATION.

xy$(SHORTVERSION)dict.pro: $(PSPRO) $(PSMAP) psfonts
  for f in $(TYPE1FONTS); do $(INSTALL) $$f $(TYPE1DIR)/$$f; done
  for f in $(PFMFONTS); do $(INSTALL) $$f $(PFMDIR)/$$f; done
  for f in $(AFMFONTS); do $(INSTALL) $$f $(AFMDIR)/$$f; done
  for f in $(PSPRO); do $(INSTALL) $$f $(PSHEADERDIR)/$$f; done
  for f in $(PSMAP); do $(INSTALL) $$f $(PSHEADERDIR)/$$f; done

# DOCUMENTATION.
documentation: manual TeXnical
#
# Manuals use the geometry package which we store in the archive in original form as required by the LaTeX license.
#
geometry.sty: geometry.zip
rm -fr geometry
unzip geometry.zip

cd geometry && $(LATEX) geometry.ins
mv geometry/geometry.sty .
clean::
rm -fr geometry
realclean::
rm -f geometry.sty
#
MANUALDATA = ross.eps kris.eps import1.eps geometry.sty
MANUALSOURCES = xyguide.man xydoc.sty latin1.sty xydoc.bib xyrefer.man $(KERNELSOURCES) $(LATEXSTYLES) $(OPTIONSOURCES) $(OPTIONDATA) $(MANUALDATA) barrdoc.latex
#
GUIDE = xyguide.pdf
REFER = xyrefer.pdf barrdoc.pdf
#
guide: $(GUIDE)
refer: $(REFER)
manual: $(GUIDE) $(REFER)
#
# Each is different...
#
# Guide is generated with dvipdfmx specials.
#
xyguide.pdf: xyguide.man macros psfonts xydoc.sty latin1.sty xydoc.bib dvipdfmx.def
$(CP) xyguide.man xyguide.tex
$(RM) xyguide.aux xyguide.bbl xyguide.ind xyguide.toc

echo "\PassOptionsToPackage{dvipdfmx}{color}\PassOptionsToPackage{dvipdfmx}{graphics}\PassOptionsToPackage{dvipdfmx}{hyperref}" > xydoc.front

echo "\xyoption{pdf}\pdftrue" > xydoc.back

$(LATEX) xyguide
$(BIBTEX) xyguide
$(LATEX) xyguide
$(LATEX) xyguide
$(MAKEINDEX) xyguide
$(LATEX) xyguide
$(LATEX) xyguide
$(DVIPDFMX) xyguide.dvi
#
## Alternate guide generation with straight pdfTeX.
#
xyguide.pdf: xyguide.man macros psfonts xydoc.sty latin1.sty xydoc.bib dvipdfmx.def
$(CP) xyguide.man xyguide.tex
$(RM) xyguide.aux xyguide.bbl xyguide.ind xyguide.toc

echo "" > xydoc.front

echo "\xyoption{pdf}" > xydoc.back

$(LATEX) xyguide
$(BIBTEX) xyguide
$(LATEX) xyguide
$(LATEX) xyguide
$(MAKEINDEX) xyguide
$(PDFLATEX) xyguide
$(BIBTEX) xyguide
$(PDFLATEX) xyguide
$(PDFLATEX) xyguide
$(PDFLATEX) xyguide
$(PDFLATEX) xyguide
$(PDFLATEX) xyguide
### Reference manual is generated with dvips and ps2pdf specials.
#
xyrefer.pdf: xyrefer.man macros psfonts xydoc.sty latin1.sty xydoc.bib dvipdfmx.def
$(KERNELSOURCES) $(OPTIONSOURCES) $(MANUALDATA)
$(CP) xyrefer.man xyrefer.tex
$(RM) xyrefer.aux xyrefer.bbl xyrefer.ind xyrefer.toc

echo "\PassOptionsToPackage{dvips}{color}\PassOptionsToPackage{dvips}{graphics}\PassOptionsToPackage{ps2pdf}{hyperref}" > xydoc.front

echo "\xyoption{dvips}\xyoption{ps}" > xydoc.back

$(LATEX) xyrefer
$(BIBTEX) xyrefer
$(LATEX) xyrefer
$(LATEX) xyrefer
$(MAKEINDEX) xyrefer
$(LATEX) xyrefer
$(LATEX) xyrefer
$(LATEX) xyrefer

APPENDIX D. DISTRIBUTION SUPPORT FILES
D.3. GENERATION

677  $(\LaTeX)$ xyrefer
678  $(\dvipsp)$ < xyrefer.dvi > xyrefer.ps
679  $(\ps2pdf)$ xyrefer.ps xyrefer.pdf
680  
681  #
682  dvi2ps.def:: wget http://www.ctan.org/tex-archive/macros/latex/contrib/dvi2ps/dvi2ps.def
683  
684  # Barr's diagxy documentation.
685  barrdoc.pdf: barrdoc.latex
686  touch barrdoc.ind
687  $(\pdf\LaTeX)$ barrdoc.latex
688  $(\pdf\LaTeX)$ barrdoc.latex
689  $(\makeindex)$ barrdoc
690  $(\pdf\LaTeX)$ barrdoc.latex
691  $(\pdf\LaTeX)$ barrdoc.latex
692  
693  # TeXNICAL DOCUMENTATION.
694  
695  .PHONY: TeXnical
696  
697  TeXNICALSOURCES = xysource.man COPYING.patch FONTCOPYING.patch
698  
699  TeXNICALAUX = \TRAILER \CATALOG \README \INSTALL \MAKE \COPYING.tex \FONT\COPYING.tex
700  
701  TeXnical: $(TeXNICAL)
702  
703  TeXnical = xysource.pdf xypdf.pdf
704  TeXnical: $(TeXNICAL)
705  
706  # Literate programming sources are generated with dvipdfmx specials.
707  xysource.pdf: xysource.man macros.tmp macros fonts $(SOURCES) $(TeXNICALAUX)
708  $(\cp)$ xysource.man xysource.tex
709  $(\rm)$ xysource.aux xysource.bbl xysource.ind xysource.toc
710  echo "\PassOptionsToPackage{dvipdfm}{color}\PassOptionsToPackage{dvipdfmx}{graphics}\PassOptionsToPackage{dvipdfm}{hyperref}" > xydoc.front
711  echo "\xyoption{pdf}" > xydoc.back
712  $(\LaTeX)$ xysource
713  $(\bib\LaTeX)$ xysource
714  $(\LaTeX)$ xysource
715  $(\makeindex)$ xysource
716  $(\LaTeX)$ xysource
717  $(\LaTeX)$ xysource
718  $(\dvipdfmx)$ xysource.dvi
719  
720  COPYING.tex: COPYING COPYING.patch FONTCOPYING.patch
721  patch COPYING -o COPYING.tex < COPYING.patch
722  
723  FONTCOPYING.tex: FONTCOPYING FONTCOPYING.patch
724  patch FONTCOPYING -o FONTCOPYING.tex < FONTCOPYING.patch
725  
726  # Macro index...
727  
728  macros.tmp: $(KERNELSOURCES) $(LATEXSTYLES) $(OPTIONSOURCES) $(METAFONTS) $(SOURCES)
729  -$(\rm)$ macros.tmp
730  for f in $(KERNELSOURCES) $(LATEXSTYLES) $(OPTIONSOURCES) $(METAFONTS); do $(GAWK) '
731  func err(a) { printf("%s,%d: %s\n",FILENAME,FNR,a) >>\"/dev/stderr\";\}
732  func pr(a,i) { \n733  printf("%I",a);\n734  prf يعرف(f){ t = substr($$0,i);\n735  if (match(t,/[\-]\|\[a-zA-Z@]*|\.\[\[^\|\]|{\[^{}]|\|{})*]/) != 1) \n736  err("Weird TeX token");\}
737  if (match(t,/[\-]\|\[a-zA-Z@]*|\.\[\[^\|\]|{\[^{}]|\|{})*]/) != 1) \n738  err("Weird TeX cs name token");\}
739  else pr(substr($$0,1,\"LENGTH\"),i);\}
740  if (match(t,/[\-]\|\[a-zA-Z@]*|\.\[\[^\|\]|{\[^{}]|\|{})*]/) != 1) \n741  err("Weird TeX cs name token");\]
742  else pr(substr($$0,1,\"LENGTH\"),i);\}
743  \"\"
744  \"\"
745  \"\"
746  \"\"
747  \"\"
APPENDIX D. DISTRIBUTION SUPPORT FILES

748 \newif{^[A-Za-z@]} { prtdef(7); next };
749 \xydef{^[A-Za-z@]} { prtdef(8); next };
750 \xylet{^[A-Za-z@]} { prtdef(9); next };
751 \global\let{^[A-Za-z@]} { prtdef(12); next };
752 \xywarnifdefined{^[A-Za-z@]} { prtdef(15); next };
753 \ifx\undefined{^[A-Za-z@]} { prtdef(17); next };
754 \xynew@{^[{}]*} { prtdef(index($$0,"}+1); next };
755 \xydefcsname@/ { prtcsdef(14); next };
756 \xyletcsnamecsname@/ { prtcsdef(20); next }
757 done | sed -e 's,\{,\\otherebgroup,' -e 's,\},\\otheregroup,' > macros.tmp
758 #
759 \printf("\macroentry{%s}{%s}{%d}
",FILENAME,FNR) };\#
760 # Finally the separate pdf option documentation.
761 # xypdf started as a LaTeX package so is generated from 'docstrip' sources.
762 xypdf.pdf: xypdf.dtx
763 $(LATEX) xypdf.dtx
764 $(LATEX) xypdf.dtx
765 $(DVIPDFMX) xypdf
766 @$(RM) xypdf.dvi
767 #
768 # Everything.
769 #
770 DOCUMENTS = $(GUIDE) $(REFER) $(TeXNICAL)
771 # HOME PAGE.
772 #
773 WWWOURCES = Xy-pic.html Xy-logo.doc Xy-logo.xy
774 #
775 Xy-logo.tex: Xy-logo.doc
776 Xy-logo.dvi: Xy-logo.tex Xy-logo.xy macros; $(LATEX) Xy-logo
777 Xy-logo.ps: Xy-logo.dvi; $(DVIPS) Xy-logo.dvi -oXy-logo.ps
778 #
779 PNMCRP = ./pnmrawtopcropwhite | pnmcrop
780 PGMTOPNG = pgmtoppm rgb:1/1/1 | pnmtopng -interlace -transparent rgb:1/1/1
781 #
782 Xy-logo.png: Xy-logo.ps pnmrawtopcropwhite
783 echo ' ' | \n784 gs -sDEVICE=pbmraw -r300 \n785 -sOutputFile='$($PNMCROP) >Xy-logo.pbm' \n786 -q -Xy-logo.ps
787 cat Xy-logo.pbm | $(PGMTOPNG) >Xy-logo.png
788 #
789 pnmrawtopcropwhite: pnmrawtopcropwhite.c
790 #
791 .PHONY: install.www
792 #
793 install.www: Xy-pic.html Xy-logo.png
794 scp Xy-pic.html Xy-logo.png krisrose@tug.org:public_html
795 scp Xy-pic.html krisrose,xy-pic@web.sourceforge.net:htdocs/index.html
796 scp Xy-logo.png krisrose,xy-pic@web.sourceforge.net:htdocs
797 #
798 # DISTRIBUTION TREE.
799 #
800 # Creates what goes on CTAN etc.
801 #
802 DIST = xy-$\{VERSION\}
803 DISTSRC = xy-$\{SHORTVERSION\}$src
804 DISTFONTDIR = $(DIST)/$(PK)fonts/$${mode}$$${dpi}
805 #
806 # Files.
807 # SOURCES = $(MACROSOURCES) $(FONTSOURCES) $(PSSOURCES) $(MANUALSOURCES) $(TeXNICALSOURCES) $(SUPPORT) $(WWW SOURCES)
808 #
809 # Targets:
810 #
811 .PHONY: dist
812 #
D.3. GENERATION

dist: $(NOTES) $(DOCUMENTS) $(SOURCES) $(MACROS) $(TEXFONTS) $(TYPE1FONTS) $(PFMFONTS) $(AFMFONTS) $(PSPRO) $(PSMAP) $(PDFENC) $(SUPPORT) $(RM) $(DIST)

$(TAR) -C .. `for f in $(SOURCES); do echo src/$$f; done` | $(GZIP) >$(DISTSRC).tar$(GZ)


$(MKDIR) $(DIST) && for f in $(NOTES); do $(INSTALL) $$f $(DIST)/$$f; done

$(MKDIR) $(DIST)/doc && for f in $(DOCUMENTS) $(DISTSRC).tar$(GZ); do $(INSTALL) $$f $(DIST)/doc/$$f; done

$(MKDIR) $(DIST)/texinputs && for f in $(MACROS) ; do $(INSTALL) $$f $(DIST)/texinputs/$$f; done

$(MKDIR) $(DIST)/texfonts && for f in $(TEXFONTS) ; do $(INSTALL) $$f $(DIST)/texfonts/$$f; done

$(MKDIR) $(DIST)/mfinputs && for f in $(METAFONTS) ; do $(INSTALL) $$f $(DIST)/mfinputs/$$f; done

$(MKDIR) $(DIST)/type1 && for f in $(TYPE1FONTS) ; do $(INSTALL) $$f $(DIST)/type1/$$f; done

$(MKDIR) $(DIST)/pfm && for f in $(PFMFONTS) ; do $(INSTALL) $$f $(DIST)/pfm/$$f; done

$(MKDIR) $(DIST)/afm && for f in $(AFMFONTS) ; do $(INSTALL) $$f $(DIST)/afm/$$f; done

$(MKDIR) $(DIST)/ps && for f in $(PSPRO) ; do $(INSTALL) $$f $(DIST)/ps/$$f; done

$(MKDIR) $(DIST)/map && for f in $(PSMAP) ; do $(INSTALL) $$f $(DIST)/map/$$f; done

$(MKDIR) $(DIST)/enc && for f in $(PDFENC) ; do $(INSTALL) $$f $(DIST)/enc/$$f; done

$(MKDIR) $(DIST)/support && for f in $(SUPPORT) ; do $(INSTALL) $$f $(DIST)/support/$$f; done

find $(DIST) -type f -exec $(MD5SUM) '{}' ';' | $(SIGN) >MANIFEST

$(CP) MANIFEST $(DIST)

zip -qr xy$(SHORTVERSION).zip $(DIST)

$(MD5SUM) xy$(SHORTVERSION).zip | $(SIGN) >xy$(SHORTVERSION).zip.sign

tar -c $(DIST) | $(GZIP) >xy$(SHORTVERSION).tar$(GZ)

$(MD5SUM) xy$(SHORTVERSION).tar$(GZ) | $(SIGN) >xy$(SHORTVERSION).tar$(GZ).sign

# Xy-pic the MOVIE :)

# .SUFFIXES: .texmovie .gif

# .texmovie.gif:
$(LATEX) '\documentclass[dvips]{movie}' \ 
'\usepackage[all]{xy}' \ 
'\begin{document}' \ 
'\input{$*.texmovie}' \ 
'\end{document}'

G[- -f texput.dvi ] && $(MV) texput.dvi $*.dvi && $(MV) texput.size $*.size

G[- -f movie.cls.dvi] && $(MV) movie.cls.dvi $*.dvi && $(MV) movie.cls.size $*.size

G[- -f movie.dvi ] && $(MV) movie.dvi $*.dvi && $(MV) movie.size $*.size

./dvitogif89a $*.dvi

# .texmovie.dvi:
$(LATEX) '\documentclass{article}' \ 
'\usepackage[movie,all]{xy}' \ 
'\begin{document}' \ 
'\maketitle' \ 
'\title{Storyboard for ``$*''}' \ 
'\input{$*.texmovie}' \ 
'\end{document}'

#

dragon.gif: dragon.texmovie movie.cls dvitogif89a
dragon.dvi: dragon.texmovie movie.cls
dragon.texmovie.movie.cls

#

sesam.gif: sesam.texmovie movie.cls dvitogif89a
sesam.dvi: sesam.texmovie movie.cls
nenesima.texmovie.movie.cls
nenesima.movie.cls

gasket.dvi: gasket.texmovie movie.cls dvitogif89a
gasket.texmovie.movie.cls
gasket.texmovie.movie.cls

gasket.texmovie.movie.cls

gasket.texmovie.movie.cls

gasket.texmovie.movie.cls

gasket.texmovie.movie.cls

# TAGS for Emacs.

#

TAGS: $(SOURCES) $(DOCUMENTS) $(SOURCES) $(MACROS)

-$(RM) TAGS

for f in `for ff in $(SOURCES); do echo src/$$f; done |sort -u`; do

do

$(GAWK) \nfunc err(a) { printf("%s,Xd: Xa\n",FILENAME,FNR,a) >>"/dev/stderr");
func prt(a,n) { printf("%s\177%d,\n",s,FNR,a);n};
func prtdef(i) { t = substr($$0,i);if (match(t,/[a-zA-Z0-9]*\{|\}\[\]/) != 1) \n
APPENDIX D. DISTRIBUTION SUPPORT FILES

err("Weird TeX token");
else {print(substr($$0,1,i+RLENGTH),i);}
func prtcedef(i) { t = substr($$0,i);
if (match(t,/{(\[^{}]|\{|\})*}|{(\[^{}]|\{|\})*{\[^{}]|\{|\})*\[^{}]|\{|\})*}) != 1) 
err("Weird TeX cs name token");
else {print(substr($$0,1,i+RLENGTH),i);}
func fin() { c += length($$0) + 1; next;}
BEGIN { c = 0;}
"\let[A-Za-z@]/ |
"\def[A-Za-z@]/ { prtdef(5); fin() ;}
"\edef[A-Za-z@]/ |
"\newif[A-Za-z@]/ |
"\xydef@[A-Za-z@]/ |
"\xyfont@[A-Za-z@]/ |
"\global\let[A-Za-z@]/ |
"\xywarnifdefined[A-Za-z@]/ |
"\ifx\undefined[A-Za-z@]/ |
"\xynew@[A-Za-z\]*]/ |
"\xydefcsname@[}/ |
"\xyletcsnamecsname@[}/ |
while (first = index(s, "???=")) {
  s = substr(s,first+4); i += first+4;
if (j = index(s, ")")) {print(substr($$0,1,i+j+1),i);}
else {print("Unmatched \"s\" "); fin() ;}
"\[A-Za-z@]/ |
"\{$$0,\"\"+1; fin() ;|
"\xydefcsname@/ |
"\xyletcsnamecsname@/ |
while (first = index(s, ")")) {
  s = substr(s,first+4); i += first+4;
if (j = index(s, ")")) {print(substr($$0,1,i+j+1),i);}
else {print("Unmatched \"s\" "); fin() ;}
"\$f >TAGS.tmp ;
wcc -c TAGS.tmp $(GAWK) "\"f\n%s,%s\n",""$$x",""$$y\""\"f\"">>TAGS;
cat TAGS.tmp >>TAGS ;
done
$(RM) TAGS.tmp

# LOG.
# Log: Makefile,v $ 
# Revision 3.74 2013/10/06 01:12:08 krisrose
# Backpatch for 3.8.9...
# Revision 3.73 2013/10/06 00:12:29 krisrose
# Backpatch for barr...
# Revision 3.72 2013/10/02 02:04:27 krisrose
# Release 3.8.9 including Barr's diagxy feature.
# Revision 3.71 2013/08/26 04:07:03 krisrose
# Version bump - next version will be 3.8.9.
# Revision 3.70 2012/05/24 00:30:38 krisrose
# Release 3.8.8 with xypdf fix by Norbert Preining.
# Revision 3.69 2011/08/28 22:19:06 krisrose
# Font fix (stroke to outline) by Daniel.
# Revision 3.68 2011/03/31 06:04:02 krisrose
# Introduced !B vector that extracts original baseline offset.
# Revision 3.67 2011/03/14 20:14:00 krisrose
# Preparing for release 3.8.6.
# Revision 3.66 2011/02/11 04:16:21 krisrose
# Preparing release 3.8.5 with Daniel's latest xypdf.
# Revision 3.65 2010/07/27 23:10:08 krisrose
# Include geometry as used...
# Revision 3.64 2010/07/27 09:49:34 krisrose
# Started xyling (and address updates).
D.3. GENERATION

# Revision 3.63 2010/07/26 11:21:02 krisrose
# 3.8.3 version oopse
# Limit rounding by mf2pt1.
# Revision 3.61 2010/07/06 22:43:56 krisrose
# Release 3.8.3.
# Revision 3.60 2010/06/23 19:40:17 krisrose
# Releaseing 3.8.2
# Revision 3.59 2010/06/19 01:59:06 krisrose
# The never ending cleanup continues...
# Revision 3.58 2010/06/17 22:39:25 krisrose
# First 3.8.2 with fixed PFB font map name and version...
# Revision 3.57 2010/06/10 18:45:49 krisrose
# Reference to GPL by URL.
# Revision 3.56 2010/06/10 15:08:54 krisrose
# Slight reorganization of release.
# Revision 3.55 2010/06/08 07:28:43 krisrose
# Identify .map file.
# Revision 3.54 2010/06/07 04:16:52 krisrose
# Fix forgotten enc files.
# Revision 3.53 2010/06/03 03:33:45 krisrose
# Avoid creating empty pdf release directory.
# Revision 3.52 2010/06/02 20:34:48 krisrose
# Enforce encoding on .a/pfm files.
# Revision 3.51 2010/05/21 18:17:59 krisrose
# Use explicit paper size – but manuals still not centered properly.
# Revision 3.50 2010/05/21 15:25:48 krisrose
# Release should not contain Windows executables so removed doc2tex.com.
# Revision 3.49 2010/05/17 23:29:21 krisrose
# Experiment: generate all the Type1 fonts with METAPOST.
# Revision 3.48 2010/05/13 23:26:05 krisrose
# TeXnical documentation not included in standard package.
# Revision 3.47 2010/05/09 22:11:05 krisrose
# Loose /usr/bin prefixes.
# Revision 3.46 2010/05/06 18:26:36 krisrose
# Reference cleanup.
# Revision 3.45 2010/05/06 17:46:29 krisrose
# Ross Moore’s e-mail address updated.
# Many obsolete files degraded to Historic.
# Revision 3.44 2010/05/06 04:12:56 krisrose
# Documentation fixes.
# Revision 3.43 2010/05/06 02:00:59 krisrose
# Hyperlinked index.
# Force dvipdfmx as pdftrue.
# Small adjustments to tips and automaton example.
# Proper bookmark special cases for PS and Xy commands.
# Revision 3.42 2010/05/05 04:13:55 krisrose
# Include signed mdSums of archives.
APPENDIX D. DISTRIBUTION SUPPORT FILES

# Revision 3.41 2010/05/04 22:48:29 krisrose
# Attempt to just generate PDF files.
# Revision 3.40 2010/05/04 08:23:00 krisrose
# Updating documentation to use dvipdfmx.
# Revision 3.39 2010/05/04 01:19:33 krisrose
# Make source distribution. No PK fons included.
# Revision 3.38 2010/05/03 18:06:23 krisrose
# Keep MANIFEST in src.
# Revision 3.37 2010/04/29 06:03:15 krisrose
# Add zip and tar.gz archive to dist.
# Revision 3.36 2010/04/29 02:12:59 krisrose
# Clean out obsolete LaTeX2HTML.
# Revision 3.35 2010/04/29 01:27:46 krisrose
# Distribution omits source (since it is not freely available on SourceForge).
# Revision 3.34 2010/04/29 01:25:38 krisrose
# Makefile included in sources. Consolidated a single INSTALL file.
# Revision 3.33 2010/04/28 23:48:08 krisrose
# make reallclean fix.
# Revision 3.32 2010/04/28 20:12:36 krisrose
# Include home page in release.
# Revision 3.31 2010/04/28 19:48:03 krisrose
# make reallclean and .cvsignore missed a few files.
# Revision 3.30 2010/04/28 07:14:30 krisrose
# New Xy-pic home page installed.
# Revision 3.29 2010/04/26 22:01:48 krisrose
# Documentation fixes (hyperref and other things).
# Revision 3.28 2010/04/26 05:56:56 krisrose
# Link fixes in progress...
# Revision 3.27 2010/04/26 03:47:22 krisrose
# All documentation typesets with pdf option.
# Revision 3.26 2010/04/26 01:45:23 krisrose
# First proper integration of xypdf into Xy-pic "make dist".
# Revision 3.25 2010/04/25 21:48:05 krisrose
# First proper integration of xypdf into Xy-pic "make dist".
# Revision 3.24 2010/04/22 16:03:50 krisrose
# Cleanup of obsolete files.
# Revision 3.23 2010/04/22 14:52:09 krisrose
# Documentation up to date. Install PDF encoding files and XYLU metric files.
# Revision 3.22 2010/04/22 09:11:26 krisrose
# Include MANIFEST with md5sums.
# Revision 3.21 2010/04/22 08:57:58 krisrose
# make dist works again!
# Revision 3.20 2010/04/22 08:10:35 krisrose
# Documentation can be generated in DVI, PS, and PDF (without xypdf)...
# Revision 3.19 2010/04/21 23:26:07 krisrose
# License patch fixup for TeXnical documentation.
# Revision 3.18 2010/04/21 18:41:41 krisrose
D.3. GENERATION

# Adjustments preparing for xypdf.dtx and friends!

Revision 3.17 2010/04/21 18:41:02 krisrose
# Adjustments preparing for xypdf.dtx and friends!

Revision 3.16 2010/04/20 20:36:43 krisrose
# Documentation updates.

Revision 3.15 2010/04/20 17:21:06 krisrose
# Experiments with xypdf.sty and mf2pt1.

Revision 3.14 2010/04/17 14:45:48 krisrose
# Generate and extract Type1 fonts.

Revision 3.13 2010/04/17 04:19:41 krisrose
# Integrated xylu tips by Jeremy Gibbons.

Revision 3.12 2010/04/16 06:58:06 krisrose
# Version fixed by hand.

Revision 3.11 2010/04/16 06:06:51 krisrose
# Preparing for a new release...

Revision 3.10 2010/04/13 09:18:54 krisrose
# Generating dummy xypdftex.tex...

Revision 3.9 2010/04/13 09:17:00 krisrose
# No more RCS. LaTeX2HTML use of html.sty broken.

Revision 3.8 2010/04/13 08:10:29 krisrose
# Up to date with Kris' development directory.

Revision 3.7 1999/02/16 15:12:50 krisrose
# Interim release (Y&Y fonts now free).

Revision 3.6 1998/03/06 01:28:05 krisrose
# Releasing (with Y&Y fonts).

Revision 3.5 1997/05/28 13:05:01 krisrose
# Fixed missing breaks bug.

Revision 3.4 1997/05/18 03:04:44 krisrose
# Essential bugfixes.

Revision 3.3 1996/12/19 17:05:13 krisrose
# And movies now included!

Revision 3.2 1996/12/19 17:05:13 krisrose
# Maintenance release!

Revision 3.1 1995/09/05 08:31:32 krisrose
# Bug fix release.

Revision 3.0 1995/09/05 08:31:32 krisrose
# Releasing!

Revision 2.13 1995/07/04 15:10:01 krisrose
# Ready to release v3?

Revision 2.12 1994/10/25 11:48:25 krisrose
# Interim release just before v3 [works with AMS-LaTeX 1.2]...
APPENDIX D. DISTRIBUTION SUPPORT FILES

# Revision 2.10 1994/06/15 13:35:19 kris
# Second 3beta release [bug fixes].

# Revision 2.9 1994/06/09 14:53:07 kris
# Release 3beta.

# Revision 2.8 1994/04/08 04:30:00 kris
# Second (bug fix) 3alpha release.

# Revision 2.7 1994/03/08 02:06:01 kris
# Release 3alpha.

# Revision 2.6.9.1 1994/03/07 04:22:46 kris
# Last internal 3alpha and pre-2.7 release.

# MAJOR REORGANISATION for version 2.7...

# Revision 2.6 1992/06/24 01:23:34 kris
# Ready to release v.2.6.

# Revision 2.5 1992/02/24 03:30:54 kris
# Default magsteps now just 0, 0.5, 1, and 2.

# Revision 2.3 1992/01/13 02:12:28 kris
# Fixed installation instructions and other details.

# Revision 2.1 1992/01/02 14:54:07 kris
# Release version.

# Revision 1.6 1991/12/17 04:51:16 kris
# Version distributed with 'final draft' on Usenet.

# Tell Emacs that this is a Makefile and how it is formatted:

# Local Variables:
# mode:makefile
# fill-prefix:"
# fill-column:77

# End:
Bibliography


Index

!, 24
&, 356
', 308
(), 24
(0), 24
(0,0), 575
(1), 24
*, 24, 308, 319, 325, 356, 370, 518, 584
**, 24, 318, 370
*stylechar@P@, xynecula.doc:105
*stylechar@|@, xyline.doc:113
+, 24
,, 24, 314
-, 24, 308, 311
,, 24
.xyd, 70
/, 308, 312, 317, 326
//, 24
::, 24, 47
\:, xybarr.doc:100, xybarr.doc:99
::, 24
;:, 24
<, 24, 308, 312, 317
<> , 24
<>(.5), 308, 311
=, 24, 308, 309, 312, 318, 319
>, 24, 308, 312, 317
?, 24, 318
@, 24, 308, 368, 472
@!, 368, 472
@!!0, 368
@!1=, 368
@!C, 368
@!R, 368
@(.42, 326, 472
@), 42
@*, 370, 472
@+, 42
@-, 42
@/, 326, 472
@1, 373
@<, 327, 472
@=, 42
@?, 328, 472
@0, 42
@C, 368
@H, 368
@L, 368
@M, 368
@R, 368
@W, 368
@', 472
@i, 42
\[, xy.doc:2544
[.], 55
[=, 47, 60, 244, 253
[P; , 290
[], 55, 57
[.c], 56
[.d], 55
[.dvips], 11
[.l], 55
[.o], 55
[.r], 55
[.textures], 11
[.u], 55
\\, 356
\., 308, 314, 315, 318, 472
\., 308, 314, 315, 318, 472
\', 308, 314
\s, 462
{.styletoks@}, xy.doc:3904
{.xypatread@}, xytile.doc:263
l, 308, 472
l<, 584
|DOCMODE, xy.doc:69
\-, 308
0, 21, 24, 472
1, 472
2, 472
3, 472
10, 220

643
INDEX

\almostz@, xy.doc:660
\alphanum, xypoly.doc:427
\AMS-LaTeX, 7, 68
\AMS-LaTeX, 7
\api, 584
\appendtoholder@, xynod.doc:1091
\appendtoholder@, xynod.doc:1095
\appendtoholder@@, xynod.doc:1099
\appendtoholder@, xynod.doc:1114
\applyFIFOstyle@, xy.doc:4172
\applyLIFOstyle@, xy.doc:4200
\ar, xyarrown.doc:1224
\ar, 60, 312, 319, 320, 462
\ar@, xyarrown.doc:1246
\aranchor, xyarrown.doc:1268
\arc, xyarrown.doc:1324
\ar@curve, xyarrown.doc:1734
\ar@curve@, xyarrown.doc:1745
\ar@curveinout, xyarrown.doc:1710
\ar@curveinout@, xyarrown.doc:1708
\ar@curveslash, xyarrown.doc:1671
\ar@error, xyarrown.doc:1322
\ar@form, xyarrown.doc:1285
\ar@i, xyarrown.doc:1344
\ar@ii, xyarrown.doc:1352
\ar@iii, xyarrown.doc:1357
\ar@iv, xyarrown.doc:1363
\ar@modifiers, xyarrown.doc:1779
\ar@PATH, xyarrown.doc:1404
\ar@slashing, xyarrown.doc:1673
\ar@slide, xyarrown.doc:1794
\ar@style, xyarrown.doc:1330
\ar@upsidedown, xyarrown.doc:1805
\ar@x, xyarrown.doc:1375
\arafterPOS@, xyarrown.doc:1406
\arafterPOS@@, xyarrown.doc:1209
\arc@, xyarrown.doc:120
\arc@i, xyarrown.doc:122
\arc@bisector, xyarrown.doc:1012
\arc@Center@, xyarrown.doc:1019
\arc@Center@i, xyarrown.doc:1025
\arc@magic, xyarrown.doc:889
\arc@magic@, xyarrown.doc:888
\arc@component@, xyarrown.doc:1422
\arc@component@@, xyarrown.doc:1213
\arc@component@i, xyarrown.doc:1436
\arc@component@ii, xyarrown.doc:1438
\arc@component@x, xyarrown.doc:1466
\arc@component@type@, xyarrown.doc:1214
\arc@conn@, xyarrown.doc:1455
\arcScale@, xyarrown.doc:251
\arcScale@dim, xyarrown.doc:261
\arcTurn@, xyarrown.doc:669
\arcTurn@@, xyarrown.doc:963
\arcTurn@@@, xyarrown.doc:969
\arexit@, xyarrown.doc:1211
\arhead@, xyarrown.doc:1206
\arinit@, xyarrown.doc:1210
\arlabels@, xyarrown.doc:1208
\armodifiers@, xyarrown.doc:1207
array, 376
\arraystretch, xy.doc:6162
\arrow, xybarr.doc:1124, xynod.doc:597
\arrowhead, 324
\arrowstem, 324
\arrowtail, 324
\arrowobject, xy2cell.doc:1206
\Arrowobject@, xy2cell.doc:1207
\arrowp, xybarr.doc:1125
\arrowpp, xybarr.doc:1126
\Arrowtok@, xy2cell.doc:1446
\arSAFE, xyarrown.doc:1226
\arsavePATHAfterPOS@, xyarrown.doc:1410
\arstem@, xyarrown.doc:1205
\arstemprefix@, xyarrown.doc:1203
\artail@, xyarrown.doc:1204
\artip@, xyarrown.doc:1440
\arvariant@, xyarrown.doc:1202
\astop, xynod.doc:147
\astop, 587
\AtBeginDvi, xyidioms.doc:174
\aTip, xynod.doc:1356
\atip, xynod.doc:1351
\aTip, 588
\atip, 588
\atip@, xy.doc:5631
\atip@, xy.doc:5640
\atip@cm, xytips.doc:131
\atip@xy, xytips.doc:47
\AtTriangle, xybarr.doc:457
\AtTrianglep, xybarr.doc:458
\AtTrianglepair, xybarr.doc:1578
\AtTrianglepairp, xybarr.doc:579
\AtTrianglepairpp, xybarr.doc:581
\AtTrianglepairppp, xybarr.doc:583
\AtTrianglepairpppp, xybarr.doc:490
\AtTrianglep, xybarr.doc:459
\AtTrianglepp, xybarr.doc:461
\AtTriangleppp, xybarr.doc:353
\aturn, xyv2.doc:367
\aturn , 588
\aturn@, xy.doc:3779
\aturn@, xy.doc:5771
\AveryUNLIKELYc@ntr@1SEQUENCE@@, xyidioms.doc

B, 8
\B@, xyidioms.doc:85
\backwardSectors@, xyarc.doc:1254
banner, 14
\basefromthebase@, xy.doc:1885
\baseOrient@, xyarc.doc:324
\bblacked@, xyframe.doc:467
\belowDirection@, xy.doc:7293
\bendPLACE@, xycurve.doc:1377
\bfig, xybarr.doc:130
\bhook, xyv2.doc:365
\bhook , 588
\bhook@, xy.doc:5748
\bhook@, xy.doc:5749
\blackened@, xyframe.doc:459
\blackened@, xyframe.doc:472
\blacktriangle, xypoly.doc:586
(b)ody of the option), 93
\bbox@, xyidioms.doc:113
\bpt, 584
\bprevPLACE@, xycurve.doc:1378
\breakend, xycurve.doc:2932
\breakend@start, xycurve.doc:2868
\breakendcontinue@, xycurve.doc:2887
\breakendcontinue@i, xycurve.doc:2894
\breakendfound@, xycurve.doc:2922
\breakstartcontinue@, xycurve.doc:2801
\breakstartcontinue@i, xycurve.doc:2807
\breakstartfound@, xycurve.doc:2820
\bssegment@i, xycurve.doc:4193
\bssegment@ii@i, xycurve.doc:4226
\bssegment@ii@iii, xycurve.doc:4177
\bssegment@iv, xycurve.doc:4203
\bssegment@y@iv, xycurve.doc:4236
\bssegment@i, xycurve.doc:4259
\bssegment@i@iv, xycurve.doc:4275
\bssegment@z, xycurve.doc:4297
\bssegment@z@i, xycurve.doc:4313
\bsplinescancel@, xycurve.doc:3451
\bsplineconnect@, xycurve.doc:4011
\bsplined@, xycurve.doc:4109
\bsplined@, xycurve.doc:4401
\bsplineiiiconnect@, xycurve.doc:3959
\bsplineivconnect@, xycurve.doc:3967
\bsplinesegment@, xycurve.doc:4327
\bsplinesegment@iii, xycurve.doc:4350
\bsplinesegment@iv, xycurve.doc:4376
\bstartPLACE@, xycurve.doc:1376
\bstop, xyv2.doc:348
\bstop , 587
\bthisPLACE@, xycurve.doc:1379
\bTip, xyv2.doc:357
\btip, xyv2.doc:352
\bTip , 588
\btip , 588
\btip@, xy.doc:5632
\btip@, xy.doc:5641
\btip@cm, xytips.doc:132
\btip@xy, xytips.doc:48
\btriangle, xybarr.doc:1449
\btrianglep, xybarr.doc:450
\btrianglepp, xybarr.doc:451
\btriangleppp, xybarr.doc:453
\btrianglepppp, xybarr.doc:337
\bturn, xyv2.doc:368
\bturn , 588
\bturn@, xy.doc:5776
\bturn@, xy.doc:5777
\buildcircle@, xycurve.doc:4603

C, 24, 42
c, 22
c, 24
\c@poly@count, xynecula.doc:106
\Calong@, xy.doc:7941
\cancelPSdict@, xyps-ps.doc:488
\car, xybarr.doc:133
cartesian coordinate system, 22
category code, 4
\Cb@reak@, xy.doc:7943
\CC@ntercept@, xycurve.doc:1663
CD, 24
\cD, xyv2.doc:197
\cD , 585
\centerobject@, xy.doc:3741
\cf@romcontrols@, xyarc.doc:507
\cf@romid@, xy.doc:2905
\cf@romp@, xy.doc:1635
\cf@roms@, xy.doc:2690
\cf@romthe@, xy.doc:1666
\cf@romthep@, xy.doc:1670
\cH, xyv2.doc:198
\cH , 585
\change@oxy@, xy.doc:861
INDEX

\docm@, xy.doc:580
\docm@i, xy.doc:81
\docm@ii, xy.doc:82
\docm@iii, xy.doc:84
\docm@iv, xy.doc:89

DOCMODE, 1, 7

\doctype, xy.doc:11
\doEllipseSectors@, xyarc.doc:1283
\doSquares, xybarr.doc:714
\dosquares, xybarr.doc:680
\doii, xy.doc:6853
\doMergeBoth@, xyarc.doc:488
\doMergeBothArc@, xyarc.doc:482
\doMergeEnd@@, xyarc.doc:457
\doMergeEndArc@@, xyarc.doc:499
\doMergeEndArc@@@, xyarc.doc:451
\doMergeStart@@, xyarc.doc:473
\doMergeStartArc@@, xyarc.doc:467
\domorphism, xybarr.doc:149
\dontleave@, xy.doc:774
\doPSspecialRotate@@, xyps-r.doc:142
\doShow, xy.doc:752
\doskewEllipse@, xyps.doc:1332
\dosolidcircle@, xycurve.doc:4534
\dosolidcircle@@, xycurve.doc:4542
\doSpecialRotate@, xyrotate.doc:1390
\dosplineplotpt@, xycurve.doc:3428
\doSplineSegment@, xycurve.doc:4847
\doSplineStraight@, xycurve.doc:4843
\doStraightArc@@, xyarc.doc:503
\dotcorner@, xycorner.doc:4709
\dotcorner@@, xyframe.doc:387
\dotcorner@@@, xycorner.doc:4707
\dotframed, xyv2.doc:502
\dotframed , 590
\Dotted, xyv2.doc:409
\dotted, xyv2.doc:408
\Dotted , 587
\dotted , 587
\dottedcircle@, xycurve.doc:4658
\dottedSpread@, xy.doc:5376
\dottedSpread@@, xy.doc:5384
\dottedwith, xyv2.doc:312
\dottedwith , 587
\double@, xy.doc:5476
\doublecircle@, xycorner.doc:4646
\dovSquares, xybarr.doc:785
\dovsquares, xybarr.doc:749
\Dp, xyv2.doc:220

\Dp , 586
\dparenthesized, xyframe.doc:819
\draw@polyframe, xynecula.doc:145
\draw@polygon, xynecula.doc:212
\drDirection@, xy.doc:7990
\langle driver\rangle, 94
\driverextensioncomplain@, xy.doc:6931
\Drop, xyv2.doc:270
\drop, xy.doc:4541
\Drop , 586
\drop , 66
\Drop@, xy.doc:2110
\Drop@@, xy.doc:3238
\drop@Twocell, xy2cell.doc:1408
\dropentry@, xymatrix.doc:657
\droplabel@, xyarrow.doc:942
\Droprule@, xy.doc:1513
\droptwocelldrop@, xy2cell.doc:375
\droptwocelllabel@, xy2cell.doc:366
\droptwocelllabel@@, xy2cell.doc:367
\droptwocelltext@, xy2cell.doc:373
\FSIZE , 585

\Dtriangle, xybarr.doc:484
\Dtriangle, xybarr.doc:441
\Dtriangle, xybarr.doc:482
\Dtriangle@, xybarr.doc:442
\Dtriangle@, xybarr.doc:605
\Dtriangleairpair, xybarr.doc:606
\Dtriangleairpairr, xybarr.doc:608
\Dtriangleairpairppp, xybarr.doc:610
\Dtriangleairpairpppp, xybarr.doc:556
\Dtriangle@, xybarr.doc:483
\Dtriangle@, xybarr.doc:443
\Dtriangle@, xybarr.doc:485
\DDtriangle@, xybarr.doc:445
\Dtriangle@, xybarr.doc:407
\Dtriangle@, xybarr.doc:321

\dumpPSdict, xyps.doc:649
\dumpPSdict {<filename>}, 268
\dumpPSdict@@, xyps-ps.doc:187
\dumpsizefile, movie.cls:283
\dvdirv@, xydvidrv.doc:41
\dvdirvEmTeX@, xydvidrv.doc:58
\dvdirvEmTeX@@, xydvidrv.doc:97
\dvdirvLine@, xydvidrv.doc:66
\dvdirvLine@@, xydvidrv.doc:65
\dvmessage@, xy.doc:6940
\dvips@, xys.doc:269
\dvips@@, xydvips.doc:41
\ellipseScale@dim, xyarc.doc:276
\ellipsesolidframed@, xyframe.doc:654
\ellipticARC@, xyarc.doc:921
\ellsave@, xyarc.doc:1157
\Emalternative@, xymatrix.doc:164
\emptyspline@, xycurve.doc:3364
\endcrv@, xycurve.doc:741
\endcrv@@, xycurve.doc:275
\endsname@, 4
\endcurve, xycurve.doc:1981
\enddiagram@, 590
\endellipse@, xyarc.doc:64
\endgraph, xygraph.doc:170
\endpatread@, xytile.doc:360
\endxy, xy.doc:1143
\endxy@, 20–22, 66
\endxycurve@, xycurve.doc:1978
\endXYdict@, xyps-ps.doc:469
\enter@, xymatrix.doc:768
\entryalignment used prior to version 3.8, 371
\entryheight@, 368
\entrymargin@, 368
\entrymodifiers@, 370
\entrywidth@, 368
\entry@, xymatrix.doc:605
\entry@, xymatrix.doc:612
\entry@i, xymatrix.doc:619
\entry@norm, xymatrix.doc:642
\entry@star, xymatrix.doc:623
\entry@star@i, xymatrix.doc:626
\entry@star@ii, xymatrix.doc:633
\entrybox, xymatrix.doc:1221
\entrybox@, 372
\entrymodifiers@, xymatrix.doc:1208
\entrymodifiers@, 370
\entrymodifiers@, xymatrix.doc:1219
\epi, xybarr.doc:874
\epileft, xybarr.doc:877
\Error@, 15
\{"escape\}, 474
\eu@, 220
\everyentry@, xymatrix.doc:1291
\everyentry@, 373
\everyentry@, xymatrix.doc:1294
\everyxy@, xymatrix.doc:128
\everyxy@, xymatrix.doc:1131
\execution@, 20
\expandbefore@, xynecula.doc:51
\extension@, 94
\extractparameter@, xymovie.doc:62
\fileheader@, 1
\filepattern@, xytile.doc:309
\filled@, xyframe.doc:943
\filled@Circle@, xymovie.doc:989
\filled@Ellipse@, xyframe.doc:990
\filled@Oval@, xyframe.doc:988
\filled@Rectangle@, xyframe.doc:987
\filled@Region@, xyframe.doc:979
\filledCircle@, xyframe.doc:969
\filledCircle@PS, xyps-f.doc:194
\filledCircle@TPIC@@, xyp-f.doc:241
\filledEllipse@, xyframe.doc:970
\filledEllipse@PS, xyps-f.doc:196
\filledEllipse@TPIC@@, xyp-f.doc:243
\filledOval@, xyframe.doc:971
\filledOval@PS, xyps-f.doc:198
\filledOval@TPIC@@, xyp-f.doc:244
\filledRectangle@, xyframe.doc:968
\filledRectangle@PS, xyps-f.doc:192
\filledRectangle@TPIC@@, xyp-f.doc:237
\findbreakwarning@, xycurve.doc:2796
\findIsegment@, xycurve.doc:1205
\findIsegment@i, xycurve.doc:1536
\findIsegment@ii, xycurve.doc:1540
\findIsegment@x, xycurve.doc:1544
\findIsegment@y, xycurve.doc:1549
\finishq@, xymatrix.doc:195
\firstbspline@, xycurve.doc:4023
\firstPS@, xyp-ps.doc:136
\firstPS@@, xyps-ps.doc:138
\firstsplinepoint@, xycurve.doc:2404
\fixedgrid@, 368
\fixedDirection@, xy.doc:7123
\fixedRadiusArc@, xyarc.doc:227
\fixedRadiusArc@i, xyarc.doc:231
\flagwarn@, xycurve.doc:716
\fonts@, 15
\fontscale@, xybarr.doc:115
\forceload@XYdict, xyps-ps.doc:612
\formatdependencies@, 7
\formats@, 7
INDEX

\lparenthesized, xyframe.doc:1844
\ltxdiagram@, xyv2.doc:1545
LU, 24
lu, 220
\m@caption, movie.cls:142
\m@height, xymovie.doc:59
\m@keys, movie.cls:132, movie.cls:93
\m@nicetotal, movie.cls:224
\m@scene, xymovie.doc:98
\m@width, xymovie.doc:60
\MacPatterns, xytile.doc:59
\makeatletter, 4
\makeatother, 4
\MakeOutlines, xy.doc:14879
\MakeOutlines, 70
math mode, 21
\mathbf, xy2cell.doc:381
\mathsurround, 356
\matrix, 356
matrix, 356
matrix orientation, 368
matrix spacing, 368
\matrixsize@, xymatrix.doc:216
maxcol0, xymatrix.doc:161
maxcolrow0, xymatrix.doc:162
maxrow0, xymatrix.doc:160
maxTPICpoints, xytpic.doc:180
\maybeunraise@, xy.doc:1156
\merge, xy2.doc:266
\merge, 586
\mergecropextents@, xyweb.doc:226
messages, 14
\middlebspline@, xycurve.doc:401
\middlespline@, xycurve.doc:1070
\mkHmax@, xymatrix.doc:525
\mkHrow@, xymatrix.doc:523
\mkHWdefaults@, xymatrix.doc:529
\mkHWmax@, xymatrix.doc:527
\mkWcol@, xymatrix.doc:524
\mkWmax@, xymatrix.doc:526
\moddefs@, xy2cell.doc:1021
(modifier), 47, 289
\modifystyle@, xy.doc:4124
\modmap@, xy2cell.doc:1009
\modmapobject, xy2cell.doc:1053
\modmapobject@, xy2cell.doc:1054
\modPSbox@, xyps-ps.doc:441
\modPSconnect@, xyps-ps.doc:426
\modPSdrop@, xyps-ps.doc:416
\modXYstyle@, xy.doc:1963
\modXYstyle@, xy.doc:1969
\modXYtransform@, xytile.doc:204
\mon, xybarr.doc:873
\monleft, xybarr.doc:576
\morepatread@, xytile.doc:301
\morphism, xybarr.doc:135, xyv2.doc:611
\morphism, 591
\morphism@, xybarr.doc:136
\morphismpp, xybarr.doc:137
\morphismppp, xybarr.doc:139
\morphismpppp, xybarr.doc:143
(move), 474
\MovieHop, xymove.doc:241
\MovieHop, xymovie.doc:265
\MovieHop, xymovie.doc:246
\MovieHop, xymovie.doc:270
\MovieHop, xymovie.doc:254
movie, 259
\MovieSetup, movie.cls:137, xymovie.doc:566
\MovieSetup, 259
\MultipleDrivers, xy.doc:6710
\MultipleDrivers, 94
\myPOS, xy2cell.doc:1285
\mypostamble, xypdf.ins:150
\newdir, xy.doc:5805
\newdir, 84, 324
\newdir, xy.doc:5807
\newdriver, xy.doc:16819
\newdriver, 96
\newgraphescape, xymovie.doc:196
\newgraphescape, 382
\newgraphescape@, xymovie.doc:605
\newlinechar, 15
\newxyColor, xcolor.doc:252
\newxyColor, xcolor.doc:251
\newxyColor, 242
\newxyColor@, xcolor.doc:270
\newxyPattern, xytile.doc:116
\newxypattern, xytile.doc:113
\newxypattern, 250
\newxypattern@, xytile.doc:1122
\newxypattern@, xytile.doc:114
\newxyPSshape, xyps-ps.doc:1682
\newxyshape, xyps-ps.doc:660
\newxystyle, xy.doc:4098
next, movie.cls:1380, movie.cls:29, movie.cls:30,
xy.doc:418, xy.doc:6847, xy.doc:6882, xy.sty:45,
xybarr.doc:151, xybarr.doc:233, xybarr.doc:236,
INDEX

xybarr.doc:260, xybarr.doc:289, xygraph.doc:495
xybarr.doc:293, xybarr.doc:306, xybarr.doc:810
xybarr.doc:310, xybarr.doc:323, xybarr.doc:812
xybarr.doc:327, xybarr.doc:339, xybarr.doc:817
xybarr.doc:343, xybarr.doc:355, xybarr.doc:817
noDictmessage@, xyps-PS.doc:491
NoDVIPScolor@, xydvips.doc:86
NoDVIPScolor@, xydvips.doc:101
NoDVITOPScolor@, yxdvips.doc:208
NoEMlinewidth, yxemtex.doc:211
NoEMspecials, yxemtex.doc:69
Noinsert@, xy.doc:6013
Nointerceptwarning@, xycurve.doc:1711
NoisyDiagrams, xyv2.doc:562
NoisyDiagrams, 591
NoisyEMTeX, yxemtex.doc:245
NoisyMatrices, xymatrix.doc:220
NoisyPIC, yxypic.doc:129
None, xyv2.doc:298
None, 587
NooldOzTeXColor@, xy17oztex.doc:91
NooldTexturesColor@, xy16textures.doc:85
NoOutlines, xy.doc:4886
NoOutlines, 70
NoOzTeXColor@, xyoztex.doc:96
NoPScolor, xyps-c.doc:74
NoPSframes, xyps-f.doc:74
NoPSlines, xyps-l.doc:26
NoPSrotate, xyps-r.doc:89
NoPSspecials, xy.doc:311
NoPSspecials, 263
NoPStiles, xyps-t.doc:260
NoResizing, yxrotate.doc:119
NoRules, xy.doc:5224
NoRules, 75
notempty, yxidioms.doc:119
NoTexturesColor@, xytextures.doc:94
notip, xyv2.doc:344
notip, 587
NoTips, xytips.doc:193
NoTips, 221
NoTPICframes, xyp-f.doc:162
NoTPICspecials, xypic.doc:62
notrelaxorelse@, xy.doc:1434
NoXDVIcolor@, xyxdvi.doc:103
NoXDVIColordvi@, xyxdvi.doc:118
inter@, xy.doc:771
nudgepos@, xy2cell.doc:827
nudgepos@, xy2cell.doc:844
\preentry@, xymatrix.doc:363
\preentry@@, xymatrix.doc:382
\preentry@@@, xymatrix.doc:422
\preentry@@@i, xymatrix.doc:427
\preentry@@@norm, xymatrix.doc:463
\preentry@@@star@i, xymatrix.doc:487
\preentry@@@star@ii, xymatrix.doc:490
\preentry@@@x, xymatrix.doc:498
\preShape@, xyrotate.doc:192
\preStyle@, xy.doc:3990
\prevEdge@, xy.doc:3134
\prevEdgefromtheEdge@, xyframe.doc:105
\prevhowclose@, xycurve.doc:1747
\prevXypreStyle@, xy.doc:4075
\prevXypostStyle@, xy.doc:4074
\prexyescape@, xyps-s.doc:172
\prexyline@, xyline.doc:593
\prexyline@@, xyline.doc:589
\prexylinecap@, xyline.doc:315
\prexylinecap@@, xyline.doc:311
\prexylinejoin@, xyline.doc:340
\prexylinejoin@@, xyline.doc:336
\prexylinemiter@, xyline.doc:365
\prexylinemiter@@, xyline.doc:361
\preXYstyle@, xy.doc:4992
\preXYtransform@, xyps-r.doc:112, xyrotate.doc:118
\preXYtransform@@, xyrotate.doc:187
privacy, 3
\processArcDirections@, xyarc.doc:1149
\processArcDirections@@, xyarc.doc:1159
\processdirections@, xyarc.doc:1145
\PSdict@@, xyps-ps.doc:62
\PShead@, xynecula.doc:108
\PSignore@, xyps-ps.doc:446
\PSinclude@@, xyps-ps.doc:65
\PSincrease@, xyps.doc:499
\PSmacro@, xyps-ps.doc:452
\PSmacro@@, xyps-ps.doc:61
\PSmessage@, xyps-ps.doc:445
\PSraw@, xyps-ps.doc:66
\PSread@, xyps-ps.doc:64
\PSspecial@@, xyps-ps.doc:63
\PSspecialdict@@, xyps-ps.doc:68
\PSspecials@, xyps.doc:447
\PSspecials@@, xyps.doc:452
\PSstracing, xyps-ps.doc:449
\ptriangle, xybarr.doc:425
\ptriangle@, xybarr.doc:426
\ptrianglepp, xybarr.doc:437
\ptriangleppp, xybarr.doc:439
\pushback, xybarr.doc:618
\pure@crv, xycurve.doc:276
\q, xypoly.doc:508
\q, 47
\qartPi@, xypic.doc:347
\qc, xyv2.doc:447
\qc, 589
\qcount@, xymatrix.doc:174
\qcount@@, xymatrix.doc:175
\qspline, xycurve.doc:4772
\qtriangle, xybarr.doc:443
\qtriangle@, xybarr.doc:434
\qtrianglepp, xybarr.doc:445
\qtriangleppp, xybarr.doc:437
\rbraced, xymatrix.doc:304
\quadarc@, xyarc.doc:927
queue, 18
\queue@, xymatrix.doc:171
\queue@@, xymatrix.doc:172
\quotemash, xy.doc:6103, xy.doc:6202, xy.doc:6222
\quotient@, xy.doc:813
\quotient@@, xy.doc:825
\quotepTK@, xy.doc:702
\quotepTK@@, xy.doc:809
R, 8, 21
R, 24, 42
\R@, xyidioms.doc:86
\R@c, xy.doc:616
\R@p, xy.doc:624
\Rc, 22
\R@p, 22
\(radius\), 86
\ratchet, xybarr.doc:89
\rbraced, xyframe.doc:779
\Rc, xyv2.doc:215
\Rc, 586
RD, 24
\PaDirection@, xy.doc:7095
\readoutline@, xy.doc:4899
\readoutline@@, xy.doc:4877
\readsplineparams@, xycurve.doc:2192
\readxycurve, xycurve.doc:1921
\readxycurve@, xycurve.doc:1920
\realinstallxyps@, xyps-ps.doc:177
\ReCompileAllDiagrams, xyv2.doc:797
INDEX

\sinPIon, xypoly.doc:665
\sinThreePIon, xypoly.doc:682
\sinTwoPIon, xypoly.doc:673
(size), 47
\skewCircle@,xyarc.doc:1322
\skewEllB@,xyarc.doc:1376
\skewEllipse@,xyarc.doc:1317
\skipspecials@, xy.doc:72, xy.doc:73
\sleave@, xy.doc:2755
(slide), 24
\SloppyCurves, xycurve.doc:2289
\sloppyendcrv@, xycurve.doc:728
\sm@accumext, xysmart.doc:420
\sm@advancecount@, xysmart.doc:261
\sm@computeext, xysmart.doc:386
\sm@conn, xysmart.doc:119
\sm@connect@, xysmart.doc:1441
\sm@connect0@, xysmart.doc:503
\sm@drawseg, xysmart.doc:456
\sm@drawseglst, xysmart.doc:448
\sm@drop@, xysmart.doc:506
\sm@maxcost, xysmart.doc:154
\sm@maxnull, xysmart.doc:155
\sm@roundcount@, xysmart.doc:278
\sm@shovext, xysmart.doc:355
\sm@straight, xysmart.doc:485
\sm@stri, xysmart.doc:499
\sm@trycrlist, xysmart.doc:295
\sm@trycrlis@0i, xysmart.doc:361
\sm@trycircles, xysmart.doc:215
\sm@tryclear, xysmart.doc:367
\smapp@, xy.doc:2770
\smappp@, xy.doc:2778
\smapxy@, xy.doc:2776
\smapxy@0i, xy.doc:2780
\smconn, xysmart.doc:513
\Solid, xyv2.doc:301
\solid, xyv2.doc:300
\Solid, 587
\solid, 587
\solid@, xy.doc:3176
\solidcircle@, xycurve.doc:14541
\solidcorner@, xyframe.doc:360
\solidhrule@, xy.doc:5272
\solidhrule@pre, xy.doc:5275
\solidhrule@typeset, xy.doc:5294
\solidhrule@typeset0, xy.doc:5301
\solidpoint@, xy.doc:5920
\solidSpread@, xy.doc:5196
\solidvrule@, xy.doc:5233
\solidvrule@pre, xy.doc:5236
\solidvrule@typeset, xy.doc:5255
\solidvrule@typeset0, xy.doc:5262
\space@, xydioms.doc:117
spacing, 368
\spline@find, xycurve.doc:3140
\spline@knot, xycurve.doc:3177
\spline@end@, xycurve.doc:3070
\spline@end00, xycurve.doc:3073
\spline@find, xycurve.doc:3114
\splineadvance@00, xycurve.doc:2354
\splinealong@, xycurve.doc:1237
\splinealong@0, xycurve.doc:1264
\splineBRec0, xycurve.doc:2761
\splinebox@, xycurve.doc:2135
\splinebreak@, xycurve.doc:1385
\splinebreakcanc0, xycurve.doc:3450
\splineBreakEndtest@, xycurve.doc:2439
\splineBreakStarttest@, xycurve.doc:2436
\splineBrec0, xycurve.doc:2735
\splineBSrec0, xycurve.doc:2745
\splinecancel0, xycurve.doc:3442
\splineclosest@, xycurve.doc:1594
\splineconn0, xycurve.doc:2330
\splineConnect@, xycurve.doc:4835
\splineconnect@0, xycurve.doc:1063
\splinecontinue, xycurve.doc:2490
\splinecontinue@0, xycurve.doc:2384
\splinecontinue@00, xycurve.doc:2387
\spline@0@0, xycurve.doc:2416
\splineDadvance@, xycurve.doc:3058
\splineDadvance@00, xycurve.doc:3060
\splinedashed@0, xycurve.doc:3005
\splinebidotted@, xycurve.doc:3312
\splinedefaulttol0, xycurve.doc:2282
\splinedepth@0, xycurve.doc:2146
\splinedotted@0, xycurve.doc:3309
\splinedouble0ed@, xycurve.doc:3296
\splineDrop@0, xycurve.doc:4831
\splinedrop0@, xycurve.doc:2315
\splineEdge, xycurve.doc:3221
\splineendScan@, xycurve.doc:2526
\splineEndtest, xycurve.doc:2444
\splineEndtest@0, xycurve.doc:2433
\splineErec0, xycurve.doc:2964
\splineextra@0, xycurve.doc:2339
\splinef@breakpt, xycurve.doc:2778
INDEX

\splinezeroend, xycurve.doc:2589
\splinezeroirstart, xycurve.doc:2583
\splitRadius@, xyarc.doc:301
\splitRadius@dim, xyarc.doc:292
\spop@, xy.doc:2744
\spread@@, xyv2.doc:564
\spreaddiagramcolumns, xyv2.doc:563
\spreaddiagramrows, xyv2.doc:563
\spushc@, xy.doc:2725
\spushid@, xy.doc:2729
\Square, xybarr.doc:654
\Square, xybarr.doc:1250
\Squarep, xybarr.doc:655
\Squarepp, xybarr.doc:251
\Squarepp, xybarr.doc:256
\Squareppp, xybarr.doc:658
\Squarepppp, xybarr.doc:254
\Squarepppp, xybarr.doc:649
\Squarepppp, xybarr.doc:231
\squarify, xyv2.doc:440
\squarify, 588
\squash, xyv2.doc:423
\squash, 588
\squiggle@, xy.doc:5412
\squiggle@, xy.doc:5423
\squiggles, 5439
\squiggles, 15
\squine@start@, xycurve.doc:10658
\squineconnect@, xycurve.doc:3514
\squined@, xycurve.doc:3668
\squined@@, xycurve.doc:3672
\squinedadvance@@@, xycurve.doc:3593
\squinedecast@, xycurve.doc:3578
\squinedwhich@, xycurve.doc:3471
\squineladvance@, xycurve.doc:3512
\squineinfo@, xycurve.doc:3632
\squinelpt@, xycurve.doc:1510
\squineplot@maybe, xycurve.doc:3595
\squinered@, xycurve.doc:3589
\squineres@, xycurve.doc:3642
\squinesegment@, xycurve.doc:3517
\squineseparams@, xycurve.doc:3664
\squinetestcvxhull@, xycurve.doc:1604
\squinewhich@, xycurve.doc:3863
\ssize, 585
\Ssolid, xyv2.doc:302
\Ssolid, 587
\sssize, 585
\ssz, xycurve.doc:202
\STACK@, xy.doc:2707
\STACK@load, xy.doc:2740
\startxycurve@, xycurve.doc:1915
\startxycurve@@, xycurve.doc:1909
\state, 22
\\stdsplineconn@, xycurve.doc:2331
\\step, 474
\stop, xyv2.doc:346
\stop, 587
\stopper@, xy.doc:5783
\stopper@@, xy.doc:5785
\straight@, xy.doc:6057
\straight@typeset, xy.doc:8802
\straightArc@, xyarc.doc:1309
\straightCbreak@, xy.doc:8104
\straightcheckoverlap@, xy.doc:8078
\straightClast@, xy.doc:8114
\straighth@, xy.doc:8160
\straightv@, xy.doc:8187
\straightalong@, xycurve.doc:1021
\straightbreak@, xycurve.doc:1056
\straightconnect@, xycurve.doc:1083
\straightintercept@, xycurve.doc:1055
\straightlast@, xycurve.doc:1057
\straightreset@, xycurve.doc:1099
\straightshavec@, xycurve.doc:1099
\straightshavep@, xycurve.doc:1005
\straitslidec@, xycurve.doc:1002
\straitslidep@, xycurve.doc:1002
\stripRCS, xy.doc:411
\stripRCS, 14
\stripRCS@, xy.doc:412
\stripRCS@@, xy.doc:441
\stripRCStypenew@, xy.doc:1059
\style, 57
\style [bevel], xyline.doc:250
\style [butt], xyline.doc:254
\style [miter], xyline.doc:252
\style [projcap], xyline.doc:256
\style [roundcap], xyline.doc:255
\style [roundjoin], xyline.doc:251
\style [thicker], xyline.doc:118
\style [thinner], xyline.doc:117
\style option, 11
\\styledbox@, xy.doc:3918
\supermorphism, xy2cell.doc:159
\swap, xyv2.doc:268
\swap , 586
\swap , xy.doc:1645
\swapdimen , xy.doc:1641
\swaptoks@ , xy.doc:1643
system dependencies, 7
\textPi@ , xytpic.doc:348
\testAND@ , xy.doc:115
\testbreakedges@ , xycurve.doc:3011
\tests@ , xy.doc:2692
\testsplineedges@ , xycurve.doc:2483
\TeXX reference point, 21
\Text , xyv2.doc:166
\Text , 585
\text , 585
\text@ , xyv2.doc:170
\text@i , xyv2.doc:174
\text@ii , xyv2.doc:184
\textC , xyv2.doc:168
\Textures@ , xyps.doc:270
\Textures@@ , xytextures.doc:46
\TexturesColor@@ , xytextures.doc:92
\TexturesColor@push , xytextures.doc:75
\TexturesColours@ , xytextures.doc:77
\texturesCrayon@ , xytextures.doc:114
\texturesCrayon@@ , xytextures.doc:113
\texturesCurrpt@ , xytextures.doc:176
\texturesCurrpt@@ , xytextures.doc:177
\TexturesDict@ , xytextures.doc:161
\texturesFrames@ , xytextures.doc:232
\texturesFrames@@ , xytextures.doc:230
\TexturesInclude@ , xytextures.doc:163
\texturesLine@ , xytextures.doc:218
\TexturesLine@@ , xytextures.doc:216
\TexturesMacro@ , xytextures.doc:160
\texturesPS@ , xytextures.doc:192
\TexturesPS@@ , xytextures.doc:122
\TexturesPStypes@ , xytextures.doc:146
\TexturesPSunload@ , xytextures.doc:123
\TexturesRaw@ , xytextures.doc:164
\TexturesRotate@ , xytextures.doc:204
\TexturesRotate@@ , xytextures.doc:202
\TexturesSpecial@ , xytextures.doc:159
\TexturesSpecialDict@ , xytextures.doc:162
\TexturesTiles@ , xytextures.doc:246
\TexturesTiles@@ , xytextures.doc:244
\theF , movie.cls:157, xymovie.doc:76
\theF@ , movie.cls:158, xymovie.doc:77
\theFrame , movie.cls:154
\thelineno@ , xy.doc:471
\thematrixCOORD@@ , xymatrix.doc:802
\thematrixprefix@@ , xymatrix.doc:803
\theMOVE@ , xygraph.doc:216
\theScene , movie.cls:155, xymovie.doc:75
\three , xybarr.doc:926
\threee , xybarr.doc:924
\threepp , xybarr.doc:922
\threeepp , xybarr.doc:920
\threeeppp , xybarr.doc:918
\threeepppp , xybarr.doc:905
\tildeARC@ , xyarcs.doc:903
\Tip , xyv2.doc:354
\Tip@ , xyv2.doc:350
\Tip@ , 588
\tip@ , 588
\Tip@ , xy.doc:3504
\Tip@@ , xy.doc:5661
\Tip@@@ , xy.doc:5639
\Tip@@@ , xytips.doc:152
\Tip@@@ , xytips.doc:130
\Tip@@@ , xytips.doc:146
\Tip@@@ , xy.doc:5634
\Tip@xy , xytips.doc:50
\Tip@xy , xytips.doc:146
\tipfamily cm , xytips.doc:121
tipfamily eu , xytips.doc:142
tipfamily lu , xytips.doc:176
tipfamily@ , xytips.doc:75
\tipjot@ , xy.doc:5832
\tipjot@xy , xytips.doc:53
\TIPS , xytips.doc:49
tips, 15
\tipsize@ , xytips.doc:76
\to , xybarr.doc:870
\todl , xyv2.doc:703
\todr , xyv2.doc:704
\toks@ , 18
\toks@ii , xyidioms.doc:187, xyidioms.doc:207
\told , xyv2.doc:1699
\topleft , xybarr.doc:875
\tolu , xyv2.doc:700
\toop , xybarr.doc:1699
\ToPOS , xy.doc:4564
\topp , xybarr.doc:168
\toppp , xybarr.doc:167
\topppp , xybarr.doc:158
\toradians@ , xy.doc:2054
\tord , xyv2.doc:701
INDEX

Vtriangle, xybarr.doc:466  Vtrianglepair, xybarr.doc:587  Vtrianglepairp, xybarr.doc:588  Vtrianglepairpp, xybarr.doc:590  Vtrianglepairppp, xybarr.doc:592  Vtrianglepairpppp, xybarr.doc:592  Vtrianglepp, xybarr.doc:467  Vtriangleppp, xybarr.doc:469  Vtrianglepppp, xybarr.doc:371  Vtwist, xyknot.doc:653  Vtwistneg, xyknot.doc:661  Vuncross, xyknot.doc:396  Vunder, xyknot.doc:483  Vunover, xyknot.doc:493  Vuntwist, xyknot.doc:669  \W@, xyidioms.doc:115  \W@max, xymatrix.doc:520  \W@maxin, xymatrix.doc:497  \W@maxout, xymatrix.doc:498  Warning, 15  warning messages, 3  \Wcol@C, xymatrix.doc:536  \Wcol@C@, xymatrix.doc:496  \Wcol@HWmax, xymatrix.doc:542  \Wcol@in, xymatrix.doc:494  \Wcol@max, xymatrix.doc:539  \Wcol@out, xymatrix.doc:495  \wdz@, xyidioms.doc:112  \whichCurveObject@, xy2cell.doc:479  \whichframe@, xymatrix.doc:1013  \whichinfo, xy2cell.doc:825  \whichnudge, xy2cell.doc:824  \whichoptions@, xymatrix.doc:1014  \whichPSspecials@, xyps.doc:309  width, 259  \writeoutline@, xy.doc:4888  \writeoutline@, xy.doc:4876  \writePSdict@, xyps-ps.doc:532  \X@base, 22  \X_ybase, 22  \X_c, 22  \X_p, 22  \X_max, 21  \X_min, 21  \xbendd, xyknot.doc:1673  \xbendl, xyknot.doc:1665  \xbendr, xyknot.doc:1661  \xbendu, xyknot.doc:1669  \xc, xys.doc:210  \xc, xybarr.doc:494, xybarr.doc:315, xybarr.doc:547, xybarr.doc:560  \xc, 586  \xcap, xyknot.doc:1894  \xcaph, xyknot.doc:1556  \xcapv, xyknot.doc:1552  \xdashed, 591  \xdotted, 591  \xdouble, 591  \xdvi@, xys.doc:55  \xdvi@Color@, xys.doc:56  \xdviColor@, xys.doc:101  \xdviCrayola@, xys.doc:135  \xdviCrayola@, xys.doc:133  \xdviCurrpt@, xys.doc:212  \xdviCurrpt@, xys.doc:213  \xdviDict@, xys.doc:181  \xdviFrames@, xys.doc:269  \xdviFrames@, xys.doc:267  \xdviInclude@, xys.doc:182  \xdviLine@, xys.doc:256  \xdviLine@, xys.doc:254  \xdviMacro@, xys.doc:180  \xdviPS@, xys.doc:224  \xdviPS@, xys.doc:152  \xdviPS@unload, xys.doc:154  \xdviPSTypes@, xys.doc:165  \xdviRaw@, xys.doc:183  \xdviRotate@, xys.doc:236  \xdviRotate@, xys.doc:234  \xdviRotScale@, xys.doc:243  \xdviSpecial@, xys.doc:179  \xdviTiles@, xys.doc:282  \xdviTiles@, xys.doc:280  \xdviTPIC@, xys.doc:295  \xdviTPIC@, xys.doc:293  \xline, 591  \Xmax, xys.doc:236
\xyvchull@, xycurve.doc:2069
\xyvchulldrop@, xycurve.doc:2086
xyd.mf, 16
xyd2.mf, 16
xydash10, 15
\xydashfont, xy.doc:562
\xydashfont , 15, 16, 74, 75, 82
\xydashh@, xy.doc:564
\xydashl@, xy.doc:563
\xydashw@, xy.doc:565
\xydate, xy.doc:419
\xydate , 14
\xydecagon@, xypoly.doc:1020
\xydef@, xy.doc:38, 3
\xydefultdriver@, xyps-ps.doc:195
\xydefcsname@, xy.doc:200
\xydelayimport@, xyimport.doc:113
XYdict@, xyps-ps.doc:598
XYdict@@, xyps-ps.doc:599
\xydist@, xycurve.doc:3044
\xydodecagon@@
\xyenvelope@, xydocmode@, xy.doc:16396
\xyendinput, xy.doc:5585
\xyendinput , 93, 96
\xyerror@, xy.doc:479
\xyerror@RC, xymatrix.doc:865
\xyescape@, xyp-s.doc:69
\xyescapeSpecial@, xyp-s.doc:75, xyp-s.doc:76
\xyever@, xy.doc:16493
\xyeveryrequest, xy.doc:6499
\xyeveryrequest , 92
\xyeverywithoption, xy.doc:6500
\xyeverywithoption , 92
\xyexportwarning@, xyimport.doc:126
\xyextern@, xyimport.doc:116
\xyextern@i, xyimport.doc:118
\xyextern@ii, xyimport.doc:122
\xyextern@x, xyimport.doc:129
\xyFN@, xyidioms.doc:107
\xyfont@, xy.doc:559
\xygetfilepatterns@, xytile.doc:266
\xygetknotPLACE@@, xyknot.doc:1180
\xygetpolyprefix@, xypoly.doc:273
\xygetpostBREAK@, xyknot.doc:1201
\xygraph, xygraph.doc:160
\xygraph , 376
\xygreet@, xy.doc:421
\xyhc@ch@, xyknot.doc:1566
\xyhc@ch@c, xyknot.doc:412
\xyhc@cs@neg@, xyknot.doc:418
\xyheptagon@@, xypoly.doc:2918
\xyhexagon@@, xypoly.doc:2992
\xyhloop, xyknot.doc:1617
\xyhover, xyknot.doc:1546
\xyhtwist, xyknot.doc:591
\xyhtwistneg, xyknot.doc:597
\xyhuncross, xyknot.doc:424
\xyhunder, xyknot.doc:552
\xyhunover, xyknot.doc:558
\xyhunover, xyknot.doc:558
\xyidioms.doc, 7
\xyidiomsloaded, xyidioms.doc:24
\xyignore, xy.doc:4685
INDEX

\xykxbendd, xyknot.doc:1749
\xykxbendl, xyknot.doc:1737
\xykxbendr, xyknot.doc:1731
\xykxbendu, xyknot.doc:1743
\xykxcaph, xyknot.doc:1597
\xykxcapv, xyknot.doc:1591
\xyxoverh, xyknot.doc:628
\xyxoverv, xyknot.doc:622
\xyxunoverh, xyknot.doc:641
\xyxunoverv, xyknot.doc:634
\xykz, xyknot.doc:1135
\xykzbdndh, xyknot.doc:1703
\xykzbdndv, xyknot.doc:1724
\xylargePolygon, xypoly.doc:1160
\xylattice, xyweb.doc:1160
\xylattice@, xyweb.doc:162
\xylet@, xy.doc:39, 3
\xyletcsnamecsname@, xy.doc:210
\xylinecap@, xyline.doc:220
\xylinejoin@, xyline.doc:221
\xylinemiter@, xyline.doc:222
\xylineSpecial@, xyline.doc:610
\xylinethick@, xyline.doc:131
\xylinewidth@, xyline.doc:586
\xylinewidth@, xyline.doc:555
\xylinewidth@i, xyline.doc:136
\xyLoadDrivers@, xy.doc:6712
\xyloaded, xy.doc:23
\xylocalColor@, xycolor.doc:183
\xylocalpattern@, xytilde.doc:212
\xylowtolerance@, xycurve.doc:2291
\xymakeADD@, xy.doc:305
\xymath@, xy.doc:111
\xymatrix, xymatrix.doc:236
\xymatrix, 60, 355
\xymatrix@addop, xymatrix.doc:980
\xymatrix@addop@, xymatrix.doc:984
\xymatrix@addop@@, xymatrix.doc:940
\xymatrix@addop@x, xymatrix.doc:987
\xymatrix@at, xymatrix.doc:942
\xymatrix@C, xymatrix.doc:991
\xymatrix@change, xymatrix.doc:1178
\xymatrix@fix, xymatrix.doc:3963
\xymatrix@fix@, xymatrix.doc:971
\xymatrix@fix@x, xymatrix.doc:977
\xymatrix@fix@x, xymatrix.doc:996
\xymatrix@i, xymatrix.doc:253
\xymatrix@insert, xymatrix.doc:1691
\xymatrix@insert@, xymatrix.doc:695
\xymatrix@L, xymatrix.doc:1994
\xymatrix@M, xymatrix.doc:1993
\xymatrix@measureit, xymatrix.doc:550
\xymatrix@measureit@@, xymatrix.doc:566
\xymatrix@mods, xymatrix.doc:1164
\xymatrix@mods@error, xymatrix.doc:1205
\xymatrix@mods@x, xymatrix.doc:1186
\xymatrix@mods@xx, xymatrix.doc:1198
\xymatrix@prefix, xymatrix.doc:240
\xymatrix@pretypeset, xymatrix.doc:346
\xymatrix@R, xymatrix.doc:1990
\xymatrix@rotation, xymatrix.doc:2804
\xymatrix@set, xymatrix.doc:1174
\xymatrix@setup, xymatrix.doc:245
\xymatrix@typeset, xymatrix.doc:584
\xymatrix@W, xymatrix.doc:997
\xymatrixcolsep, xymatrix.doc:1016
\xymatrixcolsep@, xymatrix.doc:1013
\xymatrixcompile, xymatrix.doc:278
\xymatrixcompile, 356
\xymatrixcompile@, xymatrix.doc:279
\xymatrixCOORD@, xymatrix.doc:805
\xymatrixCOORD@after, xymatrix.doc:861
\xymatrixCOORD@error, xymatrix.doc:833
\xymatrixCOORD@i, xymatrix.doc:811
\xymatrixCOORD@i@d, xymatrix.doc:822
\xymatrixCOORD@i@l, xymatrix.doc:823
\xymatrixCOORD@i@r, xymatrix.doc:824
\xymatrixCOORD@i@u, xymatrix.doc:821
\xymatrixCOORD@ii, xymatrix.doc:826
\xymatrixCOORD@other, xymatrix.doc:837
\xymatrixCOORD@x, xymatrix.doc:845
\xymatrixCOORD@xx, xymatrix.doc:853
\xymatrixnocompile, xymatrix.doc:222
\xymatrixnocompile, 356
\xymatrixprefix@, xymatrix.doc:238
\xymatrixrowsep, xymatrix.doc:1015
\xymatrixrowsep@, xymatrix.doc:1012
\xymatrixsavedCOORD@, xymatrix.doc:344
\xymidcross@, xyknot.doc:355
\xymiterSpecial@, xyline.doc:373
\xynamestyle@, xy.doc:4052
\xynamestyle@, xy.doc:4058
\xynametransform@, xyrotate.doc:254
\xynew@, xy.doc:40, 3
\xyNoColor@, xycolor.doc:178
\xyNoColor@, xycolor.doc:180
\xynoedgespline@, xycurve.doc:2490
\xynolinecap@, xyline.doc:545
\xytriangle@, xypoly.doc:583
\xytrigerror@, xypoly.doc:960
\xyuncatcodes, xy.doc:45
\xyuncatcodes, 4
\xyundecagon@, xypoly.doc:9062
\xyundefinedEffect@, xyps-ps.doc:670
\xyundefinedLine@, xyline.doc:540
\xyundefinedRotate@, xyrotate.doc:652
\xyundefinedStyle@, xy.doc:1408
\xyunload@, xy.doc:6948
\xyunloadEM@, xymtex.doc:124
\xyunloadTPIC@, xytpic.doc:214
\xyunsupportwarning@, xy.doc:6895
\xyunsupportwarnings@, xy.doc:6909
\xyUseResizing@, xyrotate.doc:134
\xyUseResizing@, xyrotate.doc:135
\xyvcap, xyknot.doc:1565
\xyvcross, xyknot.doc:1499
\xyvcrossneg, xyknot.doc:445
\xyvector, xypicture.doc:55
\xyVECTOR@, xy.doc:1419
\xyverbose, xy.doc:6435
\xyverbose, 66, 67
\xyverbose@, xy.doc:4639
\xyversion, xy.doc:416
\xyversion, 14
\xyviitoi, xylv2.doc:668
\xyviitoii, xylv2.doc:669
\xyvloop, xyknot.doc:1616
\xyvover, xyknot.doc:543
\xyvtwist, xyknot.doc:682
\xyvtwistneg, xyknot.doc:685
\xyvuncross, xyknot.doc:421
\xyvunder, xyknot.doc:549
\xyvunover, xyknot.doc:555
\xyvunvtwist, xyknot.doc:688
\xywarnifdefined, xy.doc:35
\xywarnifdefined , 3
\xywarning@, xy.doc:476
\xywidthchar@, xyline.doc:143
\xywith@@, xy.doc:6453
\xywithoption, xy.doc:6457
\xywithoption , 9, 92
\xywithrun@, xy.doc:6467
\xywithtest@, xy.doc:6464
\xywrite@, xy.doc:699
\xyxbend, xyknot.doc:1693
\xyxbendl, xyknot.doc:1689
\xyxbendr, xyknot.doc:1688
\xyxbendu, xyknot.doc:1690
\xyxcap, xyknot.doc:1568
\xyxcapv, xyknot.doc:1567
\xyxoverh, xyknot.doc:568
\xyxoverv, xyknot.doc:562
\xyxunderh, xyknot.doc:571
\xyxunderv, xyknot.doc:565
\xyxunover, xyknot.doc:574
\xyxunoverh, xyknot.doc:580
\xyxunoverv, xyknot.doc:577
\xyxy@ix@, xy.doc:1126
\xyxy@ix@, xy.doc:1880
\xyxbendh, xyknot.doc:1683
\xyzbendh, xyknot.doc:1686
Y, 21
y, 24, 39
Y@c, xy.doc:612
Y@max, xy.doc:637
Y@min, xy.doc:635
Y@origin, xy.doc:628
Y@p, xy.doc:620
Y@xbase, xy.doc:630
Y@ybase, xy.doc:632
Y@origin: 22
Y@base: 22
Y@base: 22
Y@p: 22
Y@max: 21
Y@min: 21
Y@c, xylv2.doc:211
Y@c, 586
Y@max, xylv2.doc:237
Y@max: 586
Y@min, xylv2.doc:235
Y@min: 586
Y@origin, xylv2.doc:228
Y@origin: 586
Y@p, xylv2.doc:218
Y@p: 586
Y@xbase, xylv2.doc:230
Y@xbase: 586
Y@ybase, xylv2.doc:232
Y@ybase: 586
\zybendh, xyknot.doc:1643
\zybendv, xyknot.doc:1655
zero position, 21
\zerocorner, xyframe.doc:358
\zerodash@i, xycurve.doc:4682
\zerodivide@, \textit{xy.doc:2476}
\zerodivide@@, \textit{xy.doc:2474}
\zerodivide@message, \textit{xy.doc:2475}
\zerodivideLimit, \textit{xy.doc:2484}
\zerodivideLimit@, \textit{xy.doc:2482}
\zerodivideLimit@@, \textit{xy.doc:2478}
\zerodot, \textit{xy.doc:5331}
\zerodot, 17, 77
\zerodot@i, \textit{xycurve.doc:4677}
\zerodotbox@, \textit{xy.doc:655}
\zeroe, \textit{xy.doc:7417}
\zeroe, 49
\zeroit@, \textit{xy.doc:1452}
\Zoom@, \textit{movie.cls:251}
\zz@, \textit{xy.doc:661, 17}