Tabu Package Updates

January 12, 2019

The tabu package patches internal commands of many packages and is liable to stop working if those internal interfaces change.

Unfortunately the original author appears to be out of contact and the package had not been updated for some years.

This manual documents version 2.8 from 2011, but the distributed package contains fixes reported since that time, see https://github.com/tabu-fixed/tabu
Abstract
This package defines a single environment \texttt{tabu} to make all kinds of tabulars in text or in math mode provided that they do not split across pages.

An environment \texttt{longtabu} – based on D. Carlisle \texttt{longtable} package – is also provided to make tabulars that can stretch out on several pages, while keeping some features (not all of them) of the \texttt{tabu} environment.

\texttt{tabu} is more flexible that \texttt{tabular}, \texttt{tabular*}, \texttt{tabularx} and \texttt{array} and extends the possibilities. All tabulars in this document were made with the \texttt{tabu} environment, \textit{of course}... The implementation is optimised to minimise the measurements required to put all together.

\texttt{\texttt{\textsc{tabu}} \texttt{b<}} likes colors too, with special lines that are able to keep the alignment of the surrounded text... and also like numbers with the possibility to embed \texttt{sunitx} S (or s) columns. \texttt{\texttt{\textsc{tabu}} \texttt{b<}} does not modify any of the macro defined by \texttt{array.sty} or in the \texttt{\LaTeX} kernel\footnote{based on D. Arseneau. Finally \texttt{longtabu} is based on \texttt{longtable}.}

\texttt{\texttt{\textsc{tabu}} \texttt{b<}} requires \texttt{\LaTeX} and the standard package \texttt{array.sty}. Natural widths of columns are computed (but not printed ) by the code of \texttt{varwidth} by D. Arseneau. Finally \texttt{longtabu} is based on \texttt{longtable}.

Contents
Summary of the features provided by \texttt{\textsc{tabu}} \texttt{b<} .................................................. 3
1 Examples and counterexamples .......................................................... 5
  1.1 “Locally global” settings and their scopes .................................. 5
  1.2 \texttt{X} column widths computation .......................................... 6
  1.3 Inserting Verbatim material (\texttt{fancyvrb}) ............................... 7
  1.4 Maths inside \texttt{tabu} \texttt{X} columns .................................. 7
  1.5 Embedding \texttt{sunitx} \texttt{S} columns inside \texttt{X} columns ........... 8
2 The \texttt{tabu} environment ............................................................... 8
  2.1 \texttt{tabu}, \texttt{tabu to} and \texttt{tabu spread} .............................. 8
  2.2 \texttt{longtabu}, \texttt{longtabu to} and \texttt{longtabu spread} ............... 9
  2.3 \texttt{tabu} \texttt{X} columns – Mastering horizontal space ................. 10
    2.3.1 \texttt{X} columns with “\texttt{tabu spread}” .......................... 10
    2.3.2 Negativ width coefficients for \texttt{X} columns ..................... 10
    2.3.3 Multicolumn in \texttt{tabu} ............................................ 12
  2.4 \texttt{\texttt{\textbackslash tabulinesep}} and \texttt{\texttt{\textbackslash extrarowsep}} – Mastering vertical space ... 12
  2.5 \texttt{tabu} in math mode ....................................................... 14
3 Lines leaders and colors inside \texttt{tabu} ........................................ 14
  3.1 First important remark .......................................................... 14
  3.2 Vertical lines: \texttt{|} has an optional parameter ......................... 14
  3.3 Multiple \texttt{\texttt{\textbackslash firsthline}} and \texttt{\texttt{\textbackslash lasthline}} ....... 15
  3.4 More style for lines ............................................................ 16
  3.5 Automatic horizontal lines and row colors .................................. 17
4 Modifying the font and the alignment in one row: \texttt{\texttt{\textbackslash rowfont}} ............... 18
5 Saving and restoring a \texttt{tabu}: \texttt{\texttt{\textbackslash savetabu}}, \texttt{\texttt{\textbackslash loadtabu} and \texttt{\texttt{\textbackslash preamble}}} ............. 19

\footnote{This documentation is produced with the DocStrip utility, and required \texttt{\texttt{\textsc{tabu}} \texttt{b<}} with its \texttt{linegoal} option.

--- To get the package, run: \texttt{etex tabu.dtx}
--- To get the documentation run (thrice): \texttt{pdflatex tabu.dtx}
--- To get the index, run: \texttt{makeindex \texttt{-s gind.ist} tabu.idx}

The \texttt{.dtx} file is embedded into this pdf file thank to \texttt{embedfile} by H. Oberdiek.

1. Inside the \texttt{tabu} environment a few macros are modified... this was compulsory!}
6 Some other features ................................................................. 20
  6.1 Printing numbers inside \texttt{numprint} and \texttt{sunitx} .......... 20
     6.1.1 \texttt{tabudecimal} ................................................. 20
     6.1.2 You should know how it works... .................................. 20
  6.2 Paragraph indentation ....................................................... 21
  6.3 \texttt{delarray} shortcuts .................................................. 22

7 Differences between \texttt{tabu}, \texttt{tabular}, \texttt{tabularx} and \texttt{longtable} ..................................................... 22
  7.1 Paragraph indentation ....................................................... 22
  7.2 Custom environments ....................................................... 22
  7.3 Inversion of tokens ........................................................ 22
  7.4 Improved process for rewriting columns \textit{(for keen readers)} .... 23

8 The package options ............................................................. 24
  8.1 The \texttt{debugshow} package option ..................................... 24
  8.2 The \texttt{delarray} package option ...................................... 24
  8.3 The \texttt{linegoal} package option ...................................... 24

9 Corrections of some bugs \textit{(available only inside \texttt{tabu})} .......... 24
  9.1 Correction for \texttt{colortbl} and \texttt{arydshln}: compatibility with \texttt{delarray} ................ 24
  9.2 Correction for \texttt{arydshln}: \texttt{@} columns ......................... 25

10 To do for even better \texttt{tabus} ............................................ 25

11 TECHNICAL NOTICE AND IMPLEMENTATION .............................. 26
  11.1 Drawing a tabular - The $\TeX$\$\times$ approach ..................... 26
  11.2 Algorithms ....................................................................... 26
  11.3 The \texttt{tabu} strategies ................................................ 30
  11.4 Identification and Requirements ....................................... 31
  11.5 Flow chart of expansion .................................................. 31
  11.6 Some constants .................................................................. 33
  11.7 Rules, colors and vertical adjustment .................................... 38
  11.8 The entry inside \texttt{tabu} ................................................ 47
  11.9 The rewriting process: inside the \texttt{\textasciitilde@mkpream group} .... 51
  11.10 Implementing the strategy at the exit of the \texttt{\textasciitilde@mkpream group} ............. 59
  11.11 One trial after the other (\texttt{\textasciitilde@strategy}) ................. 61
  11.12 The algorithms: Measuring the \texttt{tabu} box ......................... 64
  11.13 Measuring the natural width of columns \textit{(varwidth} code from D. Arseneau) .......... 69
  11.14 Measuring the height and depths of rows ............................. 70
  11.15 \texttt{\textasciitilde@tabuphantomline} .................................... 71
  11.16 Horizontal lines inside \texttt{tabu}: \texttt{\textasciitilde@tabucline}, \texttt{\textasciitilde@firsthline} and \texttt{\textasciitilde@lasthline} .................. 72
  11.17 Numbers in \texttt{tabu} .................................................. 75
  11.18 Verbatim inside \texttt{tabu} with X columns ............................... 77
  11.19 \texttt{\textasciitilde@savetabu} ............................................. 78
  11.20 \texttt{\textasciitilde@rowfont} ................................................ 79
  11.21 Taking care of footnotes and \texttt{\textasciitilde@arraybackslash} ............. 84
  11.22 Corrections ....................................................................... 85
  11.23 Package options and Initialisation ....................................... 87

12 References ............................................................................ 91

13 History .................................................................................. 91
[2011/02/26 v2.8] ................................................................. 91
[2011/02/25 v2.7] ................................................................. 91
[2011/02/24 v2.6] ................................................................. 91
[2011/02/19 v2.5] ................................................................. 91
[2011/02/17 v2.4] ................................................................. 91
[2011/02/13 v2.3] ................................................................. 91
[2011/02/12 v2.2 - New implementation - Absolutely no modification of \texttt{array.sty}] ........ 91
[2011/01/19 v2.1] ................................................................. 92
[2011/01/18 v2.0] ................................................................. 92
[2011/01/15 v1.9] ................................................................. 92
[2010/12/28 v1.8] ................................................................. 92
[2010/12/18 v1.7] ................................................................. 93
[2010/12/07 v1.5] ................................................................. 93
[2010/11/22 v1.4] ................................................................. 93
[2010/11/18 v1.3] ................................................................. 93
[2010/11/15 v1.2] ................................................................. 94
[2010/10/28 v1.1] ................................................................. 94

14 Index ............................................................................... 94
Summary of the features provided by $\tau_k b_c$

$\text{tabu}$ is like $\text{tabular}$ in text mode and like $\text{array}$ in math mode when there is no $X$ column in its preamble.

$\text{longtabu}$ is like $\text{longtable}$ with the possibility to use $\text{tabu}$ $X$ columns and vertical lines with the extended syntax.

$\{\text{tabu}\} \to \langle \text{dimen} \rangle$ specifies the target width of the whole tabular. This is like $\text{tabular*}$ with an automatic stretchability that can be overwritten with $@\{\text{extracoalsep} \langle \text{dimen} \rangle\}$ in front of the preamble.

$\{\text{tabu}\} \text{ spread} \langle \text{dimen} \rangle$ has no equivalent in $\LaTeX$: the final width is $\langle \text{dimen} \rangle$ wider than the natural width that can be obtained with $\text{spread Opts.}$

$\langle \text{width, color} \rangle$ vertical lines have an optional parameter.

$X[\text{coef, align, type}]$ $X$ columns widths are adjusted in order for the whole tabular to fit the target width. The target width is a dimension either:
- directly specified with $\{\text{tabu}\} \to \langle \text{dimen} \rangle$
- computed from the natural width: $\{\text{tabu}\} \text{ spread} \langle \text{dimen} \rangle$
- by default $\text{linspacing}$ (or $\text{linegoal}$ with the $\text{linegoal}$ package option).

$\text{coef}$ scales the widths of the $X$ columns, if there are more than one $X$ column.

$\text{align}$ is either $r$, $c$, 1 or $j$ (or $R$ $C$ $L$ $J$) and $\text{type}$ can be $p$ (default), $m$ or $b$.

$X[\$]$ makes a math $X$ column ($\text{ie. } >\{\$\}X<\{\$\}$)

$X[\$$]$ display math $X$ column: $>\{\displaystyle \}X<\{\$$\}$

$X[\langle - \text{coef, align, type} \rangle]$ $X$ columns widths are first computed with the absolute value: $|\text{coef}|$. Then the width is made narrower down to the natural width of the column if possible.

In any case, the final width does not exceed the one obtained with $X[|\text{coef}|]$.

$X[X \text{ options}][S[S \text{ options}]]$ Embed a $\text{sunit}$ $S$ column into a $\text{tabu}$-$X$ column.

$\text{everyrow} \{\text{code}\}$ Allows to add horizontal lines automatically for every row.

The settings can be changed inside the $\text{tabu}$

$\text{rowfont} \{\text{align}\}{\langle \text{font spec} \rangle}$ Modify the font and optionally the alignment of each cell in one row.

$\text{tabulinesep} = \langle \text{dimen} \rangle$ More control on vertical spacing of lines in a way very close to $\text{cellspace}$'s method (dynamic vertical spacing adjustment).

$\text{extrarowsep} = \langle \text{dimen} \rangle$ Control vertical spacing ($\text{extrarowheight}$ and $\text{extrarowdepth}$): fixed vertical spacing adjustment.

$\text{tabulinesep}$ generally gives better results.

$\text{tabudecimal} \{\langle \text{usermacro} \rangle \}$ a help to align numbers easily inside a column.

$\text{savetabu} \{\text{user-name}\}$ Saves the $\text{tabu}$ preamble and its parameters. The command must appear at the end of a line.

Makes a $\text{tabu}$ of exactly the same shape as the one saved with $\text{savetabu}$. All parameters ($\text{target}$, $\text{preamble}$, $\text{stretch etc.}$) are restored.

This command is put alone in the preamble in place of the columns specifications.

$\text{usetabu} \{\text{user-name}\}$ Makes a $\text{tabu}$ with the same preamble as the one saved with $\text{savetabu}$. The only $\text{preamble}$ is restored, not the $\text{target}$ nor any other parameter.

This command is put alone in the preamble in place of the columns specifications.
Summary of the features provided by $\text{T}_{\text{Rb}}$

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{\tabulinestyle {line spec}}</td>
<td>Sets the current line style to be used for</td>
</tr>
<tr>
<td>\texttt{\newtabulinestyle {name=spec,...}}</td>
<td>Defines a line style for use with \texttt{\tabucline [name]} or with \texttt{</td>
</tr>
<tr>
<td>\texttt{\tabucline {spec}{start-stop}}</td>
<td>Draws a line comparable to \texttt{\hline}. The line \texttt{(spec)} can contain information for making a dash or dotted line (f.ex. \texttt{[on 3pt off 6pt]}) and a color name. The line spec can also be defined with \texttt{\newtabulinestyle}</td>
</tr>
<tr>
<td>\texttt{\taburulecolor {rule color}}</td>
<td>sets the color for rules \texttt{(\hline, \texttt{\firstline})}</td>
</tr>
<tr>
<td>\texttt{\taburowcolors {skip}{number}{first .. last}}</td>
<td>Sets the color series to make alternate background colors for rows</td>
</tr>
<tr>
<td>\texttt{\tabuphantomline}</td>
<td>inserts a phantom (ie.invisible) line inside the \texttt{tabu} May be usefull with \texttt{\multicolumn} in some cases.</td>
</tr>
<tr>
<td>\texttt{\tracingtabu = 0, 1, 2, 3, 4}</td>
<td>Reports informations in the .log file about the steps of the algorithm for \texttt{tabu X} columns, and the informations saved by \texttt{\savetabu}</td>
</tr>
</tbody>
</table>
1 Examples and counterexamples

Let’s begin in colors!

\(\text{T}_{\text{\LaTeX}}\) provides facilities to put horizontal and vertical leaders in a tabular. The package \texttt{xcolor} must be loaded of course. Background colors for cells are left to package \texttt{colortbl} which is fully compatible with \(\text{T}_{\text{\LaTeX}}\).

1.1 “Locally global” settings and their scopes

\(\text{T}_{\text{\LaTeX}}\) observes \TeX{} grouping levels for the settings of rule colors (\texttt{\taburulecolor}) and styles (\texttt{\tabulinestyle}), and \texttt{\everyrow}. There is however a subtility for nested \texttt{tabu} environments as described in this example:

Listing 1: Locally global settings and their scopes

```
\begin{tabular}{|X|X|}
\hline
\taburulecolor{gray!50}\{\red\} \arrayrulewidth=1pt
\begin{tabular}{|X|X|} \hline
\taburulecolor{yellow}\{\blue\}
Here the lines \& are drawn in \blue \& \\taburulecolor{green}\hline
But starting from here \& they are \green\ coloured!\ \\hline
And now a nested \tabu & \begin{tabular}{(X)} \firsthline\hline
guess what colour \& \hline
is used for rules?\ \\hline\\lasthline\hline
\end{tabu} \ \\hline
\end{tabular}
% Inside the group, rule colors are blue
\hline
% After the group, rule colors are red again!
\begin{tabular}{X}\hline\hline\indent\end{tabular}
\end{tabular}
```

Here the lines are drawn in blue
But starting from here they are green coloured!

And now a nested \texttt{tabu}

guess what colour is used for rules?

Inside the group, rule colors are blue After the group, rule colors are red again!

The “rules” are the following:

- If outside of a \texttt{tabu} environment, the settings are local to the \TeX{} group. Every tabular drawn inside this group will inherit from the settings of that group.

- If \texttt{\taburulecolor} (or \texttt{\everyrow} or \texttt{\tabulinestyle}) is used inside a cell of the tabular, this is the same: the settings a local to that cell, and any nested tabular will inherit from the setting of that cell.

- When used after the end of a row, the settings are globally changed from that point until the end of the tabular, or until a new setting is set at the end of a further row (\TeX{}nically, this is done inside a \texttt{\noalign} group). But a nested \texttt{tabu} does not inherit from this “global” setting, and inherits from the settings of the \TeX{} group instead.

If \texttt{\arrayrulecolor} or \texttt{\doublerulesepcolor} (from package \texttt{colortbl}) are used instead of \texttt{\taburulecolor} then colors are globally overwritten.
A counterexample from the \texttt{xcolor} package: \texttt{\rowcolors} does not like \texttt{\cline}, \texttt{\cmidrule} etc.\footnote{2}

\begin{tabular}{cc} \toprule test & row 1 \\ test & row 2 \\ test & row 3 \\ test & row 4 \\ test & row 5 \\ test & other row 6 \\ test & other row 8 \\ test & other row 9 \\ \bottomrule \end{tabular}

The \texttt{\rownum} counter is not reliable in the case of \texttt{\cline} or \texttt{\cmidrule}.

In addition, the first coloured row is yellow, while one could have expected it green...

For $\tau_{bU}$ color changes are called at \texttt{\everyrow}:

\begin{tabular}{cc} \toprule test & row 1 \\ test & row 2 \\ test & row 3 \\ test & row 4 \\ test & row 5 \\ test & other row 6 \\ test & other row 7 \\ test & other row 8 \\ \bottomrule \end{tabular}

$\tau_{bU}$ does not use “real” alternate colors but colorseries provided by package \texttt{xcolor}. This allow some gradations:

\begin{tabular}{cc} \toprule test & Row number 1=1 \\ test & Row number 2=2 \\ test & Row number 3=3 \\ test & Row number 4=4 \\ test & Row number 5=5 \\ test & Row number 6=6 \\ test & Row number 7=7 \\ test & Row number 8=8 \\ test & Row number 9=9 \\ test & Row number 10=10 \\ \bottomrule \end{tabular}

\subsection*{1.2 X column widths computation}

The new algorithm implemented in version 2.8 requires only one measure of the width of the table in any case. This speeds up the convergence of the algorithm.

\begin{tabular}{|c|c|c|c|}
| \dotfill | & Text | & Text | & Text |
| Text | & Text | & Text |
| 1X=17.5\,mm | 2X=35\,mm | 3X=52.5\,mm | 1X=17.5\,mm |
\end{tabular}

\[ X = (140\,mm - 8 \times \texttt{\tabcolsep} - 5 \times \texttt{\arrayrulewidth}) / 7 = 17.4896\,mm \]

\footnote{2. Because color changes are done at \texttt{\everycr}, which is not exactly the same as $\tau_{bU}$ \texttt{\everyrow}!}
1.3 Inserting Verbatim material (fancyvrb)

Though the content of the `tabu` environment is collected for measuring purpose, it is possible to insert verbatim material with the `tabu*` variant of the environment. The content is then carefully collected and re-scanned (with `\scantokens`). During the process, the `\%` letter is read with the category code it has been given at the entry inside the environment (it is possible to say `\makeatletter` before `\begin{tabu*}`).

Example:

It is possible to insert Verbatim material with some `\csname` control sequences `\endcsname` inside a `tabu` and inside `X` columns. Negativ coefficients work well too, adjusting the width of the `X` column to the natural width if it is finally less than the width computed with the absolute value of the coefficient.

A complete Verbatim environment is also admissible.

But you must use the star form of the environment: `tabu*` which uses `\scantokens`.

Verbatim environments must be put alone on their lines (in the input file) for nothing is allowed after `\begin{Verbatim}` or `\end{Verbatim}`. Another point to know is that `\begin` and `\end` control sequences should match otherwise, you must enclose the Verbatim environment inside braces.

This is related to the fact that `tabu` collects its body, and looks for matching pairs of `\begin ... \end`!

`tabu*` is useless when nested inside another `tabular`. The star form of the environment should be used only for the outermost table! Comments are removed, unless the `%` character is given a category code of 12 (or 11) before the entry inside the environment.

```
\begin{tabu*}
| a & b & c | d & e & f |
\end{tabu*}
```

Here a small `\Verbatim` insertion

And this is a complete % with some comments

Verbatim environment % every now and then

It’s not possible to insert a `lstlisting` environment presently, but you can save such an environment in a `\vbox` and insert it inside the `tabu` of course.

1.4 Maths inside `tabu` `X` columns

```
\begin{tabu}{|X|}
\hline
\hline
\end{tabu}
```

```
\begin{tabu}{|X|}
\hline
\hline
\end{tabu}
```

```
\begin{tabu}{|X|}
\hline
\hline
\end{tabu}
```

```
\begin{tabu}{|X|}
\hline
\hline
\end{tabu}
```

```
\begin{tabu}{|X|}
\hline
\hline
\end{tabu}
```

```
\begin{tabu}{|X|}
\hline
\hline
\end{tabu}
```

```
\begin{tabu}{|X|}
\hline
\hline
\end{tabu}
```

```
\begin{tabu}{|X|}
\hline
\hline
\end{tabu}
```
1.5 Embedding \texttt{sunitx} $S$ columns inside $X$ columns

A $S$ column from \texttt{sunitx} can be embedded into a $X$ column of $T_{\texttt{\textbackslash{}tab}\texttt{\textbackslash{}b}}$... with the following limitations:

- The $X$ column must be centered: $X[c]$ to keep the alignment,
- The optional alignment parameter of $\texttt{\textbackslash{}rowfont}$ must not be used.

\begin{verbatim}
\newcolumntype{Y}{[group-four-digits=true,\
  round-mode=places,\
  round-precision=2,\
  round-integer-to-decimal=true,\
  per-mode=symbol,\
  detect-all]}
\newcolumntype{Z}{X[c]{[group-four-digits=true,\
  round-mode=places,\
  round-precision=2,\
  round-integer-to-decimal=true,\
  per-mode=symbol]}}
\begin{tabu}spread 8pt{|*2{Y|}c} % Table 1
  \rowfont{bfseries}
  \hline
  \text{January} & \text{February} & \text{...} \\
  12.32 & 745.32 & \text{...} \\
  21.13 & 0 & \text{...} \\
  213.324 & 12.34 & \text{...} \\
  2143.12 & 324.325 & \text{...} \\
  \hline
\end{tabu}
\end{verbatim}

Column widths are not exactly the same

\begin{verbatim}
\begin{tabu}spread 8pt{|*2{Z|}c} % Table 2
  \rowfont{bfseries}
  \hline
  \text{January} & \text{February} & \text{...} \\
  12.32 & 745.32 & \text{...} \\
  21.13 & 0 & \text{...} \\
  213.324 & 12.34 & \text{...} \\
  2143.12 & 324.325 & \text{...} \\
  \hline
\end{tabu}
\end{verbatim}

Column widths are exactly the same

$\texttt{\textbackslash{}tab\textbackslash{}column}$ is there to say $T_{\texttt{\textbackslash{}tab\textbackslash{}b}}$ that the column type has to be treated with a high priority in the rewriting process.

Another possibility to print number is provided with $\texttt{\textbackslash{}tab\textbackslash{}decimal}$.

2 The \texttt{tabu} environment

2.1 \texttt{tabu}, \texttt{tabu to} and \texttt{tabu spread}

The $\texttt{tabu}$ environment behaves mostly like $\texttt{tabular}$: the preamble is parsed by the macros in $\texttt{array.sty}$ and some measures are performed before printing. $\texttt{tabu}$ improves $\texttt{tabular}$ and $\texttt{array}$:

- footnotes and index words are allowed inside $\texttt{tabu}$, unlike $\texttt{tabular}$, footnote links are not broken when compiled with $\texttt{hyperref}$. The syntax $\texttt{\footnote{\text{\textbackslash{}text}}}{}$ is allowed in $\texttt{tabu}$ and $\texttt{longtabu}$ (this is not implemented for $\texttt{longtable}$ yet...)
- $X$ columns are implemented with an \textit{optional} parameter for the \texttt{width-coefficient} (which can be \texttt{negativ}: see next section), the \texttt{alignment} ($r$, $c$, $l$, or $j$, and $R$, $C$, $L$ or $J$ for
ragged2e settings) and the column type \(p, m, \) or \(b\).

\textbf{tabu} has a default target width when used with \(X\) columns, making nesting even easier.

- You are used to the \texttt{tabular} environment in text mode, and \texttt{array} environment in math mode, but \texttt{tabu} works in both modes and its name does not change... \(X\) columns are also possible in math mode; \texttt{delarray} shortcuts for delimiters are available in both math and text modes.

- A \texttt{tabu} environment can contain another \texttt{tabular} of any kind: \texttt{tabular}, \texttt{tabular*}, \texttt{tabularx} or \texttt{tabu} itself can be placed in any cell of a \texttt{tabu}. Conversely, \texttt{tabu} can be placed in a \texttt{tabular}, \texttt{tabularx} etc..

- \texttt{tabu} provides facilities for vertical and horizontal lines, and for the insertion of verbatim text inside \(X\) columns.

- \texttt{tabu} is fully compatible with \texttt{colortbl}, \texttt{delarray}, \texttt{hhline}, \texttt{makecell}, \texttt{booktabs}, \texttt{siunitx}, \texttt{dcolumn}, \texttt{warpcol}, etc. When you are inside a \texttt{tabu} environment, you can use \texttt{\raggedleft}, \texttt{\raggedright} and \texttt{\centering} without special care about \texttt{\arraybackslash} and conversely \texttt{\} has its “normal” meaning inside a list of items that may appear in a \(X\) column...

\begin{verbatim}
\texttt{\begin{tabu} to} \langle \texttt{dimen} \rangle \texttt{is like} \texttt{tabular*} \texttt{but the inter-columns space is given a stretchability of 1fil, in other words} \texttt{@\{\extracolsep \{Opt plus 1fil\}\}} \texttt{is inserted by default at the beginning of the tabular preamble, unless another value for} \texttt{\extracolsep} \texttt{is specified. Therefore} “\texttt{tabu to} fills in width the specified} \langle \texttt{dimen} \rangle.
\end{verbatim}

\begin{verbatim}
\texttt{\begin{tabu} spread} \langle \texttt{dimen} \rangle \texttt{does a tabular whose width is} \langle \texttt{dimen} \rangle \texttt{wider than its natural width.} \texttt{@\{\extracolsep \{Opt plus 1fil\}\}} \texttt{is inserted by default if} \langle \texttt{dimen} \rangle > 0.
\end{verbatim}

2.2 \texttt{longtabu}, \texttt{longtabu to} and \texttt{longtabu spread}

\begin{verbatim}
\texttt{\begin{longtabu}} \texttt{[l | c | r\} \{tabular preamble\}
\texttt{\begin{longtabu}} \texttt{to} \langle \texttt{dimen} \rangle \texttt{[l | c | r\} \{tabular preamble\}
\texttt{\begin{longtabu}} \texttt{spread} \langle \texttt{dimen} \rangle \texttt{[l | c | r\} \{tabular preamble\}
\end{verbatim}

\texttt{longtabu} is just like \texttt{tabu} but page breaks are allowed between rows of the table. \texttt{longtabu} is based on the \texttt{longtable} package which must be loaded, and all features of the \texttt{longtable} environment works inside \texttt{longtabu}: \texttt{\endhead}, \texttt{\endfirsthead}, \texttt{\endfoot}, \texttt{\endlastfoot} and \texttt{\caption}.

\texttt{longtabu} enhances the \texttt{longtable} environment with the possibility to use \(X\) columns and line specifications for horizontal and vertical rules. \texttt{longtabu} is thus much easier than \texttt{ltxtable}.

The following commands provided for \texttt{tabu} do not work with \texttt{longtabu}:

| \begin{tabular}{l|c|l}
| \textbf{tabu command} & \begin{tabular}{c} \textbf{Not available} \end{tabular} & \begin{tabular}{c} \textbf{Not (yet) implemented} \end{tabular} & \textbf{Comment} \\
| \hline
| \texttt{\tabucline} & \star & \texttt{\tabucline} does not care of page breaks presently: use \texttt{\hline} instead.
| \texttt{\usetabu} & \times & \texttt{\usetabu} but \texttt{\savetabu} and \texttt{\preamble} work.
| mathematical mode & \times & \texttt{longtabu} is not designed to work in math mode.
| \texttt{\delarray} shortcuts & \times & \texttt{\delarray} is not designed to work in math mode.
| \texttt{\tabuphantomline} & \times & \texttt{\tabuphantomline} is useless inside \texttt{longtabu}
| \hline

\begin{tabular}{l}
However, \texttt{\tabu} \(X\) columns, \texttt{\rowfont}, \texttt{\extrarowsep}, \texttt{\tabulinesep}, \texttt{\tabudecimal}, \texttt{\tabucline} (with restrictions on page breaks), \texttt{\taburulecolor}, \texttt{\tabulinestyle}, \texttt{\taburowcolors}, \texttt{\preamble}, \texttt{\longtabu to} , \texttt{\longtabu} \texttt{spread} work inside \texttt{longtabu}.
\end{tabular}
2.3 tabu X columns – Mastering horizontal space

*tabu* X columns can be viewed as an enhancement of *tabularx* X columns, but do not interact with them, for they are defined only for a short time during the parsing of the preamble:

  This means that the first X column will be two and a half wider than the second one or that the first X column width will be 1/3 of the whole tabular width.

<table>
<thead>
<tr>
<th>X[2.5]</th>
<th>X</th>
</tr>
</thead>
</table>

- **negativ width coefficients** can be given to X columns: ex. X[-2.5]X[1] or X[-2.5]X or X[-5]X[2]
  In this case, the first X column will be **at most** two and a half wider than the second one, and if the **natural width** of the first X column is finally less than 2.5 × (the width of the second column) then it will be narrowed down to this natural width.

The following **tabus** have the same preamble:

\begin{tabular}{|X|}
\end{tabular}

<table>
<thead>
<tr>
<th>X[2.5]</th>
<th>X</th>
</tr>
</thead>
</table>

Negativ coefficients make X columns close to standard 1, c and r columns.

- **horizontal alignment specification** is made easier with X[5,r]X[2,c] for example. Vertical alignment can be specified as well with X[5,r,m]X[2,p,c] (commas are not required, but X[2cm] or X[4pc] could be misunderstood – not by *TkX* by you...).

<table>
<thead>
<tr>
<th>Modifier</th>
<th>Meaning</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, c, r, j, L, C, R, J</td>
<td>left, centered, right, justified</td>
<td>j</td>
</tr>
<tr>
<td>p, m, b</td>
<td>X column is converted into p, m or b column</td>
<td>p</td>
</tr>
<tr>
<td>$</td>
<td>X[$] is a shortcut for: ›{$}X&lt;{$}</td>
<td></td>
</tr>
<tr>
<td>$$</td>
<td>X[$$] is a shortcut for: ›{$\text{displaystyle } }X&lt;{$}</td>
<td></td>
</tr>
</tbody>
</table>

- **tabu** X columns can be spanned with \multicolumn.
- **tabu** X columns can be used with “**tabu spread**” for small tabulars.
- **tabu** X columns can contain any type of tabular, tabular*, tabularx or tabu without special care about the syntax. *tabu* can also be put inside *tabular*, tabular* and *tabularx*. As long as *tabu* with X columns has a **default target**, nesting *tabu* with X columns is easy. Furthermore, the default global alignment of a nested *tabu* is t (for top) while the default global alignment of a *tabu* in a paragraph is c (for centered).
- The “algorithm” (or the arithmetic) to get the target width for **tabu** X columns is the same as the one used by *tabularx*. \texttt{\textbackslash halign} is the “tolerance” for the whole tabular width. We use \texttt{\textbackslash e-TkX \textbackslash dimexpr} instead of \texttt{\textbackslash linegoal} primitives (with round/truncate bias correction).
- Convergence to the target width is optimised: the \texttt{\textbackslash halign} preamble is not re-built at each trial, but only expanded again, until the target is reached. Though optimized, the process is the same as the one implemented for *tabularx* and in particular the content of the \texttt{\textbackslash tabu} environment is collected as soon as a \texttt{\textbackslash tabu} X column is found in the preamble. This implies restrictions on catcode modifications and verbatim text inside a \texttt{\textbackslash tabu} with X columns.
- If the width of the whole tabular is not specified with “**tabu to**” it is considered to be \texttt{\textbackslash linegoal}. The \texttt{\textbackslash linegoal} package option makes the default width equal to \texttt{\textbackslash linegoal}. Compilation must then be done with pdf\texttt{\textbackslash e-TkX} either in pdf or dvi mode, and package \texttt{linegoal} is loaded. \texttt{\textbackslash linegoal} requires pdf\texttt{\textbackslash e-TkX} for its \texttt{\textbackslash pdfsavepos} primitive and the \texttt{zref-savepos}: if the \texttt{tabu} is not alone in its paragraph \textit{ie.}if the target is not \texttt{\textbackslash linegoal}, then two compilations (or more) are required to get the correct target.
  Default target for nested \texttt{tabu} environments is always \texttt{\textbackslash linegoal}, which equals to the column width inside p, m, b and X columns.
- As long as the \texttt{\textbackslash halign} content is expanded more than once, protections against counters
incrementation, whatstis (\textit{write}) index entries, footnotes etc. are set up: the mechanism of \texttt{tabularx} is reimplemented and enhanced for \texttt{tabu X} columns. \texttt{\tabuDisableCommands} can be used to neutralize the expansion of additional macros during the trials.

X columns with \texttt{“tabu spread”}

\texttt{tabu X} columns can be used with \texttt{“tabu spread”} to adjust the column widths of tabulars that contain only small pieces of text. The question is: how to make a tabular the width of the line, with 6 columns; the columns 1, 2, 5 and 6 are of equal widths and the widths of columns 3 and 4 are only one half. As possible solution:

\hline
1 & 2 & 3 & 4 & 5 & 6 \\
\hline
\end{tabu}

But the text in each cell is very short: one single character, and you prefer the table to be tight, but don’t know the exact width of the whole:

\hline
1 & 2 & 3 & 4 & 5 & 6 \\
\hline
\end{tabu}

But now it’s definitely too narrow, then give it some more space:

\hline
1 & 2 & 3 & 4 & 5 & 6 \\
\hline
\end{tabu}

\texttt{tabu spread} is useless with long columns: the following tabular was made with this preamble:

\begin{tabu}{|X[9]|X[4]|X|}
\hline
\texttt{\begin{tabu}{|X[-1m]|X[c m]|}}
\tabucline - \\
\end{tabu}
\end{tabu}
\begin{tabu}{|X[-1]|X[-1]|}
\hline
alpha & beta \\
\hline
\end{tabu}

\texttt{tabulinestyle} \{3pt ForestGreen\}

\begin{tabu}{|X[1-1]|X[c m]|}
\end{tabu}
\tabucline - \\
\end{tabu}
\begin{tabu}{|X[-1]|X[-1]|}
\hline
alpha & beta \\
\hline
\end{tabu}

\begin{tabu}{|X[9]|X[4]|X|}
\hline
\texttt{\begin{tabu}{|X[-1m]|X[c m]|}}
\tabucline - \\
\end{tabu}
\end{tabu}
\begin{tabu}{|X[-1]|X[-1]|}
\hline
alpha & beta \\
\hline
\end{tabu}

There the text was too long, and \texttt{tabu spread} behaves as if you didn’t give it a target.

The result of this example is the same as if one had written \begin{tabu}{to\linewidth}.

In the preamble, \texttt{\{\}} means that the margin is removed.

Negativ width coefficients for X columns

3. Capital of the U.K. (too see a linked footnote)
Multicolumn in \tabu

\tabuphantomline

The process of \multicolumn implies the \TeX primitive \omit which discards the tabular preamble for the spanned columns. Discarding the preamble means discarding the information about the widths of the columns. This explains why the following example does not work properly:

\begin{tabu}{|X|X|X[2]|} \tabucline- \multicolumn2{|c|}{Hello} \ & World \ \tabucline- \end{tabu}

The correct result can be obtained by the mean of a phantom line, that will remain invisible unless your preamble contains special \@ or ! columns that prints some text:

\begin{tabu}{|X|X|X[2]|} \tabucline- \multicolumn2{|c|}{Hello} \ & World \ \tabucline- \tabuphantomline \end{tabu}

Hello | World

Remember you may need \tabuphantomline in conjunction with \savetabu and \usetabu with \multicolumn. Even if it is possible to add a \tabuphantomline in any line of the \tabu, it is a good practice to append it at the end of the \tabu, for it may introduce indesirable side effects on vertical alignment otherwise, when \tabu is nested inside another tabular.

In particular, \tabuphantomline should not be followed by \cr or \ or \tabularnewline...

The need for this command could disappear in a future release, but this requires a complete new implementation of \multicolumn...

2.4 \tabulinesep and \extrarowsep – Mastering vertical space

\tabulinesep sets the minimal vertical space allowed between the cell content and the cell border. The macro may be prefixed by \global (even inside a \noalign group)\footnote{However \tabulinesep is not a dimension ! You can’t test, for example, \ifdim \tabulinesep > 0pt ! Test \tabulinesep and \belowtabulinesep instead, if needed.}.

It is possible to set the “top limit” (a \TeX dimension called \abovetabulinesep) and the “bottom limit” independently with the syntaxes:

\tabulinesep =\langle dimen \rangle \tabulinesep =^\langle dimen \rangle \tabulinesep =_\langle dimen \rangle \tabulinesep =^\langle dimen \rangle _\langle dimen \rangle

These parameters can be used in text and math modes to give more vertical space between lines, especially when using math formulæ.

Examples (with \tracingtabu = 3 and \interfaces-\papergraduate to see the struts):
\texttt{\extrarowsep} is a soft parameter, and leads to rows which do not share the same height.

\[
\texttt{\extrarowsep} = \langle \text{dimen} \rangle
\]
\[
\texttt{\extrarowsep} = \_\langle \text{dimen} \rangle
\]
\[
\texttt{\extrarowsep} = \_\langle \text{dimen} \rangle \_\langle \text{dimen} \rangle
\]
\[
\texttt{\extrarowsep} = \langle \text{dimen} \rangle \_\langle \text{dimen} \rangle
\]
\[
\texttt{\extrarowsep} = \_\langle \text{dimen} \rangle \langle \text{dimen} \rangle
\]

\texttt{\extrarowsep} is an extra vertical space which is added to each row, inconditionally. \texttt{array.sty} provides the \TeX\ dimension \texttt{\extrarowheight} and \texttt{\extrarowdepth} in addition. As a result, the rows can share the same height/depth but the spacing is not dynamic. \texttt{\tabulinesep} can be used even with positive values for \texttt{\extrarowsep}, for \texttt{tabu} inserts only one strut per row and vertical spacing computations are possible in all cases.

The macro can be prefixed by \texttt{\global} as well, even inside a \texttt{\noalign} group\(^5\).

Set \texttt{\extrarowheight} and \texttt{\extrarowdepth} to different values, with the syntaxes:

\begin{itemize}
  \item \texttt{\extrarowsep} = \langle \text{dimen} \rangle \quad \text{sets} \ \texttt{\extrarowheight} \ \texttt{\extrarowdepth} \text{is unchanged}
  \item \texttt{\extrarowsep} = \_\langle \text{dimen} \rangle \quad \text{sets} \ \texttt{\extrarowdepth} \ \texttt{\extrarowheight} \text{is unchanged}
  \item \texttt{\extrarowsep} = \_\langle \text{dimen} \rangle \_\langle \text{dimen} \rangle \quad \text{sets} \ \texttt{\extrarowdepth} \ \texttt{\extrarowheight}.
\end{itemize}

Both \texttt{\extrarowheight} and \texttt{\extrarowdepth} are scaled by \texttt{\arraystretch} (a scaling \texttt{macro}\(^6\) of \texttt{array.sty}) if \texttt{\arraystretch} > 1...

These parameters can be used in text and math modes.

Examples (with \texttt{\tracingtabu} = 3 and \texttt{\interfaces\-\papergraduate} to see the struts):

\begin{itemize}
  \item \texttt{\extrarowsep} = 3mm
  \item \texttt{\extrarowsep} = 0mm
\end{itemize}

---

5. However \texttt{\extrarowsep} is not a dimension! You can’t test, for example, \texttt{\ifdim \extrarowsep > 0pt}! Test \texttt{\extrarowheight} and \texttt{\extrarowdepth} instead, if needed.

6. \texttt{\arraystretch} is not a dimension but a macro that stores a scaling factor.
2.5 tabu in math mode

On the left, you can see the famous Maxwell-Lorentz equations for electromagnetic field in vacuum, published in 1873.

\[
\begin{align*}
\text{div } \mathbf{E} &= \frac{\rho}{\varepsilon_0} \\
\text{div } \mathbf{B} &= 0 \\
\text{rot } \mathbf{E} &= -\frac{\partial \mathbf{B}}{\partial t} \\
\text{rot } \mathbf{B} &= \mu_0 \mathbf{j} + \mu_0 \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t}
\end{align*}
\]

In this example, the big \texttt{tabu} is: \texttt{\begin{tabu} to\linewidth {XX[-1$]}}. The nested \texttt{tabu} (in math mode) uses \texttt{delarray} shortcut: its preamble is: \texttt{\begin{tabu}{rl}}. \texttt{\tabulinesep} has been set to 2pt. Horizontal rules are \texttt{booktabs \toprule} and \texttt{\bottomrule}.

<table>
<thead>
<tr>
<th>array</th>
<th>tabu spread 1em</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\alpha \ \beta)</td>
<td>(\alpha \ \beta)</td>
</tr>
<tr>
<td>(\gamma \ \delta)</td>
<td>(\gamma \ \delta)</td>
</tr>
</tbody>
</table>

Here, vertical lines are made with \texttt{delarray} shortcuts: \$\begin{tabu} spread 1em \{|cc|\}$

Vertical lines inside the tabular preamble gives:

\[
\begin{tabu}{|c|c|}
\hline
\alpha & \beta \\
\gamma & \delta \\
\hline
\end{tabu}
\]

This was an example of \texttt{\savetabu...\usetabu} to keep the alignment.

3 Lines leaders and colors inside \texttt{tabu}

3.1 First important remark

The features provided in this section are quite experimental: they are not generally taken for good typography. You can use \texttt{Tb} with package \texttt{booktabs} for example, which provides properly designed commands for horizontal rules in tabulars. \texttt{arydshln} is pretty good too, but it modifies a huge amount of macros of \texttt{array.sty}, something that \texttt{Tb} does not.

Lines in \texttt{tabu} printed in this document are mostly made with \texttt{booktabs}.

3.2 Vertical lines: \texttt{|} has an optional parameter

Inside \texttt{tabu} environment, the vertical line marker \texttt{|} has an \texttt{optional} argument which is the width of the vertical rule. The default width remains \texttt{\arrayrulewidth} of course. The optional argument for \texttt{|} can also contain the name of a color. \texttt{color names} are only possible, not a color specification by the mean of a color model. The width of the line if specified, must come before the color name and... as for \texttt{X} columns parameters, commas are optional.

Example:

\[
\begin{tabu}{||[5pt]c||[5pt]|}
\hline
Hello & World \\
\hline
\end{tabu}
\]

This example was printed inside a \texttt{tabu} whose preamble is: \texttt{X[-1m]X[m]X[-2m]}

It is not a necessary to protect the optional argument with braces: \{[\ldots]\}, because \texttt{Tb} takes care the \texttt{|} token to be rewritten before any other column type (the same for \texttt{tabu X} columns, and \texttt{siunitx S} columns). The rewriting process is divided into three stages under control inside a \texttt{tabu} environment.
3.3 Multiple `\firstline` and `\lastline`

```
\firstline [extratabsurround] \hline  make multiple lines!
\firstline [extratabsurround]\hline
\lastline [extratabsurround]
\lastline [extratabsurround]\hline
```

`\firstline` and `\lastline` are defined in `array.sty` and can be used to preserve the alignment of text, when using horizontal lines. Besides, the optional argument can be used to change (locally) the `\extratabsurround` dimension.

The example of `array` documentation is:

<table>
<thead>
<tr>
<th>Tables with no versus line commands used</th>
<th>Tables with no versus line commands used</th>
</tr>
</thead>
<tbody>
<tr>
<td>tables with some line commands used</td>
<td>tables with some line commands used</td>
</tr>
<tr>
<td>with <code>\firstline</code> and <code>\lastline</code></td>
<td>with <code>\hline</code> (text alignment is not preserved)</td>
</tr>
</tbody>
</table>

Now with `tabu` you can make double, triple (or more) `\firstline` or `\lastline` as in:

#### Top alignment

```
\begin{tabu}{t|c}
with no line commands \ used
\end{tabu}
```

versus tables

```
\begin{tabu}{t|c}
\firstline \hline \hline \hline \\
with some line commands \ used. \\
\lastline \hline \hline \hline \\
\end{tabu}
```

#### Bottom alignment

```
\begin{tabu}{t|c}
with no line commands \ used
\end{tabu}
```

versus tables

```
\begin{tabu}{t|c}
\firstline \hline \hline \hline \\
with some line commands \ used. \\
\lastline \hline \hline \hline \\
\end{tabu}
```

`\firstline` is equivalent to: `\firstline` `\hline` `\hline`

But the optional argument must come in *first position*: `\firstline [extratabsurround]` ...

The same for `\lastline`.

In yellow you can see the `extratabsurround` strut, because `\tracingtabu = 3` for this `tabu`
3.4 More style for lines

\taburulecolor {⟨rule color⟩}
\taburulecolor |{(double rule sep color)}|{⟨rule color⟩}

\taburulecolor sets (in a “locally-global” way) the color to be used for \hline, \firsthline, \lasthline and also vertical lines if the standard line style is used (the standard line style is active after \tabulinestyle {} or after \tabureset {}).

The optional parameter enclosed by vertical bars: |{(double rule sep color)}| is the color to set between two adjacent rules. If not specified, double (or triple...) rules are separated by a vertical space (\vskip).

Here is \taburulecolor{lime}|{DarkSlateBlue}\arrayrulewidth=1mm \doublerulesep=2mm
\begin{tabu}spread 0pt {X[-1]} \firsthline \hline \hline \hline \end{tabu} \text{TabU package} \par
And the next paragraph follows...

And the next paragraph follows...

\tabulinestyle {⟨line style specification⟩}

\tabulinestyle sets the line style for vertical (1) and horizontal lines (\ie \tabucline: \hline, \firsthline etc. are not modified by \tabulinestyle)

The line specification is of the form:

3pt rule color on 4pt dash color off 5pt gap color
rule color on 4pt dash color off 5pt gap color
on 4pt dash color off 6pt gap color
3pt rule color
on 4pt dash color
off 5pt
3pt

Named style defined by \newtabulinestyle

Well... any parameter is optional. Obviously the rule color is the same as the dash color and the former overwrites the latter if both are given.

Your color names can contain spaces but:

- If the first character in the line specification is not a letter, then it is taken as a dimension: the thickness of the line. Otherwise, the default thickness is used \ie \arrayrulewidth.

- Your color names must not contain any series of characters that match one the patterns:
  on
  off?

where ? is a character of category 12, different from ! and possibly preceded by spaces. I don’t think this is a real limitation...

\newtabulinestyle {⟨style=line spec., style=line spec., ...⟩}\text{babel}

This command defines a line style to be used in the first optional argument of \tabucline (horizal lines) or the optional argument of \hline (vertical lines) or with \tabulinestyle (locally-global style).

Style names and color names are babel-protected.

\tabucline [style or spec.]{start-end}

\tabucline is an attempt to give a versatile command to make horizontal lines:

- \tabucline is pretty good with vertical lines even if the thickness of the line grows up,
• \tabucline takes care of \extrarowheight,
• \tabucline can make horizontal dashed lines, with a pgf/TikZ syntax:
  \tabucline [{(width) on(dash) off(gap)}{(first column)-(last column)}]
• alternatively, you can give \tabucline a \hbox to make a leader with it: The \{spec.\}
  must then begin with \hbox, \box or \copy,
• finally you can give \tabucline a color name, after the line specification.

Any parameter can be omitted.

| \tabucline [1pt on 1.5pt off 2pt]{1-4} | draws a horizontal dashed line of width 1pt. Dashes are 1.5pt long and gap width is 2pt. The line is drawn between columns 1 and 4. Here there are only 2 columns and the line stops at column 2. |
|\tabucline [1.5pt]{-} | draws a horizontal solid line of width 1.5pt between the first and the last column. |
|\tabucline {2-} | draws a horizontal solid line of width \arrayrulewidth between the second column and the last one. |
|\tabucline [on 2pt red]{-5} | draws a horizontal dashed line between columns 1 and 5 of width \arrayrulewidth. Dashes are 2pt long and gap width is 4pt (the default). |

Define the line style

\newtabulinestyle {myline=0.4pt on 2.5pt off 1pt red} \tabucline [myline]{-}

Or use a leader or a box to make a leader with it directly in the argument of \tabucline
\tabucline {\hbox {\$\scriptstyle \star \$}}{1-3}

Dashed or dotted

And below is the default

\begin{tabu}{cX[2mc]X}
\everyrow {\tabucline [1pt] -}
\end{tabu}

3.5 Automatic horizontal lines and row colors

\newtabulinestyle {myline=0.4pt on 2.5pt off 1pt red} \tabucline [myline]{-}

\everyrow {code}

\newtabulinestyle {myline=0.4pt on 2.5pt off 1pt red}

\everyrow {code}

\newtabulinestyle {myline=0.4pt on 2.5pt off 1pt red}

\everyrow {code}

\newtabulinestyle {myline=0.4pt on 2.5pt off 1pt red}

\everyrow {code}

\newtabulinestyle {myline=0.4pt on 2.5pt off 1pt red}

\everyrow {code}

\newtabulinestyle {myline=0.4pt on 2.5pt off 1pt red}

\everyrow {code}

\newtabulinestyle {myline=0.4pt on 2.5pt off 1pt red}

\everyrow {code}

\newtabulinestyle {myline=0.4pt on 2.5pt off 1pt red}

\everyrow {code}

\newtabulinestyle {myline=0.4pt on 2.5pt off 1pt red}

\everyrow {code}

\newtabulinestyle {myline=0.4pt on 2.5pt off 1pt red}

\everyrow {code}

\newtabulinestyle {myline=0.4pt on 2.5pt off 1pt red}

\everyrow {code}

\newtabulinestyle {myline=0.4pt on 2.5pt off 1pt red}

\everyrow {code}

\newtabulinestyle {myline=0.4pt on 2.5pt off 1pt red}

\everyrow {code}

\newtabulinestyle {myline=0.4pt on 2.5pt off 1pt red}

\everyrow {code}

\newtabulinestyle {myline=0.4pt on 2.5pt off 1pt red}

\everyrow {code}

\newtabulinestyle {myline=0.4pt on 2.5pt off 1pt red}

\everyrow {code}

\newtabulinestyle {myline=0.4pt on 2.5pt off 1pt red}

\everyrow {code}

\newtabulinestyle {myline=0.4pt on 2.5pt off 1pt red}

\everyrow {code}

\newtabulinestyle {myline=0.4pt on 2.5pt off 1pt red}

\everyrow {code}

\newtabulinestyle {myline=0.4pt on 2.5pt off 1pt red}

\everyrow {code}

\newtabulinestyle {myline=0.4pt on 2.5pt off 1pt red}

\everyrow {code}

\newtabulinestyle {myline=0.4pt on 2.5pt off 1pt red}

\everyrow {code}

\newtabulinestyle {myline=0.4pt on 2.5pt off 1pt red}

\everyrow {code}

\newtabulinestyle {myline=0.4pt on 2.5pt off 1pt red}

\everyrow {code}

\newtabulinestyle {myline=0.4pt on 2.5pt off 1pt red}

\everyrow {code}
\texttt{\taburowcolors \{first line\} \{number\} \{first .. last\}}

\texttt{\taburowcolors} sets the alternate colors to be used on every row of the tabular. The command can be used before a \texttt{tabu} environment or inside it, at the end of a row.

The optional parameter \{first line\} tells the first row from which background colors are starting – this optional parameter has no effect when \texttt{\taburowcolors} is used at the end of a row: background are starting immediately in this case.

\{number\} is the number of colors in the color series. If not specified, it defaults to 2 (for alternate rows color).

Finally \{first\} and \{last\} are the first and the last colors in the color series.

Example:

\begin{verbatim}
\taburowcolors{2}{3}{Crimson!30 .. ForestGreen!40}
\taburulecolor{GreenYellow}{OrangeRed}
\arrayrulewidth=1pt \doublerulesep=1.5pt
\everyrow{\hline \hline}
\begin{tabu} {X[-1]X}
This is & just a test \\
and i think & it will \\
look & rather bad \\
for & i’ve not \\
chosen & the colors \\
with care. & i can’t \\
say & less...
\taburowcolors{2}{Crimson .. ForestGreen}
1 & This is Crimson \\
2 & This is ForestGreen \\
3 & This is Crimson \\
4 & This is ForestGreen \\
\end{tabu}
\end{verbatim}

\texttt{\tabureset}

To go back to “standard” parameters, $\mathtt{\tau b c}$ provides the command \texttt{\tabureset} which basically does:

\begin{verbatim}
\setlength{\tabulinesep}{0pt} \setlength{\extrarowsep}{0pt} \setlength{\extratabsurround}{0pt}
\setlength{\tabulinestyle}{} \setlength{\everyrow}{} \setlength{\taburulecolor}{|{}|}
\end{verbatim}

4 Modifying the font and the alignment in one row: \texttt{\rowfont}

\texttt{\rowfont \{alignment\} \{font specification\}}

Inside a \texttt{tabu} environment, you can modify the font for each cell in a row. \texttt{\rowfont} has priority over column font specification, exactly like \texttt{\rowcolor} (package colortbl) has priority over \texttt{\columncolor}.

The alignment of each cell in one row can also be changed to:

\begin{verbatim}
1 = left \hspace{1cm} \text{or for \texttt{ragged2e} settings:} \hspace{1cm} L
C = center \hspace{1cm} C
r = right \hspace{1cm} R
j = justify \hspace{1cm} J
\end{verbatim}

Any other value for the optional \{alignment\} parameter is silently ignored. If \texttt{ragged2e} is not loaded, L R C and J are synonymous with the lowercase equivalent.
5 Saving and restoring a tabu

\savetabu{(user-name)}

The command \savetabu can be used at the end of any line of a \tabu environment to save the parameters of a \tabu environment. The saving is always global. This allows to easily make tabulars which share exactly the same shape throughout your document. This can also be used as a kind of tabbing environment which is able to remember the tabs positions...

If the \(\langle user-name\rangle\) has been used before, an info is displayed in the .log file and the previous settings are overwritten.

With the \texttt{tracingtabu} > 0, informations about the saved parameters are reported in the .log file.

Recalling saved parameters are done with \texttt{usetabu} (complete recovery) or \texttt{preamble} (partial recovery of the preamble only).

\usetabu{(user-name)}

\texttt{usetabu} is the complement of \texttt{savetabu}: it can be put alone in the \tabu preamble instead of the usual columns specifications to restore any previous settings saved with \texttt{savetabu}.

The \(\langle user-name\rangle\) must exist otherwise, you get an error.

\texttt{usetabu} is a help to make several tabulars of exactly the same shape, same target, same preamble. The only parameter that can be changed is the optional vertical position parameter for the whole tabular.

\texttt{usetabu} does not work with longtabu.

\texttt{usetabu} locally restores:

- the preamble\(^7\).
- the vertical position \([c]\), \([b]\) or \([t]\), unless another position is specified.
- the target width of the \tabu in points: the saved target width does not contain any control sequence: it is fixed and stored in points.
- the width of \tabu \(X\) columns: those widths are not calculated any more – even in the case of negativ coefficients – and \(X\) columns are directly transformed into \(p\), \(m\) or \(b\) columns of the same widths as the ones that where calculated at the time of \texttt{savetabu}
- \texttt{tabcolsep} (or \texttt{arraycolsep} in math mode) \texttt{extrarowheight}, \texttt{extrarowdepth}, \texttt{arraystretch} and \texttt{extratabsurround}
- \texttt{arrayrulewidth}, \texttt{doublerulesep} and the parameters for \texttt{everyrow} \texttt{taburulecolor}, \texttt{tabulinestyle}, and \texttt{taburowcolors}
- \texttt{minrowclearance}, (package \texttt{colortbl})

\texttt{abovetabulinesep} and \texttt{belowtabulinesep} are not restored, because they are related to the content of the tabular rather than to its shape.

Example:

\begin{verbatim}
\tabcolsep=12pt \extrarowsep=\text{mm}
\tabulinestyle{on 1pt ForestGreen}
\end{verbatim}

\(^7\) The complete \texttt{halign-preamble} is restored.
If one day you use \texttt{tabu}, you will have the idea to restore a \texttt{tabu} while modifying its target, or adding new columns... \texttt{savetabu} and \texttt{usetabu} have not been thought for this purpose, and you may have unexpected results.

\preamble{(user-name)}

\preamble can also be used after \texttt{savetabu}. This is a variant of \texttt{usetabu} that locally restores:

- the \texttt{tabu} (or \texttt{longtabu}) preamble.
- the vertical position [c], [b] or [t] (or [l] or [r] for \texttt{longtabu}), unless another position is specified.
- the \texttt{tabu} / \texttt{longtabu} target width, unless another target is specified.

Any other tabular parameter is not restored.

Put \preamble{(user-name)} alone inside the \texttt{tabu} (or \texttt{longtabu}) preamble in place of the usual columns specifications.

\preamble works exactly as if you defined a custom environment for \texttt{tabu}.

\preamble works with \texttt{longtabu}.

Example (continued...):

\begin{tabular}{|XX|X\{c\}|} \hline This & is & \texttt{tabu} & package \hline \end{tabular}

\begin{tabular}{|X\{c\}|} \hline This \texttt{tabu} & package \hline \end{tabular}

\texttt{tabcolsep}, rule colors etc. are not restored from \texttt{savetabu}: the only \texttt{tabu} preamble is restored.

\section{Some other features}

\subsection{Printing numbers inside \texttt{tabu} with \texttt{numprint} and \texttt{siunitx}}

\tabudecimal

\texttt{tabudecimal} provides a \textit{facility} to print numbers inside columns. This facility is not implemented to replace \texttt{siunitx} \texttt{S} and \texttt{s} columns or \texttt{numprint} \texttt{n} and \texttt{N} columns or other packages that provide alignment such as \texttt{warpcol}, \texttt{dcolumn} or \texttt{rccol}. It just make easy to apply a macro you get already on each number in a column of a \texttt{tabu}.

\tabudecimal has been developed mainly because it makes possible to align numbers inside \texttt{tabu} \texttt{X} columns.
\tabudecimal{\langle user-macro\rangle}

\tabudecimal\ can be used in the preamble of a \tabu before a column specification. The \langle user-macro\rangle\ is a macro with one parameter that has to be defined before.

Example with \numprint:\n
\def\usertabu#1{\numprint{\officialeuro}{\zap@space #1 \@empty}}
\nrounddigits{2}\ nprounddigits{0}\ npthousandsep{,}\ npunitseparator{-}

\rowfont{[c]{\bf } January \& February \\}
12.324 \& 745.32 \\ 
21.13 \& 0 \\ 
213.3245 \& 12.342 \\ 
2143.12 \& 324.325 \\ 
\end{tabu}

Example with \SI:\n
\def\usertabu#1{\SI{\langle user-macro\rangle\times[r]}}
\n\begin{tabu}spread 0pt{\!|\[GreenYellow\]\*2{>{\tabudecimal \usertabu}X[\!r]\!|\[GreenYellow\]}}
\rowfont{January \& February \;}
12.32 E/kg & 745.32 E/kg \; \n21.13 E/kg & 0.00 E/kg \; 
213.32 E/kg & 12.34 E/kg \; 
2143.12 E/kg & 324.33 E/kg \; 
\end{tabu}

As you can see, the columns widths are exactly the same, whatever their content.

Here \tabulinesep has been set to 3pt.

You should know how it works...

Yes you should know how it works to avoid problems. \tabu\ has a small scanner based on \futurelet\ to grab all numbers, blank spaces, commas and dots + and – sign and also the letter e and E for exponents. The scanner stops as soon as something else than a number, blank space, comma, dot, +, –, e, E is found, and even if it is a macro that contains a number.

This explains why there is \zap@space in the definition of \usertabu: because the scanner scans blank spaces and because \numprint does not allow blank spaces in its mandatory argument, quite strangely...

6.2 Paragraph indentation

\tabu\ takes care of paragraph indentation when it is used with X columns and its default target, no matter if it has been loaded or not with the \linegoal\ option. Example with \LaTeX\ default: \parindent = 20pt.

This is \tabu\ with its default target in an indented paragraph.

This is \tabu\ with its default target, preceded by \noindent

This is \tabularx\ with target: \linewidth in an indented paragraph.

This is \tabularx\ with target: \linewidth, preceded by \noindent
6.3 delarray shortcuts

When you enclose your tabular with math delimiters using delarray shortcuts, \( \texttt{\textbackslash N b} \) tries to reach its target for the whole: the tabular and the delimiter(s). You can see the difference:

| This is \texttt{\textbackslash tabu} with delarray shortcuts for parenthesis around. | This is \texttt{\textbackslash tabu} with delarray shortcuts for curly brackets around. |
| This is \texttt{\textbackslash tabularx} with delarray shortcuts for parenthesis around. | This is \texttt{\textbackslash tabularx} with delarray shortcuts for curly brackets around. |

Here \texttt{\textbackslash tabulinesep} = 3mm

7 Differences between \texttt{\textbackslash tabu, \textbackslash tabular, \textbackslash tabularx} and \texttt{\textbackslash longtable}

7.1 Paragraph indentation

See Paragraph indentation

7.2 Custom environments

Unlike \texttt{\textbackslash tabularx}, it is possible to define your own environment using \texttt{\tabu}:

\begin{verbatim}
\newenvironment{foo}
  {\begin{tabu}{X[1.2]|[1pt gray]X}}
  {\end{tabu}}
\end{verbatim}

\texttt{\textbackslash tabu} environment, even when X columns are used, may appear in the definition of your custom \texttt{\textbackslash tabular} environment.

You can also use the commands \texttt{\textbackslash savetabu \preamble} (or \texttt{\textbackslash usetabu}) for this purpose.

7.3 Inversion of tokens

When you typeset the following \texttt{\textbackslash tabular}:

\begin{verbatim}
\begin{tabular}{|>{\bfseries}>{ before }l<{ one }<{ two }|}
cell content
\end{tabular}
\end{verbatim}

You get the following result: \begin{verbatim} before cell content two one \end{verbatim} | before cell content two one |

\( \rightarrow \) The word \texttt{before} is not bold, and \texttt{two} comes before \texttt{one}.

The reason is explained in the documentation of \texttt{array.sty}, and is related to the \texttt{array} environment in math mode when using \texttt{\newcolumntype}.

This rather strange inversion of tokens may be justified in math mode (otherwise, errors may occur) but not in text mode in our opinion. Inside a \texttt{\tabu} environment, when not in math mode, the tokens are not reversed and you get the intuitively expected result:

\begin{verbatim} before cell content one two \end{verbatim} | before cell content one two |

In math mode however, tokens are in the reverse order in the \texttt{\tabu} environment like they are in the \texttt{\array} environment.
7.4 Improved process for rewriting columns (for keen readers)

Any tabular that does not split across pages is made with the following process:

initialisation
\hbox
\@array
\begingroup
@mkpream \{preamble\}
\endgroup
\halign \{@preamble \ldots \text{tabular content} \}

For more details, see the Flow chart of expansion.

@mkpream works in two times inside a (semi-simple) group:

First the rewriting process:
Each special column in the tabular preamble is transformed into one the columns defined by array.sty.

Second the building of the \halign preamble:
The “rewritten preamble” is parsed and transformed in a preamble for the \TeX primitive \halign. The result is stored into the @preamble macro.

Any special columns of \texttt{tabu} are defined only inside the “@mkpream” group.

In the following example, you get an error with \texttt{tabular} and no error with \texttt{tabu}. With \texttt{tabular}, and \texttt{siunitx S} column, the rewriting process is as follow:

Inside \texttt{tabular}:
1) Rewrite S: not found because inside {...}
2) Rewrite *
3) Rewrite n column defined by package numprint
Then the ‘n’ in green is rewritten \rightarrow problem

Inside \texttt{tabu}:
1) Rewrite *
2) Rewrite | (there is none here)
3) Rewrite *
4) Rewrite |
5) Rewrite S

The process of rewriting columns is usually longer inside \texttt{tabu} than inside \texttt{tabular}, but conversely \texttt{tabu} with X columns is optimised compared to \texttt{tabularx}, because the preamble is built only once, and not rebuilt before each trial as \texttt{tabularx} does. Thus \texttt{tabu} is much quicker than \texttt{tabularx}.

The process of rewriting is very sensitive to the order in which columns are actually rewritten. This becomes critical when columns are defined with an optional argument like \texttt{tabu X} and | columns or \texttt{siunitx S} column.
8 The package options

8.1 The debugshow package option

\texttt{\textbackslash tracingtabu = 1, 2, 3 or 4}

The control sequence \texttt{\textbackslash tracingtabu} has the same effect as the \texttt{\textbackslash debugshow} option:

- $\tau_\text{abc}$ will report the widths it computes at each attempt to read the target, when X columns are used.
- Saved informations on the \texttt{tabu} are reported in the \texttt{.log} file when \texttt{\textbackslash savetabu} is used.

\texttt{\textbackslash tracingtabu = 2} gives more information on the measures of the natural widths.
\texttt{\textbackslash tracingtabu = 3} shows the struts inserted inside the \texttt{tabu} environment and gives more information about the measures of the height and depth of every row.
\texttt{\textbackslash tracingtabu = 4} displays information on the insertions made by \texttt{\textbackslash tabucline}.

Typical information in the \texttt{.log} file:

\begin{verbatim}
(tabu) Try tabu X tabu Width Target Coefs Update
(tabu) 1) 386.67296pt 797.34592pt 386.67296pt 2.0pt -205.33649pt
(tabu) 2) 181.33647pt 386.67294pt 386.67296pt 2.0pt 0.00002pt
(tabu) 2) Target reached (hfuzz=0.1pt) ****************
\end{verbatim}

What does it mean?

1) The first attempt was performed with $X=386.67296pt$.
The \texttt{tabu} width ($797.34592pt$) exceeded the target by $410.67296pt$.
Thus $X$ has been updated: $410.67296pt/2 = 205.33649pt$ and then:

\[ X = 386.67296pt - 205.33649pt = 181.33647pt \]

2) The second attempt lead to a \texttt{tabu} width of $386.67294pt$: the target is reached.
The final width of each $X$ column is the product of \texttt{tabu X} by its width coefficient.

8.2 The delarray package option

\texttt{delarray} option has the single effect to load \texttt{delarray.sty} for delimiters shortcuts around \texttt{tabu}.
Delimiters shortcuts work both in math and text mode.

8.3 The linegoal package option

With the \texttt{linegoal} option, the default target for \texttt{tabu} with $X$ columns is \texttt{\textbackslash linegoal} instead of \texttt{\textbackslash linewidth}. The \texttt{linegoal} package must be loaded and compilation must be done with \texttt{pdflatex}, otherwise, a warning is displayed and the \texttt{linegoal} option has no effect: the default target remains \texttt{\textbackslash linewidth}. \texttt{linegoal} works with \texttt{pdflatex} in \texttt{pdf} mode and \texttt{in dvi} mode.

If for some reason, you wish to turn down the \texttt{linegoal} option in your document, you can say (in a group for example):

\begin{verbatim}
\let\tabudefaulttarget=\linewidth
\end{verbatim}

In any case, specifying the target overwrites the default:

\begin{verbatim}
\begin{tabu} to\linewidth
\end{tabu}
\end{verbatim}

9 Corrections of some bugs \textit{(available only inside tabu)}

9.1 Correction for \texttt{colortbl} and \texttt{arydshln}: compatibility with \texttt{delarray}

Both \texttt{colortbl} and \texttt{arydshln} forget the control sequence \texttt{\textbackslash arrayright} in their implementation, quite strangely because both of them take care of \texttt{\textbackslash arrayleft}. As a result, \texttt{delarray} shortcuts for delimiters around a tabular does not work if \texttt{colortbl} and/or \texttt{arydshln} are loaded.

Those control sequences are used by the \texttt{delarray} package to put variable size delimiters around
the array:
\begin{tabu} \{X\}
\end{tabu}

\left \{ \begin{tabu}{X}
\end{tabu} \right .

9.2 Correction for \texttt{arydshln}: © columns

A bug in \texttt{\textbackslash adl\%xarraydashrule}: t-arg columns (class 1) and ©-arg columns (class 5) should be treated the same as far as rules are concerned.

With this correction, the “known problem number 1” in \texttt{arydshln} documentation is solved.

10 To do for even better \texttt{tabu}

In decreasing order of priority:

\begin{itemize}
\item Make double \texttt{\tabucline} compatible with \texttt{colortbl} \texttt{\doublerulesepcolor}
\item Multiple \texttt{\tabucline} between different columns: extended specs:
  \texttt{\tabucline [line spec]{start-stop, start-stop}[line spec]{start-stop} ...}
\item Reimplement \texttt{\multicolumn} in order to allow the \texttt{X} token in \texttt{\multicolumn} preamble. Provide \texttt{\multicell} to allow spanning columns and rows at the same time.
\item Presently, \texttt{longtabu} with \texttt{X} columns works only if \texttt{\LTchunksize} is greater than the number of rows. I compiled a \texttt{longtabu} of 56 pages on my PC with \texttt{\LTchunksize} = 2000 without problem. Presently \texttt{\LTchunksize} is set to 10000 during trials when \texttt{longtabu} contains \texttt{X} columns.
\item Make \texttt{\tabucline} work with page breaks (one line on the top of the page, one line on the bottom of the previous).
\end{itemize}
11 Technical notice and Implementation

11.1 Drawing a tabular - The $\mathcal{T}_R$ approach

$\mathcal{T}_R$ has a different approach than almost any other package providing facilities for tabulars. \texttt{colortbl} and \texttt{arydshln} both put the cells contents into a box for measuring purpose, and then use the dimensions of each box to make their setups:

- \texttt{colortbl} needs the dimensions of the box to put a rule in the background of the cell,
- \texttt{arydshln} needs the dimensions to set the length of its leaders (dash lines).

This is achieved by modifying the macros defined in \texttt{array.sty} to insert columns inside the \texttt{\halign} preamble. Instead, $\mathcal{T}_R$ proceeds as follow:

1. It first measures (if there are some negative width coefficients, or if \texttt{tabu spread} is used) the natural widths of the cells / the columns,
2. Then it always measures the height and depth of each cell / row,
3. Thereafter, the tabular is printed exactly as if \texttt{array.sty} was entitle to print it: no “extra” boxing of the cells material. The measurements have been stored and can be used to set the struts (only one per row) and the lengths of vertical leaders.
4. No macros of \texttt{array.sty} is modified at stage 3.

$\mathcal{T}_R$ material inserted in the tabular for vertical leaders, \texttt{\rowfont} etc. is put inside the special “free” tokens provided by \texttt{array.sty}:

- A vertical leader is put inside a ! column: !{\texttt{vertical leader}}
- Changing font and alignment in one row requires some setup in > tokens: >{\texttt{rowfont material}}.

This way, the commands of \texttt{array.sty} that build each column definition (or preamble, in the sense of \texttt{\halign}) are never modified.

11.2 Algorithms

tabu to target

The algorithm of $\texttt{\tabu@arith}$ computes the desired widths to reach the target. In any case, only one measure of the tabular is required to get the widths for all columns. Here we describe the method with an example and some equations too to show that this handle all cases in generality.

Notations and initialisation of $X$

In the case of \texttt{tabu to the target} $T = 300$ is given : it is the target specified by the user or the default \texttt{tabu} target which is $\texttt{\linewidth} − \texttt{(parindent correction)}$ or $\texttt{\linegoal}$. Each $X$ column has a width coefficient which is given too (or default to 1). The coefficients are: $c_1, c_2, \ldots, c_n$.

$X$ is the main dimension that drives the widths of all columns with a non negative coefficient, and limit the widths of columns with a negative coefficient.

Then we have first:

<table>
<thead>
<tr>
<th>Coef $c_i$</th>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
<th>$c_4$</th>
<th>$c_5$</th>
<th>$c_6$</th>
<th>$\sum$</th>
<th>$\Delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target $T$</td>
<td>300</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Some coefficients are negative and we have to measure the natural widths of the corresponding columns, for columns always have a width:

$$\lambda_i = \begin{cases} 
  c_i \cdot X & \text{if } c_i > 0 \\
  \text{Min} ( |c_i| \cdot X, \nu_i) & \text{if } c_i < 0 \text{ with } \nu_i \leq T \quad \forall i
\end{cases}$$

$\nu_i$ is the “natural width of the column” in the sense that it is the maximum of the natural widths of each cell in the $ith X$ column, limited to the $\texttt{tabu}$ target: $\nu_i \leq T \quad \forall i$. 
The whole width of the tabular is always:

\[ \text{wd(table)} = \sum_i \lambda_i + \text{incompressible material} \]

- \tabcolsep
- vertical lines/leaders thickness
- non X columns

and should finally be equal to \( T \), by the correct computation of the \( \lambda_i \).

So what is \( X \) at first? Columns that have a non negative coefficients always have a width equal to \( \lambda_i = c_i \cdot X \) therefore, if we only have non negative coefficients, we can safely set:

\[ X = \frac{T}{\sum_i c_i} \]

then: \( \sum_i \lambda_i = \sum_i c_i \cdot X \geq T \) at the first trial. But this is not the same if some coefficients are negative, because in this case the column width \( \lambda_i \) can shrink until its natural width \( \nu_i \) and may be until to 0 pt! And then if every column has a negative coefficient, one of them can have a width close to the target \( T \). We have to ensure that the first measure of the natural widths does not limit them artificially:

\[ \forall i \ c_i < 0 \implies |c_i| \cdot X \geq T \]
\[ \exists c_i > 0 \implies \sum_{c_i > 0} c_i \cdot X \geq T \]

And finally, for the measure: \( X = \max \left[ \frac{T}{\sum_{c_i < 0} |c_i|}; \frac{T}{\sum_{c_i > 0} c_i} \right] \)

<table>
<thead>
<tr>
<th>Coef ( c_i )</th>
<th>( c_1 )</th>
<th>( c_2 )</th>
<th>( c_3 )</th>
<th>( c_4 )</th>
<th>( c_5 )</th>
<th>( c_6 )</th>
<th>( \sum )</th>
<th>( \Delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target ( T )</td>
<td>300</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( X )</td>
<td>300</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \nu_i )</td>
<td>10</td>
<td>300</td>
<td>80</td>
<td>80</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \lambda_i )</td>
<td>10</td>
<td>300</td>
<td>80</td>
<td>80</td>
<td>600</td>
<td>900</td>
<td>1970</td>
<td>1800</td>
</tr>
</tbody>
</table>

**First step of the algorithm: reducing the width** After having measured the table we get: \( \text{wd(table)} = 2100 \). The incompressible material is \( 2100 - 1970 = 130 \) wide and the gap to the target is \( \Delta = 2100 - 300 = 1800 \).

We now choose a new value for \( X \):

\[ \sum_i \lambda_i = \sum_i \min (\nu_i; c_i \cdot X) + \sum_i c_i \cdot X \leq \sum_i |c_i| \cdot X \]

Let’s try \( X' = \frac{\sum \lambda_i - \Delta}{\sum_i |c_i|} \) so that \( \sum \lambda_i' \leq \sum_i |c_i| \cdot X' \leq \sum \lambda_i - \Delta \):

<table>
<thead>
<tr>
<th>Coef ( c_i )</th>
<th>( c_1 )</th>
<th>( c_2 )</th>
<th>( c_3 )</th>
<th>( c_4 )</th>
<th>( c_5 )</th>
<th>( c_6 )</th>
<th>( \sum )</th>
<th>( \Delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target ( T )</td>
<td>300</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( X )</td>
<td>300</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \nu_i )</td>
<td>10</td>
<td>300</td>
<td>80</td>
<td>80</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \lambda_i )</td>
<td>10</td>
<td>300</td>
<td>80</td>
<td>80</td>
<td>600</td>
<td>900</td>
<td>1970</td>
<td>1800</td>
</tr>
<tr>
<td>( X' )</td>
<td>11.33</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( X' = \frac{1970 - 1800}{15} = \frac{170}{15} = 11.33 \) \( X' \) Note that the computation of \( X' \) does not involve any measurement.
Here we are in the case where the table width:

\[
\text{wd(table)} = \sum \lambda_i + \text{incompressible material} = T + \Delta
\]

\[
\Rightarrow \sum \lambda'_i + \text{incompressible material} \leq \sum \lambda_i - \Delta + I = T
\]

Without any measure, we can say that the final table width will be less than the target, if we choose \(X'\). The free space to share among the \(X\) columns (computed with \(X'\)) is now \(\Delta' = T - (\sum \lambda'_i + I) = 300 - (168.67 + 130) = -1.33\), where \(I\) is the incompressible material.

**Giving space** We say that a column is *saturated* (i.e., full) if its natural width is greater than \(|c_i| \cdot X\), or all the same that \(\lambda_i < \nu_i\). We also will consider that the columns with \(c_i > 0\) have a “natural width” which is always equal to \(c_i \cdot X\); in other words, a column with a non negative coefficient is always saturated.

Giving space (or “refunding” space) to the columns must be done in priority to the saturated columns. If all columns are finally underfull, then we will distribute the extra space to each, according a distribution rule. But this case can only occur if \(\forall i \ c_i < 0\) because we first chose \(X\) so that:

\[
X \geq \frac{T}{\sum_{c_i > 0} c_i}
\]

and hence, the sum of the widths of the “non negative” columns exceeds the target.

Let’s rank the columns widths:

\[
\lambda'_1, \lambda'_2, \lambda'_3, \nu_4
\]

we first give space
to the saturated columns
1, 2 and 3

Because of the saturation, the total amount of space to give: \(|\Delta'|\) shall be shared among the columns according to their widths coefficients. We shall not give too much space: the columns shall remain saturated. Let \(0 < \epsilon \leq |\Delta'|\) the amount of space to give, then after the operation:

\[
\lambda_1'' + \lambda_2'' + \lambda_3'' = \lambda'_1 + \lambda'_2 + \lambda'_3 + \epsilon
\]

\[
= |c_1| \cdot X' + |c_2| \cdot X' + |c_3| \cdot X' + \epsilon
\]

Let’s say \(X'' = X' + \frac{|\Delta'|}{\sum |c_i|_{saturated}}\) then it’s possible, without any any measure, to compute:

\[
\sum_{c_i \text{saturated}} \lambda''_i + \nu_4 \leq \sum_{c_i \text{saturated}} |c_i| \cdot X'' + \nu_4 \leq \sum |c_i| \cdot X' + \Delta' = \sum \lambda'_i + \Delta' \leq T - I
\]
Or for clarity: \( \sum \lambda_i'' + I = \text{wd}(\text{table}) \leq T \) and the new free space to share is now:

\[
\Delta'' = \left| T - \left( \sum \lambda_i'' + I \right) \right|
\]

At each step of the computation, and without any measure but the first, \( X \) grows, \( \Delta \) decreases, and finally the target is reached for \( X \) such that \( \Delta \leq \text{hfuzz} \).

<table>
<thead>
<tr>
<th>Coef ( c_i )</th>
<th>( c_1 )</th>
<th>( c_2 )</th>
<th>( c_3 )</th>
<th>( c_4 )</th>
<th>( c_5 )</th>
<th>( c_6 )</th>
<th>( \sum \Delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target ( T )</td>
<td>300</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>( X )</td>
<td>300</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \nu_i )</td>
<td>10</td>
<td>300</td>
<td>80</td>
<td>80</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \lambda_i )</td>
<td>10</td>
<td>300</td>
<td>80</td>
<td>80</td>
<td>600</td>
<td>900</td>
<td>1970</td>
</tr>
<tr>
<td>( X' )</td>
<td>11.33</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \lambda_i' )</td>
<td>10,00</td>
<td>22,67</td>
<td>56,67</td>
<td>22,67</td>
<td>22,67</td>
<td>34,00</td>
<td>168,67</td>
</tr>
<tr>
<td>( X'' )</td>
<td>11.43</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \lambda_i'' )</td>
<td>10,00</td>
<td>22,86</td>
<td>57,14</td>
<td>22,86</td>
<td>22,86</td>
<td>34,29</td>
<td>170,00</td>
</tr>
</tbody>
</table>

Now if the width of the table is less that the target, because 1) every column has a negative coefficient and 2) their natural widths are so small than the tabular don’t fill the wanted horizontal space, the algorithm artificially raise the natural widths, according to a linear distribution:

\[
\lambda_i' = \lambda_i + \Delta \cdot \frac{\lambda_i}{\sum \lambda_i} = \nu_i + \Delta \cdot \frac{\nu_i}{\sum \nu_i} = \nu_i \cdot \left( 1 + \frac{\Delta}{\sum \nu_i} \right)
\]

**tabu spread dimen**

The case of **tabu spread** is interesting and quite complex...

Here, the aim of the game is to give a target to the table, depending on its natural width. **tabu** has a default target (**\\linewidth** in general, but it is possible to **\let \tabudefaulttarget** to another value... for example **\lingoal**) which is a maximum for the final target of **tabu spread**. The case where the spread is 0pt is not simpler nor more difficult.

If every column has a negative coefficient, it’s rather easy because either the table exceeds the target, and then the new target will be the default target (the maximum), or the table width is less than the default target and we fix the new target to be that width + the spread, in the limit of the default target.

The condition that must hold on coefficient is not restrictive if every column has a negative coefficient because if you say, for example:

\[
X = \text{Max}_{i} \frac{\nu_i}{|c_i|}
\]

then:

\[
\sum_i \lambda_i = \sum_i \text{Min} (\nu_i; |c_i| \cdot X)
\]

is true. It’s always possible to find a \( X \) such that the behaviour annouced in the documentation is observed !

Then let’s get some non negative coefficients. The natural widths of such columns must be measured, but the natural width of the tabular is not the same, for the proportions between column widths – expressed by their positive coefficient \( c_i \) – must be respected.

The real natural width of the tabular, which observe the proportions between columns with a non negative coefficient is:

\[
wd(\text{table}) + \text{Max}_{c_i \geq 0} \left( \frac{\nu_i}{c_i} \right) \times \sum_{c_i > 0} c_i - \sum_{c_i > 0} \nu_i > wd(\text{table})
\]
This quantity is computed, $\tau_{\mathcal{B}}b\mathcal{C}$ adds the spread and fix the new target to the sum, in the limit of the default target.

Then $X$ is initialized such that: $X = \operatorname{Max} \left[ \frac{T}{c_{i} < 0} \frac{T}{\sum_{c_{i} > 0} c_{i}} \right]$

and the algorithm described in the former section works, without any new measurement of the tabular. Unless this was not possible or deemed inconvenient for clarity, the code is presented in the same order it executes.

11.3 The tabu strategies

<table>
<thead>
<tr>
<th>Not nested (outer)</th>
<th>Nested</th>
</tr>
</thead>
<tbody>
<tr>
<td>\count@ condition</td>
<td>\count@ condition</td>
</tr>
<tr>
<td>\tabu@endrewrite</td>
<td>3 no X column</td>
</tr>
<tr>
<td>4 X columns</td>
<td>1 no X column</td>
</tr>
<tr>
<td>0 outer is in mode 0</td>
<td>3 X column</td>
</tr>
<tr>
<td>1 outer in mode 0 =&gt; print</td>
<td>2 from 1 in \tabu@endrewrite if outer in mode 4</td>
</tr>
<tr>
<td>2 outer in mode 3</td>
<td>3 or 4 needs trials</td>
</tr>
<tr>
<td>3 or 4 needs trials</td>
<td>4 Horizontal measure (nested in coef&lt; 0 or spread)</td>
</tr>
<tr>
<td>0 print out</td>
<td>1 Exit in vertical measure (outer in mode 3)</td>
</tr>
<tr>
<td>3 Vertical measure</td>
<td>2 Exit with a rule (outer in mode 4)</td>
</tr>
<tr>
<td>4 Horizontal measure</td>
<td>4 Horizontal measure</td>
</tr>
</tbody>
</table>

Path followed by the outermost tabu

Yes

X column?

No

4 4 4 3

Path followed by a nested tabu

Yes

X column?

No

4 4 4 1 (outer in mode 3) 2 (outer in mode 4)

1 (outer in mode 3) 2 (outer in mode 4)
11.4 Identification and Requirements

\( \mathbb{T}_{\mathcal{b}b} \) requires \texttt{array.sty} and \texttt{varwidth.sty}. The package namespace is \texttt{tabu@}.

\begin{itemize}
  \item \texttt{ProvidePackage(tabu)[2011/02/26 v2.8 - flexible LaTeX tabulars (FC)]}
  \item \texttt{RequirePackage(array)[2008/09/09]}
  \item \texttt{RequirePackage(varwidth)[2009/03/30]}
\end{itemize}

Minimal catcode acertaining for loading \( \mathbb{T}_{\mathcal{b}b} \) in good conditions:

\begin{itemize}
  \item \texttt{AtEndOfPackage[\tabu@AtEnd \let\tabu@AtEnd \@undefined]}
  \item \texttt{def \tabu@AtEnd[\tabu@AtEnd}
  \item \texttt{\catcode\1 \the\catcode\1}
  \item \texttt{\catcode\1=1}
  \item \texttt{\catcode\33=12}
  \item \texttt{\catcode\58=12}
  \item \texttt{\catcode\124=12}
  \item \texttt{\catcode\36=3}
  \item \texttt{\catcode\38=4}
  \item \texttt{\catcode\32=10}
  \item \texttt{\catcode\94=7}
  \item \texttt{\catcode\95=8}
\end{itemize}

11.5 Flow chart of expansion

\begin{itemize}
  \item \texttt{begin \{tabu\}} \texttt{(\begingroup)}
  \item \texttt{\tabu} \texttt{\rightarrow} \texttt{\{ sets \tabu@target}
  \item \texttt{\tabu} \texttt{\rightarrow} \texttt{\{ loads \tabu setup and prepares \tabu@cleanup and \tabu@reset}
  \item \texttt{\tabu@begin}
  \item \texttt{\if \tabu@target > 0 then prepare}
  \item \texttt{@extracolsep \{\@flushglue \}}
  \item \texttt{creates the \@arstrutbox}
  \item \texttt{\begingroup}
  \item \texttt{\@mkpream: parse the preamble to create \@preamble}
  \item \texttt{\xdef \@preamble macro}
  \item \texttt{\endgroup}
  \item \texttt{\@arrayleft}
  \item \texttt{\vtop or \vbox or \vcenter \bgroup... \halign \{ preamble...}
  \item \texttt{\end \{tabu\}}
  \item \texttt{\if in math mode}
  \item \texttt{\endtabu \rightarrow \endtabular \rightarrow \endarray \rightarrow}
  \item \texttt{\if in text mode}
  \item \texttt{\endgroup end of envir}
\end{itemize}
The important part of the job is made inside the dashed box above: `\@mkpream` expands the columns definitions, which can be user defined. Hopefully, it does its job inside a group, therefore a user-column can set a macro to be expanded `\aftergroup`. This implementation allows many modifications in the tabular preparation, without any change in the macros of `array.sty`.

\begin{quote}
\begin{verbatim}
\begingroup \@mkpream
\{ Finds a X column \}
\begin{array}{l}
\text{Parse the optional parameter for X}
\text{Set \aftergroup \tabu@prep@TRIAL}
\text{\dots \@mkpream \dots \text{\textit{... builds the \halign preamble}}}
\end{array}
\xdef \@preamble
\endgroup
\end{verbatim}
\end{quote}

\textbf{tabu spread with X column}

In the case of “tabu spread” with X columns, the process is the same as the one described for “tabu to” with X columns. However, the first trial is different because we have first to measure the natural width of the tabular. The process is the following:

- `\tabu@target` is first set to `\linewidth` (or `\linegoal` with the `linegoal` package option).
- The X column corresponds to a `\vbox` with `\hsize` fixed to `\tabu@target`.
- Inside this `\vbox` the cell content is written into a `\hbox` whose width is limited to `\tabu@target`. This `\hbox` is captured into the box register `\tabu@box`.
- At the end of the cell, the `\badness` of the `\hbox` is checked:
  - if the `\badness` is > 1000 then the text is too long and “tabu spread” is useless: tabu to `\tabu@target` give the same result.
  - Otherwise, we get the natural width of the cell content by:
    \begin{verbatim}
    \setbox \tabu@box \hbox \{\unhbox \tabu@box\}
    \end{verbatim}
- At the end of the first trial, `\tabu@spreadarith` checks if:
  \[
  \text{width(tabular) + spread < \linewidth (or \linegoal)}
  \]
  - if not, then tabu to `\tabu@target` give the same result
  - Otherwise, the target for tabu to will be:
    \[
    \text{width(tabular) + spread} - \sum_i \text{natural widths } X_i + \text{Max}_i \left( \frac{\text{natural width } X_i}{\text{coef}_i} \right) \times \sum_i \text{coef}_i
    \]
    minimal natural width that can be obtained with the given coefs

And the next trial will be done as if the user called “tabu to” with this target.
11.6 Some constants

Here we define the constants used by $\mathbb{T}_{\mathbb{R}^{\mathbb{C}}}$: \TeX registers and a few helper macros.

When working inside a tabular (i.e. `\halign`) each cell is a \TeX group. Probably the most important property of each register defined here is whether it is global or not. A local register does not suffer, never, any global assignment.

\TeX registers

**\taburow** \TeX counter that globally stores the value of the current row. It is updated at `\everyrow`, rather than at `\everycr`. `\thetaburow` expands to the (arabic) number.

This counter can be read by the user, but she **must not change its value** because it is used internally to store the height/depth of every row, for vertical spacing adjustment (and vertical leaders).

**\tabu@nbcols** \TeX counter that – locally – saves the total number of columns of the \tabu. Special \@ and \! columns are not counted (they are not real columns for `\halign`, but only insertions into the preamble).

The value is used by `\tabucline` to ensure that the leader does not jut out over the last column...

**\tabu@cnt** \TeX counter that – locally – stores the number of trials. Incidentally, it is also temporarily used to parse the width coefficient for \(X\) columns, during the rewriting process.

**\tabu@Xcol** \TeX counter that – locally – stores the number of \tabu \(X\) columns. Defined while rewriting the \(X\) token, it is used in the specification of the width of the column `{\tabu@hsize \{Rank of the X column\}\{coef\}}`.

It is also used to store the natural width of \(X\) columns (in the cases of a negative coefficient or if \tabu spread is used).

**\tabu@alloc** A global counter whose initial value (\(-1\)) is incremented for each nested \tabular. The end of the outermost \tabular globally resets the value to \(-1\). `\tabu@nested` stores locally the value of `\tabu@alloc` and is therefore the “index” of the current \tabular (the one that is actually in construction).

This influences the initialisation process (cf. `\tabu@setup` and `\tabu@init`).

**\tabu@start** They are used locally by `\tabucline` and `\everyrow` while parsing the parameters: this is, for clarity, the local name for `\@tempcnta` and `\@tempcntb`.

\begin{verbatim}
21 \newcount \c@taburow \def\thetaburow \{\number\c@taburow\}
22 \newcount \tabu@nbcols
23 \newcount \tabu@cnt
24 \newcount \tabu@Xcol
25 \let\tabu@start \@tempcnta
26 \let\tabu@stop \@tempcntb
27 \newcount \tabu@alloc \tabu@alloc\=\m@ne
28 \newcount \tabu@nested
29 \def\tabu@alloc@{\global\advance\tabu@alloc \@ne \tabu@nested\tabu@alloc}
\end{verbatim}

**\tabu@target** \TeX dimen that – locally – stores the \tabu target (either “to” or “spread”).

**\tabu@spreadtarget** \TeX dimen that – locally – stores the \tabu spread given by the user.

**\tabu@naturalX** \TeX dimen that – globally – stores the total natural widths of the \(X\) columns, in the cases of negative coefficients and/or \tabu spread. The value is reset to 0pt at `\everyrow`, and maxima/minima are stored into the macro `\tabu@naturalXmin` and `\tabu@naturalXmax`: those are required for the algorithm of \tabu spread `{\tabu@spreadarith}`.

**\tabucolX** \TeX dimen that – locally – stores the width corresponding to the preamble token \(X[1]\): the standard width of \(X\) columns.

**\tabu@DELTA** This is for clarity, the local name of `{\@tempdimc}` in `{\tabu@arith}`.

**\tabu@thick** They are used locally by `\tabu@getline`, while parsing the parameters for a line specification. This is for clarity, the local name for `{\@tempdimc}`, `{\tempdimb}` and `{\tempdimc}`.

8. Package `xcolor` defines the `\rownum` \TeX counter, which is globally updated at `\everycr`. Hence this `\rownum` counter is not reliable in case the user invokes `\cline` or `\cmidrule` for example...
\tabuxsum \TeX\ dimen that – locally – stores the sum of all width coefficients for \textit{X} columns. This is required to fix the initial value for \texttt{\tabucolX} and then in the algorithms (\texttt{\tabuarith} and \texttt{\tabuarithnegcoef}).

\extrarowdepth \texttt{array.sty} defines \texttt{\extrarowheight} as a \TeX\ dimen register: the extra height to be finally added to each row of a table. \texttt{\tabb<} defines \texttt{\extrarowdepth} in addition: the \textit{extra depth}. Though \texttt{\extrarowheight} and \texttt{\extrarowdepth} can be set by the user, the official interface is \texttt{\extrarowsep}.

\abovetabulinesep \belowtabulinesep \TeX\ dimensions \texttt{\abovetabulinesep} and \texttt{\belowtabulinesep} store the minimum allowed vertical space between the contents of the cells and their borders. Their values are ignored if non positive. Though they can be set by the user, the official interface is \texttt{\tabulinesep}.

The philosophy and the technics are similar to the one provided by the \texttt{cellspace} package. However, limitations of \texttt{cellspace} are lifted (nested \texttt{tabu} environments, use of colors... see the \texttt{cellspace} limitations in the revision history). \texttt{\tabb<} inserts only one strut per line, whose name is \texttt{\@arstrut}.

\tabustrutrule The \TeX\ dimen \texttt{\tabustrutrule} is here only for debugging purpose: its value must be 0 pt. It behaves mostly like \TeX\ primitive \texttt{\overfullrule}, and allow to see the struts introduced in the tabular, and to control vertical spacing. Setting \texttt{\tabustrutrule} to a positive value has no effect unless \texttt{\tracingtabu} is \geq 3. The official interface is \texttt{\tracingtabu} = 3.

\tabuxthebody This token stores – locally – the collected content of the \texttt{tabu} environment during the measuring process.

\tabu@footnotes Token that globally stores the footnotes inside the \texttt{tabu} environment, for \texttt{\insert} does not work inside such a level of groupings...

\tabuxbox Stores – loally – the whole \texttt{tabu} when an attempt to adjust \texttt{X} columns is performed.

\tabu@arstrutbox While the \texttt{\@arstrutbox} may redefined globally at the end of each line (for vertical spacing adjustment), we define a new box and \texttt{\let \@arstrutbox} to be that box inside the \texttt{tabu} environment.

Hence, the \texttt{\@arstrutbox} used by other tabular environment does not suffer any modification.

\tabu@hleads \tabu@vleads Those boxes are used to built horizontal and vertical leaders. In order not to rebuild the boxes every time a leader is inserted, the box is globally defined if a line style is specified (via \texttt{[line style]} or \texttt{\tabucline [line style]...} or \texttt{\tabulinestyle \{line style\}}).
Switches

\iftabu@colortbl

The global switch \iftabu@colortbl is used by \rowfont when modifying the alignments, because colortbl changes the glues put inside the \halign preamble to make standard alignments. This switch is set At Begin Document.

\iftabu@siunitx

Global switch set \AtBeginDocument. true if siunitx package is detected.

\iftabu@measuring

This switch is somewhat magic in the sense that it has several meanings... It is temporarily set to true by \tabu@arith in the trial group, to say that the tabu did not reach its target yet. It is also set to true in the \@mkpream group when the first X column is encountered in the preamble. Finally, it is true in the trialS group when the outermost tabular is in strategy number 2 or number 3.

\iftabu@spread

A switch whether “tabu spread” is used or not. A nested tabu inside a X column whose coefficient is negative has a default target set to spread 0pt.

\iftabu@negcoef

A switch set to true in case of negativ coef (natural width if less than X[coef]).

\iftabu@everyrow

A very important global switch: true when outside any tabu environment, true as well when inside a cell of a tabu, but globally set to false at \everycr and therefore inside any \noalign command. This allows to insert leaders (by \omit \span \omit \cr \noalign {...}) or first/last line corrections only once, even if \everycr is executed more than once.

\iftabu@long

Finally the switch \iftabu@long is set to true inside longtabu and to false inside tabu. This is convenient because some setup are slightly different between tabu and longtabu.

\iftabuscantokens

\iftabuscantokens is the switch for whether or not tabu will use \scantokens. Though the user can set \iftabuscantokens to \iftrue or \iffalse, the official interface is tabu*.

It does not make sense to use \scantokens in a nested tabu: only the outermost tabu can use \scantokens, for the environment body must be collected with care!

\iftabuscantokens is the helper macro for scanning tokens.

\iftabu@rescan

Two macros which are needed when scanning tokens with \futurelet.

This gobbles the character number 10 in ASCII (~J in \TeX).

Checks if the current environment is tabu or longtabu (for \multicolumn inside tabu).

Computes the modulo (for \taburowcolors). The method is taken from H.O. intcalc package.
Trimming spaces at low cost...

Sanitize an argument (babel compliant).

The character | may have a special category code inside the document, depending on the language setting or for example, | can be the delimiter shortcut for verbatim. We use \scantokens to allow an \ifx test even if the category code of | changes along the compilation.

Commands like \everyrow, \taburulecolor, \tabulinestyle, \taburowcolors can be expanded either in a cell or outside a \tabu environment or at the end of a row, inside a \noalign group.

To avoid the insertion of an empty math atom (equivalent to \hbox to0pt{}) we open a semi-simple group rather than a math group if not in \noalign. \toks is used to define the local-to-the-TeX-group setting (post-fixed by @L).
Rebuild the \arstrutbox
\tabu@arstrut
\tabu@rearstrut

The macros rebuilds the \arstrutbox (a \hbox). With the debug variants when \tracingtabu = 3 and \tabustrutrule > 0.

\def\tabu@arstrut {
\global\setbox\@arstrutbox
\hbox{\vrule
height \arraystretch \dimexpr\ht\strutbox+\extrarowheight
\depth \arraystretch \dimexpr\dp\strutbox+\extrarowdepth
\width \z@}%%
}\tabu@arstrut

\def\tabu@rearstrut {
\@tempdima \arraystretch\dimexpr\ht\strutbox+\extrarowheight \relax
\@tempdimb \arraystretch\dimexpr\dp\strutbox+\extrarowdepth \relax
\ifodd \ifdim \ht\@arstrutbox=\@tempdima
\ifdim \dp\@arstrutbox=\@tempdimb 0 \fi\fi
\tabu@mkarstrut
\fi
}\tabu@rearstrut

This is the “debug” version of \tabu@arstrut: used when \tracingtabu = 3 or more to show the struts inserted in the tabular.

\def\tabu@DBG #1{
\ifdim \tabustrutrule>\z@ \color{#1}\fi
}\def\tabu@DBG@arstrut {
\global\setbox\@arstrutbox
\hbox to\z@{\hbox to\z@{\hss
{\tabu@DBG{cyan}\vrule
height \arraystretch \dimexpr\ht\strutbox+\extrarowheight
\depth \z@
\width \tabustrutrule}
kern-\tabustrutrule
{\tabu@DBG{pink}\vrule
height \z@
\depth \arraystretch \dimexpr\dp\strutbox+\extrarowdepth
\width \tabustrutrule}}}%
\tabu@DBG@arstrut

\def\tabu@save@decl{
\toks\count@ \expandafter{\the\toks\expandafter\count@}
\@nextchar}
\def\tabu@savedecl{
\ifcat$\d@llarend
\else
\let\save@decl \tabu@save@decl \fi % no inversion of tokens in text mode
}\tabu@savedecl

No inversion on tokens in the tabu preamble, when not in math mode.
\def\tabu@finalstrut #1{\unskip\ifhmode\nobreak\fi\vrule height\z@ depth\z@ width\z@}

Disable some commands during trials
\def\tabuDisableCommands {
\g@addto@macro\tabu@trialh@@k }
\let\tabu@trialh@@k \@empty
\def\tabu@write \write
\let\tabu@immediate \immediate
\def\tabu@WRITE{
\begingroup
\def\immediate\write{\aftergroup\endgroup
A trick (from the Ti\TeX-book) to forbidd \write when a trial is done on the \halign.
\def\tabu@nowrite #1#{\afterassignment}\toks@}
\def\tabu@immediate\write{\begingroup
\let\immediate\write \aftergroup\endgroup
\let\immediate\write
\let\immediate\write}

Disable footnotes during trials.
\def\tabu@nowrite #1#{\afterassignment}\toks@}
\let\tabu@immediate\write
\let\tabu@immediate\write
\let\tabu@immediate\write
\let\tabu@immediate\write
\let\tabu@immediate\write
For optimisation purpose, color changes are deactivated during trials, for they do not affect the measures.

siunitx S and s columns management

A macro that encloses the definition of `\tabu@celllalign`, in order to check if the column is a `siunitx` S (or s) column, and neutralise the setup of `\rowfont` in this case, for `siunitx` provides its own `key=value` options to set fonts inside S (or s) columns.

The macro may be prefixed by `\global`.

\extrarowsep makes the assignment for both `\extrarowheight` and `\extrarowdepth`.

The macro may be prefixed by `\global`.

\extrarowsep \extrarowsep makes the assignment for both `\extrarowheight` and `\extrarowdepth`.\extrarowsep makes the assignment for both `\extrarowheight` and `\extrarowdepth`.
The macro may be prefixed by \texttt{global}.

Utility macros to implement the possibility to prefix a macro by \texttt{global}.

\begin{verbatim}
\def\tabu@Gsave #1#2#3#4{\xdef#1{#1\toks#2\the\currentgrouplevel{\global#3\the#3\global#4\the#4}}}
\def\tabu@Grestore#1#2{\toks#2{}#1\toks\currentgrouplevel\expandafter{\expandafter}\the\toks#2\relax
\ifcat$\the\toks\currentgrouplevel$\else\global\let#1\@empty \global\let#2\z@\the\toks\currentgrouplevel\fi}
\end{verbatim}

Setting code for every row

As long as \texttt{tabu} needs to execute some code at \texttt{\everycr}, it's not difficult to provide a command to give the user the opportunity to execute its own arbitrary code. However, \texttt{\everyrow} will be used almost only with \texttt{\hline} (or \texttt{\tabucline} or \texttt{\midrule}).

\texttt{\everyrow} can be changed anywhere inside the \texttt{tabu}: at the end of a row, or even inside a cell.

The settings are saved in a “locally-global” way...

\begin{verbatim}
234 \newcommand\everyrow[1]{
235 \tabu@start \z@ \tabu@stop \z@ \tabu@evrstartstop
236 }% \everyrow
237 \def\tabu@evrstartstop {
238 \afterassignment \tabu@evrstartstop \tabu@stop=
239 \ifx \@let@token
240 \afterassignment\tabu@everyr@w \toks@
241 \else \afterassignment\tabu@everyr@w \toks@
242 \fi}
243 }% \tabu@evrstartstop
244 \def\tabu@everyr@w {%
245 \xdef\tabu@everyrow{%
246 \noexpand\tabu@everyrowfalse
247 \let\noalign \relax
248 \noexpand\tabu@rowfontreset
249 }% \tabu@rowfontreset
250 \let\noexpand\tabu@doclinestyle \noexpand\tabu@docline@evr
251 \let\noexpand\tabu@docline \noexpand\tabu@docline@evr
252 \let\noexpand\tabu@docline@evr {
253 \ifodd 1
254 \else \expandafter\def\expandafter\tabu@docline@evr\expandafter{\the\toks@}
255 \fi
256 }% \tabu@docline@evr
257 \ifodd 1
258 \else \tabulinestyle{\the\toks@}
259 \fi
260 }% \tabu@everyrowfalse
261 \def\tabu@evrline{
262 \tabulinestyle{\the\toks@}
263 \fi
264 }% \tabulinestyle
265 \def\tabulinestyle[1]{
266 \expandafter\def\expandafter\tabulinestyle\expandafter{\the\toks@}}%
267 \def\tabulinestyle[2]{
268 \expandafter\def\expandafter\tabulinestyle\expandafter{\the\toks@}}%
269 \def\tabulinestyle[3]{
270 \expandafter\def\expandafter\tabulinestyle\expandafter{\the\toks@}}%
271 \def\tabulinestyle[4]{
272 \expandafter\def\expandafter\tabulinestyle\expandafter{\the\toks@}}%
273 }% \tabulinestyle
\end{verbatim}

Setting line styles and colors

\begin{verbatim}
234 \newcommand\newtabulinestyle[1]{%
235 \expandafter\def\expandafter\newtabulinestyle\expandafter{\the\toks@}}%
236 }% \newtabulinestyle
237 \def\newtabulinestyle[1]{%
238 \expandafter\def\expandafter\newtabulinestyle\expandafter{\the\toks@}}%
239 }% \newtabulinestyle
240 \def\newtabulinestyle[2]{%
241 \expandafter\def\expandafter\newtabulinestyle\expandafter{\the\toks@}}%
242 }% \newtabulinestyle
243 \def\newtabulinestyle[3]{%
244 \expandafter\def\expandafter\newtabulinestyle\expandafter{\the\toks@}}%
245 }% \newtabulinestyle
246 \def\newtabulinestyle[4]{%
247 \expandafter\def\expandafter\newtabulinestyle\expandafter{\the\toks@}}%
248 }% \newtabulinestyle
\end{verbatim}

All the job is done by \texttt{\tabu@getline}. New line style specifications are always defined globally, and can be overwritten without warning...

\begin{verbatim}
234 \newcommand\newtabulinestyle[1]{%
235 \expandafter\def\expandafter\newtabulinestyle\expandafter{\the\toks@}}%
236 }% \newtabulinestyle
237 \def\newtabulinestyle[1]{%
238 \expandafter\def\expandafter\newtabulinestyle\expandafter{\the\toks@}}%
239 }% \newtabulinestyle
240 \def\newtabulinestyle[2]{%
241 \expandafter\def\expandafter\newtabulinestyle\expandafter{\the\toks@}}%
242 }% \newtabulinestyle
243 \def\newtabulinestyle[3]{%
244 \expandafter\def\expandafter\newtabulinestyle\expandafter{\the\toks@}}%
245 }% \newtabulinestyle
246 \def\newtabulinestyle[4]{%
247 \expandafter\def\expandafter\newtabulinestyle\expandafter{\the\toks@}}%
248 }% \newtabulinestyle
\end{verbatim}

The job is done by \texttt{\tabu@getline}. The settings as usual, are stored in a “locally-global” way...

\begin{verbatim}
234 \newcommand\tabulinestyle[1]{%
235 \expandafter\def\expandafter\tabulinestyle\expandafter{\the\toks@}}%
236 }% \tabulinestyle
237 \def\tabulinestyle[1]{%
238 \expandafter\def\expandafter\tabulinestyle\expandafter{\the\toks@}}%
239 }% \tabulinestyle
240 \def\tabulinestyle[2]{%
241 \expandafter\def\expandafter\tabulinestyle\expandafter{\the\toks@}}%
242 }% \tabulinestyle
243 \def\tabulinestyle[3]{%
244 \expandafter\def\expandafter\tabulinestyle\expandafter{\the\toks@}}%
245 }% \tabulinestyle
246 \def\tabulinestyle[4]{%
247 \expandafter\def\expandafter\tabulinestyle\expandafter{\the\toks@}}%
248 }% \tabulinestyle
\end{verbatim}
\iftabu@everyrow
 \toks@\expandafter[\expandafter \def \expandafter
 \tabu@ls\ls@L\expandafter[\expandafter{\tabu@thestyle}\ignorespaces]%
 \gdef\tabu@ls@{\{\tabu@ls\ls@L}\
 \else
 \global\let\tabu@ls\ls@G \tabu@thestyle
 \gdef\tabu@ls@{\{\tabu@ls\ls@G}\
 \fi
 \tabu@everyrow@egroup
 \% \tabulinestyle

\textbf{\taburulecolor} provides \texttt{\arrayrulecolor}, but the definition is global and must be restored manually after the table. \texttt{\taburulecolor} works with the same scheme as \texttt{\everyrow}: even if the definition of the rules colors must be global (because we can be changed inside the tabular) the value is not restored globally at the end of the environment.

Instead, \texttt{\tabu@arc@L} stores locally the color definition (\textit{i.e.} its definition is relative to the group level before the entry inside the \texttt{tabu} environment).

This is the same for \texttt{\doublerulesepcolor} (which may be given as an optional argument to \texttt{\taburulecolor}): \texttt{\color@tab} makes the definition global, while \texttt{\tabcolor@arc@L} keeps grouping level into mind ("locally-global" settings).
The aim of the game is to define the process that will be executed at \everyrow.

After that, the usual process for “locally-global” settings is plugged into \tabu@cleanup and \tabu@reset...
Simply – and locally – reset the default values for \tabulinesep (0pt), \extrarowsep (0pt), \extratabsurround (0pt), \tabulinestyle {}, \everyrow {} and \taburulecolor {}.

\newcommand*{\tabureset} {\tabulinesep=\z@ \extrarowsep=\z@ \extratabsurround=\z@ \tabulinestyle{} \everyrow{} \taburulecolor||{} \taburowcolors{}%}

Parsing line styles

\tabu@getline This macro parses a line specification argument of the form:

\begin{quote}
3pt BlanchedAlmond on 4pt Crimson off 2pt ForestGreen
\end{quote}

Note that Crimson will overwrite BlanchedAlmond in this case: the color for the line dash may be specified after the line width or after the line dash length.

The process uses \scantokens on the argument given by the user, which is first expanded in a context where the babel switch \if@save@actives is set to true. Then \scantokens is used on the argument in a group where the letter “o” is active, and defined to be a macro which rewrites the line specification. Incidentally, the comma is active too, and expands to a space. This way the initial argument is “genetically modified”, so that it becomes very easy to assign dimensions (thickness, dash length and gap length) and colors separately.

For example: 3pt BlanchedAlmond on 4pt Crimson will be expanded in a context where “o” is active (and equal to \tabu@oxiii, the xiii suffix means “active” ie.\catcode=13).

Then the “o” in BlanchedAlmond is rewritten as follow:

1. “o” sees “n” after itself, then it expands \tabu@onxiii.
2. \tabu@onxiii sees a character whose catcode is not other, then the rewriting process is aborted, and “ond” is rewritten as “ond” where the “o” is not active but the usual letter “o”.

The next “o” is rewritten as follow:

1. “o” sees “n” after itself, then it expands \tabu@onxiii.
2. \tabu@onxiii sees a space (which is active): it calls back itself again,
3. \tabu@onxiii sees a character whose catcode is other: then the sequence “on\texttt{3}” is rewritten as:

\begin{quote}
\texttt{\tabu@ \tabu@on =4pt Crimson}
\end{quote}

Finally the whole argument is rewritten as:

\begin{quote}
\texttt{\tabu@ \tabu@thick =3pt BlanchedAlmond \tabu@ \tabu@on =4pt Crimson \tabu@ \tabu@}
\end{quote}

Define \tabu@ as an appropriate macro which uses \afterassignment to:

1. Assign the corresponding dimension (thickness, dash length or gap length).
2. Collect the rest until the next \tabu@, trim spaces and check if the color exists.

Limitation: A color name must not contain a sequence that matches one of the patterns:

\begin{quote}
...on\langle a\ character\ of\ category\ 12\rangle\ldots\quad\text{or}\quad...off\langle a\ character\ of\ category\ 12\rangle\ldots
\end{quote}

But this “limitation” is not too heavy, I suppose...
The result is \texttt{\tabu@thestyle}: a \texttt{tabu} line style to be used to rewrite a \texttt{l} column, for \texttt{\tabucline}.

We use locally the LaTeX defined dimen registers \texttt{\@tempdima}, \texttt{\@tempdimb} and \texttt{\@tempdimc}. For clarity, their names are \texttt{\tabu@thick}, \texttt{\tabu@on} and \texttt{\tabu@off} here...

\begin{verbatim}
def\tabu@getline #1\{\begingroup 
\csname \ifcsname if@safe@actives\endcsname % <babel>
@safe@activestrue\else
relax\fi \endcsname
\edef\tabu@temp{#1}\tabu@sanitizearg{#1}\@tempa
\let\tabu@thestyle \relax
\ifcsname tabu@linestyle@\@tempa \endcsname
\edef\tabu@thestyle{\endgroup \def\tabu@thestyle{\expandafter
\noexpand\csname tabu@linestyle@\@tempa\endcsname}\
\tabu@thestyle}
\else \expandafter\tabu@definestyle \tabu@temp \@nil \fi
\endgroup\}
\end{verbatim}

\texttt{\tabu@definestyle} Here is the \texttt{\scantokens} stuff.

\begin{verbatim}
\def\tabu@definestyle #1#2\@nil {\endlinechar \m@ne \makeatletter
\tabu@thick \maxdimen \tabu@on \maxdimen \tabu@off \maxdimen
\let\tabu@c@lon \@undefined \let\tabu@c@loff \@undefined
\ifodd 1\ifcat .#1\else\ifcat\relax #1\else 0\fi\fi % catcode 12 or non expandable cs
\def\tabu@temp{\tabu@getparam{thick}}%
\else \def\tabu@temp{\tabu@getparam{thick}\maxdimen}%
\fi{\def\tabu@ tellin\leaderstyle \tabu@thick}{\the\tabu@on}{#1}
{\the\tabu@off}{#2}\
\def\expandafter\:\expandafter{\:}% line spec rewritten now ;-)
\def\;{\def\:}%
\scantokens\expandafter{\expandafter\;\expandafter}{\:}% space is now inactive (catcode 10)
\let\tabu@ \tabu@getcolor :% all arguments are ready now ;-}
\ifdefined\tabu@c@lon \else \let\tabu@c@lon\@empty \fi
\ifx \tabu@c@lon\@empty \def\tabu@c@lon{\CT@arc@}\fi
\ifdefined\tabu@c@loff \else \let\tabu@c@loff\@empty \fi
\ifdim \tabu@on=\maxdimen \ifdim \tabu@off<\maxdimen
\tabu@on \tabulineon \fi\fi
\ifdim \tabu@off=\maxdimen \ifdim \tabu@on<\maxdimen
\tabu@off \tabulineoff \fi\fi
\ifodd 1\ifdim \tabu@off=\maxdimen \ifdim \tabu@on=\maxdimen 0 \fi\fi
\in@true % <leaders>
\else \in@false % <rule>
\fi
\ifdim\tabu@thick=\maxdimen \def\tabu@thick{\arrayrulewidth}%
\else \edef\tabu@thick{\the\tabu@thick}%
\fi
\edef \tabu@thestyle ##1##2{\endgroup
\def\tabu@thestyle{\ifin@ \noexpand\tabu@leadersstyle {\tabu@thick}{\the\tabu@on}{##1}\
{\the\tabu@off}{##2}\
\else \noexpand\tabu@rulesstyle {##1\vrule width \tabu@thick}\
{##1\leaders \hrule height \tabu@thick \hfil}\fi
\endgroup}
\end{verbatim}
We have to define the active “o” character, which looks for the next tokens, trying to find a pattern like on(category 12) or off(category 12) (possibly with – active – spaces between on or off and the next character of catcode 12).

The rewritten stuff.

\TABU@getcolor \TABU@ \TABU@on = (3pt) Crimson \TABU@
When a style is executed, it expands either \tabu@leadersstyle or \tabu@rulesstyle depending on whether or not it contains leaders (dashed lines) or simple rules (solid lines): \TeX internals allow to insert solid lines easily inside a tabular, while inserting leaders is more complex.

\tabu@leadersstyle eventually rebuilds the (horizontal and vertical) leaders boxes, and then define two macros: \tabu@thevleaders and \tabu@thehleaders, suitable to draw vertical and horizontal lines respectively. Incidentally, \tabu@leaders is defined to be the parameters for the leaders.

\tabu@rulesstyle only defines the two macros \tabu@thevrule and \tabu@thehrule. The control sequence \tabu@leaders is undefined so that we know if the style contains a leader or a rule.

\begin{verbatim}
def\tabu@leadersstyle #1#2#3#4#5{\def\tabu@leaders{{#1}{#2}{#3}{#4}{#5}}
\ifx \tabu@leaders\tabu@leaders@G \else \tabu@LEADERS{#1}{#2}{#3}{#4}{#5}\fi}
def\tabu@rulesstyle #1#2{\let\tabu@leaders @undefined \gdef\tabu@thevrule{#1}\gdef\tabu@thehrule{#2}}
\end{verbatim}

Here the two leaders boxes \tabu@hleads and \tabu@vleads are built, as well as the leaders macros \tabu@thehleaders and \tabu@thevleaders.

\begin{verbatim}
def\tabu@LEADERS #1#2#3#4#5{%% width, dash, dash color, gap, gap color
{\let\color \tabu@color % => during trials -> \color = \tabu@nocolor
{\def\@therule{\vrule}\def\@thick{height}\def\@length{width}%%
\def\@box{\hbox}\def\@unbox{\unhbox}\def\@elt{\wd}%%
\def\@skip{\hskip}\def\@ss{\hss}\def\tabu@leads{\tabu@hleads}%%
\tabu@l@@d@rs {#1}{#2}{#3}{#4}{#5}%%
\global\let\tabu@thehleaders \tabu@theleaders
}{\def\@therule{\hrule}\def\@thick{width}\def\@length{height}%%
\def\@box{\vbox}\def\@unbox{\unvbox}\def\@elt{\ht}%%
\def\@skip{\vskip}\def\@ss{\vss}\def\tabu@leads{\tabu@vleads}%%
\tabu@l@@d@rs {#1}{#2}{#3}{#4}{#5}%%
\global\let\tabu@thevleaders \tabu@theleaders
}{\global\let\tabu@leaders@G{{#1}{#2}{#3}{#4}{#5}}}}%
}
def\tabu@therule #1#2{\@therule \@thick#1\@length\dimexpr#2/2 \@depth\z@}
def\tabu@l@@d@rs #1#2#3#4#5{%% width, dash, dash color, gap, gap color
{\global\setbox \tabu@leads=\@box{\{#3\@therule{#1}{#2}}%
\ifx\#5\#5\else{#5\@therule{#1}{#4*2}}\fi
{#3\@therule{#1}{#2}}}}%
def\tabu@leaders0G{(#1){#2}{#3}{#4}{#5}}%
def\global{\let\tabu@leaders\tabu@leaders0}$
\end{verbatim}
11.8 The entry inside \texttt{tabu}

\texttt{\textbackslash tabu, \textbackslash endtabu, \textbackslash longtabu and \textbackslash endlontabu}

\texttt{\textbackslash endtabu} is \texttt{\textbackslash endtabular} or \texttt{\textbackslash endarray} in math mode.

\begin{verbatim}
539 \newcommand*{\tabu}{\tabu@longfalse
540 \ifmmode \def\tabu@\longtable
541 \else \def\tabu@\tabular
542 \expandafter\let\csname tabu\textbackslash end\endcsname \tabu
543 \expandafter\def\csname endlongtabu\textbackslash end\endcsname{\endlongtabu}
544 \let\LT@startpbox \tabu@LT@startpbox % \everypar{ array struts }
545 \endverbatim

\begin{verbatim}
557 \expandafter\def\csname longtabu\textbackslash end\endcsname{\tabuscantokenstrue \longtabu}
558 \def\tabu@nolongtabu{\PackageError{tabu}{longtabu requires the longtable package}\@ehd}
\end{verbatim}

Setting the \texttt{tabu target}

\texttt{\textbackslash tabu@settarget \textbackslash tabu@begin}

The macro sets \texttt{\tabu@target} (a dimen) to the value specified for “\texttt{tabu to}” or “\texttt{tabu spread}”.  

\begin{verbatim}
560 \def\tabu@settarget {\futurelet\@let@token \tabu@sett@rget }
561 \def\tabu@sett@rget {\tabu@target \z@
562 \ifcase \ifx \bgroup\@let@token \z@ \else \else
563 \ifx \@sptoken\@let@token \@ne \else
564 \if t\@let@token \tw@ \else
565 \if s\@let@token \thr@@\else
566 \z@\fi\fi\fi\fi
567 \expandafter\tabu@begin
568 \or \expandafter\tabu@gobblespace\expandafter\tabu@settarget
569 \or \expandafter\tabu@to
570 \or \expandafter\tabu@spread
571 \fi
572 }% \tabu@sett@rget
573 \def\tabu@to to{\def\tabu@halignto{to}\tabu@gettarget}
574 \def\tabu@spread spread{\def\tabu@halignto{spread}\def\tabu@gettarget}
575 \def\tabu@gettarget {\futureassignment{\tabu@linegoaltarget \tabu@target }\tabu@linegoaltarget
576 \def\tabu@linegoaltarget {\futurelet\tabu@temp \tabu@linegoalt@rget }
577 \def\tabu@linegoalt@rget {\%
Flexible \LaTeX\ tabulars
\tabu@setup \tabu@init is expanded only when \tabu is not nested. In this case, and if \parindent > 0, and if \tabudefaulttarget = \linewidth, the correction of the default target for paragraph indentation is executed (see paragraph indentation).

\tabu@setup
\def\tabu@setup{\tabu@alloc@
  \ifcase \tabu@nested
    \ifmmode \else \iftabu@spread\else \ifdim\tabu@target=\z@ \let\tabu@afterendpar \par \fi\fi\fi
    \def\tabu@align{c}\tabu@init \tabu@indent
  \else % <nested tabu>
    \def\tabu@align{t}\let\tabudefaulttarget\linewidth
    \let\tabu@aligndefault\t
  \fi
  \let\tabu@thetarget\tabudefaulttarget \let\tabu@restored@undefined
  \edef\tabu@NC@list{\the\NC@list}\NC@list{\NC@do \tabu@rewritefirst}%
  \everycr{}
  \let\@startpbox\tabu@startpbox % for nested tabu inside longtabu...
  \let\@endpbox\tabu@endpbox % idem " " " " "
  \let\@tabarray\tabu@tabarray % idem " " " " "
  \tabu@setcleanup \tabu@setreset
}\tabu@init
\ifcase \tabu@nested
  \ifmmode \else \iftabu@spread\else \ifdim\tabu@target=\z@ \fi\fi\fi
  \def\tabu@align{c}\tabu@init \tabu@indent
\else % <nested tabu>
  \def\tabu@align{t}\let\tabudefaulttarget\linewidth
  \let\tabu@aligndefault\t
\fi
\def\tabu@setup{\tabu@alloc@
  \ifcase \tabu@nested
    \ifmmode \else \iftabu@spread\else \ifdim\tabu@target=\z@ \let\tabu@afterendpar \par \fi\fi\fi
    \def\tabu@align{c}\tabu@init \tabu@indent
  \else % <nested tabu>
    \def\tabu@align{t}\let\tabudefaulttarget\linewidth
    \let\tabu@aligndefault\t
  \fi
  \let\tabu@thetarget\tabudefaulttarget \let\tabu@restored@undefined
  \edef\tabu@NC@list{\the\NC@list}\NC@list{\NC@do \tabu@rewritefirst}%
  \everycr{}
  \let\@startpbox\tabu@startpbox % for nested tabu inside longtabu...
  \let\@endpbox\tabu@endpbox % idem " " " " "
  \let\@tabarray\tabu@tabarray % idem " " " " "
  \tabu@setcleanup \tabu@setreset
}\tabu@init
\def\tabu@init{\tabu@starttimer \tabu@measuringfalse \tabu@starttimer \tabu@measuringfalse
  \edef\tabu@firsthline{\the\dimexpr\fuzz+1sp}\global\tabu@footnotes()\%
  \let\@xhline\tabu@xhline\let\@arstrutbox\tabu@arstrutbox
  \edef\tabu@LT@@hline{\the\NC@list}\NC@list{\NC@do \tabu@rewritefirst}%
  \edef\tabu@firstline{\the\dimexpr\fuzz+1sp}\global\tabu@footnotes()\%
  \let\@xline\tabu@xline\let\@arstrutbox\tabu@arstrutbox
  \edef\tabu@LT@@hline{\the\NC@list}\NC@list{\NC@do \tabu@rewritefirst}%
  \edef\tabu@firsthline{\the\dimexpr\fuzz+1sp}\global\tabu@footnotes()\%
  \let\@xhline\tabu@xhline\let\@arstrutbox\tabu@arstrutbox
  \edef\tabu@LT@@hline{\the\NC@list}\NC@list{\NC@do \tabu@rewritefirst}%
  \edef\tabu@firstline{\the\dimexpr\fuzz+1sp}\global\tabu@footnotes()\%
  \let\@xline\tabu@xline\let\@arstrutbox\tabu@arstrutbox
  \edef\tabu@LT@@hline{\the\NC@list}\NC@list{\NC@do \tabu@rewritefirst}%
  \edef\tabu@firsthline{\the\dimexpr\fuzz+1sp}\global\tabu@footnotes()\%
  \let\@xhline\tabu@xhline\let\@arstrutbox\tabu@arstrutbox
  \edef\tabu@LT@@hline{\the\NC@list}\NC@list{\NC@do \tabu@rewritefirst}%
  \edef\tabu@firstline{\the\dimexpr\fuzz+1sp}\global\tabu@footnotes()\%
  \let\@xline\tabu@xline\let\@arstrutbox\tabu@arstrutbox
  \edef\tabu@LT@@hline{\the\NC@list}\NC@list{\NC@do \tabu@rewritefirst}%
  \edef\tabu@firsthline{\the\dimexpr\fuzz+1sp}\global\tabu@footnotes()\%
  \let\@xhline\tabu@xhline\let\@arstrutbox\tabu@arstrutbox
  \edef\tabu@LT@@hline{\the\NC@list}\NC@list{\NC@do \tabu@rewritefirst}%
  \edef\tabu@firstline{\the\dimexpr\fuzz+1sp}\global\tabu@footnotes()\%
  \let\@xline\tabu@xline\let\@arstrutbox\tabu@arstrutbox
  \edef\tabu@LT@@hline{\the\NC@list}\NC@list{\NC@do \tabu@rewritefirst}%
We have to save locally (in the group of the environment) the current value of the last global assignments to \CT@arc@, \CT@drsc@, \tabu@ls@ etc.

\tabu@cleanup

Restoration will be done globally after the box that contains the tabular by \tabu@cleanup.
At the beginning of each trial, we have to restore the current value that were active at the entry in the \texttt{tabu} environment (for they could have been globally overwritten inside the tabular).

The same must occur when using \texttt{\usetabu} as a preamble. Values are restored locally inside the \texttt{tabu} box. \texttt{\tabu@setreset} defines \texttt{\tabu@reset} to be expanded at the beginning of each trial and when \texttt{\usetabu} is used.
11.9 The rewriting process: inside the \texttt{"\@mkpream\ group"}

New column types and private (new) column types

\texttt{\tabu@newcolumntype}  A helper macro to create new column types for \texttt{tabu}.

The column types \textbf{are not appended} to \texttt{\NC@list} in order to keep them local to \texttt{tabu}.

\texttt{\def\tabu@newcolumntype \#1\%}  \texttt{\expandafter\tabu@new@columntype} \texttt{\csname NC@find@\string\#1\expandafter\endcsname \csname NC@rewrite@\string\#1\endcsname}  \texttt{\def\#1\##1\#3}{\NC@{##1}}  \texttt{\let\#2\relax \newcommand*\#2}{\tabu@new@columntype}

\texttt{\tabu@privatecolumntype}  Columns types defined with \texttt{\tabu@privatecolumntype} are "mounted" only inside the \texttt{\@mkpream} group of \texttt{tabu}.

\texttt{\def\tabu@privatecolumntype \#1\%}  \texttt{\expandafter\tabu@private@columntype} \texttt{\csname NC@find@\string\#1\expandafter\endcsname \csname NC@rewrite@\string\#1\endcsname \csname tabu@NC@find@\string\#1\endcsname \csname tabu@NC@rewrite@\string\#1\endcsname}  \texttt{\global\tabu@alloc} \texttt{\tabu@alloc@save}  \texttt{\everyrow\expandafter{\tabu@evr@L}}  \texttt{\def\tabu@reset}{\tabu@savedparams}  \texttt{\let\CT@arc@\tabu@arc@L}  \texttt{\let\CT@drsc@\tabu@drsc@L}  \texttt{\let\tabu@ls@\tabu@ls@L}  \texttt{\let\tabu@rc@\tabu@rc@L}  \texttt{\global\tabu@alloc} \texttt{\tabu@alloc@save}  \texttt{\let\c@taburow\z@}
High priority columns

\tabucolumn puts a user-defined column in high priority in the \texttt{tabu} rewriting process.

Rewriting vertical lines and leaders

\texttt{\textbackslash tabu@vlinearg} The macro that parses the optional argument of \textbackslash \texttt{tabu@vlinearg}...
Vertical lines and leaders in the `\multicolumn` preamble

A special rewrite to allow `|...|` in `\multicolumn` preamble inside `tabu` environment.

As long as `\multicolumn` begins with `\omit` (via `\multispan`) special care has to be taken: everything shall be purely expandable until `\omit`.

`\multicolumn` is not an environment: no group is opened apart the `\@mkpream` group. We open a semi simple group for `\multicolumn` when inside `tabu`, in order for the setup to be local (in case a user would try to embed a `tabular` inside the argument of `\multicolumn`...)

Rewriting `tabu` X columns

This is the rewrite macro for `tabu` X columns. Such a column has an optional argument: the width coefficient for the `tabu` X column whose default value is 1, and may be some alignments parameters. The coefficient is used in the expression: `p \dimexpr ⟨coef⟩ \tabucolX`
This macro is expanded by during the rewriting process in case a X column is found.

\tabu@Xsum (a dimen) stores the sum of the (absolute) width coefficients.

For the first X column found in the preamble, a special setup occurs:

- if the default target is used (no target specified or \tabu spread with X columns), the target: \tabu@target is set to the default, with a message in the .log file.

- \@halign is \let to \relax to avoid its expansion in \edef \@preamble just after \@mkpream. Indeed as long as we have to measure the natural width of the tabular, \@halign must be empty for trial steps.

- The rest of the setup is made \aftergroup (ie after \edef \@preamble which occurs inside a group) by \tabu@prep@TRIAL.

\def\tabu@rewriteX #1#2#3{\tabu@Xarg {#1}{#2}{#3} \iftabu@measuring
  \else \tabu@measuringtrue \fi % first X column found in the preamble
  \let\@halignto \relax \let	abu@halignto \relax
  \iftabu@spread \tabu@spreadtarget \tabu@target \z@ \else \tabu@spreadtarget \z@ \fi
  \iftabu@target=\z@ \setlength\tabu@target{\tabu@thetarget} %<default coef = 1>
  \else \tabu@message{\tabu@message@target}\fi
  \fi}
\def\tabu@rewriteXrestore #1#2#3{\let\@halignto \relax \def	abu@rewritten{l}\let\tabu@rewritten{}}

A tedious (and fastidious) macro to parse the optional argument of X columns. The aim is to build \tabu@rewritten which expands to the column specification:

>{alignment} p or m or b {\dimexpr coef \tabucolX \relax}

After that array.sty make it easy: \expandafter \NC@find \tabu@rewritten
Final step: the whole optional argument has been read, then builds the rewritten column specification.
Depending on the sign of the coefficient, and of the stored value for the natural width of the column the X cell belongs to, \texttt{\tabu@hsize} returns the wanted width for the \textit{par-box} that contains the cell content.

\begin{Verbatim}
\def\tabu@hsize #1#2{% 
  \ifdim #2\p@<\z@ 
    \ifdim \tabucolX=\maxdimen \tabu@wd{#1}\else 
      \ifdim \tabu@wd{#1}<#2\tabucolX \tabu@wd{#1}\else -#2\tabucolX\fi 
    \fi 
  \else #2\tabucolX \fi 
}\end{Verbatim}

Rewritting \texttt{\usetabu} and \texttt{\preamble}

The rewritting process is very simple, when all the job has been done cleverly at the time of \texttt{\savetabu}!!

The \texttt{\savetabu} macro is a bit more complex...

\begin{Verbatim}
\def\tabu@privatecolumntype \usetabu \[1]{% 
  \ifx\&\#1\&\tabu@saveerr{}\else 
    \@ifundefined{tabu@saved@\string#1}{\tabu@saveerr{#1}}{\csname tabu@saved@\string#1\expandafter\endcsname\expandafter\@ne}\fi 
}\end{Verbatim}

\begin{Verbatim}
\def\tabu@privatecolumntype \preamble \[1]{% 
  \ifx\&\#1\&\tabu@saveerr{}\else 
    \@ifundefined{tabu@saved@\string#1}{\tabu@saveerr{#1}}{\csname tabu@saved@\string#1\expandafter\endcsname\expandafter\z@}\fi 
}\end{Verbatim}

Controlling the rewritting process

This new column type is not really a column type! It is always added to a \texttt{\tabu} preamble in order to do some setup before any other column is rewritten by \texttt{\@mkpream}.

Thus, \texttt{\NC@list} is simply set to {\texttt{\NC@do \tabu@rewritefirst}}. The rewritting of \texttt{\tabu@rewritefirst} will restore the original list \texttt{\NC@list}.

This “column type” sets:

- \texttt{\tabu@select} to be expanded \texttt{\aftergroup} (after the closing of the \texttt{\@mkpream} group. All the thick is there: all information collected during the rewritting of X columns (and vertical lines or leaders) can be \textit{reinjected} into the group below the \texttt{\@mkpream} group, by the mean of the
\textbf{Technical notice and implementation} \tabularx (globally defined).

- The private columns types are loaded by \texttt{\tabu@rewritefirst}: they will be rewritten afterwards, during the rewriting loop. This way, \texttt{X} column definition for \texttt{tabu} are only available during the rewriting process of the \texttt{tabu} preamble, making it possible (and easy) to embed a \tabularx inside a cell of a \texttt{tabu}.

- \texttt{\save@decl} is modified inside the \texttt{\@mkpream} group, if \texttt{tabu} is in text mode.

\begin{verbatim}
995 \ifthenelse{\isif@long}{% 
996 \aftergroup \tabu@longpream % <the whole implementation is here !> 
997 \else \aftergroup \tabu@pream 
998 \fi
999 \let\tabu@ \relax \let\tabu@hsize \relax 
1000 \let\tabu@Xcoefs @{\empty} \let\tabu@savels \relax 
1001 \let\tabu@Xcol \z@ \let\tabu@cnt \tw@
1002 \edef\tabu@temp{\the\@temptokena}
1003 \def\tabu@\textbar{} \def\tabu@lines{} 
1004 \NC@list{\NC@do S \NC@do S} \NC@list{\NC@do X} 
1005 \NC@list{\NC@do X \NC@do X \NC@do X \NC@do X} 
1006 \edef\tabu@\textbar@endrewrite\tabu@rewritemiddle\tabu@rewritelast % * X S <original>
1007 \NC@list{\NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X} 
1008 \NC@list{\NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X} 
1009 \NC@list{\NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X} 
1010 \NC@list{\NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X} 
1011 \NC@list{\NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X} 
1012 \NC@list{\NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X} 
1013 \NC@list{\NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X} 
1014 \NC@list{\NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X} 
1015 \NC@list{\NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X} 
1016 \NC@list{\NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X} 
1017 \NC@list{\NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X} 
1018 \NC@list{\NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X \NC@do X} 
\end{verbatim}

This new column type is rewritten after \texttt{X} columns, because it is declared by when the column \texttt{\tabu@rewritemiddle} is actually rewritten. In the case where \texttt{\tabu@target} is > 0 (either because of “\texttt{tabu to}” or “\texttt{tabu spread}” has been called) and if there is no \texttt{X} column, then \texttt{\extracolsep \@flushglue} is added at the beginning of the preamble.

To avoid duplicate margin in the \texttt{tabu} we have to test the next token in the preamble. If the next token is \texttt{|} or \texttt{!} then no margin must be added and \texttt{\extracolsep \@flushglue} can be inserted at the beginning of the preamble.

Otherwise, we must insert \texttt{\extracolsep \@flushglue} in order to keep the margin.
The end of the rewriting process: determining the \texttt{tabu} strategy

\texttt{\textbackslash tabu@endrewrite} Determines the strategy to be executed \texttt{\textbackslash aftergroup} (at the closing of the \texttt{\textbackslash @mkpream group}):

0) There is no real strategy: \texttt{tabu} behaves like \texttt{tabular}, there no X column, and no need to measure the vertical dimensions of the cells (no dynamic spacing, no vertical leader). In case a target has been given to \texttt{tabu}, it behaves like \texttt{tabular} and an infinite stretchability is given to the column inter-space. This is done (if required) by \texttt{\textbackslash tabu@extracolsep}.

1) Measuring natural width of some (or all) columns is compulsory for \texttt{tabu spread} of X columns with negativ coefficients. Thereafter, the strategy nr 2 will bring into play.

2) Measuring the natural width is not necessary, or has been done before. But \texttt{tabu} contains X columns and trials have to be performed to reach the desired target, adjusting the \texttt{\textbackslash tabucol} dimension accordingly. Then, the strategy nr 3 may bring into play, if vertical measure is required.

3) Vertical measure of the cells is required, for vertical spacing adjustment or vertical leaders. This step can be done only if the width are known.

\texttt{3} The \texttt{tabu} is finished and ready to be printed!!

\begin{verbatim}
1032 \def\tabu@endrewrite {\%
1033  \let\tabu@temp NC@find
1034  \ifx @arrayright\relax \let@arrayright @empty \fi
1035  \count0=\%
1036  \ifx @finalstrut\tabu@finalstrut \z@ \% outer in mode 0 print
1037  \iff@tabu@measuring
1038    \edef\tabu@mkpreambuffer{\tabu@mkpreambuffer \\
1039      \tabu@target \csname tabu@the\tabu@nested.T\endcsname
1040      \tabucolX \csname tabu@the\tabu@nested.X\endcsname
1041      \edef@halignto {\ifx @arrayright\empty to \tabu@target\fi}}%
1042  \fi
1043  \else\iff@tabu@measuring 4 \% X columns
1044    \edef\tabu@mkpreambuffer{\tabu@\{\tabu@mkpreambuffer \\
1045      \tabu@target \the\tabu@target
1046      \tabu@spreadtarget \the\tabu@spreadtarget\}%
1047      \def\noexpand\tabu@Xcoefs\{\tabu@Xcoefs\%
1048      \edef\tabu@halignto {\ifx @arrayright\empty to \tabu@target\fi}}%
1049    \let\tabu@Xcoefs \relax
1050  \else\iff@tabu@nested \thr@@ \% outer, no X
1051    \iff@tabu@afterendpar \relax
1052    \else \@ne \% inner, no X, outer in mode 1 or 2
1053    \fi
1054    \ifdefined\tabu@usetabu
1055      \else \ifdim\tabu@target=\z@ \%
1056      \else \let\tabu@temp \tabu@extracolsep
1057      \fi\fi
1058    \fi
1059  \fi
1060  \edef\tabu@mkpreambuffer{\count0 \the\count0 \tabu@mkpreambuffer}%
1061  \tabu@temp
1062 \}% \tabu@endrewrite
\end{verbatim}

\texttt{\textbackslash tabu@extracolsep} Inserts \texttt{\textbackslash extracolsep} \texttt{\textbackslash @flushglue} in front of the preamble, unless another value for \texttt{\textbackslash extracolsep} has been specified.

\texttt{\textbackslash @flushglue} is Opt plus 1fil.

\begin{verbatim}
1063 \def\tabu@extracolsep{\@defaultunits \expandafter\let
1064 \expandafter\tabu@temp \expandafter=\the\emptokena \relax\@nil
1065 \ifx \tabu@temp@emptoken \@defaultunits
1066 \expandafter\tabu@gobblespace \expandafter\tabu@extracolsep
\end{verbatim}
11.10 Implementing the strategy at the exit of the \@mkpream group

Here we check if trials are required or not: depending on the value of \count@ (set at \@endrewrite, and injected here by \@mkpreambuffer), on \@iftabu@measuring (nested trials).

When trials are required, \@tabu@select give control to \@tabu@setstrategy (to prepare the neutralisation of commands, save counters etc).

When trials are not required, we just have to expand \@tabuthepreamble, after having set up the \everyrow stuff properly (for vertical adjustment or vertical measure, if needed).

\tabu@select

This is the long version for \longtabu: the material to collect until \@preamble is different !

\def\tabu@select {%
General setup for trials: neutralisation of \write etc.

This is the general setup for trials: the \texttt{tabu} will be expanded more than once, thus some protections are set: the value of global counters are saved, footnotes have a special setup, \texttt{hbadness} and \texttt{hfuzz} are neutralised etc.

The initial value for \texttt{\tabucolX} is computed with the coefficients stored into \texttt{\tabu@Wvoefs}: \texttt{\tabu@{coef1} \tabu@{coef2} \tabu@{coef3}} etc.

is very suitable for loops on the column width coefficients (without the need of \texttt{@for} or whatsoever).
Collecting the tabu body

The macro collects the stuff inside \@array: depending on the global vertical alignment parameter for the whole tabular, the tabular is built inside a \vbox, \vtop or \vcenter (the default – unless tabu is nested).

At this time, we define \tabu@trial (which inherits from the \vbox, \vtop or \vcenter) and \tabu@Xfinish as well.

The mechanism is the same as \collect@body (also defined in environ.sty). The content of the tabular is captured inside \toks, expanded by \tabu@trial.

11.11 One trial after the other (\tabu@strategy)

Switching between the strategies

This macro does some specific setup depending on the strategy (1, 2 or 3), and orders to finish when all measurements are done.

This consists in a switch (\ifcase) which is done before the trials by \tabu@strategy, and after the trials by \tabu@endtrial.
\newcommand\tabu@quickend
\begingroup
\tabu@evr{\tabu@verticalinit}\tabu@cellalign@def{\tabu@verticalmeasure}%
\def\tabu@cellralign{\tabu@verticalspacing}%
\expandafter\tabu@measuring
\else % case 4 = horizontal measure
\begingroup
\global\let\tabu@elapsedtime\tabu@message@etime
\long\def\multicolumn##1##2##3{\multispan{##1}}%
\let\tabu@startpboxORI\@startpbox
\iftabu@spread
\def\tabu@naturalXmax{\z@}%
\let\tabu@naturalX\tabu@naturalXmax
\let\@startpbox\tabu@startpboxmeasure
\else\iftabu@negcoef
\let\@startpbox\tabu@startpboxmeasure
\else\let\@startpbox\tabu@startpboxquick\fi\fi
\expandafter\tabu@measuring
\fi
\expandafter\tabu@endtrial
\endarray$\egroup % got \tabu@box
\def\tabuendlongtrial{% no @ allowed for \scantokens
\LT@echunk \global\setbox\@ne\hbox{\unhbox\@ne}\kern\wd\@ne
\LT@get@widths
}% \	abu@strategy
\tabu@measuring Expands \tabu@trial with the whole content of the environment stored in \toks@ by \tabu@collectbody. At the end of the trial, \count@ will be reassigned to the value it had before the trial. Then \tabu@endtrial will choose the algorithm depending on the strategy number, and set the new strategy number (into \count@ again) for the next step.

\tabu@trial This is the starting point of trials: \halign is expanded here.

\tabu@longtrial This is the long version of \tabu@trial for longtabu. Almost the same apart for the math group and the end (a longtable environment does not finish with \endarray).

\def\tabu@measuring{\expandafter\tabu@trial\expandafter\count@\the\count@\tabu@endtrial}
\def\tabu@trial{\iftabu@long\tabu@longtrial\else\tabu@shorttrial\fi}
\def\tabu@shorttrial{\setbox\tabu@box\hbox\bgroup\tabu@seteverycr\ifx\tabu@savecounters\relax\else\let\tabu@savecounters\relax\tabu@clckpt\fi\iftabuscantokens\tabu@rescan\else\expandafter\@secondoftwo\fi\expandafter{\expandafter\tabuthepreamble\the\tabu@thebody\tabu@endtrial}}
\def\tabu@longtrial{\setbox\tabu@box\hbox\bgroup\tabu@seteverycr\ifx\tabu@savecounters\relax\else\let\tabu@savecounters\relax\tabu@clckpt\fi\iftabuscantokens\tabu@rescan\else\expandafter\@secondoftwo\fi\expandafter{\expandafter\tabuthepreamble\the\tabu@thebody\tabuendlongtrial}}
\def\tabuendlongtrial{% no @ allowed for \scantokens
\LT@echunk \global\setbox\@ne\hbox{\unhbox\@ne}\kern\wd\@ne\LT@get@widths
}}
align resets \everycr to an empty token. This macro sets \everycr for the tabu environment: a bridge around \align is built: \everycr redefines itself \afterassignment!

When the algorithm said the tabular was ready to be printed, \tabu@endoftrials closes the trials group and prints the tabular...

The required values (column widths, struts etc.) are injected into the group by the mean of the buffer \tabu@bufferX (locally defined).

This closes the group in which all the trials are done.
Quick exit after having measuring the natural width of a nested `tabu`.

Depending on the strategy that was just applied, \texttt{\textbackslash\texttt{tabu\textbackslash endtrial}} chooses the algorithm and determines the number of the strategy for the next step.

The algorithms: Measuring the \texttt{tabu} box

At the end of each trial, we call \texttt{\textbackslash\texttt{tabu\textbackslash arith}} (or \texttt{\textbackslash\texttt{tabu\textbackslash spreadarith}}) to computes the widths and update the values.

At the exit, \texttt{\textbackslash\texttt{iftabu\textbackslash measuring}} is set to \texttt{\textbackslash\texttt{iftrue}}: a further trial is necessary, or \texttt{\textbackslash\texttt{iffalse}}: the target width is reached.

The arithmetic of \texttt{X} columns: the \texttt{tabu} to case

This is a loop against the width coefficients. There is no \texttt{@for} or \texttt{@whiles} because \texttt{\textbackslash\texttt{tabu\textbackslash Xcoefs}} stores the series in the form: \texttt{\textbackslash\texttt{tabu@\{coef1\}}} \texttt{\textbackslash\texttt{tabu@\{coef2\}}} \texttt{\textbackslash\texttt{tabu@\{coef3\}}}.

Thus, just \texttt{\textbackslash\texttt{let \textbackslash\texttt{tabu}}\textbackslash\texttt{}} to be \texttt{\textbackslash\texttt{tabu\textbackslash arith\textbackslash negcoef}} and expand \texttt{\textbackslash\texttt{tabu\textbackslash Xcoefs}}!

The aim of the game is to **neutralize** some \texttt{X} columns: when their natural width are less than coef\times\texttt{\textbackslash\texttt{tabu\textbackslash colX}}.
General algorithms for tabu to with X columns.

% \tabu@arith
% \def\tabu@givespace #1#2{% here \tabu@DELTA < \z@}
% \ifdim \tabu@DELTA<\z@
% \let\tabu@ \tabu@givespace \tabu@Xcoefs
% \advance\@tempdima \@tempdimb \advance\@tempdima -\tabu@DELTA % for message
% \else % need for narrower X columns
% \tabucolX =\dimexpr \@tempdima -\tabu@DELTA \relax % always < 0 here
% \fi
% \edef\tabu@bufferX{\endgroup \tabu@cnt \the\tabu@cnt}
The arithmetic of \( \mathbf{X} \) columns for \textbf{tabu spread} 

Algorithm for \textbf{tabu spread} with \( \mathbf{X} \) columns: the aim of the game is to compute the target (relative to the natural width of the tabular) and go to \texttt{\tabu@arith} afterwards.

\begin{verbatim}
1382 \def\tabu@spreadarith {
1383 \dimen@ \z@ \@tempdima \tabu@naturalXmax \let\tabu@ \tabu@spread@arith \tabu@Xcoefs
1384 \edef\tabu@naturalXmin {\the\dimexpr\tabu@naturalXmin*\dimen@/\p@}\
1385 \@tempdimc = \dimexpr \wd\tabu@box - \tabu@naturalXmax + \tabu@naturalXmin \relax
1386 \iftabu@measuring
1387 \tabu@target = \dimexpr \@tempdimc + \tabu@spreadtarget \relax
1388 \edef\tabu@bufferX{\endgroup \tabucolX \the\tabucolX \tabu@target \the\tabu@target}\
1389 \else
1390 \tabu@message(\tabu@message@spreadarith)\
1391 \ifdim \dimexpr \@tempdimc + \tabu@spreadtarget > \tabu@target
1392 \tabu@message(\tabu@target useless here: default target used)\
1393 \else too large: reduced to fit default target fi.\)\
1395 \else
1396 \tabu@target = \dimexpr \@tempdimc + \tabu@spreadtarget \relax
1397 \tabu@message(\tabu@target useless here: default target used)\
1398 \else too large: reduced to fit default target fi.\)
1399 \fi
1400 \begingroup \let\tabu@wddef \@gobbletwo
1401 \@tempdimb \@tempdima
1402 \tabucolX@init
1403 \tabu@arithnegcoef
1404 \wd\tabu@box = \dimexpr \wd\tabu@box + \@tempdima - \@tempdimb \relax
1405 \expandafter\endgroup \expandafter\tabucolX \the\tabucolX
1406 \tabu@arith
1407 \fi
1408 \def\tabu@spreadarith \ifnum\tabu@nested=\z@ \else \fi
1409 \iftabu@measuring \advance\dimen@ \z@ \advance\@tempdimb \z@ \relax
1410 \else \advance\@tempdimb \tabu@wd{#1} \relax
1411 \fi
1412 \def\tabu@spreadarith \fi
1413 \def\tabu@message@defaulttarget {
1414 \ifnum\tabu@nested=\z@ \string\linegoal = \the\tabu@thetarget \else - \the\tabu@thetarget \fi =
1416 \iffx\tabudefaulttarget\linewidth \string\linewidth \else
1417 \iffx\tabudefaulttarget\linegoal \string\linegoal \else
1419 \fi
1420 \else (tabu) Default target (nested): \fi
1421 \the\tabu@target \on@line
1422 \iftabu@nested=\z@ \string\linegoal \fi
1423 \def\tabu@message@target {\string\linegoal \the\tabu@target \fi
1424 \the\tabu@target \on@line, page \the\c@page}
1425 \def\tabu@message@arith {\tabu@header
\end{verbatim}
11.13 Measuring the natural width of columns (varwidth code from D. Arseneau)

\texttt{\textbackslash tabu@startpboxmeasure} The important job is done at the end: by \texttt{\textbackslash tabu@endpboxmeasure}.

When \texttt{\textbar{tabu spread}} is used with X columns, the first trial must measure the natural width of the columns. When X columns have negative coefficient, the natural is computed after the target has been reached, with the absolute coefficients.

Nested trials may occur (\texttt{\textbar{tabu spread}} inside a X column with negative coefficient for example).

For the further trials, the standard scheme for X column is used: the natural width is measured only once.

pdfTEX font expansion is disabled inside the \texttt{varwidth} environment (we set \texttt{\textbackslash pdfadjustspacing} to 0).

\begin{verbatim}
def\tabu@startpboxmeasure #1{% entering vtop \
def\tabu@temp{\expandafter\@secondoftwo \ifx\tabu@hsize #1\else\relax\fi}\% 
  \ifodd 1\ifx \tabu@temp\@empty 0 \else \% starts with \tabu@hsize ? \n    \iftabu@spread \else \% if spread -> measure \n      \ifdim \tabu@temp\p@>\z@ 0 \fi\fi\fi\% if coef>0 -> do not measure \n  \let\@startpbox \tabu@startpboxORI \% restore immediately (nesting) \n  \tabu@measuringtrue \% for the quick option... \n  \tabu@Xcol =\expandafter\@firstoftwo \ifx\tabu@hsize #1\fi \% 
  \ifdim \tabu@temp\p@>\z@ \ifdim \tabu@wd\tabu@Xcol<\tabu@target \n    \tabu@target=\tabu@temp\tabu@colX \fi\fi \% 
  \setbox\tabu@box \hbox \bgroup \begin{varwidth}\tabu@target \n    \let\FV@ListProcessLine \tabu@FV@ListProcessLine \% \hbox to natural width... \n    \narrowragged \arraybackslash \parfillskip \@flushglue \n    \ifdef\pdfadjustspacing \pdfadjustspacing\z@ \fi \n  \begin{group} \aftergroup\tabu@endpboxmeasure \n    \ifdef\cellspacetoplimit \tabu@cellspacepatch \fi \n  \else \expandafter\@gobble \n    \tabu@startpboxquick{#1}\% \@gobble \bgroup \n  \fi \n% \tabu@startpboxmeasure 
\def\tabu@endpboxmeasure {\% \@finalstrut \@arstrutbox \n  \begin{varwidth} \vtop \n    \let\FV@ListProcessLine \tabu@FV@ListProcessLine \% \hbox to natural width... \n    \narrowragged \arraybackslash \parfillskip \@flushglue \n    \ifdef\pdfadjustspacing \pdfadjustspacing\z@ \fi \n  \begin{group} \aftergroup\tabu@endpboxmeasure \n    \ifdef\cellspacetoplimit \tabu@cellspacepatch \fi \n  \else \expandafter\@gobble \n    \tabu@startpboxquick{#1}\% \@gobble \bgroup \n  \fi \n% \tabu@startpboxmeasure 
\end{varwidth} \egroup \% <got my \tabu@box>
\end{group} \% \vtop (measure) restore \tabu@target }
\end{verbatim}

\texttt{\textbar{tabu@endpboxmeasure}} The cell has been built inside a box: we have to get its dimensions, and update \texttt{\textbar{tabu@naturalX}}, \texttt{\textbar{tabu@naturalXmin}} and \texttt{\textbar{tabu@naturalXmax}} accordingly (for \textbar{tabu spread}), and even store (globally) each column width: the column width is the maximum width of the cells it contains.

\begin{verbatim}
def\tabu@endpboxmeasure {% 
  \@finalstrut \@arstrutbox \n  \begin{varwidth} \vtop \n    \let\FV@ListProcessLine \tabu@FV@ListProcessLine \% \hbox to natural width... \n    \narrowragged \arraybackslash \parfillskip \@flushglue \n    \ifdef\pdfadjustspacing \pdfadjustspacing\z@ \fi \n  \begin{group} \aftergroup\tabu@endpboxmeasure \n    \ifdef\cellspacetoplimit \tabu@cellspacepatch \fi \n  \else \expandafter\@gobble \n    \tabu@startpboxquick{#1}\% \@gobble \bgroup \n  \fi \n% \tabu@startpboxmeasure 
\end{varwidth} \egroup \% <got my \tabu@box>
\end{group} \% \vtop (measure) restore \tabu@target }
\end{verbatim}
11.14 Measuring the height and depths of rows

Starting point for vertical measure of every cell. Only the maxima/minima are stored, for \( \mathcal{T}_{\mathcal{H}b} \) must know the height/depth of every row.
Vertical spacing adjustment with struts for \( p \), \( m \), or \( b \) columns.

```latex
\def\tabu@verticalsp@pmb{% 
\par \expandafter\egroup 
\@tempdimc \the\prevdepth 
\@tempdima \dimexpr \ht\tabu@box+\abovetabulinesep \relax 
\ifdim\tabustrutrule<\z@ \tabu@debug{\tabu@message@verticalsp}\fi 
\ifdim \tabu@ht<\@tempdima \tabu@htdef{\the\@tempdima}\fi 
\ifdim \tabu@dp<\@tempdimb \tabu@dpdef{\the\@tempdimb}\fi 
\let\@finalstrut \@gobble 
\hrule height\@tempdima depth\@tempdimb width\hsize 
\box\tabu@box 
}% 
\tabu@verticalsp@pmb 
```

Initialisation of \tabu@ht and \tabu@dp. Done at \everyrow.

```latex
\def\tabu@verticalinit{% 
\ifnum \c@taburow=\z@ \tabu@reset ! \fi 
\advance\c@taburow \@ne 
\tabu@htdef{\the\ht\@arstrutbox}\tabu@dpdef{\the\dp\@arstrutbox}% 
\advance\c@taburow \m@ne 
}% 
\def\tabu@htdef {
\expandafter\xdef \csname tabu@nested.H\the\c@taburow\endcsname}
\def\tabu@ht {
\csname tabu@nested.H\the\c@taburow\endcsname}
\def\tabu@dpdef {
\expandafter\xdef \csname tabu@nested.D\the\c@taburow\endcsname}
\def\tabu@dp {
\csname tabu@nested.D\the\c@taburow\endcsname}
```

This updates the \@arstrutbox at \everyrow (\ie \everycr) in order to adjust the vertical spacing of cells.

```latex
\def\tabu@verticaldynamicadjustment {% 
\advance\c@taburow \@ne 
\extrarowheight \dimexpr\tabu@ht - \ht\strutbox - \ht\@arstrutbox \relax 
\extrarowdepth \dimexpr\tabu@dp - \dp\strutbox - \dp\@arstrutbox \relax 
\let\arraystretch \@empty 
\advance\c@taburow \m@ne 
}% 
\def\tabu@reset{% 
\tabu@htdef \def\tabu@ht {
\expandafter\xdef \csname tabu@nested.H\the\c@taburow\endcsname}
\tabu@dpdef \def\tabu@dp {
\expandafter\xdef \csname tabu@nested.D\the\c@taburow\endcsname}
```

This updates the \@arstrutbox at \everyrow (\ie \everycr) in order to adjust the vertical spacing of cells.

This macro inserts a phantom line in front of a tabu. This is necessary when you use \usetabu with \tabu X column, with a single line containing \multicolumn...

```latex
\def\tabuphantomline{% 
\setbox\@arstrutbox \box\voidb@x 
\let\tabu@cellalign \tabu@cellalign 
\let\tabu@cellalign \tabu@cellalign 
\let\tabu@cellleft \tabu@cellleft 
\let\tabu@cellright \tabu@cellright 
\let\tabu@thevline \tabu@thevline 
\let\tabu@cellalign \@empty 
\let\tabu@cellalign \@empty 
\let\tabu@cellalign \@empty 
\let\tabu@thevline \relax 
\edef\tabu@temp{\def\tabu@temp\tabu@multispan \tabu@nbcols{\noindent \&}}% 
```

```
```
```
11.16 Horizontal lines inside \texttt{tabu}: \texttt{\tabucline}, \texttt{\firsthline} and \texttt{\lasthline}

Horizontal lines: multiple \texttt{\firsthline} / \texttt{\lasthline}

\texttt{\firsthline} and \texttt{\lasthline} are \texttt{let} to \texttt{\tabu@firsthline} and \texttt{\tabu@lasthline} inside the \texttt{tabu} environment. This allows to duplicate horizontal lines, while keeping the alignment:

\texttt{\firsthline \firsthline \firsthline} is allowed inside \texttt{tabu} and is the same as:

\texttt{\firsthline \hline \hline}.

\texttt{\def\tabu@hlineAZ #1#2\{\noalign{\ifnum0='}\fi \dimen@ \z@ \count@ \z@ \toks@lbrace\def\tabu@hlinecorrection{#1}\def\tabu@temp{#2}\toks@rbrace\def\tabu@hlineAZsurround[1][\extratabsurround\{\extratabsurround\}]{\extratabsurround\let\tabucline \tabucline@scan \let\hline \tabu@hlinescan \let\firsthline \hline \expandafter \futurelet \expandafter \tabu@temp \expandafter \tabu@nexthlineAZ \tabu@temp\}}\def\tabu@hlinescan {\tabu@thick \arrayrulewidth \tabu@xhlineAZ \hline}}

Here we go, inside a \texttt{\noalign} group, we collect the next tokens:

1. first the option,
2. and then the next tokens if they are \texttt{\hline} or \texttt{\firsthline}.

The code to be executed at the end of the \texttt{\noalign} group is built into \texttt{\toks@}.
\def\tabu@firsthlinecorrection{% 
\count@ = number of \hline -1
\edef\firsthline{% <local in \noalign>
\omit \hbox to\z@{\hss\noexpand\tabuDBG{yellow}\vrule
height \the\dimexpr\ht\@arstrutbox+\dimen@+
depth \dp\@arstrutbox
width \tabastrutrule}hss\cr
\noalign{\vskip -\the\dimexpr \dimen@+\@tempdimb+\dp\@arstrutbox}%
\the\toks0
}\ifnum0='{\fi
\expandafter}\firsthline % we are then !
} % \tabu@firsthlinecorrection

\def\tabu@lasthlinecorrection{%
\@tempdimc \dimexpr \dp\@arstrutbox+\dimen@+
\edef\lasthline{% <local in \noalign>
\the\toks0
\noalign{\vskip -\the\dimexpr \dimen@+\@tempdimb+\dp\@arstrutbox}%
\omit \hbox to\z@{\hss\noexpand\tabuDBG{yellow}\vrule
depth \the\dimexpr \dp\@arstrutbox+\@tempdimb+\dimen@
+\extratabsurround-\@tempdimc
height \z@
width \tabastrutrule}hss\cr
\ifnum0='{\fi
\expandafter}\lasthline % we are then !
} % \tabu@lasthlinecorrection

\def\tabu@LT@@hline{\ifx\LT@next\hline
\global\let\LT@next \@gobble
\ifx \CT@drsc@\relax
\gdef\CT@LT@sep{\noalign{\penalty-\@medpenalty\vskip\doublerulesep}}% 
\else \advance\dimen@ \dimexpr \@tempdimm+\doublerulesep \relax
\fi
\fi
\expandafter}\LT@next % we are then !
} % \tabu@LT@@hline

Allowing colored rules even if colortbl is not loaded.
\def\tabu@LT@hline{%
\ifx\LT@next\hline
\global\let\LT@next \@gobble
\ifx \CT@drsc@\relax
\gdef\CT@LT@sep{% 
\noalign{\penalty-\@medpenalty\vskip\doublerulesep}}% 
\else
\fi
\fi
\expandafter}\LT@next % we are then !
} % \tabu@LT@hline

\tabu@firsthlinecorrection This is the “correction macro” for \firstline, i.e. a strut and a skip are inserted before the first \hline.
\tabu@lasthlinecorrection This is the “correction macro” for \lastline, i.e. a strut and a skip are inserted after the last \hline.
Horizontal lines: \texttt{tabucline}

\texttt{\tabucline [style or spec.]{start-end}}

\texttt{tabucline} appears only at the end of a line: this is the place where we can insert a \texttt{noalign} group. The line to be inserted inside the \texttt{tabu} is build inside this \texttt{noalign} group.

\texttt{\tabu@start} and \texttt{\tabu@stop} store the limits for the line: they are, for clarity, the local name of \texttt{\@tempcnta} and \texttt{\@tempcntb}.

\begin{verbatim}
1776 \let\tabu@start \@tempcnta
1777 \let\tabu@stop \@tempcntb
1778 \newcommand*{\tabucline}{\noalign{\ifnum0='}\fi \tabu@cline}
1779 \newcommand*{\tabu@cline[2]}{\tabu@startstop{#2}}
1780 \def\tabu@startstop #2{
1781 \ifnum \tabu@start<\z@ \omit
1782 \tabu@multispan \tabu@start \span \omit &
1783 \tabu@multispan \tabu@stop \span \omit
1784 \tabu@thehline\cr
1785 \tabu@tracinglines{(tabu:tabucline) Style: #1^^J\the\toks@}^J
1786 \fi}
1787 \futurelet \tabu@temp \tabu@xcline
\end{verbatim}
11.17 Numbers in tabu

\tabudecimal

\tabudecimal is \tabudecimal inside the tabu environment.
\ifcase 0\ifx \tabu@temp \@empty \else 
 2\fi \fi \relax
\let \tabu@getdecimal \@empty
\expandafter \tabu@skipdecimal
\else \expandafter \tabu@gobblespace \expandafter \tabu@scandecimal
\fi
\else \expandafter \tabu@scandecimal
\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
\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
\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
\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
\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
\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
\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
\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
\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
\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
\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
\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
\fi
\fi
\fi
\fi
\fi
\fi
\fi
\fi
\fi
11.18 Verbatim inside tabu with X columns

\tabu@verbatim

Setup to be done before \scantokens to allow verbatim inside the tabu environment.

\verbatim

\def\tabu@verbatim{% \let\verb \tabu@verb \let\FV@DefineCheckEnd \tabu@FV@DefineCheckEnd \def\tabu@verbatim{% \let\verb \tabu@verb \let\FV@DefineCheckEnd \let\FV@@DefineCheckEnd \let\FV@@@@DefineCheckEnd \edef\FV@EndScanning{% \def\noexpand\next{\noexpand\end{\FV@EnvironName}}\global\let\noexpand\FV@EnvironName\relax \noexpand\next} % \tabu@verbatim \def\tabu@FV@CheckEnd#1{% \expandafter\FV@@CheckEnd\detokenize{#1\end{}}\@nil} \edef\tabu@FV@@@CheckEnd\detokenize{\end{}} \begingroup\catcode'\[1 \catcode'\[2 \edef\x[\endgroup

Compatibility with \LaTeX’s kernel \verb command

\tabu@verb

The \verb macro from the latex kernel expands \@ifstar in a context where the space token □ has a category code of 12.

This is not compatible with \scantokens since \scantokens adds a space after each control sequence, including \verb:

\verb +some verbatim text+ becomes:
\verb □+some verbatim text+

and thus, the space token □ is set as the \verb delimiter.

We therefore use (a silly) \@ifstar in order to gobble the possible space token.

\verbatim

\def\tabu@ltx@verb \verb
\def\tabu@verb{\@ifstar {\tabu@ltx@verb*} \tabu@ltx@verb}

Compatibility with the fancyvrb package

\tabu@FV@DefineCheckEnd

This is quite the same issue as for \LaTeX \verb command: a space is inserted after each control sequence scanned by \scantoken.

This leads to a break in the macro that checks the end of a Verbatim environment, since this macro basically checks for a line that conforms to the pattern:

#1\end{#2}#3

while with \scantokens, such a line becomes:

#1\end{□}#3

in a context where the space token is not of category 10 (space).

Thus we replace the end-check for the Verbatim environment by a check on the detokenized-line (with ε-\TeX \texttt{\detokenize}):

\verbatim

\def\tabu@fancyvrb {% \def\tabu@FV@DefineCheckEnd {\def\FV@DefineCheckEnd\tabu@FV@DefineCheckEnd}%

\verbatim

\def\tabu@FV@DefineCheckEnd #1{% \edef\x[\endgroup
\tabu@FV@ListProcessLine  This macro replaces \FV@ListProcessLine when measuring the natural width of a \texttt{Verbatim} environment (see \tabu@startpboxmeasure)

\begin{verbatim}
1932 \def\noexpand \tabu@FV@CheckEnd {\detokenize\end{}\end{}\detokenize{}}
1933 \x \nil {\def \@tempa (#2) {\def \@tempb (#3) {}}}
\end{verbatim}

11.19 \savetabu

\savetabu  When this command is called by the user, the \texttt{tabu} preamble and target are globally stored into a macro \tabu@saved\langle user-name\rangle.

\begin{verbatim}
1947 \newcommand*{\savetabu}[1][]{\noalign{\vsetabu\savepreamble@\saveparams@}}
1948 \x {\tabu@sanitizearg{#1}{\tabu@target}{\tabu@preamble}}
1949 \ifx {\tabu@temp}{\empty} {\texttt{The tabu will not be saved}}\else
1950 \ifdefined{\tabu@restored}{\let\csname \tabu@saved\tabu@temp \endcsname \tabu@restored}
1951 \edef{\tabu@preamble}{\noexpand\the\toks0}{\noexpand\the\toks1}
1952 \edef{\tabu@savedparams}{\noexpand\the\toks0}
1953 \edef{\tabu@savedpream}{\noexpand\the\toks1}
1954 \fi
1955 \fi
1956 \end{verbatim}
11.20 \rowfont

Setting font and alignment specification

\rowfont uses the control sequences \tabu@celllalign, \tabu@cellleft, \tabu@cellright and \tabu@cellralign which have been placed on purpose into the user-defined tokens inserted in any preamble by the array package.

\tabu@celllalign and \tabu@cellralign are used to modify the alignment. If the optional [alignment] parameter of \rowfont is not specified, then those control sequences expand to #{@empty.}

\tabu@cellleft contains the font-modification information. Placement of those control sequences into the user-tokens that are inserted in the preamble by the array package is explained below under the macro \tabu@prepnext@tok.
Preparing stuff to be able to use `\rowfont`

`\tabu@prepnext@tok` will replace `\prepnext@tok` defined in `array.sty` (only inside a `tabu` environment). Its purpose is to count the number of columns, and to insert the control sequences `\tabu@cellalign`, `\tabu@cellleft`, `\tabu@cellright`, and `\tabu@cellralign` at the edge of each cell of the tabular. This is done by putting them inside the user-tokens placed around each column by the `array` package.

`\prepnext@tok` in `array.sty` initialises each user-token to an empty one, each time there is a need for a new one! The macro has a very simple definition, but its expansion is the occasion to look carefully at the counters `\count@` and `\@tempcnta` which gives us all information required to decide is the token in preparation will be finally placed on the left or on the right of a column.

When a column is inserted in the tabular preamble (`\@preamble`), the \TeX\ counter `\count@` is equal to `\@tempcnta + 1` (i.e., the right token) and the counter `\@tempcnta` is equal to `\count@` (i.e., the left token). If the column is special (i.e., `@` or `!`) `\@tempcnta` is not updated.

Thus, when a new token is “prepared” by `\prepnext@tok`:

**either:** \( i = \count@ = \@tempcnta \) : the token to prepare (i.e., `\toks < i + 1 >`) is the right one of a “normal” column. The switch `\iftabu@cellright` is set to `true`.

The previous token (\( \toks < i > = \count@ \)) is necessarily the left one of this “normal” column: we prepend `\tabu@cellalign` and append `\tabu@cellleft` to this token (\( \toks < i > \)). This token is finished and will not change afterwards.

**or:** \( i = \count@ = \@tempcnta + 1 \) : the token to prepare (\( \toks < i + 1 > \)) is either the left one of a normal column, or the single one of a special `@` or `!` column.

If the switch `\iftabu@cellright` is true, then the previous token `\toks < i >` is the right one of the last inserted column (which was a “normal” column, thus): `\tabu@cellright` `\tabu@cellralign` is appended to it, and the switch `\iftabu@cellright` is reset to `false`. May be `\prepnext@tok` will be
expanded again (by \texttt{savedecl}): if it happens, then again \texttt{\count@ = @\tempcnta + 1} (same case) but \texttt{\iftabu@cellright} is false and nothing is changed.

\textbf{else:} The token to prepare (which is \texttt{\toks < i + 1 > = \toks \count@ + 1}), cannot be the right one of a “normal” column: \texttt{\iftabu@cellright} is set to false. The fact that \texttt{\count@ - @\tempcnta} > 1 tells us that the previous token \texttt{\toks < i >} is necessarily the single one of a “special” @ or ! column. We don’t modify this token, as long as \texttt{special columns are always inserted as is: \rowfont} has no effect on special columns, nor \texttt{\rowfont}.

Thereafter, the original initialisation sequence occurs: \texttt{\advance \count@ by \@ne} and initialize the token to prepare (\texttt{\toks \count@ = \toks < i + 1 >}) to an empty one.

\begin{verbatim}
2046 \newif \iftabu@cellright
2047 \def \tabu@prepnext@tok{%
2048 \ifnum \count@<\z@ % <first initialisation>
2049 \@tempcnta \@M % <not initialized by array.sty>
2050 \tabu@nbcols \z@
2051 \let \tabu@fornoop ORI \@fornoop
2052 \iftabu@cellrightfalse
2053 \else
2054 \ifcase \numexpr \count@-@\tempcnta \relax % (case 0): prev. token is left
2055 \advance \tabu@nbcols \@ne
2056 \iftabu@cellright % before-previous token is right and is finished
2057 \tabu@cellrightfalse % <only once>
2058 \iftabu@cellrightfalse
2059 \else % special columns: do not change the token
2060 \iftabu@cellright % before-previous token is right
2061 \tabu@cellrightfalse
2062 \tabu@righttok
2063 \or % (case 1) previous token is right
2064 \tabu@cellrighttrue \let \@fornoop \tabu@lastnoop
2065 \iftabu@cellrighttrue % before-previous is not finished
2066 \tabu@cellrightfalse
2067 \tabu@righttok
2068 \fi % \iftabu@cellrighttrue
2069 \fi % \iftabu@cellrighttrue
2070 \etab@prepnext@tokORI
2071 \def \tabu@prepnext@tok{%
2072 \long \def \tabu@lastnoop#1\@@#2#3{%\backslash \tabu@lastnoopp #2\@nextchar \in@\in@@}
2073 \def \tabu@lastnoopp #1\@nextchar #2#3\in@@{%
2074 \ifx \in@#2\else
2075 \let \@fornoop \tabu@fornoop ORI
2076 \xdef \tabu@mkpreambuffer{\backslash \tabu@nbcols \the \tabu@nbcols \backslash \tabu@mkpreambuffer}%
2077 \backslash \toks0\expandafter{\expandafter{\backslash \everyrowtrue \the \toks0}}%
2078 \expandafter \prepnext@tok
2079 \fi
2080 \etab@lastnoop
2081 \def \tabu@righttok{%
2082 \advance \count@ \@ne
2083 \toks@\count@\expandafter{\the \toks@ \tabu@cellright \tabu@cellralign}%
2084 \advance \count@ \@ne
2085 \etab@righttok
2086 \def \tabu@lefttok{%\toks@\count@\expandafter{\expandafter{\backslash \tabu@celllalign
2087 \the \toks@ \tabu@cellleft} % after because of \$
2088 \etab@lefttok
\end{verbatim}
Neutralisation of glues and alignment modification

First initialisation to `\empty`.

Set up macros to modify the alignment. The skips inserted to make the standard alignment specified in the `tabular` preamble are not the same with standard `array` tabulars and `colortbl` tabulars, hence the switch to `tabu`/`colortbl`.

```
\newcommand{\tabu@cellleft}{\let\tabu@cellleft\empty}
\newcommand{\tabu@cellalign}{\let\tabu@cellalign\empty}
\newcommand{\tabu@cellright}{\let\tabu@cellright\empty}
\newcommand{\tabu@cellralign}{\let\tabu@cellralign\empty}
\newcommand{\tabu@cell@align}{% force alignment to left
  \tabu@cell@align
  \tabu@removehfil \raggedright \tabu@cellleft}
\newcommand{\tabu@cell@c}{% force alignment to center
  \tabu@cell@align
  \tabu@removehfil \centering \tabu@flush{.5}\tabu@cellleft}
\newcommand{\tabu@cell@r}{% force alignment to right
  \tabu@cell@align
  \tabu@removehfil \raggedleft \tabu@flush1\tabu@cellleft}
\newcommand{\tabu@cell@j}{% force justification (for p, m, b columns)
  \tabu@cell@align
  \tabu@justify\tabu@cellleft}
\newcommand{\tabu@justify}{%}
\newcommand{\leftskip}{\z@skip \@rightskip\leftskip \rightskip\@rightskip
\parfillskip\@flushglue
\newcommand{\tabu@cell@L}{% force alignment to left (ragged2e)
  \tabu@cell@align
  \tabu@removehfil \RaggedRight \tabu@cellleft}
\newcommand{\tabu@cell@c}{% force alignment to center (ragged2e)
  \tabu@cell@align
  \tabu@flush1\tabu@ignorehfil}
\newcommand{\tabu@cell@r}{% force alignment to right (ragged2e)
  \tabu@cell@align
  \tabu@flush1\tabu@ignorehfil}
```
\tabu@flush 1 \tabu@cellleft}
\tabu@ignorehfil
\RaggedLeft

\def\tabu@flush#1{% 
\iftabu@colortbl % colortbl uses \hfill rather than \hfil
\hskip \ifnum13<\currentgrouptype \stretch{#1}\
\else \ifdim#1pt<\p@ \tabu@cellskip 
\else \stretch{#1}\fi\fi \relax
\else % array.sty
\ifnum 13<\currentgrouptype
\hfil \hskip1sp \relax \fi
\fi

\def\tabu@removehfil{\iftabu@colortbl
\unkern \tabu@cellskip =\lastskip 
\ifnum\gluestretchorder \tabu@cellskip =\tw@ \hskip-\tabu@cellskip 
\else \tabu@cellskip \z@skip \fi
\else 
\ifdim\lastskip=1sp\unskip\fi 
\ifnum\gluestretchorder\lastskip =\@ne
\hfilneg % \hfilneg for array.sty but not for colortbl...
\fi
\fi
\fi
\def\tabu@nohfil{% \hfil -> do nothing + restore original \hfil
\def\hfil{\let\hfil \tabu@hfil}% local to (alignment template) group
\def\tabu@colortblalignments {% if colortbl
\def\tabu@nohfil
\def\tabu@colortblalignments {% if colortbl
\let\tabu@hfil \hfil
\let\tabu@hfill \hfill
\let\tabu@hskip \hskip
\def\tabu@removehfil
\let\tabu@hfil \hfil
\let\tabu@hfill \hfill
\let\tabu@hskip \hskip
\def\tabu@removehfil removes (eventually) the infinite stretchable glue inserted before the cell (in the preamble of \halign) to make the column alignment.
\let\tabu@hfil \hfil
\let\tabu@hfill \hfill
\let\tabu@hskip \hskip
\def\tabu@removehfil
\let\tabu@hfil \hfil
\let\tabu@hfill \hfill
\let\tabu@hskip \hskip
\def\tabu@removehfil removes (eventually) the infinite stretchable glue inserted after the cell (in the preamble of \halign) to make the column alignment.
11.21 Taking care of footnotes and \arraybackslash

Footnotes and hyperfootnotes

Footnotes and hyperfootnotes The macros in case hyperref is not used, or used with the option hyperfootnotes=false:

Footnotes and hyperfootnotes The macros in case hyperref is loaded with the option hyperfootnotes=true:

\centering, \raggedright, \raggedleft and @normalcr

Inside tabu environment, no need to add \arraybackslash after such commands.

\begin{verbatim}
2233 \def\tabu@latextwoe {%
2234 \def\tabu@temp##1##2##3{{\toks@\expandafter{##2##3}\xdef##1{\the\toks@}}}
2235 \tabu@temp \tabu@centering   \centering   \arraybackslash
2236 \tabu@temp \tabu@raggedleft \raggedleft \arraybackslash
2237 \tabu@temp \tabu@raggedright \raggedright \arraybackslash
2238 }% \tabu@latextwoe
2239 \def\tabu@raggedtwoe {%
2240 \def\tabu@temp##1##2##3{{\toks@\expandafter{##2##3}\xdef##1{\the\toks@}}}
2241 \tabu@temp \tabu@centering   \centering   \arraybackslash
2242 \tabu@temp \tabu@raggedleft \raggedleft \arraybackslash
2243 \tabu@temp \tabu@raggedright \raggedright \arraybackslash
2244 \tabu@temp \tabu@justifying \justifying \arraybackslash
2245 }% \tabu@raggedtwoe
2246 \def\tabu@normalcrbackslash{%\let\\@normalcr}
2247 \def\tabu@trivlist{%\expandafter\def\expandafter\@trivlist\expandafter{%\tabu@normalcrbackslash \@trivlist}}
\end{verbatim}

Utilities: tabu \fbox
\tabu@fbox works exactly like LATEX \fbox but allows the syntax: \fbox \bgroup...\egroup suitable for use inside tabular columns. \fbox is \let to \tabu@fbox at the entry inside a tabu environment.

\begin{verbatim}
2249 \def\tabu@fbox {%\leavevmode\afterassignment\tabu@beginfbox \setbox\@tempboxa\hbox}
2250 \def\tabu@beginfbox {%\bgroup \kern\fboxsep 
2251 \bgroup\aftergroup\tabu@endfbox \setbox\z@ \hbox}
\end{verbatim}

\tabu@fcolorbox works exactly like xcolor \fcolorbox but allows the syntax: \fcolorbox \{frame color\}{background color}\bgroup...\egroup suitable for use inside tabular columns. \fcolorbox is \let to \tabu@fcolorbox at the entry inside a tabu environment.

\begin{verbatim}
2254 \def\tabu@color@b@x #1#2{%\leavevmode \bgroup 
2255 \def\tabu@docolor@b@x{#1{#2\color@block{\wd\z@}{\ht\z@}{\dp\z@}\box\z@}}% 
2256 \afterassignment\tabu@begincolor@b@x \setbox\z@ \hbox 
2257 }% \tabu@color@b@x
2258 \def\tabu@begincolor@b@x {\kern\fboxsep \bgroup 
2259 \aftergroup\tabu@endcolor@b@x \set@color} 
2260 \def\tabu@endcolor@b@x {\kern\fboxsep \egroup
2261 \dimen@\ht\z@ \advance\dimen@ \fboxsep \ht\z@ \dimen@ 
2262 \dimen@\dp\z@ \advance\dimen@ \fboxsep \dp\z@ \dimen@ 
2263 \tabu@docolor@b@x \egroup}
2264 }% \tabu@endcolor@b@x
\end{verbatim}

11.22 Corrections

delarray compatability fix for colortbl and arydshln

Both colortbl and arydshln forgot the control sequence \@arrayright which must be expanded by \endarray. Originally defined for delarray, this control sequence is used by tabu environments when tabu X columns are present in the preamble.

Here is the fix. We test if \endarray contains \@arrayright before modifying the control sequence, in case colortbl and/or arydshln modify their definition.

\begin{verbatim}
2265 \def\tabu@fix@arrayright {%% \@arrayright is missing from \endarray
\end{verbatim}
arydshln @ columns

arydshln, colors without colortbl and empty p columns

arydshln redefines @endpbox for p columns. The definition is stored in \adl@act@endpbox. Here it is:

\unskip \ifhmode \nobreak 
\vrule\@width\z@\@height\z@\@depth\z@\fi \fi 
\egroup \adl@colhtdp \box\adl@box \hfil 

The \vrule inserted is exactly what package array calls: @finalstrut @arstrutbox.

However, just like in array.sty, this array-strut should be inserted inconditionnally, and \ifhmode applies only to \nobreak (misplaced \fi in arydshln definition).

Finally, arydshln is not compatible with colors in columns, such that: \p3in, Unless colortbl is also loaded, the color group is missing.
Fixed inside \texttt{tabu} environment.

\begin{verbatim}
\def\tabu@adl@act@endpbox {
  \unskip \ifhmode \nobreak \fi \@finalstrut \@arstrutbox
}

\def\tabu@adl@fix {
  \let\adl@xarraydashrule \tabu@adl@xarraydashrule % <fix> arydshln
  \let\adl@act@endpbox \tabu@adl@act@endpbox % <fix> arydshln
  \let\adl@act@@endpbox \tabu@adl@act@endpbox % <fix> arydshln
  \let\@preamerror \@preamerr % <fix> arydshln
}
\end{verbatim}

\longtable \@startpbox

The leading strut should be inserted at \texttt{everypar} in order for \texttt{\tabulinesep} to work (otherwise, \LaTeX{} is in horizontal mode and \texttt{\nointerlineskip} breaks).

\begin{verbatim}
\def\tabu@LT@startpbox #1{%
  \bgroup
  \let\@footnotetext\LT@p@ftntext
  \setlength{\hsize}{#1}\%
  \@arrayparboxrestore
  \everypar{%
    \vrule \@height \ht@arstrutbox \@width \z@
    \everypar{}%}
}\end{verbatim}

\section{Package options and Initialisation}

\texttt{\tracingtabu} and the package options

\texttt{\delarray} (package option) The \texttt{\delarray} package option is only there for convenience: it simply loads the \texttt{\delarray} package.

\begin{verbatim}
\DeclareOption{delarray}{\AtEndOfPackage{\RequirePackage{\delarray}}}
\end{verbatim}

\texttt{\linegoal} (package option) The \texttt{\linegoal} package option only sets \texttt{\tabudefaulttarget} to be equal to \texttt{\linegoal}. The required package \texttt{\linegoal} is loaded.

\begin{verbatim}
\DeclareOption{\linegoal}{%\AtEndOfPackage{\RequirePackage{\linegoal}[2010/12/07]%\let\tabudefaulttarget \linegoal% \linegoal is \linewidth if not pdf\TeX{}%}}
\end{verbatim}

\texttt{\scantokens} (package option) The \texttt{\scantokens} package option makes \texttt{\tabu} equal to \texttt{\tabu*.}

\begin{verbatim}
\DeclareOption{\scantokens}{\tabuscantokenstrue}
\end{verbatim}

\texttt{\tracingtabu} \texttt{\scantokens} (package option) The \texttt{\tracingtabu} is the same as the package option \texttt{\debugshow}.

\begin{verbatim}
\def\tracingtabu \atendofpackage{\tracingtabu=\tw@}
\end{verbatim}

\texttt{\debugshow} (package option)

\begin{verbatim}
\def\debugshow \atendofpackage{\debugshow=\tw@}
\end{verbatim}

\texttt{\begingroup} \texttt{\ifnextchar=}

\texttt{\afterassignment} \texttt{\tabu@tracing\count@}

\texttt{\afterassignment} \texttt{\tabu@tracing\count@\relax}

\texttt{\expandafter} \texttt{\tabu@tracing\expandafter\indentgroup}

\texttt{\expandafter} \texttt{\tabu@tracing\expandafter\indentgroup\the\count@ \relax}

\texttt{\tabu@tracing}

\texttt{\ifnum1>\thr@@ \let\tabu@tracinglines\message}

\texttt{\let\tabu@tracinglines\gobble}

\texttt{\fi}

\texttt{\ifnum1>\tw@ \let\tabu@DBG \tabu@DBG}

\texttt{\fi}

\texttt{\fi}

\texttt{\fi}

\texttt{\fi}

\texttt{\fi}

\texttt{\fi}

\texttt{\fi}

\texttt{\fi}

\texttt{\fi}
Initialisation and setup \AtBeginDocument

At the end of the tabu package:

- \tracingtabu is set to 0: this initialises the message commands. Eventually, the value will be overwritten by the \debugshow package option later.
- \everyrow is set to empty: this initialises the process at \everycr to the default process,
- a new empty line style is defined, to be equivalent to \hline: this creates the default leaders, which will be used if a line style specification cannot be parsed successfully.

Then this default line style is set to be the current one.

At Begin Document, a fix for arydshln and colortbl compatibility with delarray shortcuts available inside tabu: requirement for this fix is checked by \fixarrayright.

Then the switch \iftabu@colortbl is set.

Finally, the longtabu environment is defined only if the longtable package is detected.

\AtBeginDocument{\tabu@AtBeginDocument}
\def\tabu@AtBeginDocument{\let\tabu@AtBeginDocument \undefined
\ifdefined\arrayrulecolor \tabu@colortbltrue \tabu@colortblalignments % different glues are used \else \tabu@colortblfalse \fi
\ifdefined\CT@arc@ \else \let\CT@arc@ \relax \fi
\ifdefined\CT@drsc@ \else \let\CT@drsc@ \relax \fi
\let\tabu@arc@L \CT@arc@ \let\tabu@drsc@L \CT@drsc@
\ifodd1\ifcsname siunitx_table_collect_begin:Nn\endcsname % <siunitx: ok>
\expandafter\ifx \csname siunitx_table_collect_begin:Nn\endcsname\relax0\fi\fi\relax
\tabu@siunitxtrue
\else \let\tabu@maybesiunitx \tabu@nosiunitx
\fi
\tabu@siunitxfalse
\ifdefined\adl@array % <arydshln>
\else \let\tabu@adl@fix \relax \fi
\ifdefined\longtable % <longtable>
\else \let\longtabu \tabu@nolongtabu \fi
\ifdefined\cellspacetoplimit \tabu@warn@cellspace \fi
\else \let\tabu@maybesiunitx \@empty % <not siunitx: setup>
\fi
\let\tabu@siunitxtrue
\let\tabu@siunitxfalse
\fi
\ifdefined\adl@array % <arydshln>
\else \let\tabu@adl@fix \relax \fi
\ifdefined\longtable % <longtable>
\else \let\longtabu \tabu@nolongtabu \fi
\ifdefined\cellspacetoplimit \tabu@warn@cellspace \fi
\csname\ifHy@hyperfootnotes\endcsname % <hyperfootnotes>
\let\tabu@footnotetext \tabu@Hy@ftntext
\let\tabu@xfootnote \tabu@Hy@xfootnote \fi
\ifdefined\FV@DefineCheckEnd% <fancyvrb>
\tabu@fancyvrb \fi
\ifdefined\color % <color / xcolor>
\let\tabu@color \color
\def\tabu@leavevmodecolor ##1{\leavevmode ##1} \expandafter\tabu@leavevmodecolor \expandafter{\color} \else
\let\tabu@color \tabu@nocolor
\let\tabu@leavevmodecolor \@firstofone \fi
\tabu@latextwoe
\ifdefined\@raggedtwoe@everyselectfont % <ragged2e>
\tabu@raggedtwoe \else
\let\tabu@cell@L \tabu@cell@l
\let\tabu@cell@R \tabu@cell@r
\let\tabu@cell@C \tabu@cell@c
\let\tabu@cell@J \tabu@cell@j \fi
\expandafter\in@ \expandafter\@arrayright \expandafter{\endarray} \ifin@ \let\tabu@endarray \endarray \else \tabu@fix@arrayright \fi% <fix for colortbl & arydshln (delarray)>
\everyrow{} \endtabu@AtBeginDocument
\def\tabu@warn@cellspace{\PackageWarning{tabu}{Package cellspace has some limitations And redefines some macros of array.sty. \MessageBreak Please use \string\tabulinesep\space to control vertical spacing of lines inside tabu environment}}
\ProcessOption ∗ is much quicker than without the star...
\tabuscantokensfalse
\let\tabu@arc@G \relax
\let\tabu@drsc@G \relax
\let\tabu@evr@G \@empty
\let\tabu@rc@G \@empty
\def\tabu@ls@G {\tabu@linestyle@}
\let\tabu@@rowfontreset \@empty % <init>
\let\tabu@rowfontreset \@empty % <init>
\let\tabu@cellalign \@empty
\let\tabu@cellralign \@empty
\let\tabu@cellleft \@empty
\let\tabu@cellright \@empty
\let\tabu@rowfontreset \@empty
\def\tabulineon {4pt} \let\tabulineoff \tabulineon
\tabu@everyrowtrue
\ifdefined\pdfelapsedtime % <pdfTeX>
\def\tabu@pdftimer {\xdef\tabu@starttime{\the\pdfelapsedtime}} \else \let\tabu@pdftimer \relax \let\tabu@message@etime \relax \fi
\tracingtabu=\z@
\newtabulinestyle {=} \maxdimen \tabu@linestyle@
\tabulinestyle{}
\taburowcolors{}
\let\tabudefaulttarget \linewidth
\ProcessOptions* % \ProcessOptions* is quicker!

(/package)
12 References

[1] *A new implementation of \LaTeX’s tabular and array environments* by Frank Mittelbach
2008/09/09 v2.4c – Tabular extension package (FMI)
CTAN:help/Catalogue/entries/array.html

[2] *The varwidth package* by Donald Arseneau
2009/03/30 ver 0.92 – Variable-width minipages
CTAN:help/Catalogue/entries/varwidth.html

[3] *The enumitem-zref package* by \texttt{FC}
2011/02/18 ver 1.8 – Extended references for enumitem pkg
CTAN:help/Catalogue/entries/enumitem-zref.html

13 History

[2011/02/26 v2.8]
- Bug in the starred version (with \texttt{scantokens}) of the longtabu* environment.

[2011/02/25 v2.7]
- Automatic \texttt{par} after the end of the tabu environment used with its default target is removed in case of tabu spread: this was a bug.
- Some \texttt{\ignorespaces} were missing (in \texttt{\everyrow}, \texttt{\taburulecolor}, \texttt{\taburowcolors} and \texttt{\tabulinestyle}).

[2011/02/24 v2.6]
- \texttt{\savetabu} now also saves \texttt{\tabulinesep} (i.e. \texttt{\aboveatabulinesep} and \texttt{\belowatabulinesep})
- Bug fixed for custom-environments when nested.
- \texttt{\taburulecolor} works even if \texttt{colortbl} is not loaded for the tabu environment.
  This is now the same for the longtabu environment.

[2011/02/19 v2.5]
- Bug fixed for \texttt{\pdfeIanalyzed} when compilation without pdf\LaTeXX.
- Modification of \texttt{\@finalstrut} (“null-rule” added) to avoid problems with \texttt{\columncolor}.

[2011/02/17 v2.4]
- Documentation revisited

[2011/02/13 v2.3]
- Fixed two bugs for nested tabu environment: when using \texttt{\rowfont} and when tabu is nested inside longtabu

[2011/02/12 v2.2 – New implementation - Absolutely no modification of array.sty]
- $\tau_{\nu,bc}$ has been totally reimplemented, including the algorithms.
  In particular, outside of the tabu environment, absolutely none of the macros of array.sty, (and obviously none of \LaTeXX) is modified.

  The process has been completely reinvented: tabu follows a path along different modes (or strategies) measuring natural width of cells, fixing $X$ column widths, measuring vertical length of rows and then printing the final tabular. The process is optimized, especially in the case of nested tabu environments: a tabular is not built twice for measuring purpose...
  As a result, many new features are now possible... vertical leaders (dashed lines), dynamic vertical spacing adjustment, and hopefully still more in a next release.

  tabu now systematically collects the environment body. But with \texttt{scantokens}, it is possible to insert verbatim material inside the columns: use tabu* instead of tabu, for the outer most tabular.
• New: \firstline and \lastline can draw multiple lines, and there is an option to set \extratabsurround instantly, and locally.
• New: \taburulecolor with a good behaviour with groupings (like \everyrow)
• Modification: \tabulinestyle sets the line style for the tabu, \newtabulinestyle defines a new line style.

<table>
<thead>
<tr>
<th>This</th>
<th>is</th>
<th>the new</th>
<th>(\text{T}_R b\subset)</th>
<th>package</th>
</tr>
</thead>
</table>

[2011/01/19 v2.1]
• Vertical spacing had a bug with longtabu and paragraph columns.
  Fixed.
• New: \everyrow.
• Fix a bug of \rowfont when using siunitx S columns.
• Some code optimisation.
• To do (if possible): a syntax X[6mc]S[...] to “embed” siunitx S column inside tabu and longtabu X columns...

[2011/01/18 v2.0]
• Vertical spacing of lines implemented ! See \tabulinesep and \extrarowsep.
• \tabulinestyle : user defined line style can now be used inside the optional argument of the |[...]| preamble token.
• |[...]| is now allowed in \multicolumn preamble inside tabu environment.
• Bug fixed inside \tabu@prepnexttok (again !!! - a difficult case !)
• Incompatibility of package cellspace with tabu spread and tabu with negativ coefficients for X columns with has been lifted.
  However, as said in the documentation of package cellspace, S column modifier does not work in the case of nested tabulars.
  The S column modifier becomes C when the package siunitx is loaded (see siunitx documentation).
  Moreover, cellspace does not work with color or xcolor and paragraph column types !!
• Finally, cellspace redefines globally \@startpbox and \@endpbox and is therefore not fully compatible with array.sty and therefore with \(\text{T}_R b\subset\).
  For all those reasons, \(\text{T}_R b\subset\) displays a warning to discourage the use of cellspace with the tabu environment.

[2011/01/15 v1.9]
• Bug in \savetabu when used inside longtabu...
• Bug when tabu with X column is nested inside lontabu.
• Documentation (\rowfont was missing in the summary).

[2010/12/28 v1.8]
• \tracingtabu / debugshow package option:
  reporting of the time elapsed during trials (if \pdfelapsedtime and thus pdf\TeX{} is available)
  Slight modifications for better reporting on the .log file.
• Fix a bug when \savetabu is used after \multicolumn (\multicolumn globally redefines \@preamble).
• Fix a bug with \tabucline and \CT@arc@ (colortbl).
• Better privacy of columns types specifically defined for tabu.
• Improvement in the rewriting process (but only very few people should notice...)
• Documentation.
[2010/12/18 v1.7]

- Code optimisation
- Modification in the columns rewriting process (bug with some new column types defined by the user).

[2010/12/07 v1.5]

- Implementation of negative width coefficients for \texttt{X} columns (cf. \texttt{tabu X columns – Mastering horizontal space point 2}).
- Columns natural widths computation (for \texttt{tabu spread} with \texttt{X} columns and negative coefficients) is based on the code of the \texttt{varwidth} package by Donald Arseneau.
- \texttt{longtabu} is now provided, based on the \texttt{longtable} package by David Carlisle. \texttt{longtabu} can be used just like \texttt{tabu}.
- Vertical lines can be used whatever the catcode of | is.
- \texttt{\savetabu} reports saved informations in the .log (\texttt{debugshow} option).
- \texttt{\savetabu... \usetabu} now restores the \texttt{\halign} preamble rather than the \texttt{tabu} preamble! \texttt{\preamble} can be use in the \texttt{tabu} preamble to restore a \texttt{tabu} preamble.
- \texttt{\tabucline} is more robust with “special” preambles containing > or < tokens. \texttt{\tabucline} now takes care of \texttt{\arrayrulecolor} (package \texttt{colortbl}).
- \texttt{enumitem-zref} package has been added to the documentation (see the link point 1)
- Optimisation of some parts of the code.

[2010/11/22 v1.4]

- Compatibility improvement with \texttt{linegoal} for the syntax: \texttt{\begin{tabu} to\linegoal {...}}
- Hyper footnotes now work correctly.
- Fix a bug when using colored vertical lines in \texttt{tabu} in math mode.
- Fix a bug with vertical lines and \texttt{colortbl} \texttt{\arrayrulecolor} specification.
- Fix a compatibility bug with \texttt{arydshln}: when nesting a \texttt{tabular} that use vertical dashed lines (\texttt{arydshln}) inside \texttt{tabu spread} with \texttt{X} columns.

[2010/11/18 v1.3]

- Fix a bug that may appear in \texttt{\tabucline} depending on the preamble due to arbitrary \texttt{\countdef}.
- Improvement in the use of \texttt{\everycr}: no \texttt{\global} stuff. Thus bug fixed when nesting \texttt{tabu} inside \texttt{AMS-align} environment for example. Same issue with \texttt{\rowfont} which now works without global modification of \texttt{\everycr}.
- No phantom line is added to \texttt{tabu} but a command \texttt{\tabuphantomline} is provided for this purpose (required with \texttt{\multicolumn} in some cases).
- Improvement on vertical alignment.
- To do: an example file to test a wide range of possibilities...
- Documentation.
14 Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; plain numbers refer to the code lines where the entry is used.

Symbols
\& .................................................. 80
\arrayparboxrestore .................................. 2322
\arrayright .......................... 1034, 1041, 1048, 2265, 2270, 2275, 2282, 2284, 2286, 2409
\arraystretch ................................. 693
\arstrutbox 109, 117, 118, 123, 624, 691, 692, 821, 822, 1003, 1284, 1555, 1638, 1654, 1726, 1730, 1733, 1739, 1742, 1744, 2308, 2324
\defaultunitex 330, 331, 858, 1063, 1794, 1838, 1839
\endbox ........................................... 615, 638
\finalstrut ............................... 873, 1041, 1131, 1267, 1280, 1281, 1993
\finnextchar .................................. 166, 168, 177, 197, 199, 208, 237, 285, 854, 2335
\mkpream ..................................... 834
\normalcr .......................... 626, 2246
\onelevel@sanitize .............................. 97
\preamble ..................................... 1077, 1081, 1085, 1092, 1502, 1964, 1971, 2271, 2276, 2283
\raggedtwo@everyselectfont .................. 2402
\rightskip .................................. 2126
\sharp ........................................... 1978
\sp@token 64, 563, 911, 1065, 1857, 1891
\stbararray .................................... 616, 639
\tempb ....................................... 1933
\tempcnta 25, 1123, 1776, 2049, 2054
\tempcntb 26, 1777
\tempdimb 36, 116, 118, 1343, 1349, 1360, 1363, 1370, 1400, 1403, 1463, 1467, 1472, 1475, 1615, 1618, 1619, 1627, 1630, 1632, 1720, 1732, 1739, 1742, 1744
\tempdimc 34, 37, 1385, 1387, 1391, 1393, 1396, 1439, 1440, 1625, 1719, 1739, 1745
\tempstokena 746, 751, 767, 857, 861, 1017, 1020, 1064, 1849, 1865, 1869, 1900, 1903
\thefnmark .................................. 2243, 2212
\trivlist ...................................... 2247, 2248
\whilesw .................................. 1357
\xfootnote 628, 1127, 2198
\xfootnotetext 627
\xhline ....................................... 623
\{ ............................................. 1930
\} .............................................. 1930

A
\above@tabulinesep 30, 200, 202, 204, 209, 211, 213, 216, 217, 741, 1515, 1614, 1626
\adl@endarray .................................. 2269, 2281
\afterassignment .................................. 142, 173, 176, 204, 207, 238, 240, 241, 483, 575, 947, 1243, 2249, 2256, 2336, 2337
\aftergroup ..................................... 146, 163, 164, 194, 195, 657, 660, 834, 996, 997, 1243, 1546, 1830, 2181, 2251, 2259
\array ........................................ 540
\arraybackslash .................................. 1544, 2235, 2236, 2237, 2241, 2242, 2243, 2244
\arraycolsep ..................................... 734
\arrayrulecolor .................................. 2367
\arrayrulewidth 429, 736, 1510, 1682, 1706, 1707, 1765, 1769, 1772, 2298
\arraystretch .................................. 110, 111, 115, 116, 126, 131, 743, 1517, 1649

B
\baserulesep .................................. 2020
\begin ........................................ 8, 9
\below@tabulinesep 30, 200, 202, 209, 211, 213, 216, 217, 742, 1516, 1615, 1627

C
\c@footnote ..................................... 2199
\c@page .................................... 1422, 1424
\c@taburow .................................. 21, 254, 349, 351, 352, 369, 373, 685, 721, 758, 1636, 1637, 1639, 1641, 1642, 1643, 1644, 1646, 1650
\c@taburow@save .................................. 685, 721
\catcode 10, 11, 80, 92, 446, 447, 448, 449, 1929
\centering ..................................... 630, 924, 2138, 2140, 2241
\centering ..................................... 628, 920, 2294, 2210, 2211, 2235
\color ........................................ 1139
\cleaders .................................. 1802
\cline ........................................ 1702, 1707, 1712
\color ........................................ 30, 90, 122, 290, 303, 312, 320, 507, 624, 1128, 1501, 2036, 2393, 2394, 2397
\color@ ........................................ 488
\color@b@x .................................. 633
| Command Name | ... | 1865, 1893 | ... | 1868, 1895 | ... | 483, 485, 495 | ... | 417, 481 | ... | 1850, 1853, 1859 | ... | 1850, 1855, 1864 | ... | 266, 272, 388, 804, 1804 | ... | 406, 407, 463, 476, 480 | ... | 573, 574, 575 | ... | 163, 188 | ... | 1332, 1339, 1348 | ... | 60, 68, 1066, 1861 | ... | 60, 166, 169, 171, 178, 180, 197, 200, 202, 209, 211 | ... | 60, 116, 1164 | ... | 62, 63, 67 | ... | 190, 221, 224 | ... | 182, 186, 213, 217, 224 | ... | 45, 511 | ... | 619, 180, 184, 1845, 1846, 1847, 1848, 1849 | ... | 619, 180, 184, 1845, 1846, 1847, 1848, 1849 | ... | 1676, 1677, 1678, 1679, 1685 | ... | 1697, 1699, 1705 | ... | 1696, 1714 | ... | 1701, 1706 | ... | 968, 972, 999, 1336, 1532, 1533, 1538 | ... | 2167, 2183, 2187 | ... | 2166, 2188 | ... | 528, 530, 531, 532, 533, 534, 535 | ... | 2036 | ... | 60, 843 | ... | 204, 2110, 2163, 2139, 2145, 2181 | ... | 144, 147 | ... | 603 | ... | 603 | ... | 922, 926, 2121, 2123, 2125, 2128 | ... | 632, 2244 | ... | 808, 810, 817 | ... | 621, 1676 | ... | 622, 1676 | ... | 2072, 2073 | ... | 2062, 2072, 2080 | ... | 88, 89 | ... | 1132 | ... | 2233, 2238, 2240 | ... | 501, 506 | ... | 499, 500, 503, 819, 1810, 1813 | ... | 500, 522 | ... | 511, 518, 527, 531, 532, 535 | ... | 2036, 2395, 2396, 2397, 2400 | ... | 2060, 2086, 2088 | ... | 576, 577, 581 | ... | 575, 576 | ... | 798, 835, 1004 | ... | 801 | ... | 197, 199, 206 | ... | 268, 2426, 2442 | ... | 539 | ... | 996, 1077 | ... | 1212 | ... | 548 | ... | 276, 279, 702, 761, 803, 813, 1793 | ... | 278, 279, 667, 668, 671, 702, 754, 2426 | ... | 667, 701 | ... | 275, 276, 668, 749, 754, 761, 1522 | ... | 625, 1751 | ... | 554, 2318 | ... | 1910, 1911 | ... | 155, 2094, 2377 | ... | 1194, 1200, 1212 | ... | 619, 1002, 1533, 1371, 1989 | ... | 872, 1565, 1537 | ... | 878, 879, 1341, 1345, 1361, 1377, 1390, 1392, 1397, 1455, 1526, 2356, 2360 | ... | 1345, 1425 | ... | 878, 1473 | ... | 1560, 1575, 1580 | ... | 1197, 1454, 2439 | ... | 1442 | ... | 1377, 1451 | ... | 743, 1478, 1983, 1986, 2357 | ... | 1390, 1434 | ... | 874, 1423 | ... | 1461, 1616, 1628 | ... | 119, 689, 720, 2346, 2349 | ... | 689, 720 | ... | 815, 842, 1002, 1038, 1044, 1060, 1078, 1090, 1120, 1130, 2076 | ... | 60, 352, 369 | ... | 1481, 1482, 1484, 1498 | ... | 1362, 1363, 1364, 1367, 1426, 1427, 1428, 1431, 1435, 1436, 1437, 1438, 1439, 1465, 1467, 1470, 1475, 1481 | ... | 636, 843 | ... | 603 | ... | 1665, 1784, 1785, 1831, 1832, 1835 | ... | 168, 189 | ... | 195, 220 | ... | 30, 686, 717, 1203, 1563, 1565, 1566 | ... | 686, 717 | ... | 688, 718, 1201, 1202, 1383, 1385, 1437, 1565, 1566, 2433 | ... | 688, 718 | ... | 687, 719, 1202, 1384, 1385, 1438, 1567, 1568, 2432 | ... | 687, 719 | ... | 21, 1238, 1507, 1665, 1836, 1840, 1841, 1842, 1974, 2050, 2055, 2076 | ... | 613, 838, 1009, 1010, 1024, 1079, 1091 | ... | 544, 555 | ... | 961, 1150
<table>
<thead>
<tr>
<th>Command</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\tabulineoff</code></td>
<td>424, 2435</td>
</tr>
<tr>
<td><code>\tabulineon</code></td>
<td>422, 2435</td>
</tr>
<tr>
<td><code>\tabulinesep</code></td>
<td>12, 193, 385, 2418</td>
</tr>
<tr>
<td><code>\tabulinestyle</code></td>
<td>16, 272, 386, 2443</td>
</tr>
<tr>
<td><code>\tabuphantomline</code></td>
<td>12, 1652</td>
</tr>
<tr>
<td><code>\taburereset</code></td>
<td>18, 384</td>
</tr>
<tr>
<td><code>\taburow</code></td>
<td>21</td>
</tr>
<tr>
<td><code>\taburowcolors</code></td>
<td>18, 249, 327, 386, 2444</td>
</tr>
<tr>
<td><code>\tabuscleventhrowfalse</code></td>
<td>16, 283, 386</td>
</tr>
<tr>
<td><code>\tabuscleventhrowtrue</code></td>
<td>1099, 2421</td>
</tr>
<tr>
<td><code>\tabuscleventhrowstrue</code></td>
<td>547, 557, 2333</td>
</tr>
<tr>
<td><code>\tabuscleventhrowrule</code></td>
<td>39, 122, 128, 132, 1283, 1616, 1628, 1731, 1747, 2347, 2350</td>
</tr>
<tr>
<td><code>\tabuthepreamble</code></td>
<td>1086, 1095, 1104, 1109, 1113, 1220, 1229, 1252, 1256</td>
</tr>
<tr>
<td><code>\themepfn</code></td>
<td>2203, 2212</td>
</tr>
<tr>
<td><code>\thetaburow</code></td>
<td>21</td>
</tr>
<tr>
<td><code>\tracingtabu</code></td>
<td>24, 2334</td>
</tr>
<tr>
<td><code>\unhbox</code></td>
<td>510, 1234</td>
</tr>
<tr>
<td><code>\unkern</code></td>
<td>2170</td>
</tr>
<tr>
<td><code>\unskip</code></td>
<td>139, 2175, 2308</td>
</tr>
<tr>
<td><code>\unvbox</code></td>
<td>517</td>
</tr>
<tr>
<td><code>\usetabu</code></td>
<td>19, 746, 980, 987, 1011, 1988, 2010, 2017</td>
</tr>
<tr>
<td><code>\usetabu_{\text{(private\_column\_type)}}</code></td>
<td>980</td>
</tr>
<tr>
<td><code>\value</code></td>
<td>2202, 2211</td>
</tr>
<tr>
<td><code>\vbox</code></td>
<td>517, 1607, 1798</td>
</tr>
<tr>
<td><code>\vcenter</code></td>
<td>1606</td>
</tr>
<tr>
<td><code>\vskip</code></td>
<td>518, 1687, 1688, 1732, 1742, 1756, 1765, 1821</td>
</tr>
<tr>
<td><code>\vss</code></td>
<td>518</td>
</tr>
<tr>
<td><code>\vtop</code></td>
<td>820, 1531, 1570, 1605, 1799</td>
</tr>
<tr>
<td><code>\write</code></td>
<td>143, 146, 1125</td>
</tr>
<tr>
<td><code>\X_{\text{(private\_column\_type)}}</code></td>
<td>852</td>
</tr>
<tr>
<td><code>\xglobal</code></td>
<td>354, 366, 371</td>
</tr>
<tr>
<td><code>\xleaders</code></td>
<td>535, 1803, 1808</td>
</tr>
</tbody>
</table>