The **arydshln** package*

Hiroshi Nakashima  
(Kyoto University) 

2019/02/21

Abstract

This file gives \LaTeX{}’s \texttt{array} and \texttt{tabular} environments the capability to draw horizontal/vertical dash-lines.

Contents

1 Introduction 3

2 Usage 3
   2.1 Loading Package 3
   2.2 Basic Usage 4
   2.3 Style Parameters 4
   2.4 Fine Tuning 5
   2.5 Finer Tuning 5
   2.6 Performance Tuning 6
   2.7 Compatibility with Other Packages 7

3 Known Problems 9

4 Implementation 10
   4.1 Problems and Solutions 10
   4.2 Another Old Problem 13
   4.3 Register Declaration 14
   4.4 Initialization 17
   4.5 Making Preamble 22
   4.6 Building Columns 27
   4.7 Multi-columns 30
   4.8 End of Rows 32
   4.9 Horizontal Lines 33
   4.10 End of Environment 37
   4.11 Drawing Vertical Lines 38

*This file has version number v1.76, last revised 2019/02/21.
4.12 Drawing Dash-lines .............................................. 44
4.13 Shorthand Activation ........................................... 45
4.14 Compatibility with colortab .................................... 48
4.15 Compatibility with longtable ................................. 48
  4.15.1 Initialization .............................................. 49
  4.15.2 Ending Chunks ............................................ 51
  4.15.3 Horizontal Lines and p-Boxes ............................ 53
  4.15.4 First Chunk .............................................. 55
  4.15.5 Output Routine ........................................... 56
4.16 Compatibility with colortbl ................................. 60
  4.16.1 Initialization, Cell Coloring and Finalization ........... 62
  4.16.2 Horizontal Line Coloring ................................ 63
  4.16.3 Vertical Line Coloring ................................... 65
  4.16.4 Compatibility with longtable ............................ 68
1 Introduction

In January 1993, Weimin Zhang kindly posted a style \texttt{hvdashln} written by the author, which draws horizontal/vertical dash-lines in L\TeX{}’s \texttt{array} and \texttt{tabular} environments, to the news group \texttt{comp.text.tex}. The style, unfortunately, has a known problem that vertical lines are broken when an array contains tall rows.

In March of the year, Monty Hayes complained of this problem encouraging the author to make a new version \texttt{arydashln} to solve the problem. The new style also has new features, such as allowing ‘:’ to specify a vertical dash-line in preamble, and \texttt{\cdashline} being a counterpart of \texttt{\cline}.

In March 1999, Sebastian Rahtz kindly invited the style, which had been improved following the bug report from Takahiro Kubota, to be included in \TeX{} CTAN and also in the online catalogue compiled by Graham Williams. This invitation gave the style new users including Peter Ehrbar who wished to use it with \texttt{array} style in Standard \TeX{} Tools Bundle and had trouble because these styles were incompatible with each other. Therefore, the style became compatible with \texttt{array} and got additional new features.

In February 2000, Zsuzsanna Nagy reported that \texttt{arydashln} is not compatible with \texttt{colortab} style to let the author work on the compatibility issue again.

In February 2001, Craig Leech reported another compatibility problem with \texttt{longtable}. Although the author promised that the problem would be attacked some day, the issue had left long time\footnote{Two years and a half! Sorry Craig.} until three other complaints were made. Then the author attacked the problem hoping it is the last compatibility issue\footnote{But his hope was dashed as described below.}.

In May 2004, Klaus Dalinghaus found another incompatibility with \texttt{colortbl}. Although he was satisfied by a quick hack for cell painting, the author attacked a harder problem for line coloring to solve the problem\footnote{Without dreaming it is the last compatibility issue.}.

2 Usage

2.1 Loading Package

The package is usable to both \LaTeX\,2ε and \LaTeX\,2.09 users with their standard package loading declaration. If you use \LaTeX\,2ε, simply do the following.

\begin{verbatim}
\usepackage{arydashln}
\end{verbatim}

If you still love \LaTeX\,2.09, the following is what you have to do.

\begin{verbatim}
\documentstyle[\ldots,arydashln,\ldots]{\langle style\rangle}
\end{verbatim}

Only one caution given to users of \texttt{array} (v2.3m or later) and \texttt{longtable} (v4.10 or later) packages, included in Standard \TeX{} Tools Bundle, and \texttt{colortab} and \texttt{colortbl} package is that \texttt{arydashln} has to be loaded \texttt{after} \texttt{array}, \texttt{longtable}, \texttt{colortab} and/or \texttt{colortbl} done. That is, the following is correct but reversing the order of \texttt{\usepackage} will cause some mysterious error.

---

\begin{footnotesize}
\begin{enumerate}
  \item Two years and a half! Sorry Craig.
  \item But his hope was dashed as described below.
  \item Without dreaming it is the last compatibility issue.
\end{enumerate}
\end{footnotesize}
\usepackage{array} % and/or
\usepackage{longtable} % and/or
\usepackage{colortab} % or
\usepackage{colortbl}
\usepackage{arydshln}

2.2 Basic Usage

You can simply use \texttt{array} or \texttt{tabular(*)} environments with standard preamble, such as \texttt{array} and \texttt{tabular}, and standard commands \texttt{\textbackslash hline}, \texttt{\textbackslash cline} and \texttt{\textbackslash multicolumn}.

\begin{tabular}{r|c|ll}
  & B & C \\
\hline
A & AAA & BBB & CCC \\
\hdashline
\cdashline{1-2}
\multicolumn{2}{|l:}{AB} & C \\
\hdashline
\hdashline
\end{tabular}

Note that the intersections of leftmost/rightmost vertical lines and horizontal dash-lines are little bit different from those produced by ordinary \texttt{array/tabular}. That is, with very careful examination you will find that vertical lines of ordinary ones are broken with small white specks at intersections, while in the example above they have no specks. In addition, the four corners of outermost rectangular also have specks in ordinary ones, while those in the example above have perfect contacts of L-shape\footnote{The top-left/right corners had specks before v1.73, the fix in which made the topmost dash segment of a vertical dash-line a little bit shorter.}.

\texttt{\textbackslash hdashline} and \texttt{\textbackslash cdashline} are the counterparts of \texttt{\textbackslash hline} and \texttt{\textbackslash cline}.

\begin{tabular}{|l::c:r|}
\hline
A&B&C \\
\hdashline
AAA&BBB&CCC \\
\cdashline{1-2}
\multicolumn{2}{|l:}{AB} & C \\
\hdashline
\end{tabular}

\texttt{\textbackslash firsthdashline} and \texttt{\textbackslash lasthdashline} are available.

2.3 Style Parameters

You have two style parameters to control the shape of dash-lines: \texttt{\textbackslash dashlinedash} is for the length of each dash segment in a dash line; \texttt{\textbackslash dashlinegap} controls the amount of each gap between dash segments. Both parameters have a common default value, 4 pt.
2.4 Fine Tuning

Although you can control the shape of dash-lines in an \texttt{array/tabular} environment as described in §2.3, you might want to draw a dash-line of a shape different from others. To specify the shape of a vertical dash-line explicitly, you may use;

\begin{verbatim}
\{\langle\text{dash}\rangle/\langle\text{gap}\rangle}\}
\end{verbatim}

instead of ordinary ‘:’ and will have a dash-line with dash segments of \texttt{(dash)} long separated by spaces of \texttt{(gap)}.

As for horizontal dash-lines, explicit shape specifications may be given through optional arguments of \texttt{\hdashline} and \texttt{\cdashline} as follows.

\begin{verbatim}
\hdashline{\langle\text{dash}\rangle/\langle\text{gap}\rangle}
\cdashline{\langle\text{col1}\rangle-\langle\text{col2}\rangle}\{\langle\text{dash}\rangle/\langle\text{gap}\rangle}\}
\end{verbatim}

For example;

\begin{verbatim}
\begin{tabular}{|l::c;{2pt/2pt}r|}
\hline
A&B&C\\
\hdashline[1pt/1pt]
AAA&BBB&CCC\\
\cdashline{1-2}[.4pt/1pt]
\multicolumn{2}{|l;{2pt/2pt}}{AB}&C\\
\hdashline\hdashline
\end{tabular}
\end{verbatim}

will produce the following result.

\begin{verbatim}
A B C
AAA BBB CCC
AB C
\end{verbatim}

The vertical solid and dashed lines are drawn as if their width is zero, as standard \LaTeX{}'s \texttt{array} and \texttt{tabular} do, if you don’t use \texttt{array} package. Otherwise, they have real width of \texttt{\arrayrulewidth} as the authors of \texttt{array} prefers. However, you may explicitly tell \texttt{arydshln} to follow your own preference by \texttt{\ADLnullwide} if you love \LaTeX{} standard, or \texttt{\ADLsomewide} if you second the preference of \texttt{array} authors.

2.5 Finer Tuning

To draw dash-lines, we use a powerful primitive of \LaTeX{} called \texttt{xleaders}. It replicates a segment that consist of a dash and gap so that a dash-line has as many segments as possible and distributes remainder space to make the spaces between adjacent dash segments (almost) equal to each other. Therefore, you will have dash-lines with consistent steps of gaps and spaces the lines in Figure 1(1) are.

However, because of a bug (or buggy feature) of \texttt{xleaders}, there had been a small possibility that a dash segment near the right/bottom end drops, until it was fixed in the version of 3.141592\textsuperscript{5}. Though the fix ultimately made any effort to cope with the problem unnecessary, the package still gives you alternative drawing modes which you may specify by \texttt{\ADLdrawingmode{\langle m \rangle}} as follows.

\texttt{\ADLnullwide} \texttt{\ADLsomewide}
Figure 1: Drawing mode controlled by \ADLdrawingmode

\ADLdrawingmode

• \( m = 1 \)
As shown in Figure 1(1), it gives most beautiful result by \xleaders. This is default.

• \( m = 2 \)
As shown in (2) of the figure, beautiful if dash-lines are not so sparse as right/lower lines, but dash segments near the both ends may be a little bit too long as left/upper lines, because in this mode the second first/last segments are drawn by a special mechanism.

• \( m = 3 \)
As shown in (3) of the figure, beautiful if dash-lines are not so sparse as right/lower lines, but gaps near the both ends may be considerably too large as left/upper lines, because in this mode the lines are drawn by \cleaders.

It is strongly recommended to use default mode 1 unless you want to have some special effect.

2.6 Performance Tuning

Since drawing dash-lines is a hard job, you have to be patient with the fact that the performance of typesetting \array/tabular with dash-lines is poorer than that of ordinary ones. In fact, according to author’s small performance evaluation with a \tabular having nine vertical and ten horizontal dash-lines, typesetting the \tabular is approximately ten times as slow as its ordinary counterpart with solid lines.

However, this is not a really bad news, unfortunately. The real one is that loading \arydshln makes typesetting \array/tabular slower even if they only have solid lines which the package treats as special ones of dash-lines. The evaluation result shows the degradation factor is about nine. Therefore, if your document has many \array/tabular with solid lines, \LaTeX will run slowly even with quite few (or no) \array/tabular with dash-lines.

\ADLinactivate

To cope with this problem, you may inactivate dash-line functions by the command \ADLinactivate that replaces dash-lines with solid lines drawn by a faster (i.e. ordinary) mechanism. Although the inactivation does not completely solve the performance problem, the degradation factor will become much smaller and acceptable, approximately 1.5 in

\footnote{By pointing out this problem, the author got a check of $327.68 plus a significantly large amount of interest from DEK. Wow!!}

\footnote{Until the fix of \xleaders, the second bottom/rightmost segments of right/lower lines were dropped.}
the author’s evaluation. For example, the draft version of your document will have the command in its preamble, which you will remove from your final version.

Alternatively, you may do \ADLinactivate in the preamble, switch on by \ADLactivate before you really need dash-lines, and switch off again afterward. A wiser way could be surrounding \texttt{array/tabular} by \begin{ADLactivate} and \end{ADLactivate}.

If you feel it tiresome to type the long command/environment name for the activation, you may use \texttt{Array} and \texttt{Tabular(*)} environment in which dash-line functions are always active. Note that, however, since these environment names are too natural to keep them from being used by authors of other packages or yourself, name conflict could occur. If \texttt{Array} and/or \texttt{Tabular} have already been defined when \texttt{arydshln} is loaded, you will get a warning to show you have to define new environments, say \texttt{dlarray} and \texttt{dltabular}, as follows.

\newenvironment{dlarray}{\ADLactivate\begin{array}}% 
{\end{array}}
\newenvironment{dltabular}{\ADLactivate\begin{tabular}}% 
{\end{tabular}}
\newenvironment{dltabular*}{\ADLactivate\begin{tabular*}}% 
{\end{tabular*}}

On the other hand, if they are defined after \texttt{arydshln} is loaded, their definitions are silently replaced or \LaTeX{} complains of multiple definitions. The error in the latter case will be avoided by putting \texttt{\ADLnoshorthanded} just after \texttt{\usepackage{arydshln}}.

### 2.7 Compatibility with Other Packages

Users of \texttt{array} package may use all of newly introduced preamble characters, such as ‘\texttt{>}', ‘\texttt{<}', ‘\texttt{m}', ‘\texttt{b}', and all the commands such as \texttt{\extrarowheight}, \texttt{\firsthline} and \texttt{\lasthline}. The preamble characters given by \texttt{arydshln} may be included in the second argument of \texttt{\newcolumntype}.

Also users of \texttt{colortab} package may use \texttt{\LCC}/\texttt{\ECC} construct to color columns. A horizontal solid/dash line may be colored by, e.g. \texttt{\NAC\hdashline\ENAC}. The pair of \texttt{\AC} and \texttt{\EAC} may be used to color everything between them but, unfortunately, vertical lines are not. There are no ways to color vertical lines in a table having dash lines. You may color vertical lines of a ordinary table inactivating dash line functions by \texttt{\ADLinactivate}.

Another (and more convenient) table coloring tool \texttt{colortbl} may be also used simply by loading it before \texttt{arydshln}. Not only the painting commands \texttt{\rowcolor}, \texttt{\columncolor} and \texttt{\cellcolor} work well, but both solid and dash lines are also colored by the command \texttt{\arrayrulecolor} of \texttt{colortbl}\footnote{The \texttt{colortbl} manual says \texttt{\arrayrulecolor} and \texttt{\doublerulecolor} may be in \texttt{\begin{...}} in a preamble but they cause an error with the original implementation. This bug is fixed in \texttt{arydshln} and they are now usable to specify the color of the vertical (dash) lines whose specifications occur after the commands.}.

\begin{verbatim}
\begin{tabular}{|>{\columncolor{red}}l:>{\colomncolor{green}}r|}
... \\
\end{tabular}
\end{verbatim}
you will find the vertical dash line is a sequence of black (or the color of $\texttt{arrayrulecolor}$) and white segments. This problem is partly solved by declaring $\texttt{ADLnullwide}$ to conjunct the red and blue columns and to draw the dash line on their border. Unfortunately, however, $\texttt{ADLnullwide}$ does not affect the real width of horizontal (dash) lines and thus you will still see white gaps in $\texttt{hdashline}$ and $\texttt{cdashline}$. A solution is to put $\texttt{ADLnullwide}$ before you start a $\texttt{array/tabular}$\footnote{This command also makes $\texttt{cline}$ and $\texttt{cdashline}$ visible even if the row below is painted.}. With this command, a horizontal (dash) line is drawn adjusting its bottom edge to that of the row above. The command $\texttt{ADLsomewidehline}$ turns the switch to default and the top edge of a horizontal (dash) line will be adjusted to the bottom edge of the row above.

Another method to avoid white gaps is to give a color to gaps by $\texttt{dashgapcolor}$ with $\texttt{nodashgapcolor}$ arguments same as $\texttt{color}$. For example;

\begin{verbatim}
arrayrulecolor{green}\dashgapcolor{rgb}{1,1,0}
\end{verbatim}

makes colorful dash lines with green dashes and yellow gaps. The command can be placed outside of $\texttt{array/tabular}$ for dash lines in the environment, in the argument of preamble character $>$ for vertical dash lines following them, or at the beginning of a row for horizontal dash lines following the command. The command $\texttt{nodashgapcolor}$ (no arguments) nullifies the effect of $\texttt{dashgapcolor}$. Note that $\texttt{nodashgapcolor}$ is different from $\texttt{dashgapcolor}$ because the former makes gaps transparent while the later whiten them.

Usage of $\texttt{longtable}$ with $\texttt{arydshln}$ is quite simple. Just loading $\texttt{arydshln}$ after $\texttt{longtable}$ is enough to make the $\texttt{longtable}$ environment able to draw dash-lines. A shorthand activation of dash-line functions is also available by $\texttt{Longtable}$ environment. One caution to $\texttt{longtable}$ users is that the temporary results before the convergence of the column widths may be different from those without $\texttt{arydshln}$. For example, the following is the first pass result of the example shown in Table 3 of the $\texttt{longtable}$ manual.

\begin{verbatim}
1 2 3
wide multicolumn spanning 1–3
multicolumn 1–2 3
wide 1 2 3
\end{verbatim}

Since $\texttt{LTchunksize}$ is one in the example, columns of each row has their own widths and thus has vertical lines drawn at the edges of the columns. On the other hand, you will have the following as the first pass result with $\texttt{arydshln}$.

\begin{verbatim}
1 2 3
wide multicolumn spanning 1–3
multicolumn 1–2 3
wide 1 2 3
\end{verbatim}

As you see, the vertical lines are drawn at the column edges of the last row\footnote{More precisely, drawn according to the column widths established by all the chunks preceding page output.} because $\texttt{arydshln}$ draws them when it see the last row. Anyway, you may ignore temporary results and will have a compatible result when the column widths are converged like the following.
3 Known Problems

There are following known problems.

1. The new preamble specifiers ‘;’ and ‘\{\langle dash\rangle/\langle gap\rangle\}’ cannot be followed or preceded by ‘\text{\langle text\rangle}’, or you will have an ugly result. More specifically, a specifier to draw a dash-line at the left edge of a column cannot be preceded by ‘\text{\langle text\rangle}’, while that to draw at the right edge cannot be followed by ‘\text{\langle text\rangle}’.

2. If you use \texttt{array} package, the restriction of ‘\texttt{\textbackslash}’ shown above is also applied to ‘\texttt{!}’.

3. In order to make it sure that a dash-line always touches its both end, i.e. a dash-line always begins and ends with a dash segment, the amount of a gap will slightly vary depending on the dash-line length.

4. If a dash-line is too short, you will have an ugly result without overfull message. More specifically, in mode 1 or 3, a line will look to protrude beyond its column/row borders if it is shorter than a half of \texttt{\dashlinedash}. In mode 2, the minimum length to avoid the protrusion is $1.5 \times \texttt{\dashlinedash} + \texttt{\dashlinegap}$.

5. As described in §2.6, the processing speed for \texttt{array} and \texttt{tabular} environment will become slower even if dash-lines are not included.

6. As described in §2.7, \texttt{\textbackslash AC} and \texttt{\textbackslash EAC} pair of \texttt{colortab} such as \texttt{\AC\textbackslash EAC} cannot color the vertical line at &. Use \texttt{ADLinactivate} if you want to have a ordinary table with colored vertical lines. Note that you may color vertical lines with \texttt{colortbl} package.

7. There should be a number of packages whose own \texttt{array/tabular} implementations are not compatible with \texttt{arydshln}, though the author has made efforts at the compatibility. One of them is \texttt{plext} package for Japanese typesetting but it has a style file named \texttt{plextarydshln.sty} to solve the compatibility issue. So if you use the functionality of \texttt{arydshln} with \texttt{plext}, do \texttt{\usepackage{plextarydshln}} instead of \texttt{\usepackage{arydshln}}.
4 Implementation

4.1 Problems and Solutions

We have two different problems to solve; how to draw horizontal dash-lines and how to draw vertical dash-lines. The former problem is relatively easy because the technique for drawing \cline-s can be used. That is, if we know the number of columns, we can draw a dash-line across the \multispan-ed columns by \xleaders of dash. Modifying a preamble of array/tabular to count the number of columns is not hard. Since \cdashline is given beginning and ending columns, its implementation is also easy.

The latter problem, however, is much harder. Remember that array/tabular draws vertical solid lines by \vrule-s in each row without height/depth specification exploiting \TeX’s sophisticated mechanism of the rule extension in the surrounding box. Since \TeX does not have such a mechanism for \xleaders unfortunately, we at least have to know the height and depth of a row which includes vertical dash-lines. Although the height and depth are often same as those of \@arstrutbox, we will have an exceptionally tall and/or deep row that makes dash-lines broken if we assume every row has the standard height and depth.

Moreover, even if we can measure the height/depth of each row (in fact we will do as described later), drawing dash-lines in each row will not produce a good result. Look at the following two examples closely.

\begin{tabular}{|c|c|}
\hline
A & B \\
\hline
A & B \\
\hline
\end{tabular}

In the left example, two dash-lines are individually drawn in two rows. Since the first row is not so tall and deep (8.4 pt/3.6 pt) as to contain enough number of default dash segments (4 pt dash and 4 pt gap) to keep \xleaders from inserting a large space, the dash-line in the first row is sparse. On the other hand, the second row is enough tall and deep (16.8 pt/7.2 pt) and thus the dash-line in the row looks better. Thus the resulting dash-line is awful because it does not have a continuous dash/gap sequence.

The right example, which we wish to produce, is much better than the left. In this example, the dash line is drawn across two rows keeping continuous steps of dashes and gaps. In order to have this result, we have to draw the dash-line after two rows are built because it is necessary to know the total height and depth of two rows. In general, if we know the total height and depth of rows and whether a column has a dash-line, we can draw dash-lines by adding an extra row containing dash-lines. For example, the result shown above is obtained by the following row.

\begin{tabular}{|c|c|}
\hline
\omit & \omit \\
\hline
\omit & \omit \\
\hline
\end{tabular}

\smash{\langle dash-line of 36 pt high\rangle}

Note that \langle dash-line of 36 pt high\rangle have to be \texttt{\smash-ed}.

In addition to this basic scheme, we have to take the following points into account.

- A dash-line drawn by the preamble character ‘;’ will have non-default dash/gap specification.
• A column may have two or more dash-lines separated by spaces of `\doublerulesep`. Mixed sequence of solid- and dash-lines also have to be allowed.

• The first column may have dash-lines at both ends, while those of others will appear at right ends only. An exception of this rule is brought by `\multicolumn` that may have leading sequence of solid- and/or dash-line specifiers in its preamble.

• A `\multicolumn` may break or add a dash-line, or may change the dash/gap specification of a dash-line. A sequence of `\h(dash)line`-s also break dash-lines.

• If `colorthl` is in use, coloring dash/gap by `\arrayrulecolor` and `\dashgapcolor` gives another possibility of the variation of dash/gap specification.

In order to cope with them, the following data structure is constructed during rows are built.

1. The list of row information \( R = \langle r_1, r_2, \ldots, r_N \rangle \).

2. The \( i^{th} \) element of \( R \), \( r_i \), is one of the following\(^\text{11}\).
   
   - A triple \( \langle C^L_i, C^R_i, h_i \rangle \), where \( C^L_i \) and \( C^R_i \) are the lists of solid- or dash-line segments drawn at the left and right edge of columns respectively, and \( h_i \) is the height plus depth of the \( i^{th} \) row.
   - `connect(h_i)` for a `\h(dash)line` of \( h_i \) wide meaning that \( r_i \) is an empty pseudo row of \( h_i \) high and dash-lines are not broken at the row.
   - In `longtable` environment, `discard(h_i)` for a negative vertical space inserted by `\[\langle h_i \rangle\]` or `\h(dash)line` meaning \( r_i \) is an empty pseudo row of \( h_i \) high and dash-lines are not broken but may be discarded by the page break at the row.
   - `disconnect(h_i)` for a vertical gap generated by a sequence of `\h(dash)line` meaning \( r_i \) is an empty pseudo row of \( h_i \) high and dash-lines are broken at the row.

3. \( C^L_i = \langle e^L_{i,1}, e^L_{i,2}, \ldots, e^L_{i,m} \rangle \) where \( e^L_{i,j} \) corresponds to the \( j^{th} \) (leftmost is first) solid- or dash-line segment. \( C^R_i \) is similar but its elements are ordered in reverse, i.e. the rightmost segment is the first element.

4. The \( j^{th} \) element of \( C^L_i \) or \( C^R_i \), \( e^L_j, e^R_j \), is a triple \( \langle c^L_j, d^L_j, g^L_j \rangle \) where \( c^L_j \) is the column number in which the segment appears, and \( d^L_j \) and \( g^L_j \) are dash/gap specification, length and color, of the segment. For a solid line segment, the length attributes of both \( d^L_j \) and \( g^L_j \) are 0.

Then this data structure is processed to draw solid- and dash-lines at the end of the `array/tabular` as follows. Let \( e^L_j = \langle c^L_j, d^L_j, g^L_j \rangle \) be the \( j^{th} \) element of \( C^L_i \) of \( r_i \). The position \( p^L_j \) of \( e^L_j \) in the column \( c^L_j \) is defined as follows.

\[
p^L_j = \begin{cases} 
1 & \text{if } j = 1 \lor c^L_j \neq c^L_{j-1} \\
p^L_{j-1} + 1 & \text{otherwise}
\end{cases}
\]

\(^{11}\)In the real implementation, the structure of \( r_i \) is slightly different.
The following defines whether two elements \( e_j \) and \( e_{j'} \) are connected, or \( e_j \sim e_{j'} \).

\[
e_j \sim e_{j'} \iff i < i' \wedge
\]

\[
e_j = c_{j'}, d_j = d_{j'} \wedge g_j = g_{j'} \wedge p_j = p_{j'} \wedge
\]

\( \forall k(i < k < i' \rightarrow r_k \in \{\text{connect}(h_k), \text{discard}(h_k)\}) \).

With these definitions, we can classify all \( e_j \) into ordered sets \( S_1, S_2, \ldots, S_n \) as follows.

\[
k \ne k' \leftrightarrow S_k \cap S_{k'} = \emptyset
\]

\[
e_j \sim e_{j'} \leftrightarrow \exists k: e_j, e_{j'} \in S_k \wedge S_k = \{\ldots, e_j, e_{j'}, \ldots\}
\]

\[
k < k' \leftrightarrow \forall e_j \in S_k, \forall e_{j'} \in S_{k'}: (e_j < e_{j'}) \wedge
\]

\[
(c_j = c_{j'} \wedge p_j < p_{j'}) \vee
\]

\[
(c_j = c_{j'} \wedge p_j < p_{j'} \wedge i < i').
\]

Now we can draw a dash-line \( L_k = (\gamma_k, \pi_k, \delta_k, \xi_k, \tau_k, \beta_k) \) corresponding to \( S_k = \{e_j, \ldots, e_{j'}\} \) as follows.

- \( L_k \) is \( \pi_k^\text{th} \) line in the \( \gamma_k^\text{th} \) column where \( \gamma_k = c_j = \ldots = c_{j'} \), and \( \pi_k = p_j = \ldots = p_{j'} \).

- \( L_k \) has the dash specification (size and color) \( \delta_k = d_j = \ldots = d_{j'} \), and gap specification \( \xi_k = g_j = \ldots = g_{j'} \).

- The top and bottom ends of \( L_k \) are at \( \tau_k \) and \( \beta_k \) above the bottom of the \texttt{array/tabular}, where;

\[
\eta = \begin{cases} h_l & r_l = \text{connect}(h_l), \tau_k = \eta_{l-1} + \sum_{l=1}^{N} h_l, \beta_k = -\eta_{l'} + 1 + \sum_{l'=1}^{N} h_l. & \text{otherwise} \end{cases}
\]

Note that \( \eta_{l-1} \) and \( \eta_{l'+1} \) are added/subtracted so that the top/bottom of \( L_k \) is at the top/bottom edge of the horizontal lines above/below the set \( S_k \).

The row to draw \( L_1, \ldots, L_n \) is;

\[
\sigma_1 L_1 \sigma_2 L_2 \ldots L_{n-1} \sigma_n L_n \sigma_{n+1} \cr
\]

where;

\[
\sigma_1 = \text{\texttt{\small \textbackslash omit[\texttt{\textbackslash hss\textbackslash omit}]^{\gamma_{n-1}}}}
\]

\[
\sigma_{1 < k \leq n} = \begin{cases} \text{\texttt{\null}} & \text{if } \gamma_{k-1} = \gamma_k \wedge \pi_{k-1} = \pi_k \\
\text{\texttt{\textbackslash hss\textbackslash skip\textbackslash doublerulesep}} & \text{if } \gamma_{k-1} = \gamma_k \wedge \pi_{k-1} \neq \pi_k \\
\text{\texttt{\textbackslash hss\textbackslash omit}}^{\gamma_k - \gamma_{k-1}} & \text{if } \gamma_{k-1} \neq \gamma_k
\end{cases}
\]

\[
\sigma_{n+1} = \text{\texttt{\textbackslash hss\textbackslash omit}^{\Gamma - \gamma_n - 1}}\text{\texttt{\textbackslash hss}}.
\]

Note that \([x]^m\) means \(m\)-times iteration of \( x \), and \( \Gamma \) is the number of columns specified in the preamble.
Dash-lines at the right edges of columns are similarly drawn by processing $C^R_i$ with the following modifications.

\[
k < k' \leftrightarrow \forall e_{i}^{l} \in S_k, \forall e_{i'}^{l} \in S_{k'} : (c_{j}^{l} < c_{j'}^{l}) \lor \\
(c_{j}^{l} = c_{j'}^{l} \land p_{j}^{l} > p_{j'}^{l}) \lor \\
(c_{j}^{l} = c_{j'}^{l} \land p_{j}^{l} = p_{j'}^{l} \land i < i')
\]

\[
\sigma_1 = \text{hskip hss}[\text{hskip hss}]^{\gamma_1-1}
\]

\[
\sigma_{k+1} = \begin{cases} \\
\text{null} & \text{if } \gamma_{k-1} = \gamma_k \land \pi_{k-1} = \pi_k \\
\text{hskip doublerulesep} & \text{if } \gamma_{k-1} = \gamma_k \land \pi_{k-1} \neq \pi_k \\
[\text{hskip hss}]^{\gamma_k-\gamma_{k-1}} & \text{if } \gamma_{k-1} \neq \gamma_k
\end{cases}
\]

\[
\sigma_{n+1} = [\text{hskip hss}]^{\gamma_n-1}
\]

### 4.2 Another Old Problem

In the default mode 1, we draw a dash line of dash size $d$ and gap size $g$ as follows. Let $W$ be the length of the line plus \texttt{10sp}, which is unknown for us if horizontal but known for \TeX, and assume $W \geq d/2$ (or the line protrude to the column/row boarder.) At the both ends of the columns, dashes of $d/2$ long are drawn to make the dash-line \textit{touched} to the ends. Then $n = \lfloor (W - d + g)/(d + g) \rfloor$ dashes are equally distributed in the remaining space. Thus we will have:

\[
D_0(d/2) G_0(g + \epsilon') D_1(d) G_1(g + \epsilon) \ldots G_{n-1}(g + \epsilon) D_n(d) G_n(g + \epsilon') D_{n+1}(d/2)
\]

where $D_l$ and $G_l$ are dash and gap of $l$ long, $\epsilon = (W - (n + 1)(d + g))/(n + 1)$ (rounded), and $\epsilon' = (W - (n + 1)(d + g) - (n - 1)\epsilon)/2$ to compensate the rounding error on the calculation of $\epsilon$. For a horizontal line, this result will be obtained by \texttt{xleaders} as follows where $G_n^m(\epsilon)$ and $G_n^m(\epsilon')$ are the spaces inserted by \texttt{xleaders}.

\[
D_0(d/2) G_0^l(g/2) \texttt{xleaders} \texttt{hbox}\{G^r(g/2) D(d) G^l(g/2)\} \texttt{hss} G_n^r(g) D_{n+1}(d/2)
\]

\[
= D_0(d/2) G_0^l(g/2) G_0^m(\epsilon') \left(G_0^r(g/2) D_1(d) G_1^l(g/2)\right) G_n^m(\epsilon)
\]

\[
(G_1^l(g/2) D_2(d) G_2^l(g/2)) G_2^m(\epsilon)
\]

\[
\ldots
\]

\[
G_{n-1}^m(\epsilon) \left(G_{n-1}^r(g/2) D_{n-1}(d) G_{n-1}^l(g/2)\right) G_n^m(\epsilon') G_n^r(g/2) D_{n+1}(d/2)
\]

\[
= D_0(d/2) G_0(g + \epsilon') D_1(d) G_1(g + \epsilon) \ldots G_{n-1}(g + \epsilon) D_n(d) G_n(g + \epsilon') D_{n+1}(d/2)
\]

The problem is that $\epsilon'$ could be negative and old \TeX mistakenly ignored this possibility. That is, since \TeX older than 3.141592 did not put \texttt{hbox} beyond the right edge of \texttt{xleaders}, the rightmost \texttt{hbox} was omitted if $\epsilon'$ is negative.

Since it is (almost) impossible to know the length of a horizontal line, we could not cope with this problem by adding or subtracting its length. Thus we introduced \texttt{dracing mode}.

\[\texttt{12This small amount is added by \texttt{xleaders} in order to, according to the comment in \texttt{tex.web}, compensate floating point rounding error.}\]
to have imperfect solutions. In the mode 2, we draw a line by the following sequence.

\[ D_0(d/2)G_0(g/2)D_1(d)G_1'(g/2)G(-d-g) \]
\[ \text{\texttt{xleaders}} \text{\hbox}{G'(g/2)D(d)G'(g/2)}\hss \]
\[ G(-d-g)G_n(g/2)D_n(d)G_n'(g/2)G_n(g)D_{n+1}(d/2) \]

That is, \( n^{th} \) \texttt{\hbox} that could be disappeared is put twice and the first one is also overlaid for symmetrization. Therefore the length of the first and \( n^{th} \) dashes is \( d + |\varepsilon'| \) and thus could be a little bit longer than others.

On the other hand, we replace \texttt{xleaders} of mode 1 with \texttt{cleaders} for the drawing in mode 3. The result will be:

\[ D_0(d/2)G_0(g+R)D_1(d)G_1(g) \ldots G_{n-1}(g)D_n(d)G_n(g+R)D_{n+1}(d/2) \]

where \( R = (W - (n+1)(d+g))/2 \) to make the first and last gaps considerably wider than others.

\section{4.3 Register Declaration}

Here registers and switches are declared.

First of all, two \texttt{\dimen} registers \texttt{\dashlinedash} and \texttt{\dashlinegap} to control the shape of dash-lines are declared, and their default values, 4 pt for both, are assigned to them. They have aliases, \texttt{\hdashlinewidth} and \texttt{\hdashlinegap} respectively, for the backward compatibility.

1 \%\ Register Declaration
2 \%\ Register Declaration
3 \newdimen\dashlinedash \dashlinedash 4pt \%
4 \newdimen\dashlinegap \dashlinegap 4pt \%
5 \let\hdashlinewidth\dashlinedash
6 \let\hdashlinegap\dashlinegap

Next, the following six switches are declared.

- \texttt{\ifadl@leftrule} is used in the preamble analysis macro \texttt{\@mkpream} and is true during its processes leading characters for solid- and dash-lines, i.e. ‘|’, ‘;’ and ‘;’.

- \texttt{\ifadl@connected} is used to indicate the connection \( e_j \sim e_j' \). When we process \( e_j' \), the switch is true iff \( \exists e_j(e_j \sim e_j') \).

- \texttt{\ifadl@doublerule} is used to make \( \sigma_k \). When we are to make \( \sigma_k L_k \), it is true iff \( \gamma_{k-1} = \gamma_k \land \pi_{k-1} \neq \pi_k \).

- \texttt{\ifadl@zwvrule} controls the real width of vertical lines. If it is true, lines are drawn as if their width is zero following \LaTeX{}’s standard. Otherwise, their width \texttt{\arrayrulewidth} contribute to the width of columns as \texttt{\array} does.
\ifadl@zwhrule
  \ifadl@zwhrule controls the \textit{real} width of horizontal lines. If it is true, a line is drawn as if its width is zero and its bottom edge is adjusted to that of the row above by inserting \texttt{\vskip-\textbackslash arrayrulewidth} before the drawing. Thus a horizontal dash line is included in the row above and its gaps look colored properly if the row is painted. If it is false, the width \texttt{\textbackslash arrayrulewidth} contribute to the height of \texttt{array/tabular} as usual.
\fiadl@zwhrule

\ifadl@usingarypkg
  \ifadl@usingarypkg is true iff \texttt{array} has been loaded prior to \texttt{arydshln}. This switch shows us which definitions, by E\TeX or \texttt{array}, we have to modify. Its value is set by examining if \texttt{\extrarowheight}, which is introduced by \texttt{array}, is defined.
\fiadl@usingarypkg

\ifadl@inactive
  \ifadl@inactive inactivates dash-line functions if it is true. Its default value is false.
\fiadl@inactive

We also use a working switch \texttt{\atempswa}.

\newif\ifadl@leftrule
\newif\ifadl@connected
\newif\ifadl@doublerule
\newif\ifadl@zwvrule
\newif\ifadl@zwhrule
\ifx\extrarowheight\undefined \adl@usingarypkgfalse \else \adl@usingarypkgtrue \fi
\newif\ifadl@inactive \adl@inactivefalse

\ADLnullwide The switch \texttt{\ifadl@hwvrule} is turned on/off by user interface macros \texttt{\ADLnullwide} and \texttt{\ADLsomewide}. Its initial value is the complement of \texttt{\adl@usingarypkg}.

\ADLnullwidehline The switch \texttt{\ifadl@zvrule} is turned on/off by user interface macros \texttt{\ADLnullwidehline} and \texttt{\ADLsomewidehline}. Its initial value is false.

\ADLactivate The switch \texttt{\ifadl@inactive} is also turned on/off by user interface macros \texttt{\ADL inactivate} and \texttt{\ADLactivate}.

\def\ADLnullwide{\adl@zvruletrue}\def\ADLnullwide{\adl@zvrulefalse}
\ifadl@usingarypkg \ADLsomewide \else \ADLnullwide \fi
\def\ADLnullwidehline{\adl@zvruletrue}
\def\ADLnullwidehline{\adl@zvrulefalse}
\ADLsomewidehline
\def\ADLactivate{\adl@inactivefalse}
\def\ADLactivate{\adl@inactivetrue}

The following \texttt{\box} register and three \texttt{\dimen} registers are used to measure the height and depth of a row.

\adl@box
  \adl@box
  The contents of a column is packed into the \texttt{\box} register \texttt{\adl@box} to measure its height and depth.
\texttt{\@height} \texttt{\@depth}  

- The \texttt{\dimen} registers \texttt{\@height} and \texttt{\@depth} contain the height/depth of the tallest/deepest column in a row. When a column is processed, they are compared to the height and depth of \texttt{\@box} and are updated if they are less.

\texttt{\@heightsave} \texttt{\@depthsave}  

Since we have to update these registers \texttt{global-ly} to pass their values across & and we may have a column containing \texttt{array/tabular}, they are saved into \texttt{\@heightsave/\@depthsave} at the beginning of the environment and are restored at its end.

\texttt{\@finaldepth}  

- The other \texttt{\dimen} register \texttt{\@finaldepth} is set to the depth of the last row, or zero if the last vertical item is a horizontal line. This value is used to shift \texttt{array/tabular} down because we add extra two \texttt{\smash-ed} rows which make the depth of \texttt{array/tabular} zero.

We also use working \texttt{\dimen} registers \texttt{\@tempdima} and \texttt{\@tempdimb}.

28 \texttt{\newbox\@box}  
29 \texttt{\newdimen\@height \newdimen\@heightsave}  
30 \texttt{\newdimen\@depth \newdimen\@depthsave}  
31 \texttt{\newdimen\@finaldepth}  

Then the following \texttt{\count} registers are declared. Note that some of them contain dimensions measured by the unit \texttt{sp}.

\texttt{\@columns} \texttt{\@ncol}  

- \texttt{\@columns} has the number of columns specified in the preamble of the environment. Because of a complicated reason related to the compatibility with \texttt{array}, we cannot count up \texttt{\@columns} directly but increment \texttt{\@ncol} when each column of preamble is built and move its value to \texttt{\@columns} after the preamble is constructed.

\texttt{\@currentcolumn} \texttt{\@currentcolumnsave}  

- To process \texttt{\multicolumn}, we have to know the column number where it appears. Thus we have a column counter \texttt{\@currentcolumn} which is \texttt{\global}ly incremented when each column is built. Because of the \texttt{\global} assignment, the counter has to be saved/restored into/from \texttt{\@currentcolumnsave}.

\texttt{\@totalheight}  

- In the real implementation, $\tau_k$ and $\beta_k$ are calculated by the following equations rather than those shown in \S 4.1.

$$H = \sum_{l=1}^{N} h_l, \quad \tau_k = H + \eta_{k-1} - \sum_{l=1}^{k-1} h_l, \quad \beta_k = \tau_k - \eta_{k-1} - \sum_{l=k+1}^{i} h_l.$$  

\texttt{\@totalheight} contains $\sum_{l=1}^{i} h_l$ when the $i^{th}$ row is built and thus its final value is $H$. Since the data structure $R$ are represented by a text, we have to pay attention to the precision of its dimensional elements, such as $h_i$. That is, if we append $h_i$ to $R$ by expanding \texttt{\the\dimen} which has the height plus depth of $i^{th}$ row, $h_i$ will be an approximation of $\dimen$ represented by a decimal fraction with \texttt{pt}. Although the error of the approximation is quite small and may be negligible, the error must be avoided because it is avoidable by simply using \texttt{\number\dimen}. Therefore, $h_i$ is an integer and thus \texttt{\@totalheight} is too.
Because of the \texttt{global} assignment to \texttt{adl@totalheight} to pass its value across rows, it has to be saved/restored into/from \texttt{adl@totalheightsave}.

\begin{itemize}
  \item In order to check $e_i^j \sim e_i^{j'}$, the size attributes of $d_{ij}$ and $g_{ij}$ are kept in the registers \texttt{adl@dash} and \texttt{adl@gap} when we process $e_i^{j'}$. As explained above, $d_{ij}$ and $g_{ij}$ are integers and thus \texttt{adl@dash} and \texttt{adl@gap} are \texttt{count} registers.
  \item The coding of \texttt{cdashline} is similar to that of \texttt{cline} in \LaTeXe{} which uses two global \texttt{count} registers \texttt{@cla} and \texttt{@clb}. These registers are omitted from \LaTeXe{} because its \texttt{cline} is completely recoded. We could adopt new coding but it requires some other macro definitions that \LaTeXe{} does not have. Thus we simply introduce new global counters \texttt{adl@cla} and \texttt{adl@clb} for \texttt{cdashline} in order to make \texttt{cdashline} work in both \LaTeXe{} and \LaTeXe{}.
\end{itemize}

We also use working \texttt{count} registers \texttt{@tempcnta} and \texttt{@tempcntb}.

\begin{verbatim}
\newcount adl@columns \newcount adl@ncol
\newcount adl@currentcolumn \newcount adl@currentcolumnsave
\newcount adl@totalheight \newcount adl@totalheightsave
\newcount adl@dash \newcount adl@gap
\newcount adl@cla \newcount adl@clb
\end{verbatim}

\texttt{adl@everyvbox} The last register declaration is for a \texttt{toks} register named \texttt{adl@everyvbox}. In order to minimize the copy-and-modify of the codes in \LaTeXe{} and \texttt{array}, we need to use \texttt{everyvbox} in our own definition of \texttt{@array}. The register is used to save the contents of \texttt{everyvbox}.

\begin{verbatim}
\newtoks adl@everyvbox
\end{verbatim}

The other declarative stuff consists of the sequence of \texttt{let} to capture the original definitions of macros that we will modify afterward. The main purpose of them is to nullify the modification when dash-line functions are inactive, while \texttt{adl@org@cline} is also referred to in its modified version.

\begin{verbatim}
\let adl@org@arrayclassz @arrayclassz
\let adl@org@tabclassz @tabclassz
\let adl@org@classz @classz
\let adl@org@@startpbox @@startpbox
\let adl@org@@endpbox @@endpbox
\let adl@org@startpbox @startpbox
\let adl@org@endpbox @endpbox
\let adl@org@cline \cline
\end{verbatim}

4.4 Initialization

\begin{verbatim}
\let \array \LaTeXe{}’s macro \texttt{array} is modified to save and initialize registers and data structures which are \texttt{global}-ly updated in order to allow nested \texttt{array/tabular}. This saving and initializing are performed by \texttt{adl@arrayinit} as explained below. The problem in the
\end{verbatim}
modification is that the code of \@array in array is completely different from that of \LaTeX original.

The main difference is that \LaTeX builds \@preamble locally, while array does globally exploiting the fact that the lifetime of \@preamble ends before another array/tabular appears in a column. The latter implementation will work well unless the building process in \@mkpream produces something referred to after \@preamble is thrown into \TeX's stomach. In our implementation, unfortunately, the number of columns has to be counted in \@mkpream and will be referred to by \@dashline and the vertical line drawing procedure.

Thus we have to change the column counting mechanism depending on whether or not array is in use. The simplest way could be to copy the codes of \LaTeX and array and modify them appropriately examining the value of \ifadl@usingarypkg. However this solution is vulnerable to the modification of the original version and thus we wish to refuse it as far as possible.

Therefore, we use a trick with \everyvbox in which \adl@arrayinit is temporarily included to initialize registers and locally set \adl@columns to the number of columns \global-ly counted by \adl@ncol. This trick works well so far because:

- the first \vbox, \vtop or \vcenter made by \@array is the vertical box surrounding \halign, and;
- in \@array of array the box is opened after the preamble is constructed;

and will hopefully work in future.

Next, if \ifadl@inactive is true, \adl@inactivate is invoked to inactivate dash-line functions. Otherwise, \adl@activate is invoked to activate them because an inactivated array/tabular may have active children in it. Finally, \adl@noalign is made \let-equivalent to \noalign so that \arrayrulecolor, \doublerulesepcolor and \dashgapcolor are expanded with \noalign in the environment.

\@array

Another stuff for the compatibility with array is to \let a control sequence \@array be equal to \array if it is made so by array and the equality is kept. That is, with array \@array is invoked by \@tabarray and it is \let-equivalent to \@array by default, while \@array can be made different from \@array by some other package, e.g., delarray, to do some special operations defined in the package. Therefore by the conditional equalization with \ifx, our own \@array is directly invoked through \@array if the default equality is kept, while otherwise the package-dependent definition of \@array is respected.
As described in §4.3, registers updated \texttt{global}-ly, which are \texttt{adl@height}, \texttt{adl@depth}, \texttt{adl@currentcolumn} and \texttt{adl@totalheight}, are saved in \texttt{adl@arraysave}, and also given initial values. The macro also saves the following data structures and initializes them to empty lists.

- In the real implementation, the data structure $R$ is split into two lists;
  
  \begin{align*}
  \texttt{adl@rowsL} &= R^L = \langle \langle C^L_1, h_1 \rangle, \ldots \rangle \\
  \texttt{adl@rowsR} &= R^R = \langle \langle C^R_1, h_1 \rangle, \ldots \rangle \\
  \end{align*}
  
  and they are saved into \texttt{adl@rowsLsave} and \texttt{adl@rowsRsave}.

- When the $i^{th}$ row is building, $C^L_i$ and $C^R_i$ are constructed in the macros \texttt{adl@colsL} and \texttt{adl@colsR}. They are saved into \texttt{adl@colsLsave} and \texttt{adl@colsRsave}.

In the real implementation, $e^i_j$ is represented by a control sequence \texttt{@elt}, and \texttt{connect(i)} by \texttt{adl@connect}. They are made \texttt{\let}=equal to \texttt{relax} to keep them from expansion during $R$ is constructed. In \texttt{longtable} environment, \texttt{connect(i)} for negative vertical space inserted by $\langle h \rangle$ or a horizontal line has another representation \texttt{adl@discard} to indicate it corresponds to a discardable item of page breaking. Since this representation, however, is nonsense in usual \texttt{array/tabular} even if they are included in \texttt{longtable}, we define \texttt{adl@discard} as \texttt{adl@connect} so that it transforms itself into \texttt{adl@connect} when it is added to \texttt{adl@rowsL/R} by \texttt{xdef}. Note that \texttt{adl@discard} is made \texttt{\let}=equal to \texttt{relax} to inhibit the transformation at the beginning of \texttt{longtable} environment.

Then, we set to \texttt{adl@columns} to the value of \texttt{adl@ncol} locally. As explained above, this has an effect with \texttt{array} because \texttt{adl@arrayinit} is called after the preamble is generated. Without \texttt{array}, on the other hand, this assignment has no effect but safe because it is included in a group of \texttt{vbox} etc.

\begin{verbatim}
\def\adl@arrayinit{%
  \adl@arraysave
  \global\adl@height\z@ \global\adl@depth\z@
  \global\adl@currentcolumn\one \global\adl@totalheight\z@
  \gdef\adl@rowsL{}
  \gdef\adl@rowsR{}
  \gdef\adl@colsL{}
  \gdef\adl@colsR{}

  \global\adl@columns\adl@ncol%
}
\def\adl@arraysave{%
  \adl@heightsave\adl@height \\
  \adl@depthsave\adl@depth \\
  \adl@currentcolumnsave\adl@currentcolumn \\
  \adl@totalheightsave\adl@totalheight \\
  \let\adl@rowsLsave\adl@rowsL \\
  \let\adl@rowsRsave\adl@rowsR \\
  \let\adl@colsLsave\adl@colsL \\
  \let\adl@colsRsave\adl@colsR%
}
\end{verbatim}
\adlinactivate \iffalse If \texttt{ADLinactivate} has effect and thus \texttt{\ifadlinactive} is true, the macro \texttt{\adlinactivate} is called from \texttt{\@array}. This \texttt{\let}s the following control sequences be equal to their counterparts in \texttt{\LaTeX} and/or \texttt{array} package.

\@arrayclassz \@tabclassz \@classz \@@startpbox \@@endpbox \\
@@startpbox \@@endpbox \adl@cr \adl@argcr \adl@endarray

Note that \texttt{\@classz} has to be \texttt{\let-equal} to \texttt{\adl@org@classz} only if \texttt{array} is in use, because \texttt{\LaTeX} does not define \texttt{\@classz} but refers to it which is either \texttt{\@arrayclassz} or \texttt{\@tabclassz}. Yet another remark is that we have to conceal \texttt{\cr} for \texttt{\adl@cr/\adl@argcr} and \texttt{\crcr} for \texttt{\adl@endarray} by bracing them from \texttt{\LaTeX}'s \texttt{\halign} mechanism that searches them when an \texttt{array/tabular} has an nested \texttt{array/tabular}. This could be done by a tricky \texttt{\let}-assignment such as;

\iffalse{
\let\adl@cr\cr
\iffalse}\fi
\fi

but we simply use \texttt{\def} instead of \texttt{\let} because of clarity.

We also \texttt{\let} the following be no-operation or their inactive versions.

\adl@hline \adl@ihdashline \adl@cdline \adl@vlineL \adl@vlineR \\
\adl@vlineL \adl@vlineR

Note that we have to inactivate both \texttt{\adl@vlineL} and \texttt{\adl@vlineR}, because the latter is referred to when \texttt{array} is in use while the former is done otherwise. Their \texttt{R} relatives are also inactivated by the same reason.

\adl@inactivate 77 \def\adl@inactivate{\%
78 \let\@arrayclassz\adl@org@arrayclassz
79 \let\@tabclassz\adl@org@tabclassz
80 \ifadl@usingarypkg \let\@classz\adl@org@classz \fi
81 \let\@startpbox\adl@org@startpbox
82 \let\@endpbox\adl@org@endpbox
83 \let\@startpbox\adl@org@startpbox
84 \let\@endpbox\adl@org@endpbox
85 \def\adl@cr{\cr}\
86 \def\adl@argcr##1{\cr}\
87 \def\adl@endarray{\crcr}\
88 \let\adl@hline\@gobbletwo
89 \let\adl@ihdashline\adl@inactivehdl
90 \let\adl@cdline\adl@inactivecdl
91 \let\adl@vlineL\adl@inactivevl
92 \let\adl@vlineR\adl@inactivevl
93 \let\adl@vlineR\adl@inactivevl\}

\adl@activate \iffalse On the other hand, if \texttt{\ifadl@inactive} is false, the macro \texttt{\adl@activate} is called from \texttt{\@array} to make inactivated macros active again in order to cope with the case in which

\footnote{Before v1.53, \texttt{\adl@inactivate} was called from \texttt{\adl@arrayinit} and thus invoked after the preamble of \texttt{array} is built. This was incorrect of course and made inactive version of \texttt{p, m and b} produce nothing.}
<table>
<thead>
<tr>
<th>command</th>
<th>active</th>
<th>inactive</th>
</tr>
</thead>
<tbody>
<tr>
<td>l c r</td>
<td>\adl@act@classz</td>
<td>\adl@org@classz</td>
</tr>
<tr>
<td></td>
<td>\adl@act@tabclassz</td>
<td>\adl@org@tabclassz</td>
</tr>
<tr>
<td></td>
<td>\adl@act@arrayclassz</td>
<td>\adl@org@arrayclassz</td>
</tr>
<tr>
<td>with array</td>
<td></td>
<td></td>
</tr>
<tr>
<td>without array</td>
<td></td>
<td></td>
</tr>
<tr>
<td>p m b (open)</td>
<td>\adl@act@classz</td>
<td>\adl@org@classz</td>
</tr>
<tr>
<td></td>
<td>\adl@act@startpbox</td>
<td>\adl@org@startpbox</td>
</tr>
<tr>
<td></td>
<td>\adl@act@@startpbox</td>
<td>\adl@org@@startpbox</td>
</tr>
<tr>
<td></td>
<td>\adl@act@@vlineL/R</td>
<td>\adl@inactivevl</td>
</tr>
<tr>
<td></td>
<td>\adl@act@hline</td>
<td>\adl@inactivehline</td>
</tr>
<tr>
<td></td>
<td>\adl@act@ihdashline</td>
<td>\adl@inactiveihdashline</td>
</tr>
<tr>
<td></td>
<td>\adl@act@cdline</td>
<td>\adl@inactivecdline</td>
</tr>
<tr>
<td></td>
<td>\adl@act@@vlineL</td>
<td>\adl@inactivevlineL</td>
</tr>
</tbody>
</table>

Table 1: Active and Inactive Operations

An inactive array/tabular has active children in it\textsuperscript{14}. To do that, \texttt{\adl@activate} makes \texttt{\@arrayclassz} etc. \texttt{\let} equal to their active version \texttt{\adl@act@arrayclassz} etc. which will be defined (\texttt{\let}-equal to) as our own \texttt{\@arrayclassz} etc. in §4.13.

\texttt{\def\adl@activate{%}
  \let\@arrayclassz\adl@act@arrayclassz\let\@tabclassz\adl@act@tabclassz\ifadl@usingarypkg \let\@classz\adl@act@classz \fi\let\@startpbox\adl@act@startpbox\let\@endpbox\adl@act@endpbox\let\adl@cr\adl@act@cr\let\adl@argcr\adl@act@argcr\let\adl@endarray\adl@act@endarray\let\adl@hline\adl@act@hline\let\adl@ihdashline\adl@act@ihdashline\let\adl@cdline\adl@act@cdline\let\adl@@vlineL\adl@act@@vlineL\let\adl@@vlineR\adl@act@@vlineR\let\adl@vlineL\adl@act@@vlineL\let\adl@vlineR\adl@act@@vlineR}}

The summary of the activation and inactivation is shown in Table 1.

\textsuperscript{14}Before v1.54, an active array/tabular in an inactive parent was not activated.
4.5 Making Preamble

Each preamble character is converted to a part of \texttt{\textbackslash halign}'s preamble as follows.

\begin{verbatim}
\texttt{\textbackslash adl@colhtdp}

- '1', 'r' and 'c' are converted to the following (lrc).

\begin{align*}
(lrc) &::= [\texttt{\hfil}](\texttt{put-lrc})[\texttt{\hfil}] \\
(put-lrc) &::= \texttt{\setbox}\texttt{\adl@box}\texttt{\hbox}{(lrc-contents)} \\
\texttt{\adl@colhtdp} &\texttt{\unhbox}\texttt{\adl@box} \\
(lrc-contents) &::= $\relax# | #\unskip$
\end{align*}

That is, the content of a column is at first packed into the \texttt{\box} register \texttt{\adl@box}, then its height and depth are compared to \texttt{\adl@height} and \texttt{\adl@depth} by the macro \texttt{\adl@colhtdp}, and finally the box is put with leading and/or trailing \texttt{\hfil}.

\texttt{\textbackslash adl@vlineL \textbackslash adl@vlineR}

- '1', ': and \{\langle\texttt{\textbackslash dash}\rangle)/\langle\texttt{\textbackslash gap}\rangle\} are converted to the following (vline).

\begin{align*}
(vline) &::= [\texttt{\hskip\texttt{\doublerulesep}}](\texttt{\textbackslash vline-LR}) \\
(\texttt{\textbackslash vline-LR}) &::= \texttt{\textbackslash adl@vlineL}(\langle I_d\rangle)\{(\langle c\rangle)\{(\langle d\rangle)\langle g\rangle}\} | \\
\texttt{\textbackslash adl@vlineR}(\langle I_d\rangle)\{(\langle c\rangle)\{(\langle d\rangle)\langle g\rangle}\} \\
\langle d\rangle &::= 0 | \ldots \text{ for '1'} \\
\texttt{\textbackslash dashlinedash} &\ldots \text{ for ':'} \\
(\langle dash\rangle) &\ldots \text{ for '1'} \\
\langle g\rangle &::= 0 | \ldots \text{ for '1'} \\
\texttt{\textbackslash dashlinegap} &\ldots \text{ for ':'} \\
(\langle gap\rangle) &\ldots \text{ for '1'}
\end{align*}

Note that \langle c\rangle is the column number (leftmost is 1) where the character appears, and \langle I_d\rangle and \langle I_g\rangle is the color of dashes and gaps specified in \texttt{\textbackslash CT@arc@} and \texttt{\adl@\textbackslash dashgapcolor}.

Additionally, each column except for the last one has:

\begin{verbatim}
\texttt{\textbackslash global\textbackslash advance}\texttt{\adl@currentcolumn}\texttt{\textbackslash one}
\end{verbatim}

just before & to increment \texttt{\adl@currentcolumn}. Other features, such as inserting spaces of \texttt{\textbackslash arraycolsep}/\texttt{\textbackslash tabcolsep}, are as same as original scheme. This means that \{\langle\texttt{\textbackslash text}\rangle\} and !\{\langle\texttt{\textbackslash text}\rangle\} of array are not handled specially although it could interfere with drawing vertical lines. Therefore, we have the problem 1 shown in §3, which is very hard to solve. Note that the measurement of the column of ‘p’ of \texttt{\textbackslash B\textbackslash e\textbackslash T\textbackslash e\textbackslash X} original is done by (modified) \texttt{\textbackslash @\textbackslash startpbox} and \texttt{\textbackslash @\textbackslash endpbox} and thus the preamble for ‘p’ is not modified. In the case with array, however, the preambles for ‘p’ and its relatives ‘m’ and ‘b’ are modified to set \texttt{\textbackslash adl@box} to the box for them.

22
To make the preamble shown above, \texttt{\@mkpream} is modified to \texttt{\let} control sequences \texttt{\@colhtdp}, \texttt{\@vlineL} and \texttt{\@vlineR} be \texttt{\relax} in order to keep them from being expanded by \texttt{\edef} for the preamble construction. The control sequences \texttt{\@startmbox} and \texttt{\@endmbox} for \texttt{m}-columns of \texttt{array} are also made \texttt{\let}-equal to \texttt{\relax}.

Giving them their own definition is done by \texttt{\adl@preaminit} that is called using \texttt{\afterassignment} after \texttt{\@preamble} is made by \texttt{\adl@mkpream}, the original version of \texttt{\@mkpream}. If \texttt{array} is not in use, \texttt{\@mkpream} is followed by an \texttt{\edef} of \texttt{\@preamble} to add \texttt{\ialign} etc. and thus \texttt{\adl@preaminit} is properly called \texttt{after} this final assignment to make \texttt{\@preamble}.

With \texttt{array}, on the other hand, calling \texttt{\adl@preaminit} is safe because \texttt{\@mkpream} is followed by \texttt{\edef} for \texttt{\@preamble} too, but has no effect because it is in the group for \texttt{\@mkpream}. This grouping, however, gives us an easier way to give those control sequences their own definition. That is, we simply initiate them with the definitions that will be regained when the group is closed.

The modified \texttt{\@mkpream} also initializes \texttt{\adl@ncol} and \texttt{\ifadl@leftrule}, and set \texttt{\adl@columns} to the value of \texttt{\adl@ncol} locally after the preamble is made. This has an effect in the case without \texttt{array} because the body of \texttt{array/tabular} is in the same grouping context of \texttt{\@mkpream}. With \texttt{array}, on the other hand, this assignment has no effect but safe because it is included in a group of \texttt{\@mkpream}'s own.

The macro \texttt{\@addamp} is also modified to add the code for incrementing the counter \texttt{\adl@currentcolumn} to \texttt{\@preamble} with \&. The counter \texttt{\adl@ncol} is also incremented by \texttt{\@addamp} so that we can refer to its value as \texttt{⟨c⟩} of \texttt{\adl@vlineL}/\texttt{\adl@vlineR}. This increment is done \texttt{\global}-ly in order that we locally set \texttt{\adl@columns} to the counting result outside of the group for \texttt{\@mkpream} of \texttt{array}. Therefore, whether or not \texttt{array} is in use, \texttt{\adl@columns} will have a correct value and will be correctly referred to by \texttt{\hdashline} to know how many columns are specified in the preamble. Note that this \texttt{\global} assignment is safe because the life time of \texttt{\adl@ncol} is same as that of \texttt{\@preamble}.

Since the implementation of \texttt{\@testpach} and macros for class-0 characters (i.e. \texttt{l}, \texttt{r} and \texttt{c}) is completely different between \LaTeX{} and \texttt{array}, we have to have two versions switched by \texttt{\adl@usingarypkg}. 

23
With array

\@testpach Although we introduced two preamble characters `:` and `;`, we did not introduce new character class because we want to minimize the modification of original codes. Therefore, `:` and `;` is classified into class-1 together with `|`. Since these characters obviously have their own appropriate operations, \@testpach is modified so that \@arrayrule, which is invoked from \@mkpream in the case of class-1 character, is \let-equal to the macro corresponding to each character.

\begin{verbatim}
129 \ifadl@usingarypkg
130 \def\@testpach{\@chclass
131 \ifnum @lastchclass=6 \one \@chnum \one \else
132 \ifnum @lastchclass=7 \five \else
133 \ifnum @lastchclass=8 \tw@ \else
134 \ifnum @lastchclass=9 \thr@@
135 \else \z@
136 \ifnum @lastchclass = 10 \else
137 \edef\@nextchar{\expandafter\string\@nextchar}\
138 \@chnum
139 \if \@nextchar c\z@ \else
140 \if \@nextchar l\@ne \else
141 \if \@nextchar r\tw@ \else
142 \z@ \@chclass
143 \if\@nextchar \|\one \let\@arrayrule\adl@arrayrule \else
144 \if\@nextchar :\@ne \let\@arrayrule\adl@arraydashrule \else
145 \if\@nextchar ;\@ne \let\@arrayrule\adl@argarraydashrule \else
146 \if \@nextchar !6 \else
147 \if \@nextchar @7 \else
148 \if \@nextchar <8 \else
149 \if \@nextchar >9 \else
150 10
151 \@chnum
152 \if \@nextchar m\thr@@ \else
153 \if \@nextchar p4 \else
154 \if \@nextchar b5 \else
155 \z@ \@chclass \z@ \@preamerr \z@ \fi \fi \fi \fi \fi \fi
156 \fi \fi \fi \fi \fi \fi \fi \fi \fi \fi \fi \fi
157 158 \def\@classz{@classx
159 \@tempcnta \count@ 160 \prepnext@tok
161 \addtopreamble{\ifcase \@chnum
162 \end{verbatim}

\@classz In array, array and tabular share common macro for class-0 named \@classz, which also generates the preamble for `p`, `m` and `b`. Thus we modify it to measure the height and depth of the class-0 column by the macro \adl@putlrc, and to set \adl@box to the box for `p` and its relatives. Note that a m-type preamble (\@chnum = 3) has to be generated to have \adl@startmbox and \adl@endmbox in it because a \vcenter construct cannot be assigned to \adl@box by \setbox directly.

\begin{verbatim}
158 \def\@classz{@classx
159 \@tempcnta \count@
160 \prepnext@tok
161 \addtopreamble{\ifcase \@chnum
162 \end{verbatim}

24
Another stuff for compatibility is to refer to the class number for the beginning of preamble which is different between \LaTeX{} and \texttt{array}, and that for \texttt{p} or \texttt{G} to get the argument of \texttt{;} as explained later. In the case with \texttt{array}, the former is class-4 and we use \texttt{G} (class-7) for the latter.

\begin{verbatim}
\adl@class@start 4
\adl@class@iiiorvii 7
\end{verbatim}

\begin{verbatim}
Without array
\end{verbatim}

The reason why and how we modify \texttt{@testpach} of \LaTeX{} is same as those of \texttt{array}.

\begin{verbatim}
\def\@arrayclassz{
  \ifcase \@lastchclass
    \@acolampacol \or \@ampacol \or \\
  \or \or \@addamp \or \@acolampacol \or \@firstampfalse \@acol \fi}
\end{verbatim}

Since \LaTeX{} has two macros for class-0, one for \texttt{array} and the other for \texttt{tabular}, we have to modify both. Since the box for \texttt{p} is opened by \texttt{@@startpbox}, however, we may not worry about it.

\begin{verbatim}
\def\arrayclassz{\ifcase \@lastchclass \@acolampacol \or \@ampacol \or \\
  \or \or \@addamp \or \@acolampacol \or \@firstampfalse \@acol \fi}
\end{verbatim}
In \LaTeX, the beginning of preamble is class-6 and we use ‘p’ (class-3) to get the argument of ‘;’.

Hereafter, codes for \LaTeX{} and \emph{array} are common again.

The macro \texttt{\adl@putlrc} is for class-0 preamble characters to set \texttt{\adl@box} to the contents of a column, measure its height/depth by \texttt{\adl@colhtdp} and put the box by \texttt{\unhbox} (not by \texttt{\box}) in order to make the glues in the contents effective.

The preamble parts for vertical solid- and dash-lines are constructed by the macros \texttt{\adl@arrayrule} for ‘|’, \texttt{\adl@arraydashrule} for ‘:’, and \texttt{\adl@argarraydashrule} for ‘;’. The macro;

\begin{verbatim}
\adl@xarraydashrule{(cL)}{(cR)}{(d)/(g)}
\end{verbatim}

is invoked by them to perform common operations. It at first checks the preamble character is the first element of the preamble (\texttt{\@lastchclass} = \texttt{\adl@class@start}) or it follows another character for vertical line (\texttt{\@lastchclass} = 1). If this is not satisfied, the vertical line is put at the right edge of a column and thus \texttt{\ifadl@leftrule} is set to false. Then it adds \texttt{\adl@vlineR} to\texttt{\{\ifadl@zwrule}}\texttt{\} and \texttt{\ifadl@inactive} otherwise. Note that \texttt{\(cL\)} is always 1 for main preamble while \texttt{\(cR\)} is the column number given by \texttt{\adl@ncol}, but \texttt{\(cL\)} may not be 1 for the preamble of \texttt{\multicolumn} as described in §4.7. Also note that \texttt{\(I_d\)} and \texttt{\(I_g\)} are \texttt{\CT@arc@} and \texttt{\adl@dashgapcolor} respectively whose bodies are \texttt{\color} for dashes and gaps specified by \texttt{\arrayrulecolor} and \texttt{\dashgapcolor}, or \texttt{\relax} if they are not colored.

In addition, an invisible \texttt{\vrule} of \texttt{\arrayrulewidth} wide is added if both \texttt{\ADLsome wide} and \texttt{\ADLactivate} are in effect, i.e. both \texttt{\ifadl@zwrule} and \texttt{\ifadl@inactive} are false, to keep a space for the vertical line having real width.
pretend it is for ‘p’ if \texttt{array} is not in use, or ‘@’ otherwise. Then it temporally changes the definition of \texttt{@classv}, which is incidentally for the argument of ‘p’ and ‘@’ in the case without/with \texttt{array} respectively, to \texttt{adl@classvfordash} to process the argument of ‘;’ rather than that of ‘p’ or ‘@’. Then \texttt{adl@classvfordash} is invoked by \texttt{@mkpream} and it adds the argument to \texttt{@preamble}. Finally, it restores the definition of \texttt{@classv} and sets \texttt{@chclass} to 1 to indicate that the last item is a vertical line specification.

\begin{verbatim}
210 \def\adl@arrayrule{\adl@xarraydashrule{\@ne}{\adl@ncol}{{\z@/\z@}}}  
211 \def\adl@arraydashrule{\adl@xarraydashrule{\@ne}{\adl@ncol}{{\dashlinedash/\dashlinegap}}}  
212 \def\adl@argarraydashrule{\adl@xarraydashrule{\@ne}{\adl@ncol}{{}}}  
213 \let\adl@classv\@classv  
214 \def\adl@classvfordash\@addtopreamble{{\@nextchar}}\let\@classv\adl@classvfordash  
215 \@chclass\@ne 
\end{verbatim}

4.6 Building Columns

\texttt{\let\adl@colhtdp\adl@@colhtdp \let\adl@vlineL\adl@@vlineL \let\adl@vlineR\adl@@vlineR} 

If \texttt{array} is not in use, after the \texttt{@preamble} is completed, the control sequences for macros in it should regain their own definition. The macro \texttt{adl@preaminit} performs this operation for macros we introduced, \texttt{adl@colhtdp}, \texttt{adl@vlineL} and \texttt{adl@vlineR}. For the case with \texttt{array}, we will call \texttt{adl@preaminit} in \texttt{arydshln} to initiate them with the definitions as described later.

\begin{verbatim}
238 239 %% Building Columns  
240 \def\adl@preaminit{\let\adl@colhtdp\adl@@colhtdp \let\adl@vlineL\adl@@vlineL \let\adl@vlineR\adl@@vlineR} 
\end{verbatim}
For the measurement of the height and depth of a row, \adl@colhtdp compares \adl@height and \adl@depth to the height and depth of \adl@box which contains the main part of the column to be built, and \global-ly updates the registers if they are less.

\def\adl@colhtdp{\ifdim\adl@height<\ht\adl@box \global\adl@height\ht\adl@box \fi \ifdim\adl@depth<\dp\adl@box \global\adl@depth\dp\adl@box\fi}

The macro \adl@vlineL\langle Γd\rangle\langle Γg\rangle\langle c\rangle \{\langle d\rangle/\langle g\rangle\} adds the element \(e = \langle c, d, g\rangle = \elt{(c)} \{(d)\}(\langle g\rangle)\} \{\langle γd\rangle\}(\langle γg\rangle)\} \text{ to the tail of the list } \adl@colsL \text{ to construct } \mathcal{C}_i^L \text{, where } γd \text{ and } γg \text{ are the color specifications given by } \color \text{ macros in } Γ_d \text{ and } Γ_g \text{. The macro } \add@@vlineR \text{ performs similar operation but the element is added to the head of } \adl@colsR \text{ for } \mathcal{C}_i^R \text{ because it is processed right-to-left manner. The argument } ⟨d⟩ \text{ and } ⟨g⟩ \text{ are extracted by the macro } \adl@ivline \text{ which converts given dimensional values of them to integers. It also sets } ⟨d⟩ \text{ and } ⟨g⟩ \text{ to 0 (i.e. solid-line) if one of given values are not positive, in order to make it sure that one dash segment has positive length. Then it invokes } \adl@setcolor \text{ to define } \adl@dashcolor \text{ and } \adl@gapcolor \text{ with the color specification of } Γ_d \text{ and } Γ_g \text{. Since } \adl@setcolor \text{ locally expands } \color \text{ macro in } Γ_d \text{ and } Γ_g \text{ to define } \current@color \text{ that becomes the body of } \adl@dashcolor \text{ and } \adl@gapcolor \text{ with expansion, different } \color \text{ specifications of a color, such as } \color{red} \text{ and } \color{rgb}{1,0,0}, \text{ will produce a unified result such as } \{\text{rgb 1 0 0}\}. \text{ If } Γ_d \text{ or } Γ_g \text{ is } \relax \text{ which is the body of } \adl@nocolor, \gamma_d \text{ or } γ_g \text{ is also } \relax \text{ to indicate dashes are colored (or not colored) as done in outer world and gaps are transparent.}
in the group of \@mkpream, and regain their own definitions outside the group where the completed \@preamble is referred to.

\adl@preaminit

\adl@inactivevl If \ADLinactivate is in effect, \adl@vlineL/R \adl@@vlineL/R are \let-equal to \adl@inactivevl. This macro simply puts a \vrule by \vline with \color (or \relax) in its first argument and with/without negative \hskip of a half of \arrayrulewidth wide depending on \ifadl@zwvrule, discarding other arguments.

\def\adl@inactivevl#1#2#3#4{\ifadl@zwvrule \hskip-.5\arrayrulewidth \fi
\if#1\vline\fi\ifadl@zwvrule \hskip-.5\arrayrulewidth \fi}

\@@startpbox \@@endpbox \@startpbox \@endpbox \adl@startmbox \adl@endmbox

The macros to make \parbox for 'p', \@@startpbox and \@@endpbox, are modified for height/depth measurement. The code for \@@endpbox is based on that of \LaTeX \epsilon to fix the bug of \strut-ing in \LaTeX-2.09, but \@finalstrut is manually expanded because it is not available in \LaTeX-2.09.

In array, these two macros are not used but \@startpbox and \@endpbox are. Until v2.4h, the former may be untouched and the latter can be \let-equal to \@@endpbox. However in v2.4i, \color@begingroup and \color@endgroup are added to them to make the compatibility issue a little bit complicated. That is, our version of \@@endpbox would have to have \color@endgroup if and only if array is v2.4i or later because \@startpbox has \color@begingroup in these versions, if we relied on the original \@startpbox. To avoid version dependent coding, we copy the new definition of \@startpbox to ensure it has \color@begingroup and let our own \@endpbox with height/depth measurement have \color@endgroup irrespective of the version of array. Note that the assigning the box having 'p' or 'b' to \adl@box for the measurement is done in our own \\classz shown in §4.5.

As for m-type columns, we need a special care because its body \vcenter cannot be assigned directly to \adl@box by \setbox. Thus we enclose a $\vcenter{\ldots}$ construct in a \hbox and assign it to \adl@box. The construct is opened and closed by the macros \adl@startmbox and \adl@endmbox with \@startpbox and what \@endpbox of array 2.4j has, in order to perform color-grouping regardless of the version of array. The latter macro also has our own function to measure the height and depth of the \hbox by \adl@colhtdp. Note that \@startpbox in \adl@startmbox can be different from the definition made here and, more specifically, will be \adl@LTstartpbox when longtable is in use. Also note that the mechanism with \vcenter was replaced with a vertical shift of a box for 'm' in v2.4f of array, but we stick the old mechanism to avoid version dependent coding.

\def\@@startpbox#1{\setbox\adl@box\vtop\bgroup 
\hsize#1\@arrayparboxrestore}
\def\@@endpbox{\unskip \ifhmode \nobreak
\vrule\@width\z@\@height\z@\@depth\dp\@arstrutbox \fi
\par \egroup \adl@colhtdp \box\adl@box \hfil}
\def\@startpbox#1{\bgroup
\color@begingroup
15 The author had forgotten this fact until Morten Høgholm pointed out it. Thanks Morten.
4.7 Multi-columns

The macro \texttt{\multicolumn} is modified for the following.

- The macros to construct the parts of \texttt{\@preamble} for vertical lines, \adl@arrayrule, \adl@arraydashrule and \adl@arraydashrule, have to perform operations slightly different from those for main preamble. Thus they are \texttt{\def}-ined to multi-column version \adl@mcarrayrule, etc. These \texttt{\def}-initions are enclosed in a group so that they are not affected to \texttt{array} or \texttt{tabular} which may occur in the third argument of \texttt{\multicolumn}. In order to make \texttt{\@preamble} work well outside of the group containing \texttt{\@makepream}, \adl@preamble is \texttt{\global}-ly \texttt{\let}-equal to \texttt{\@preamble} just after \texttt{\@makepream} in the group and then reverse \texttt{\let}-assignment is performed just after the group is closed. These global assignment is unnecessary with \texttt{array} because \texttt{\@preamble} is constructed \texttt{\global}-ly, but safe.

Since this grouping nullifies the effect of \adl@preaminit called in \texttt{\@mkpream}, we call \adl@preaminit again after the group closing.

- In \texttt{array}, \texttt{\@addamp} to make \texttt{\@preamble} for \texttt{\multicolumn} has a different definition from that for main one. Thus it is \texttt{\let}-equal to \adl@mcaddamp whose definition is switched by \texttt{\ifadl@usingarypkg}.

- If \texttt{array} is in use, \texttt{\@preamble} has to be \texttt{\xdef}-ed once again by \texttt{\@addpreamble} with an \texttt{\@empty} argument after \texttt{\@mkpream} to expand the contents of \texttt{\toks} registers. This is performed whether or not with \texttt{array} because it is safe.

- As done in \texttt{\@array}, \texttt{\set@typeset@protect} is replaced with direct \texttt{\let}.

- If without \texttt{array}, \texttt{\@startpbox} and \texttt{\@endpbox} should be \texttt{\let}-equal to their \texttt{@@} counterparts, while should not with \texttt{array}. Thus we define \adl@activatepbox to do or not to do so depending on \texttt{\ifadl@usingarypkg}.

- The counter \adl@currentcolumn is \texttt{\global}-ly incremented by the first argument of \texttt{\multicolumn} (number of columns to be \texttt{\span}-ned).

Note that \adl@columns is modified by \texttt{\@mkpream}, but it is not referred to by \adl@mcarrayrule etc., and its value is restored before referred to by \texttt{\hdashline}, etc.
\def\multicolumn#1#2#3{\multispan{#1}\begingroup\begingroup
\def\adl@arrayrule{\adl@mcarrayrule{#1}}%
\def\adl@arraydashrule{\adl@mcarraydashrule{#1}}%
\def\adl@argarraydashrule{\adl@mcargarraydashrule{#1}}%
\let\@addamp\adl@mcaddamp
\@mkpream{#2}\@addtopreamble\@empty
\global\let\adl@preamble\@preamble \endgroup
\let\@preamble\adl@preamble
\def\@sharp{#3}\let\protect\relax
\adl@activatepbox
\adl@preaminit
\@arstrut \@preamble\hbox{\endgroup
\global\advance\adl@currentcolumn#1\ignorespaces}
\ifadl@usingarypkg
\def\adl@mcaddamp{\if@firstamp\@firstampfalse \else\@preamerror5\fi}
\let\adl@activatepbox\relax
\else
\let\adl@mcaddamp\@addamp
\def\adl@activatepbox{\let\@startpbox\@@startpbox
\let\@endpbox\@@endpbox}
\fi

The preamble parts for vertical lines are constructed by the macros \adl@mcarrayrule, \adl@mcarraydashrule and \adl@mcargarraydashrule to which the first argument \langle n \rangle of \multicolumn is passed to know the number of columns to be \spanned. They are similar to their relatives for main preamble, \adl@arrayrule, etc., but the arguments \langle c_L \rangle and \langle c_R \rangle passed to \adl@arraydashrule are;

\[c_L = c, \quad c_R = c + n - 1\]

where \(c = \text{\adl@currentcolumn}\). This makes leading vertical lines drawn at the left edge of the leftmost \spanned column and trailing ones at the right edge of the rightmost column.

\[\}
\def\adl@mcarraydashrule#1{\@tempcnta#1\advance\@tempcnta\adl@currentcolumn
\advance\@tempcnta\m@ne\adl@xarraydashrule{\adl@currentcolumn}{\@tempcnta}{{\dashlinedash/\dashlinegap}}
\def\adl@mcargarraydashrule#1{\@tempcnta#1\advance\@tempcnta\adl@currentcolumn
\advance\@tempcnta\m@ne\adl@xarraydashrule{\adl@currentcolumn}{\@tempcnta}{}\]
4.8 End of Rows

At the end of the $i^{th}$ row, we have to calculate $h_i$ which is the height plus depth of the row, and add elements $(C_L^i, h_i)$ and $(C_R^i, h_i)$ to $R^L$ and $R^R$. To do this, \texttt{\p@}\p\texttt{-s in the macros \texttt{\p@xarraycr}, \texttt{\p@xtabularcr}, \texttt{\p@xargarraycr} are replaced with our own \texttt{\adl@}\p\texttt{cr}. The macro \texttt{\p@yargarraycr(dimen)} is also modified but its \texttt{\p@cr} is replaced with \texttt{\adl@argcr(dimen)} to add (negative) \texttt{\p@dimen} to $h_i$. Note that \texttt{\p@yargarraycr(dimen)} uses ordinary \texttt{\adl@cr} because the extra vertical space of \texttt{\p@\texttt{\texttt{\p@dimen}}} is inserted to the last column.

Note that the implementation of \texttt{\p@xarraycr} is slightly different between \LaTeX{} and \texttt{\p\texttt{\texttt{\p@array}}} we have to have two versions and choose one.

- The macro \texttt{\p@\p@cr} performs \texttt{\p@cr} and then invoke the common macro \texttt{\adl@\p@cr(x)}. The argument \texttt{\p@\texttt{\p@dimen}} is the extra (negative) vertical space for \texttt{\adl@\p@argcr}, while it is 0 for \texttt{\adl@\p@cr}.

The macro \texttt{\p@\adl@\p@cr(x)} at first calculate $h_i$ as follows. The registers \texttt{\adl@\p@height = $\eta$} and \texttt{\adl@\p@depth = $\delta$} have the maximum height and depth of the columns in the row. However, they could be smaller than the height and/or depth of \texttt{\p@arstrutbox}, $\eta_s$ and $\delta_s$. If so, the height and/or depth of the row are $\eta_s$ and $\delta_s$. Therefore, $h_i$ is calculated by:

$$h_i = \max(\eta, \eta_s) + \max(\delta, \delta_s).$$

Additionally, if the extra space \texttt{\p@\texttt{\p@dimen}} is negative, a vertical space of $x$ is inserted below the row\textsuperscript{16}. Thus the integer value of $h_i + x$ is \texttt{\p@\p@\p@\p@local}ly added to \texttt{\adl@\p@totalheight}, and the elements $(C_L^i = \texttt{\adl@colsL}, h_i)$ and $(C_R^i = \texttt{\adl@colsR}, h_i)$ are added to the tail of $R^L = \texttt{\p@\p@\p@\p@rowsL}$ and $R^C = \texttt{\adl@\p@\p@rowsR}$. If $x$ is not 0 (negative), discard$(x)$ or connect$(x)$

\textsuperscript{16}Before v1.54, negative \texttt{\p@\p@\p@\p@dimen} shrinks the height of the row by $|x|$. Although the former result may be more appropriate if the row has vertical lines than the current because lines extrude to the next row now, new feature is considered compatible with original \texttt{\p@array/\p@\texttt{\texttt{\p@tabular}}}.

32
is also added after \( (C^L_i, h_i) \) according to the current environment (longtable or not). In the real implementation, \( R^L \) and \( R^C \) has the following format of \( \langle \text{rows} \rangle \).

\[
\langle \text{rows} \rangle ::= \left[ \langle \text{row} \rangle ; \right]^* \\
\langle \text{row} \rangle ::= \langle \langle \text{cols} \rangle / (h_i) \rangle \\
\langle \text{cols} \rangle ::= \left[ \text{elt}\{(c)\}(\langle d \rangle)(\langle g \rangle)\right]^* \\
\adl@connect | \quad \ldots \text{C}^L \text{ or } \text{C}^R \\
\adl@discard | \quad \ldots \text{for } \text{connect}(h_i) \\
\relax \quad \ldots \text{for } \text{disconnect}(h_i)
\]

Since \adl@discard is \texttt{def} ined as \adl@connect by \adl@arrayinit, added \adl@discard transforms itself into \adl@connect if current environment is not longtable. Otherwise, as we make \adl@discard \texttt{let}-equal to \relax when a longtable environment starts, it keeps its own form.

Then, \adl@finaldepth is set to \adl@depth if \( x \) is zero, or to zero otherwise (negative), in order to make the depth of array/tabular equal to that of the last row. Finally, \adl@colsL, \adl@colsR, \adl@currentcolumn, \adl@height and \adl@depth are reinitialized to process the next row.

4.9 Horizontal Lines

The macro \texttt{\hline} is modified to insert \texttt{\vskip-\arrayrulewidth} before drawing if \ADLnullwidehline is in effect, or to add the element \texttt{connect(}\( w \)) = \texttt{(\adl@connect/number\arrayrulewidth)} to the end of \( R^L \) and \( R^C \) by \adl@hline otherwise. The other modifications are to set \adl@finaldepth to zero for the case that the last vertical item
is \hline, and to check if it is followed by not only \hline but also \hdashline by \adl@xhline.

The macro \cline is also modified to set \adl@finaldepth to zero. As for the feature of \ADLnullwidehline, it inserts \vskip-\arrayrulewidth to shift the line up before drawing, and \vskip\arrayrulewidth after drawing to cancel the negative skip inserted by \adl@org@cline.

The macro \hdashline calls \adl@hdashline to open the \noalign construct by the well-known trick {\ifnum0='}\fi and then to invoke \adl@ihdashline checking the existence of its optional argument \{(dash)/(gap)\}. Before the invocation, it inserts \vskip-\arrayrulewidth if \ADLnullwidehline is in effect, or adds connect(w) to the end of R\ and R\(^T\). Then \adl@ihdashline closes the \noalign by \ifnum0='\fi to start the pseudo row for the horizontal dash-line. Before the dash-line is drawn by \adl@hcline which is also used for \cdashline, all the columns are \spanned by giving \adl@columns to \multispan. Finally, the \noalign is opened again and \adl@xhline is invoked to check whether \h(dash)line is followed.

If \ADLinactivate is in effect, \adl@ihdashline is \let-equal to \adl@inactivehdl. This macro simply puts a \hrule discarding its arguments after inserting \vskip-\arrayrulewidth if \ADLnullwidehline is in effect.
The macro \texttt{\textbackslash adl@xhline} is the counterpart of the original \texttt{\textbackslash xhline}. This is introduced to check the mixed sequence of \texttt{\textbackslash hline} and \texttt{\textbackslash hdashline}, and to add the element \texttt{\textbackslash disconnect(s) = (\textbackslash relax/\textbackslash doublerulesep) to the end of R^L and R^R by \texttt{\textbackslash adl@xhline} if a pair of \texttt{\textbackslash h(dash)line} is found.

\begin{verbatim}
\def\adl@xhline{\ifx\@tempa\hline \adl@ixhline\fi
\ifx\@tempa\hdashline \adl@ixhline\fi
\ifnum0='{\fi}}
\def\adl@ixhline{\vskip\doublerulesep \adl@hline\relax\doublerulesep}
\end{verbatim}

The macro \texttt{\textbackslash adl@xhline(\textit{cs})\textit{\textbackslash dimen}} \texttt{\textbackslash global-ly adds the integer value of \textit{\textbackslash dimen}} to \texttt{\textbackslash adl@totalheight} and adds the element \texttt{\langle\textit{cs}\rangle/\texttt{\textbackslash number(\textit{\textbackslash dimen})}} to the tail of \texttt{\textbackslash R^L} and \texttt{\textbackslash R^R}. The arguments \texttt{\langle\textit{cs}\rangle\textit{\textbackslash dimen}} are \texttt{\textbackslash adl@connect/\textbackslash arrayrulewidth} for \texttt{\textbackslash connect(w) or \textbackslash relax/\textbackslash doublerulesep} for \texttt{\textbackslash disconnect(s)}.

\begin{verbatim}
\def\adl@hline#1#2{\@tempcnta#2
\global\advance\adl@totalheight\@tempcnta
\xdef\adl@rowsL{\adl@rowsL(\textit{w}/\texttt{\textbackslash number(\textit{\textbackslash dimen})})}}
\end{verbatim}

The macro \texttt{\textbackslash cdashline} at first opens \texttt{\textbackslash noalign} and then invokes \texttt{\textbackslash adl@cdline} checking the existence of its optional argument \texttt{\langle\textit{dash}/\textit{\textbackslash gap}\rangle}. The macro \texttt{\textbackslash adl@cdline} first inserts \texttt{\vskip-\textbackslash arrayrulewidth} if \texttt{\textbackslash ADLnullwidehline} is in effect. Then it performs column \texttt{\span-ning by the code based on that of \textbackslash @cline in \LaTeX-2.09 because \LaTeX~2\epsilon's version will not work with \LaTeX~2.09.} The main job is done by \texttt{\textbackslash adl@hcline} after the target columns are \texttt{\span-ned by \textbackslash adl@cdlinea or \textbackslash adl@cdlineb}.

\begin{verbatim}
\def\cdashline#1{\noalign{\ifnum0='}\fi
\if\adl@zwhrule \def\@gtempa{\adl@cdline}\else \def\@gtempa{\adl@cdlineb}\fi
\vskip-\arrayrulewidth}
\end{verbatim}

If \texttt{\textbackslash ADLinactivate} is in effect, \texttt{\textbackslash adl@cdline} is \texttt{\let}-equal to \texttt{\textbackslash adl@inactivecdl}. This macro simply calls our own \texttt{\textbackslash cline}, after closing the \texttt{\textbackslash noalign} opened by \texttt{\textbackslash cdashline}.

\begin{verbatim}
\def\cdashline#1{\noalign{\if\adl@zwhrule \def\@gtempa{\adl@cdline}\else \def\@gtempa{\adl@cdlineb}\fi
\vskip-\arrayrulewidth}}
\end{verbatim}

\begin{verbatim}
\def\adl@activecdl{\def\adl@cdlinea\adl@cdlineb{\unskip \adl@hcline}}
\end{verbatim}

\begin{verbatim}
\def\adl@cdline{
\global\advance\adl@totalheight\@tempcnta
\xdef\adl@rowsL{\adl@rowsL(\textit{w}/\texttt{\textbackslash number(\textit{\textbackslash dimen})})}
\xdef\adl@rowsR{\adl@rowsR(\textit{w}/\texttt{\textbackslash number(\textit{\textbackslash dimen})})}}
\end{verbatim}

\begin{verbatim}
\def\adl@cdlinea{\multispan\adl@cla \adl@clb \unskip \adl@hcline}
\end{verbatim}

\begin{verbatim}
\def\adl@cdlineb{\multispan\adl@cla \adl@clb \unskip \adl@hcline}
\end{verbatim}
The macro \adl@hcline\[w\][\(d\)/\(g\)] draws a horizontal dash-line of dash size \(d\) and gap size \(g\) for \hdashline and \cdashline in the \spanned columns by \adl@draw. As we will discuss in §4.12, the macro requires \(d\) and \(g\) are passed through \@tempdima and \@tempdimb, and control sequences \(\langle rule\rangle\), \(\langle skip\rangle\) and \(\langle box\rangle\) are passed through its arguments to make it usable for both horizontal and vertical lines. Then the vertical space of \(w\), \(\text{--}\arrayrulewidth\) for \cdashline, is inserted if it is not 0 (for \hdashline) and \ADLnullwidehline is not in effect.

\[\text{\adl@hcline}\text{\[\#1/\#2\]\[\#3\]}\]

\(\text{\adl@hcline}\text{\[\#1/\#2\]\[\#3\]}\)

\[\text{\firsthdashline}\text{\[\#1/\#2\]}\]

\[\text{\lasthdashline}\text{\[\#1/\#2\]}\]

\[\text{\adl@firsthdashline}\text{\[\#1/\#2\]}\]

\[\text{\adl@lasthdashline}\text{\[\#1/\#2\]}\]

\[\text{\adl@firsthdashline}\text{\[\#1/\#2\]}\]

\[\text{\adl@lasthdashline}\text{\[\#1/\#2\]}\]

\[\text{\adl@firsthdashline}\text{\[\#1/\#2\]}\]

\[\text{\adl@lasthdashline}\text{\[\#1/\#2\]}\]

\[\text{\adl@firsthdashline}\text{\[\#1/\#2\]}\]

\[\text{\adl@lasthdashline}\text{\[\#1/\#2\]}\]

\[\text{\adl@firsthdashline}\text{\[\#1/\#2\]}\]

\[\text{\adl@lasthdashline}\text{\[\#1/\#2\]}\]
4.10 End of Environment

The macros to close the \texttt{array/tabular} environment, \texttt{\endarray} and \texttt{\endtabular(*)}, are modified so that they invoke \texttt{\adl@endarray} to draw vertical lines just before closing \texttt{\halign}, and \texttt{\adl@arrayrestore} to restore registers and data structures \texttt{\global}-ly modified in the environment. Note that \texttt{array} and related packages such as \texttt{delarray} define a macro \texttt{\arrayright} as the closing hook and thus we invoke it if it is defined.

\begin{verbatim}
441 \% End of Environment
442 \def\endarray{\adl@endarray \egroup \adl@arrayrestore \egroup
443  \csname @arrayright\endcsname}
444 \def\endtabular{\endarray \egroup}
445 \expandafter\let\csname endtabular*\endcsname\endtabular
\end{verbatim}

The macro \texttt{\adl@endarray} at first closes the last row by \texttt{\crcr}. If this \texttt{\crcr} has real effect, we have to invoke \texttt{\adl@@cr} to perform our own end-of-row operations. We assume that the \texttt{\crcr} is effective if either \texttt{\adl@height} or \texttt{\adl@depth} has a non-zero value\textsuperscript{17}.

The rows to draw vertical lines $L_1, \ldots, L_n$; $\sigma_1 L_1 \sigma_2 L_2 \ldots L_{n-1} \sigma_n L_n \sigma_{n+1}$ are created in \texttt{\adl@vlrowL} and \texttt{\adl@vlrowR} by \texttt{\adl@makevlrL} and \texttt{\adl@makevlrR}. In the real implementation, $L_k = \langle \gamma_k, \pi_k, \delta_k, \tau_k, \beta_k \rangle$ is represented as:
\begin{verbatim}
\adl@vl{\beta_k}{\tau_k - \beta_k}{\delta_k}{\xi_k}
\end{verbatim}

Thus \texttt{\adl@vl} is made \texttt{\relax} when the rows are constructed and to \texttt{\adl@vl} when the rows are put.

Since \texttt{\adl@makevlrL} and \texttt{\adl@makevlrR} shares common macros, they conceptually have the following interface.
\begin{verbatim}
\adl@vlrow = \adl@makevlrL/R(\adl@rows:⟨R_L or R_R⟩, \adl@currentcolumn:⟨start column⟩, \adl@addvl:⟨macro to add an element⟩)
\end{verbatim}

Thus they are invoked as:
\begin{verbatim}
\adl@vlrowL = \adl@makevlrL(\adl@rowsL, 1, \adl@addvlL)
\adl@vlrowR = \adl@makevlrR(\adl@rowsR, \adl@columns, \adl@addvlR)
\end{verbatim}

Finally, after constructed rows for vertical lines are put by \texttt{\adl@drawvl}, a vertical skip of $-\adl@finaldepth$ is inserted to move back to the last baseline, and then an invisible \texttt{\vrule} of \texttt{\adl@finaldepth} deep is put to make \texttt{array/tabular} has the depth of the last real row or zero if it ends with a horizontal line.

\textsuperscript{17}The author confesses that this rule is not strict and the introduction of a switch could improve the strictness.
The macro \texttt{\adl@arrayrestore} restores the values of registers and data structures, \texttt{\adl@height}, \texttt{\adl@depth}, \texttt{\adl@currentcolumn}, \texttt{\adl@totalheight}, \texttt{\adl@rowsL}, \texttt{\adl@rowsR}, \texttt{\adl@colsL} and \texttt{\adl@colsR}, saved by \texttt{\adl@arrayinit}.

\texttt{\adl@makevlrL} corresponds to the line (2) and (31)–(36). Its right-edge counterpart \texttt{\adl@makevlrR} has the same correspondence but the lines (1)–(2) are;

\begin{verbatim}
\def\adl@endarray{\crcr \noalign{\ifdim\adl@height=\z@\ifdim\adl@depth=\z@ \else \adl@cr\z@ \fi \else \adl@cr\z@ \fi \let\adl@vl=\relax \def\adl@vlrow{} \adl@currentcolumn\@ne \let\adl@rows\adl@rowsL \let\adl@addvl\adl@addvlL \adl@makevlrL \global\let\adl@vlrow\@adl@vlrow \def\adl@vlrow()\adl@currentcolumn\@one \let\adl@rows\adl@rowsR \let\adl@addvl\adl@addvlR \adl@makevlrR \global\let\adl@vlrow\@adl@vlrow \global\let\adl@vlrow\@adl@vlrowL} \global\let\adl@vl=\adl@@vl \let\adl@drawvl \noalign{\vskip-\adl@finaldepth} \omit\vrule@width\z@@height\z@@depth\adl@finaldepth\cr}
\adl@arrayrestore
\end{verbatim}

\subsubsection*{4.11 Drawing Vertical Lines}

Figure 2 shows the conceptual code of \texttt{\adl@makevlrL}. The correspondance of variables in the code and control sequences in the real implementation is as follows.

\begin{align*}
R^L & : \texttt{\adl@rowsL} & R & : \texttt{\@tempb} & A & : \texttt{\adl@vlrowL} \\
\Gamma & : \texttt{\adl@columns} & \gamma & : \texttt{\adl@currentcolumn} & \tau & : \texttt{\@tempcnta} & \beta & : \texttt{\@tempcntb} & \eta & : \texttt{\adl@lastconn} \\
\delta & : \texttt{\adl@dash/\@adl@dashcolor} & \xi & : \texttt{\adl@gap/\@adl@gapcolor} & H & : \texttt{\adl@totalheight} \\
\text{conn} & : \texttt{\ifadl@connected} & \text{double} & : \texttt{\ifadl@doublerule}
\end{align*}

\texttt{\adl@makevlrL} The macro \texttt{\adl@makevlrL} corresponds to the line (2) and (31)–(36). Its right-edge counterpart \texttt{\adl@makevlrR} has the same correspondence but the lines (1)–(2) are;
(1) \( A \leftarrow \emptyset; R \leftarrow R^L; \gamma \leftarrow 1; \)

(2) \( \text{while } \gamma \leq \Gamma \text{ do begin} \)

(3) \( \tau \leftarrow H; \beta \leftarrow H; \eta \leftarrow 0; \delta \leftarrow (\bot, \bot); \xi \leftarrow (\bot, \bot); \)

(4) \( \text{conn } \leftarrow \text{false}; \text{double } \leftarrow \text{false}; R' \leftarrow \emptyset \)

(5) \( \text{while } R \neq \emptyset \text{ do begin} \)

(6) \( (r,R) \leftarrow R; \)

(7) \( (C,h) \leftarrow r; \)

(8) \( \text{if } C = \emptyset \text{ then begin } \text{add}(\tau,\beta,\delta,\xi); \eta \leftarrow 0; \text{ end; } \)

(9) \( \text{elseif } C = \langle \text{connect} \rangle \text{ then } \eta \leftarrow h; \)

(10) \( \text{else begin } \)

(11) \( \langle e,C' \rangle = C; \langle e,d,g \rangle = e; \)

(12) \( \text{if } c = \gamma \text{ then begin } \)

(13) \( \text{if } d = \delta \land g = \xi \text{ then begin } \)

(14) \( \text{if } \neg \text{conn then begin } \)

(15) \( \tau \leftarrow \beta + \eta; \text{conn } \leftarrow \text{true; } \)

(16) \( \text{end; } \)

(17) \( \text{end; } \)

(18) \( \text{else begin } \)

(19) \( \text{add}(\tau,\beta,\delta,\xi); \)

(20) \( \delta \leftarrow d; \xi \leftarrow g; \tau \leftarrow \beta + \eta; \text{conn } \leftarrow \text{true; } \)

(21) \( \text{end; } \)

(22) \( \text{if } C' = \langle \langle \gamma,?,?,? \rangle,? \rangle \text{ then double } \leftarrow \text{true; } \)

(23) \( C \leftarrow C'; \)

(24) \( \text{end; } \)

(25) \( \text{else } \text{add}(\tau,\beta,\delta,\xi); \)

(26) \( \eta \leftarrow 0; \)

(27) \( \text{end; } \)

(28) \( \beta \leftarrow \beta - h; R' \leftarrow (R',\langle C,h \rangle) \)

(29) \( \text{end; } \)

(30) \( \text{add}(\tau,\beta,\delta,\xi); R \leftarrow R'; \)

(31) \( \text{if double then } \Lambda \leftarrow \langle A,\backslash hskip\backslash doublerulesep \rangle; \)

(32) \( \text{else begin } \)

(33) \( \gamma \leftarrow \gamma + 1; \)

(34) \( \text{if } \gamma > \Gamma \text{ then } A \leftarrow \langle A,\backslashhfil \rangle; \)

(35) \( \text{else } A \leftarrow \langle A,\backslashhfil&\omit \rangle; \)

(36) \( \text{end; } \)

(37) \( \text{end; } \)

(38) \( \text{end; } \)

(39) \( \text{procedure } \text{add}(\tau,\beta,\delta,\xi) \text{ begin } \)

(40) \( \text{if conn then begin } \)

(41) \( A \leftarrow \langle A,\langle \beta,\tau - \beta,\delta,\xi \rangle \rangle; \text{conn } \leftarrow \text{false; } \)

(42) \( \text{end; } \)

(43) \( \text{end; } \)

Figure 2: Conceptual Code of \texttt{adl@makevlrL}
(1) $A \leftarrow \langle \rangle; \ R \leftarrow R^R; \ \gamma \leftarrow \Gamma;$

and (31)--(36) are:

(31) \quad \text{if } \text{double } \text{then } \ A \leftarrow \langle \text{hskip}\text{doublerulesep}, A \rangle; \\
(32) \quad \text{else begin} \\
(33) \quad \gamma \leftarrow \gamma - 1; \\
(34) \quad \text{if } \gamma = 0 \text{ then } \ A \leftarrow \langle \text{hss}, A \rangle; \\
(35) \quad \text{else } \ A \leftarrow \langle \text{	extbackslash omit} \text{hss}, A \rangle; \\
(36) \quad \text{end;}

\% Drawing Vertical Lines
\def\adl@makevlrL{\adl@makevlr
\ifadl@doublerule
\edef\adl@vlrow{\adl@vlrow \hskip\doublerulesep}\% \\
\let\next\adl@makevlrL
\else
\advance\adl@currentcolumn\@ne
\ifnum\adl@currentcolumn>\adl@columns \let\next\relax \\
\edef\adl@vlrow{\adl@vlrow \hss}\% \\
\else \let\next\adl@makevlrL \\
\edef\adl@vlrow{\adl@vlrow \hss \&\omit}\% \\
\fi\fi
\next}
\def\adl@makevlrR{\adl@makevlr
\ifadl@doublerule
\edef\adl@vlrow{\hskip\doublerulesep \adl@vlrow}\% \\
\let\next\adl@makevlrR \\
\else
\advance\adl@currentcolumn\@ne
\ifnum\adl@currentcolumn=\z@ \let\next\relax \\
\edef\adl@vlrow{\hss \adl@vlrow}\% \\
\else \let\next\adl@makevlrR \\
\edef\adl@vlrow{\&\omit \hss \adl@vlrow}\% \\
\fi\fi
\next}

\adl@makevlr \quad \text{The macro } \adl@makevlr \text{ corresponds to the lines (3)–(4) and (30).}
The macro \texttt{\adl@makevlr(⟨r⟩)}; corresponds to the lines (5)–(6), and the macro \texttt{\adl@iimakevlr((⟨C⟩)/⟨h⟩)} to (7) and (28).

\texttt{\adl@endmakevlr}

\texttt{\adl@iiimakevlr}

\texttt{\adl@ivmakevlr}

\texttt{\adl@vmakevlr}

\texttt{\adl@endmakevlrcut}

\texttt{\adl@endmakevlrconn}

\texttt{\adl@connect}

\texttt{\adl@endmakevlr}

\texttt{\adl@endmakevlr}

\texttt{\adl@endmakevlr}

\texttt{\adl@endmakevlr}

\texttt{\adl@endmakevlr}

\texttt{\adl@endmakevlr}

\texttt{\adl@endmakevlr}

\texttt{\adl@endmakevlr}

\texttt{\adl@endmakevlr}

1. If $\texttt{\adl@connect} \neq ⟨⟩$ and $\texttt{\adl@connect} \neq ⟨\texttt{\adl@connect}⟩$, $\texttt{\adl@connect}$ has at least one \texttt{\adl@elt ⟨c⟩⟨d⟩⟨g⟩} which is made \texttt{\let}-equal to \texttt{\adl@ivmakevlr} by \texttt{\adl@iiimakevlr}. Thus the lines (10)–(21) and (25)–(26) are performed by \texttt{\adl@iiimakevlr}.

Then:

(a) if $c = γ$, \texttt{\adl@elt} becomes \texttt{\let}-equal to \texttt{\adl@vmakevlr} which corresponds to (22) in the case of $C' \neq ⟨⟩$. Then \texttt{\adl@vmakevlr} is invoked for (23) and to eat the sentinel \texttt{\adl@endmakevlr}. If $C' = ⟨⟩$, \texttt{\adl@endmakevlrcut} is invoked, because the sentinel \texttt{\adl@endmakevlr} is made \texttt{\let}-equal to it by \texttt{\adl@iiimakevlr}, for (23) (i.e. $C ← ⟨⟩$).

(b) if $c \neq γ$, \texttt{\adl@vmakevlr} is invoked to perform implicit $C ← C$ operation and to eat the sentinel.

2. If $\texttt{\adl@connect} = ⟨\texttt{\adl@connect}⟩$, i.e. it has only one element \texttt{\adl@connect}, the macro \texttt{\adl@connect} is invoked with $h$ because it is \texttt{\define-dl} to be \texttt{\adl@connect((⟨\texttt{\adl@connect}⟩))}. The macro performs (9) and implicit $C ← C (= ⟨\texttt{\adl@connect}⟩)$ eating the sentinel.

3. If $\texttt{\adl@connect} = ⟨⟩$, \texttt{\adl@endmakevlrcut} that is \texttt{\let}-equal to the sentinel \texttt{\adl@endmakevlr} is invoked to perform (8) and implicit $C ← C (= ⟨⟩)$.  

\texttt{\adl@iiimakevlr#1#2#3#4#5{\let\elt\adl@vmakevlr\let\next\relax
\#\drive@gap#2\edef\tempb{\@tempb(#2);}}}}
The macro \adl@addvlL correponds to the lines (38)–(42), i.e. the procedure \textit{add}. The macro \adl@addvlR performs similar operations, but its conceptual code is the following.

\begin{verbatim}
(38) \textbf{procedure} \textit{add}(τ, β, δ, ξ) \begin{align*}
(39) & \text{if } \text{conn} \text{ then begin} \\
(40) & \quad \quad \Lambda \leftarrow \langle β, τ \rangle - \langle β, δ, ξ \rangle; \ \text{conn} \leftarrow \text{false}; \\
(41) & \text{end;} \\
(42) & \text{end;} \\
\end{align*}
\end{verbatim}
After the macros \texttt{\adl@vlrowL} and \texttt{\adl@vlrowR} are constructed, they are expanded to
draw vertical lines by \texttt{\adl@drawvl}. Prior to the expansion, the macro \texttt{\adl@drawvl} globally defines
\texttt{\adl@vl@leftskip} and \texttt{\adl@vl@rightskip}, which are the amount of negative
spaces inserted to the left/right of a vertical line, as follows.

\texttt{\adl@vl@leftskip} = \begin{cases} 
\frac{\text{\arrayrulewidth}}{2} & \text{if } \texttt{\ifadl@zwrule} \\
\text{\arrayrulewidth} & \text{otherwise}
\end{cases}

\texttt{\adl@vl@rightskip} = \begin{cases} 
\frac{\text{\arrayrulewidth}}{2} & \text{if } \texttt{\ifadl@zwrule} \\
\text{\arrayrulewidth} & \text{otherwise}
\end{cases}

That is, if \texttt{\ADLnulwide} is in effect, a vertical line is surrounded by horizontal spaces of
$-\frac{\text{\arrayrulewidth}}{2}$ to adjust the center of the line to the left or right edge of its column.
Otherwise, a horizontal space $-\text{\arrayrulewidth}$ is inserted after (before) the line is drawn
to adjust its left (right) edge to the left (right) edge of the column\footnote{Before v1.54, the horizontal spaces was not inserted if \texttt{\ADLsomewide} and thus disconnected lines were
not aligned vertically.}.

Then the macros \texttt{\adl@vlrowL} and \texttt{\adl@vlrowR} are expanded. These macros will have
\texttt{\adl@vl}, which is made \texttt{\let}-equal to \texttt{\adl@vl} prior to the expansion, to draw a vertical
line. The macro \texttt{\adl@vl\textbackslash l} ($\beta$) ($\lambda$) ($\delta_l$) ($\gamma_l$) ($\delta_c$) ($\gamma_c$) ($x_l$ and $x_c$ are length and color) draws a
solid line if $\gamma_l = 0$ or a dash-line otherwise in a \texttt{\vbox} of \texttt{$\lambda$} = $\tau$ - $\beta$ high and \texttt{\raise}s it
by $\beta$. The method to draw a dash line in the \texttt{\vbox} is analogous to that for horizontal line
shown in §4.9, except that a line is surrounded by horizontal spaces of \texttt{\adl@vl@leftskip}
and \texttt{\adl@vl@rightskip}. Coloring gaps is done by drawing a vertical rule setting $\gamma_c$ by
\texttt{\set@color} prior to dash line drawing if $\gamma_c$ is not \texttt{\relax}. To color dashes or solid line,
\texttt{\set@color} with $\delta_c$ is done if it is not \texttt{\relax} before line drawing.

\begin{verbatim}
573 \def\adl@drawvl{%
574 \omit \relax \ifadl@zwvrule
575 \gdef\adl@vl@leftskip{.5\arrayrulewidth}\% 576 \global\let\adl@vl@rightskip\adl@vl@leftskip
577 \else \global\let\adl@vl@leftskip\z@ 578 \global\let\adl@vl@rightskip\arrayrulewidth
579 \fi \adl@vlrowL \cr
580 \omit \relax \ifadl@zwvrule
581 \gdef\adl@vl@leftskip{.5\arrayrulewidth}\% 582 \global\let\adl@vl@rightskip\adl@vl@leftskip
583 \else \global\let\adl@vl@leftskip\arrayrulewidth
584 \global\let\adl@vl@rightskip\z@ 585 \fi \adl@vlrowR \cr}

587 \def\adl@vlrowL#1#2#3#4#5#6{%\vbox to#2\z@{\vss\hbox{% 588 \hskip-\adl@vl@leftskip
589 \ifnum#3=\z@\else \ifx\@tempa\adl@nocolor\else
590 \raise#1sp\hbox{\let\current@color\@tempa \set@color
591 \vrule height#2sp width\arrayrulewidth}%
592 \fi\fi\fi
\end{verbatim}
4.12 Drawing Dash-lines

\texttt{\textbackslash adl@vrule}\texttt{\textbackslash adl@hrule}

As explained later, horizontal and vertical lines are drawn by a common macro \texttt{\textbackslash adl@draw} to which the length of a dash segment, \(d\), is passed through \texttt{\textbackslash tempdim\#a}. The macro also has an argument that is either \texttt{\textbackslash adl@vrule} to draw a dash for horizontal lines or \texttt{\textbackslash adl@hrule} for vertical. These two macros commonly have one argument \(f\) to draw a dash of \(f \times d\) long and of \texttt{\arrayrulewidth} wide.

\texttt{\textbackslash adl@drawi}\texttt{\textbackslash adl@drawii}\texttt{\textbackslash adl@drawiii}\texttt{\textbackslash adl@draw}

The macro \texttt{\textbackslash adl@draw} is to draw a horizontal or vertical line. It is \texttt{\let}-equal to one of \texttt{\textbackslash adl@drawi}, \texttt{\textbackslash adl@drawii} and \texttt{\textbackslash adl@drawiii} according to the drawing mode specified by \texttt{\ADLdrawingmode}. These three macros have common interface, \texttt{\@tempdima} and \texttt{\@tempdimb} for the length of dash and gap, \(d\) and \(g\), and three arguments \langle rule \rangle, \langle skip \rangle and \langle box \rangle with which \texttt{\textbackslash adl@draw} is called in the following manner.

\begin{verbatim}
\texttt{\textbackslash adl@drawi}\texttt{\textbackslash adl@vrule}\hskip\hbox... horizontal
\texttt{\textbackslash adl@drawii}\texttt{\textbackslash adl@hrule}\vskip\vbox... vertical
\end{verbatim}

The drawing methods in three modes have been explained in §4.2. More specifically, \texttt{\textbackslash adl@drawi} for mode 1, to which \texttt{\textbackslash adl@draw} is \texttt{\let}-equal by default, conceptually performs the following operations.

\begin{verbatim}
\langle rule\rangle\{1/2\} \langle skip\rangle\{g/2\}
\texttt{xlearners}(box)\{\langle skip\rangle\{g/2\} \langle rule\rangle\{1\} \langle skip\rangle\{g/2\}\}
\langle skip\rangle\{0\ plus\ lfil\ minus\ lfil\}
\langle skip\rangle\{g/2\} \langle rule\rangle\{1/2\}
\end{verbatim}

The conceptual operations of \texttt{\textbackslash adl@drawii} for mode 2 are as follows.

\begin{verbatim}
\langle rule\rangle\{1/2\} \langle skip\rangle\{g/2\}
\langle box\rangle\{\langle skip\rangle\{g/2\} \langle rule\rangle\{1\} \langle skip\rangle\{g/2\}\} \langle skip\rangle\{d\ minus\ g\}
\texttt{xlearners}(box)\{\langle skip\rangle\{g/2\} \langle rule\rangle\{1\} \langle skip\rangle\{g/2\}\}
\end{verbatim}
The macro \texttt{\adl@drawiii} for mode 3 is quite similar to \texttt{\adl@drawi} except that \texttt{\xleaders} is replaced by \texttt{\cleaders}. This replacement is done by temporarily \texttt{\let-ing} \texttt{\xleaders} be equal to \texttt{\cleaders}.

\begin{verbatim}
\def\adl@drawiii#1#2#3{{\let\xleaders\cleaders \adl@drawi#1#2#3}}
\let\adl@draw\adl@drawi
\end{verbatim}

\texttt{\ADLdrawingmode\langle m \rangle} defines the drawing mode by \texttt{\let-ing} \texttt{\adl@draw} be equal to \texttt{\adl@drawi} if \texttt{m} = 1, and so on. If \texttt{\langle m \rangle} is neither 1, 2 nor 3, it is assumed as 1.

\begin{verbatim}
\def\ADLdrawingmode#1{\ifcase #1\let\adl@draw\adl@drawi \or \let\adl@draw\adl@drawi \or \let\adl@draw\adl@drawii \or \let\adl@draw\adl@drawiii \else \let\adl@draw\adl@drawi \fi}
\end{verbatim}

\section*{4.13 Shorthand Activation}

\texttt{\adl@Array}, \texttt{\adl@Tabular}, \texttt{\adl@Tabular*} and \texttt{\adl@Longtable} start environments \texttt{array}, \texttt{tabular}, \texttt{tabular*} and \texttt{longtable} respectively, turning \texttt{\ifadl@inactive} false to activate dash-line functions. We will \texttt{\let} macros \texttt{\Array} etc. be equal to them for shorthand activation.

\begin{verbatim}
\def\adl@inactivefalse\array
\def\adl@inactivefalse\tabular
\def\adl@inactivefalse\tabular*
\def\adl@inactivefalse\longtable
\end{verbatim}
Before making `\Array` etc. \texttt{\let}-equal to `\adl@Array` etc., we have to check if these macros having too natural names have already used. This check is done by `\@ifdefinable` that will call `\notdefinable` for the complaint if undefinable. Since we want to complain with our own warning message, `\notdefinable` is temporarily `\def`-ined so that it simply `\def`-ines a macro `\adl@notdefinable` as empty. Therefore, `\adl@notdefinable` will have some definition if one of `\Array`, `\Tabular`, `\Tabular*` and `\Longtable` (if `longtable` is loaded) cannot be defined, while it will stay undefined otherwise.

If `\adl@notdefinable` is `\undefined` indicating that all `\Array` etc. are definable, we `\let` them be equal to `\adl@Array` etc. We also `\let` ending macros `\endArray` etc. be equal to `\endarray` etc. Note that `\Longtable` and `\endLongtable` are defined only when `longtable` is loaded, and `\endLongtable` is `\def`-ined as (not being `\let`-equal to) `\endlongtable` because its definition of our own is not given yet. Otherwise, we complain with a warning message put by `\PackageWarning` if it is defined (i.e. \TeX 2ε) or `\@warning` otherwise (i.e. \TeX-2.09).
\ADLnoshorthanded

If a user wishes to define an environment named \texttt{Array} or \texttt{Tabular} (or \texttt{Longtable} if \texttt{longtable} is in use) by him/herself or by loading other packages after \texttt{arydshln} is loaded, \texttt{\newenvironment} for \texttt{Array} etc. will fail because they have already been undefinable. The macro \texttt{\ADLnoshorthanded} makes them definable again by \texttt{let-ing} them and their ending counterparts be equal to \texttt{\relax}.

\begin{verbatim}
\def\ADLnoshorthanded{%
  \let\Array\relax
  \let\Tabular\relax
  \expandafter\let\csname Tabular*\endcsname\relax
  \let\endArray\relax
  \let\endTabular\relax
  \expandafter\let\csname endTabular*\endcsname\relax
  \ifx\longtable\undefined\else
    \let\Longtable\relax
    \let\endLongtable\relax\fi}
\end{verbatim}

Finally here we define \texttt{active} version of \texttt{\@arrayclassz} named \texttt{\adl@act@arrayclassz} etc. for \texttt{\adl@activate} (see \S4.4). The definitions are simply done by \texttt{let-ing} \texttt{\adl@act@arrayclassz} etc\footnote{Alternatively, we may define \texttt{\adl@act@arrayclassz} in place of \texttt{\@arrayclassz} but the author chose this way to minimize the possiblity of \texttt{enbug}.}.

\begin{verbatim}
\adl@act@arrayclassz \adl@act@tabclassz \adl@act@classz \adl@act@@startpbox \adl@act@@endpbox \adl@act@startpbox \adl@act@endpbox \adl@act@cr \adl@act@argcr \adl@act@endarray \adl@act@hline \adl@act@ihdashline \adl@act@cdline \adl@act@@vlineL \adl@act@@vlineR \adl@act@diamondline \adl@act@arrowline \adl@act@arrowlineL \adl@act@arrowlineR \adl@act@diagline \adl@act@doubleline \adl@act@doublelineL \adl@act@doublelineR
\end{verbatim}
4.14 Compatibility with \texttt{colortab}

The package \texttt{colortab} has a macro:

\begin{verbatim}
\LCC\texttt{(colorspec)}\langle\texttt{rows}\rangle\ECC
\end{verbatim}

to color \texttt{(rows)} referring \texttt{(colorspec)}. The macro \texttt{CC@}, the heart of the coloring function, first makes a box with \texttt{(rows)} using \texttt{@preamble} to measure the height of \texttt{(rows)}, then makes a row putting a heavy rule of the height in each column with a color command for the column specified by \texttt{(colorspec)}, and finally puts \texttt{(rows)} overlaying them on the colored rule. Therefore \texttt{(rows)} is processed twice by \texttt{CC@} to update \texttt{global} registers/structures incorrectly.

Thus we modify \texttt{CC@}, if the package \texttt{colortab} is provided, to save \texttt{global} stuff by \texttt{adl@arraysave} before the height measurement and restore them by \texttt{adl@arrayrestore} after that.

\begin{verbatim}
def\adl@CC@#1#2#3{\
  \ifcolortab
    \noalign{\adl@arraysave\setbox\CT@box=\vbox{#1#3\crcr\egroup}\
    \adl@arrayrestore\CT@dim=\ht\CT@box\global\advance\CT@dim by \dp\CT@box
    \def\CT@next{}\futurelet\next\CT@columncolor#2&\@nil}\
  \CT@next\cr
  \noalign{\vskip-\CT@dim}\
  \fi
\}
\ifx\ColortabLoaded\undefined\else
  \let\CC@\adl@CC@
\fi

\end{verbatim}

4.15 Compatibility with \texttt{longtable}

Making \texttt{arydshln} compatible with \texttt{longtable} is a hard job because a \texttt{longtable} consists of multiple \texttt{chunks} and each chunk is a distinct \texttt{halign}. We could draw vertical lines in each chunks as we do with ordinary \texttt{array/table}. However this straightforward solution should \textit{break} dash-lines at invisible borders of chunks and produce awful results.
Therefore, this implementation draws dash-lines in \texttt{output} routine in which we have all the rows to be put in a page. The hard part is to know which rows are being put in \texttt{output}. This problem is solved by extracting the leading part of \( R_L \) (\texttt{adl@rowsL}) and \( R_R \) (\texttt{adl@rowsR}) by the height/depth of the table fraction to be put and removing the part from \( R_L/R_R \).

### 4.15.1 Initialization

First of all, the following switch and \texttt{dimen} register are declared.

\begin{enumerate}
  \item \texttt{ifadl@LTfirstpage} is tested in \texttt{output} routine to examine if the page being put has the first fraction of a \texttt{longtable}.
  \item \texttt{adl@LTpagetotal} is set to \texttt{pagetotal} just before the first portion of a \texttt{longtable} is added to the main vertical list. Since the \texttt{box255} has items preceding the \texttt{longtable} and its first fraction, we can obtain the height of the first fraction by subtracting \texttt{adl@LTpagetotal} from the height plus depth of \texttt{box255}.
\end{enumerate}

Next, we skip everything if \texttt{longtable} is not in use, or we have undefined-error when we refer to the definitions in it. Note that since \texttt{newif} cannot be in the \texttt{ifx/\fi} construct, the declarations above are excluded.

\begin{verbatim}
\ifx\longtable\undefined\else
  \adl@LT@array
  \LT@array\adl@ discard
\fi
\end{verbatim}

Then we redefine the macro \texttt{LT@array}, which is the heart of \texttt{longtable}, saving its original definition in \texttt{adl@LT@array}. The modified \texttt{LT@array} first calls \texttt{adl@arrayinit} to initialize the global data structures, and sets \texttt{ifadl@LTfirstpage} to true. Then \texttt{adl@dashline}, \texttt{adl@idashline} and \texttt{adl@discard} are made \texttt{let}-equal to the \texttt{longtable} versions \texttt{adl@LTdashline} and \texttt{adl@LTidashline}, and \texttt{relax} (to inhibit expansion) respectively. Then the macro calls \texttt{adl@LTinactivate} if \texttt{adl@inactive} is true, and finally calls its original version \texttt{adl@LT@array}. Note that since \texttt{longtable} cannot be nested;

- \texttt{adl@arraysave} in \texttt{adl@arrayinit} is unnecessary but safe, and thus its invocation timing is not so sensitive; and
- \texttt{activator} is not required.

Also note that the assignment \texttt{adl@ncol} to \texttt{adl@columns} in \texttt{adl@arrayinit} is void and thus we will do it afterward.

\begin{verbatim}
\adl@LTinactivate
\end{verbatim}

The macro \texttt{adl@LTinactivate} first calls \texttt{adl@inactive} to do basic inactivation and then \texttt{let}-s the following control sequences be equal to their counterparts in \texttt{longtable}.

49
The macro \LT@make@row is redefined for additional initialization which must be done after the original \LT@array performs its own initialization. First, \LT@make@row itself is reset to its original version \adl@org@LT@make@row to initialize stuff only once, since \LT@make@row is called repeatedly at each chunk. Next, \adl@ncol is assigned to \adl@columns to give its value calculated in \@mkpream.

Then macros to begin/end p-boxes are made \let-equivalent to our own version because the original \LT@array has done it with longtable's own version. That is, if array is in use \@startpbox is \let-equivalent to our own \adl@LTstartpbox, while \@@startpbox and \@startpbox are \let-equivalent to another macro \adl@@LTstartpbox of our own. On the other hand, \@@endpbox and \@endpbox are commonly \let-equivalent to our own \adl@LTendpbox. Note that these our own macros indirectly invoke \color@begingroup and \color@endgroup, which are added to \LT@startpbox and \LT@endpbox of longtable bundled in latex-tools 2019-01-05, to make the color grouping effective regardless the version of longtable. Also note that we need \adl@LTendmbox to close m-boxes through our own closing macro \adl@endmbox, whose definition is kept in \adl@@endmbox, for longtable-specific operations for footnotes.

Finally, the original version \adl@org@LT@make@row is called.
Table 2: Active and Inactive `longtable` Operations

<table>
<thead>
<tr>
<th>command</th>
<th>active</th>
<th>inactive</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>p b</code> (open) with array</td>
<td><code>\adl@act@classz</code> <code>\adl@LTstartpbox</code> <code>\adl@LTstartpbox</code> <code>\LT@startpbox</code></td>
<td><code>\adl@org@classz</code> <code>\LT@startpbox</code></td>
</tr>
<tr>
<td>without array</td>
<td><code>\adl@act@classz</code> <code>\adl@LTstartpbox</code> <code>\LT@startpbox</code></td>
<td><code>\adl@org@classz</code> <code>\LT@startpbox</code></td>
</tr>
<tr>
<td><code>m</code> (open)</td>
<td><code>\adl@act@classz</code> <code>\adl@LTstartpbox</code> <code>\LT@startpbox</code></td>
<td><code>\adl@startmbox</code> <code>\LT@startpbox</code> <code>\LT@startpbox</code></td>
</tr>
<tr>
<td><code>p b</code> (close)</td>
<td><code>\adl@LTendpbox</code> <code>\LT@endpbox</code></td>
<td><code>\adl@LTendpbox</code> <code>\LT@endpbox</code></td>
</tr>
<tr>
<td><code>m</code> (close)</td>
<td><code>\adl@LTendmbox</code> <code>\LT@endpbox</code></td>
<td><code>\adl@LTendmbox</code> <code>\LT@endpbox</code></td>
</tr>
<tr>
<td><code>\hline</code></td>
<td><code>\adl@act@hline</code> <code>\@gobbletwo</code></td>
<td><code>\adl@LTihdashline</code> <code>\adl@act@hline</code> <code>\@gobbletwo</code></td>
</tr>
<tr>
<td><code>\hdashline</code></td>
<td><code>\adl@LTihdashline</code> <code>\adl@act@hline</code> <code>\@gobbletwo</code></td>
<td><code>\adl@LTihdashline</code> <code>\adl@act@hline</code> <code>\@gobbletwo</code></td>
</tr>
<tr>
<td><code>\endlongtable</code> <code>\LT@make@row</code> <code>\adl@org@LT@make@row</code></td>
<td>modified version <code>\adl@org@endlongtable</code> <code>\adl@org@LT@make@row</code> <code>\adl@org@LT@make@row</code></td>
<td><code>\adl@org@LT@make@row</code> <code>\adl@org@LT@make@row</code></td>
</tr>
<tr>
<td><code>\LT@echunk</code> <code>\adl@org@LT@echunk</code></td>
<td><code>\adl@LTechunk</code> <code>\adl@org@LT@echunk</code></td>
<td><code>\adl@LTechunk</code> <code>\adl@org@LT@echunk</code> <code>\adl@org@LT@echunk</code></td>
</tr>
<tr>
<td><code>\LT@end@hd@ft</code> <code>\LT@kill</code> <code>\LT@output</code></td>
<td><code>\adl@org@LT@end@hd@ft</code> <code>\adl@org@LT@kill</code> <code>\adl@org@LT@output</code></td>
<td><code>\adl@org@LT@end@hd@ft</code> <code>\adl@org@LT@kill</code> <code>\adl@org@LT@output</code></td>
</tr>
</tbody>
</table>

760 \let\@@endpbox\adl@LTendpbox
761 \let\@endpbox\adl@LTendpbox
762 \let\adl@@endmbox\adl@endmbox
763 \let\adl@endmbox\adl@LTendmbox
764 \adl@@endmbox\adl@endmbox
765 \let\@@endpbox\adl@LTendpbox
766 \let\@endpbox\adl@LTendpbox
767 \let\adl@@endmbox\adl@endmbox
768 \let\adl@endmbox\adl@LTendmbox
769 \adl@org@LT@make@row}
770 \NL

The summary of the activation and inactivation specific to `longtable` is shown in Table 2.

### 4.15.2 Ending Chunks

When a chunk is closed with \crr, we have to add the information of the last row to \R \L if the row is not finished by an explicit \\. This is done by \adl@LTlastrow as we did at the first job of \adl@endarray. Two chunk closing macros, \endlongtable and \LT@echunk, are modified to call \adl@LTlastrow before its original job done by \adl@org@endlongtable and \adl@org@LT@echunk respectively. Note that \adl@LTlastrow only has \crr and \noalign and thus another \crr in original \endlongtable and \LT@echunk is no-operation as desired. Also note that \adl@LTlastrow is called twice from \endlongtable, once from \LT@echunk in the original version, but it is safe because the first call makes \adl@height and \adl@depth zero and thus the second become no-operation.
Another chunk ending macro is \LT@end@hd@ft{⟨box⟩} to close a header/footer called by \endfirsthead, \endhead, \endlastfoot and \endfoot with an argument ⟨box⟩ being \LT@firsthead, \LT@head, \LT@lastfoot and \LT@foot respectively. In order to maintain the information of rows $R^L/R = \adl@rowsL/R$ of headers/footers separately from the main one, the modified \LT@end@hd@ft saves them together with \adl@totalheight to weirdly named macros;

\adl@LTth{⟨box⟩}
\adl@rowsL{⟨box⟩}
\adl@rowsR{⟨box⟩}

after closing the last row by \adl@LTlastrow. The \string representation of the macros looks like;

\adl@LTth\LT@firsthead

and so on. The saving operation is done by the macro \adl@LTfsave{⟨box⟩}(info) and is equivalent to;

\global\let\(info)⟨box⟩=⟨info⟩

After the saving, three global variables are reinitialized. Calling \adl@LTlastrow twice, once from the original version through \LT@echunk is safe as described above.

The additional job for yet another chunk closer \LT@kill to kill a template row is a little bit harder. Since the row information might have been added by an explicit \\ preceding
\kill, we have to remove it from the tail of \adl@rowsL/R, and subtract its \ht from \adl@totalheight because \kill-ed row may be in header/footer definition. To do that, modified \LT@kill first ensures the information addition by \adl@LTlastrow, then traverses \adl@rowsL/R adding its non-last elements to \@tempb by the loop of \adl@LTkill, and assigns \@tempb to \adl@rowsL/R globally by \adl@LTkillend when \adl@LTkill finds the tail. The macro \adl@LTkillend also sets the \ht of the last element to \@tempcnta, which is subtracted from \adl@totalheight globally. Finally, the original version \adl@org\LT@kill is called.

\let\adl@org\LT@kill\LT@kill
\def\LT@kill{
\adl@LTlastrow 
\noalign{
\def\@tempb{}
\expandafter\adl@LTkill\adl@rowsL\@nil\adl@rowsL
\def\@tempb{}
\expandafter\adl@LTkill\adl@rowsR\@nil\adl@rowsR
\global\advance\adl@totalheight-\@tempcnta}\
\adl@org\LT@kill}

\def\adl@LTkill#1;#2{
\def\@tempa{#2}
\ifx\@tempa\@nnil
\def\next{\adl@LTkillend#1}\
\else\edef\@tempb{\@tempb#1;}
\def\next{\adl@LTkill#2}\fi
\next}
\def\adl@LTkillend(#1/#2)#3{\global\let#3\@tempb\@tempcnta#2\relax}

4.15.3 Horizontal Lines and p-Boxes

The macro \LT@hline, longtable version of \hline, is redefined to add pseudo row information to RL/R and to check mixed sequence of \hline and \hdashline\footnote{In the original longtable, a sequence of three \hline-s are not recognized. This buggy feature is fixed in this implementation.}. We also define the macro \adl@LTihdashline[⟨dash⟩/⟨gap⟩] and its inactive counterpart \adl@LTinactivehd as the longtable version of \adl@ihdashline and \adl@inactivehd. These two macros, the main part of \hdashline, are redefined to make it possible that \hdashline can be broken into two part by \TeX’s page breaker.

These three macros call a common routine \adl@LTdashline after defining \adl@LTdhline which makes a row of horizontal (dash) line drawn by \multispan and \leaders\hrule or \adl@hcline[⟨dash⟩/(⟨gap⟩)].

Note that we define \adl@LTihdashline to make \adl@hdashline \let-equal to it in longtable environments because its version without longtable performs a part of the job done by \adl@LTdhline as shown soon.

\def\LT@hline{\noalign{\ifnum0='}\fi
\gdef\adl@LTdhline{\multispan{\LT@cols}\unskip
\leaders\hrule\@height\arrayrulewidth\hfill\cr}\
\adl@LTdhline}
\def\adl@LTihdashline#1{\noalign{\ifnum0='}\fi
\gdef\adl@LTihdashline#1{\noalign{\ifnum0='}\fi

%% Compatibility with longtable: horizontal lines and p-boxes
\let\adl@LTdashline\adl@LTihdashline
\let\adl@LTinactivehd\adl@LTihdashline
\let\adl@LTdhline\adl@LTdhline
The macro `\adl@LThdline` called by above three macros first inserts a vertical penalty
10000 to inhibit page break between the horizontal line and preceding row. Then it inserts
`\vskip-\arrayrulewidth` with another break inhibitor if `\ADLnullwidehline` is in effect, or
adds the pseudo row information `connect(\arrayrulewidth)` to $R^{L/R}$ by `\adl@hline`\textsuperscript{21}. Next, it draws a horizontal (dash) line by `\adl@LThdlrow` and checks if the following control sequence is `\hline` or `\hdashline` by `\futurelet` and `\adl@LTxhline`. If `\hline` or `\hdashline` is the next token, `\adl@LTxhline` is called to insert a vertical penalty of $-\@medpenalty$ and a vertical space of `\doublerulesep`. The macro `\adl@LTxhline` also adds `disconnect(\doublerulesep)` to $R^{L/R}$ and makes `\adl@LThdlrow` void. Otherwise, `\adl@LThdline` inserts a vertical penalty of $-\@lowpanalty$ and a vertical space of $-\arrayrulewidth$ and draws the horizontal (dash) line again by `\adl@LThdlrow`. Thus a page can be broken between two overlaid horizontal (dash) lines\textsuperscript{22}. Two pseudo row information, `discard(-\arrayrulewidth)` for the negative vertical space which may be discarded and `connect(\arrayrulewidth)` for the second horizontal line, are also added to $R^{L/R}$.

\textsuperscript{21} Or do noting if inactive and thus it is \let-equal to `\@gobbletwo`.
\textsuperscript{22} If the page is broken, the horizontal line at the beginning of the succeeding page has a width even if `\ADLnullwidehline` is in effect.
Macros for opening/closing p-boxes are fairly simple. The macro \LTstartpbox\{\(w\}\} is \texttt{let}-assigned to \(\@startpbox\) by \LT@make@row to open a p-box of \(w\) wide by \LT@act@startpbox and performs a footnote related operation introduced by \texttt{longtable}, when \texttt{array} is in use. If not, this macro is invoked from \LTstartpbox which is \texttt{let}-equal to \(\@startpbox\) and is to assign the p-box to \LT@box. Since \LT@act@startpbox is for opening p-box with \texttt{array}, it has \texttt{color@begingroup} in it and thus the color grouping is effective regardless the version of \texttt{longtable} or \texttt{array}.

On the other hand, the closing macro \LTendpbox for p (or d)-boxes is \texttt{let}-equal to \(\@endpbox\) and \(\@@endpbox\) for the cases with/without \texttt{array}, and performs the footnote operations after doing our own ones by \LT@act@endpbox, which also has \texttt{color@begingroup} for version-independent color grouping.

As for m-boxes, the opening operation is done by \LTstartmbox in which \LTstartpbox = \LTstartmbox is invoked for the footnote-related operation. On the other hand, the closing operation is done by \LTendmbox, which is made \texttt{let}-equal to \LT@endmbox by our \LT@make@row to perform our own operations by \LT@endmbox in which the original definition of \LT@endmbox is kept by \LT@make@row too. Since \LTstartpbox and \LTendmbox have \texttt{color@begingroup} and \texttt{color@endgroup}, the color-grouping is done regardless of the version of \texttt{longtable} and \texttt{array}.

\begin{verbatim}
\def\LTstartpbox{\setbox\LT@box\vtop\LTstartpbox}
\def\LTstartpbox#1{\LTact@startpbox{#1}\let\@footnotetext\LT@p@ftntext}
\def\LTendpbox{\LTact@endpbox \the\LT@p@ftn \global\LT@p@ftn{}}
\def\LTendmbox{\LT@endmbox \the\LT@p@ftn \global\LT@p@ftn{}}
\end{verbatim}

4.15.4 First Chunk

\LT@start\ The macro \LT@start which puts (first) head and controls the page break of the first page is modified for the following.

- After it inserts a vertical skip \LT@pre, \texttt{endgraf} is performed so that the skip contributes to \texttt{pagetotal}.\footnote{This modification is necessary for the original \texttt{longtable}, or it underestimates the room of the first page and leaves head and foot only.}

- When the \texttt{box2} is \texttt{vsplit} to get first item of the first chunk, \texttt{vbadness} is saved into \texttt{@tempcnta}, set to 10000 to avoid unnecessary \texttt{underfull} message, and restored from \texttt{@tempcnta}.

- The \texttt{dimen} register \LT@pagetotal is set to \texttt{pagetotal} to know the total height of the items preceding \texttt{longtable}. Since the assignment is performed after the inserted \texttt{endgraf} and the intentional page break, it should have real total height.

- The box \LT@firsthead is put by \texttt{copy} rather than \texttt{box} because it is referred to in the \texttt{output} routine.
This macro does not have an inactive counterpart because the modification shown above is desirable (first two) or not-harmful\textsuperscript{25} (last two) to the original version.

\begin{verbatim}
\% Compatibility with longtable: first chunk\%
\newcommand{\LT@start}{
  \let\LT@start\endgraf
  \endgraf \penalty\z@ \vskip\LTpre \endgraf
  \dimen@ \pagetotal
  \advance\dimen@ \ht\ifvoid\LT@firsthead\LT@head\else\LT@firsthead\fi
  \advance\dimen@ \dp\ifvoid\LT@firsthead\LT@head\else\LT@firsthead\fi
  \advance\dimen@ \ht\LT@foot
  \dimen@ii \vfuzz \@tempcnta \vbadness
  \vfuzz \maxdimen \vbadness \@M
  \setbox\tw@ \copy\z@
  \setbox\tw@ \vsplit\tw@ to \ht\@arstrutbox
  \setbox\tw@ \vbox{\unvbox\tw@}
  \vfuzz \dimen@ii \vbadness \@tempcnta
  \advance\dimen@ \ht
  \ifdim\ht\@arstrutbox>\ht\tw@\@arstrutbox\else\tw@\fi
  \ifdim\dp\@arstrutbox>\dp\tw@\@arstrutbox\else\tw@\fi
  \ifdim\dimen@z@ \vfill \break \fi
  \global\adl@LTpagetotal \pagetotal
  \global\@colroom \@colht
  \ifvoid\LT@foot\else
    \advance\dimen@ \ht\LT@foot
    \global\advance\@colroom \ht\LT@foot
    \advance\dimen@ \ht\LT@foot \pagegoal \dimen@
    \ifvoid\LT@foot\else
      \copy\ifvoid\LT@firsthead \LT@head \else \LT@firsthead \fi
      \output{\LT@output}}
\end{verbatim}

4.15.5 Output Routine

The\LT@output\LT@output routine is the heart of the longtable compatible implementation. The macro \LT@output which is set to \output by \LT@start\LT@start is modified from its original (and thus inactive) version \adl@org\LT@output\LT@output as follows.

- Three fractions of the original version to compile the final output image of the table portion into \texttt{box255} or the main vertical list are modified to set the image into \texttt{box255} unconditionally and to call \adl@LTdraw\langle foot\rangle\langle tail\rangle which is the real heart of the compatible implementation. The argument \langle foot\rangle is \LT@foot or \LT@lastfoot

\textsuperscript{25}Logically, at least.
according to the portion of the longtable to be output. The argument \texttt{(tail)} is \texttt{\vss} if the last item is it which is not included in \texttt{\box255} yet, or \texttt{@empty} otherwise. Since \texttt{\adl@LTdraw} builds final output image drawing vertical (dash) lines in \texttt{\box255}, it is put to the main vertical list if the longtable portion is the last one.

- Since the boxes \texttt{\LT@head}, \texttt{\LT@foot} and \texttt{\LT@lastfoot} are referred to in \texttt{\adl@LTdraw}, they are put by \texttt{\copy} rather than \texttt{\box}.

\begin{verbatim}
881 \% Compatibility with longtable: output routine
882 \let\adl@org@LT@output\LT@output
883 \def\LT@output{%
884 \ifnum\outputpenalty <-\@Mi
885 \ifnum\outputpenalty > -\LT@end@pen
886 \LT@err{floats and marginpars not allowed in a longtable}\@ehc
887 \else
888 \setbox\z@\vbox{\unvbox\@cclv}%
889 \ifdim\ht\LT@lastfoot\p@\ht\LT@foot
890 \dimen\pagegoal
891 \advance\dimen-\ht\LT@lastfoot
892 \ifdim\dimen<\ht\z@
893 \setbox\@cclv\vbox{\unvbox\z@\copy\LT@foot}%
894 \adl@LTdraw\LT@foot\vss
895 \@makecol
896 \@outputpage
897 \setbox\z@\vbox{\copy\LT@head}%
898 \fi
899 \else
900 \global@colroom@colht
901 \global@vsize@colht
902 \setbox\@cclv\vbox{\unvbox\z@}
903 \ifvoid\LT@lastfoot\LT@foot\else\LT@lastfoot\fi%
904 \adl@LTdraw\LT@lastfoot@empty \box\@cclv
905 \fi
906 \else
907 \setbox\@cclv\vbox{\unvbox\@cclv\copy\LT@foot}%
908 \adl@LTdraw\LT@foot\vss
909 \@makecol
910 \@outputpage
911 \global@vsize@colroom
912 \copy\LT@head
913 \fi
914 \fi
915 \fi
916 \adl@LTdraw The macro \texttt{\adl@LTdraw(foot)(tail)} draws vertical (dash) lines onto the image in \texttt{\box255}.
\adl@LTinit First it measures the total height \(H\) (\texttt{\adl@totalheight}) of longtable rows in \texttt{\box255} and the total height \(H_b\) (\texttt{@tempdim}) of its body which consists of the rows without the header and footer, as follows where \(H_{255}\), \(H_b\) and \(H_f\) are the height plus depth of \texttt{\box255}.
\adl@LTheadL
\adl@LTheadR
\adl@LTfootL
\adl@LTfootR
\end{verbatim}
and the effective header and footer of the page respectively.

\[
\begin{align*}
T &= \begin{cases} 
\text{\texttt{\adl@LTpagetotal}} & \text{if \texttt{\ifadl@LTfirstpage}} \\
0 & \text{otherwise}
\end{cases} \\
t &= \begin{cases} 
\text{\texttt{\topskip glue}} & \text{if \texttt{\longtable} is the first item of the page} \\
0 & \text{otherwise}
\end{cases} \\
H &= H_{255} - t - T \\
H_b &= H - H_h - H_t
\end{align*}
\]

The hard part is to measure \( t \) because it is not \texttt{\topskip} but that minus the first box of \texttt{\box255}. Thus we do not measure \( t \) but remove it from the box by the following tricky way. First we copy \texttt{\box255} items into \texttt{\box0} adding a \texttt{\hrule} of 1 sp high as its first item. Then \texttt{\box0} is \texttt{\vsplit} to 1 sp setting \texttt{\splittopskip} to 0. Since the \texttt{\topskip} glue is the first item of \texttt{\box255} and the \texttt{\vsplit} discards it at the breakpoint, \texttt{\box0} must have all the items in \texttt{\box255} lead by 0 (\texttt{\splittopskip} glue) rather than \texttt{\topskip} glue. Thus the height of \texttt{\box0} is \( H_{255} - t \).

Subtraction of \( H_h \) and \( H_t \) is done by the macro \texttt{\adl@LTinit{⟨hf⟩}{⟨box⟩}}, where \( ⟨hf⟩ \) is head or foot and \( ⟨box⟩ \) is one of \texttt{\LT@firsthead}, \texttt{\LT@head} and \( ⟨foot⟩ \) (\texttt{\LT@lastfoot} or \texttt{\LT@foot}). This macro also copies the contents of weirdly named structure such as \texttt{\adl@rowsL\LT@head} into \texttt{\adl@LTheadL} and so on\(^{26}\) if \( ⟨box⟩ \) is not void. Otherwise, \texttt{\adl@LTheadL} etc. is kept to their initial value, \texttt{\@empty}.

Next, we make rows for vertical lines by \texttt{\adl@makevL/R} after extracting the leading part of \( R_{L/R} \) corresponding to the body by the macro \texttt{\adl@LTsplit{⟨R_{L/R}⟩}{⟨R_{L/R}⟩}{⟨R_{L/R}⟩}}, where \( R_{h/R} \) and \( R_{f/R} \) are \texttt{\adl@LTheadL} and so on. Since the macro defines \texttt{\adl@rows} given to \texttt{\adl@makevL/R} to the sequence of \( R_{h/R} \), the extracted part of \( R_{L/R} \) and \( R_{f/R} \), the rows for vertical lines for all the rows including header and footer are build in \texttt{\adl@vlrowL} and \texttt{\adl@vlrowR} as in the ordinary case without \texttt{\longtable}.

Then the rows are put into \texttt{\box0} by calling \texttt{\LT@bchunk} with \texttt{\adl@drawvL} (line drawing) and \texttt{\LT@save@row} (column widths adjustment), saving/restoring counters \texttt{\LT@rows} and \texttt{\LT@chunks} which \texttt{\LT@bchunk} globally updates. Since we refer to potentially immature \texttt{\LT@save@row} here, some weird looking vertical lines could be drawn but the result after convergence should be correct. Finally, the contents of \texttt{\box255} followed by the vertical lines in \texttt{\box0} are put back into \texttt{\box255} keeping its original depth and adding \( ⟨\text{tail}⟩ \) (\texttt{\vss} or nothing) to its end.

\(^{26}\)Copying by \texttt{\edef} can be replaced by \texttt{\let} with many \texttt{\expandafter} but it is not comprehensible.
The macro \texttt{\textbackslash adl@LTsplit} moves leading elements in \( R^{L/R} \) into \( R' \) \texttt{\textbackslash adl@rows} until total heights of the elements summed in \( h \) \texttt{\textbackslash@tempdimb} reaches to \( H_b \). Before moving, however, we have to remove \texttt{\textbackslash advance\textbackslash@tempdimadp@cclv} although \( h \) must become \( H_b \) exactly in usual case, we stop the loop when \( h \geq H_b \) to avoid accidental overrun in unusual cases.
discardable item(s)\footnote{Must be only one but the implementation allows two or more.} from the top of $R^{L/R}$. Since an element for a discardable item is *disconnect* (\relax) or *discard* (\adl@discard), we check the first part of the element by \ifx-comparison with \adl@LTrowrelax and \adl@LTrowdiscard whose bodies are \relax and \adl@discard if the longtable portion does not have a header ($R^{L/R}_h$ is $\emptyset$). Otherwise, the discardable item was not discarded because the first item of the page is not it but the header.

Note that since moving from $R^{L/R}$ to $R'$ is done by \edef and \adl@discard is \defined as \adl@connect in \adl@LTdraw, non-discardable *discard* transforms into *connect* in $R'$. Also note that since the remaining part of $R^{L/R}$ is \defined as the body of $\@tempb$ which is globally \let-assigned to $R^{L/R}$ again, \adl@discard survives in the new $R^{L/R}$.

\section*{4.16 Compatibility with \texttt{colortbl}}

The implementation to make \texttt{arydshln} compatible with \texttt{colortbl} consists of the following three (almost independent) issues.

Cell coloring is the easiest part because it does not affect dash line drawing. Another reason of the easiness is that \texttt{colortbl} packs each cell in a box to measure its height for
painting in the modified version of \@classz. Thus we do not need to code \@classz for both of \colortbl and \arydshln, but may sneak our own height/depth measurement into \@classz of \colortbl. Almost everything we have to pay attention to is the compatibility of the initialization and finalization of \colortbl and \arydshln.

**Horizontal line coloring** is relatively easy because it is almost enough to insert coloring macro \CT@arc@ before the line drawing. A little bit complicated part is the gap coloring which is done by drawing a solid line of gap color before dash line is drawn.

**Vertical line coloring** is the hardest part but almost everything is done in previous sections to attach dash/gap color to each vertical line segment $e_i^j$ in the list $C^L_i$ and $C^R_i$ of the $i$-th row information $r_i$. What we do here is to fix the bugs of \arrayrulecolor and \doublerulesepcolor in \colortbl implementation and to add \dashgapcolor. If you put \arrayrulecolor in \>{...} construct to specify the color of the vertical lines following the construct as the manual of \colortbl says, you will have an error message “Misplaced \noalign” because the macro is expanded with \noalign in a column body. Even if you somehow remove \noalign to avoid the error, you will have a mysterious line coloring as follows:

- If you have \arrayrulecolor before the \array/\tabular starts, \arrayrulecolor in the preamble has no effect to vertical lines but decides the color of horizontal lines except for those at the top of the environment. Additional \arrayrulecolor at the beginning of a row has no effect to vertical lines (as expected) but decides horizontal lines following it (also as expected). The effect of \doublerulesepcolor is same as \arrayrulecolor.

- Otherwise, i.e. without \arrayrulecolor outside the environment, \arrayrulecolor in the preamble decides the color of vertical and horizontal lines except for verticals preceding columns in the first row and horizontal at the top of the environment. Additional \arrayrulecolor at the beginning of a row decides all the vertical and horizontal lines following it. On the other hand, \doublerulesepcolor acts as if \doublerulesepcolor\{white\} is done outside the environment.

The reason of the mysterious behavior is as follows. An \arrayrulecolor, which globally \def-ines a macro \CT@arc@ with a body containing \color, in the preamble is not expanded nor evaluated in the preamble construction phase but done when the first (and succeeding) row is build. On the other hand, \CT@arc@ attached to vertical line drawing is expanded in the preamble construction phase. Thus if \CT@arc@ has been defined before the environment starts, vertical lines are colored following the outside definition. Otherwise, since \CT@arc@ is \let-equal to \relax, it remains unchanged in the preamble construction phase and expanded when each row is build referring to its definition that \arrayrulecolor modifies in the row building phase. Since the macro \CT@drsc@ defined by \doublerulesepcolor is examined if it is \relax or not in the preamble construction phase, \doublerulesepcolor in the preamble has no effect regardless the existence of the outside definition.

Thus we have to expand and evaluate \arrayrulecolor and \doublerulecolor in the preamble construction phase to define \CT@arc@ and \CT@drsc@. We also have
to initialize $\texttt{CT@arc}$ as an expandable but non-operative token (e.g. a macro with a body of $\texttt{relax}$ as we do) to make it is expanded in the preamble construction phase rather than the row building.

### 4.16.1 Initialization, Cell Coloring and Finalization

First of all, we initialize the macro $\texttt{CT@arc}$, which will be \def-ined as $\texttt{color}$ to specify the color of solid lines and dash segments by $\texttt{arrayrulecolor}$, with a body of $\texttt{relax}$ because it will be referred to by the vertical line drawing process even if $\texttt{colortbl}$ is not in use. We also initialize the macro $\texttt{adl@dashgapcolor}$ for the color of gaps of dash lines similarly. Note that these macros are not $\texttt{let}$-equal to $\texttt{relax}$ but have bodies of $\texttt{relax}$ so that they are replaced with $\texttt{relax}$ in the preamble construction phase rather than surviving with their own name.

Next we examine if $\texttt{colortbl}$ is in use by $\texttt{@ifpackageloaded}$, and skip everything if not, or we have some errors especially when $\texttt{array}$ is not in use.

Then we redefine $\texttt{adl@inactivate}$ and $\texttt{adl@activate}$ referring their original version $\texttt{adl@org@inactivate}$ and $\texttt{adl@org@activate}$ so that they make $\texttt{CT@setup}$ $\texttt{let}$-equal to its original version $\texttt{adl@CT@setup}$ if $\texttt{ADLinactivate}$ is in effect, or to our own version $\texttt{adl@act@CT@setup}$ which will be defined soon. New $\texttt{adl@activate}$ also $\texttt{inactivates}$ $\texttt{@startpbox}$ and $\texttt{@endpbox}$ because our own ones for column height/depth measurement is inappropriate with $\texttt{colortbl}$ as explained soon.

Cell coloring is done by $\texttt{@classz}$ preamble of $\texttt{colortbl}$ in which a column is packed in $\texttt{box0}$. On the other hand, our own $\texttt{@classz}$ one with $\texttt{array}$ packs the column in $\texttt{adl@box}$ so that we measure its height and depth. Thus we have choices; to insert height/depth measurement into $\texttt{colortbl}$'s version; or to insert coloring into our own version. Since the code of height/depth measurement is much simpler than the coloring, we choose the first way. Thus the macro $\texttt{adl@act@CT@setup}$, which is $\texttt{let}$-equal to $\texttt{CT@setup}$ and is invoked from $\texttt{@classz}$ preamble after the column is packed into $\texttt{box0}$, measures the height and depth of $\texttt{box0}$ and sets $\texttt{adl@height}$ and/or $\texttt{adl@depth}$ to them if they break the records as $\texttt{adl@colhtdp}$ does with $\texttt{adl@box}$, after it invokes its original version $\texttt{adl@CT@setup}$. Note that we compare $\texttt{adl@height}$ with the height of $\texttt{box0}$ plus $\texttt{minrowclearance}$
because it is the real height. Also note that we could insert the measurement code into the modified version of colortbl's \@classz placing it just before the \box0 is put where \ht0 plus \minrowclearance is calculated, but did not because the author wished to make it clear that \@classz is modified only for the bug fix of \arrayrulecolor and \doublerulesepcolor (and to introduce \dashgapcolor).

\let\adl@CT@setup\CT@setup
\def\CT@setup{
\@tempdima\ht\z@ \advance\@tempdima\minrowclearance
\ifdim\adl@height<\@tempdima \global\adl@height\@tempdima \fi
\ifdim\adl@depth<\dp\z@ \global\adl@depth\dp\z@\fi}
\let\adl@act@CT@setup\CT@setup
\adl@activatepbox

Another job for cell coloring is to make \CT@x@color \(x \in \{\text{cell, column, do}\}\) \let\=-equal to \relax before the body of \multicolumn is put so that the \columncolor in the environment preamble does not affect the \span-ned column. Note that resetting \CT@cell@color will be unnecessary (but safe) because it is always reset after its invocation. Also note that resetting \CT@row@color in colortbl's \multicolumn is a buggy feature because it should be effective, and thus we remove it. Although we have our own \multicolumn for dash lines, we keep it unchanged. Instead we redefine \adl@activatepbox, which is usually \relax with \array, to do the color resetting to minimize recoding.

\def\adl@activatepbox\{\let\CT@cell@color\relax
\let\CT@column@color\relax
\let\CT@do@color\relax
\adl@CT@start
\CT@start
\adl@dashgapcolor@save
\adl@CT@end
\CT@end
\endarray
\endArray
Yet another job is the save/restore of color information at the beginning and end of the environment. Since this is done by \CT@start and \CT@end, we modify them to save/restore \adl@dashgapcolor to/from \adl@dashgapcolor@save referring their original version \adl@CT@start and \adl@CT@end. We also modify our own \endarray and its shorthand active version \endArray so that \CT@end is invoked at the end of environment together with \arrayright if it is defined. Note that we may not modify \endtabular because it refers \endarray. Also note that \CT@start is invoked from \@tabarray which we keep unchanged.

\def\adl@CT@start\CT@start
\def\CT@start\{\adl@CT@start \let\adl@dashgapcolor@save\adl@dashgapcolor
\def\adl@CT@end\CT@end
\endarray
\endArray
\def\endarray\{\adl@endarray \egroup \adl@arrayrestore \CT@end \egroup
\csname @arrayright\endcsname
\ifx\adl@notdefinable\undefined \let\endArray\endarray \fi

4.16.2 Horizontal Line Coloring

To color \hline and inactivated \hdashline, we modify our own \hline and \adl@inactivehd1 inserting the line coloring macro \CT@arc@ before drawing by \hrule and
pushing the coloring/drawing into a group. We also modify \adl@ixhline to draw a colored horizontal rule of \doublerulesep wide with the color defined in \CT@drsc@ if it is not \relax, rather than to insert a vertical skip. Note that the \cline coloring is done by colortbl's \cline renamed as \adl@org@cline and invoked from our own one.

\begin{verbatim}
def\hline{\noalign{\ifnum0='}\fi
  \ifadl@zwhrule \vskip-\arrayrulewidth
  \else \adl@hline\adl@connect\arrayrulewidth \fi
  {\CT@arc@ \hrule\@height\arrayrulewidth}\
  \global\adl@finaldepth\z@
  \futurelet\@tempa\adl@xhline}
def\adl@inactivehhdashline[#1/#2]{\ifadl@zwhrule \vskip-\arrayrulewidth \fi
  {\CT@arc@ \hrule\@height\arrayrulewidth}\
  \futurelet\@tempa\adl@xhline}
def\adl@ihdashline{\ifx\CT@drsc@\relax \vskip \else
  \CT@drsc@\hrule\@height \fi \doublerulesep}\
  \adl@hline\relax\doublerulesep}
def\adl@act@ihdashline
def\adl@cdline{\ifadl@zwhrule \vskip-\arrayrulewidth \fi
  \adl@hclinesetup{#1}{#2}\
  \adl@hcline{-\arrayrulewidth}}
def\adl@act@cdline
def\adl@hclinesetup#1#2{\global\adl@cla#1\relax
  \global\advance\adl@cla\m@ne
  \ifnum\adl@cla>\z@ \global\let\@gtempa\adl@cdlinea
  \else \global\let\@gtempa\adl@cdlineb\fi
  \global\adl@clb#2\relax
  \ifnum0='\fi}
def\adl@cdlinea{\multispan\adl@cla &
  \multispan\adl@clb \unskip}
def\adl@cdlineb{\multispan\adl@clb \unskip}
\end{verbatim}

To draw a horizontal dash line with colored dashes and also colored gaps, we drastically modified \adl@ihdashline for \hdashline and \adl@cdline for \cdashline. First, they invoke \adl@hclinesetup that makes the prefix of a \multispan-ned row from the first to last columns for \hdashline or given columns for \cdashline. Then the line is drawn by the modified version of \adl@hcline. We have to declare these macros are active ones again.

The macro \adl@hclinesetup\(f\langle t\rangle\) makes the prefix of a \multispan-ned row from the column \(f\) to \(t\) and \global-ly defines it as \gttempa. This is done by a code very similar to original \adl@hcline (and thus \LaTeX-2.09's \cline) but the invocation of \adl@hcline is removed from \adl@cdlinea and \adl@cdlineb, one of which is \gttempa.

\begin{verbatim}
def\adl@hclinesetup#1#2{\global\adl@cla#1\relax
  \global\advance\adl@cla\m@ne
  \ifnum\adl@cla>\z@ \global\let\@gtempa\adl@cdlinea
  \else \global\let\@gtempa\adl@cdlineb\fi
  \global\adl@clb#2\relax
  \ifnum0='\fi}
def\adl@cdlinea{\multispan\adl@cla &
  \multispan\adl@clb \unskip}
def\adl@cdlineb{\multispan\adl@clb \unskip}
\end{verbatim}

64
The modified version of \adl@hcline draws a colored horizontal dash line of dash size \(d\) and gap size \(g\) and insert vertical skip of \(w\). First it \span-s columns by \@gtempa and checks if the body of \adl@dashgapcolor is something other than \relax. If so, i.e., it has \color, \adl@paintdashgap is invoked to draw a horizontal rule of \color by \leaders as the background of the dash line, to insert \nobreak (for \longtable) and a negative space for canceling the width of the rule, and to \span the columns again. Then \adl@hcline draws the colored dash line, over the background if the gaps are colored, by inserting \CT@arc@ before the invocation of \adl@draw.

\begin{verbatim}
1056 \def\adl@hcline#1[#2/#3]{\@gtempa
1057 \ifx\adl@dashgapcolor\adl@nocolor \else \adl@paintdashgap \fi
1058 \@tempdima#2\relax \@tempdimb#3\relax
1059 \CT@arc@ \adl@draw\adl@vrule\hskip\hbox\cr
1060 \noalign{\global\adl@finaldepth\z@ \ifdim#1=\z@\else
1061 \ifadl@zwhrule\else \vskip#1\fi\fi\}
1062 \def\adl@paintdashgap{{\adl@dashgapcolor
1063 \leaders\hrule\@height\arrayrulewidth\hfill}\cr
1064 \noalign{\penalty\@M \vskip-\arrayrulewidth}\@gtempa}
1065
\end{verbatim}

4.16.3 Vertical Line Coloring

A bug of colorbl's \arrayrulecolor and \doublerulesepcolor is that they are defined like:

\begin{verbatim}
1066 \def\arrayrulecolor{\adl@defcolor\CT@arc@}
1067 \def\doublerulesepcolor{\adl@defcolor\CT@drsc@}
1068 \def\dashgapcolor{\adl@defcolor\adl@dashgapcolor}
1069 \def\adl@defcolor{\adl@idefcolor}
1070 \def\adl@idefcolor#1#2#{\adl@noalign{\gdef#1{\color#2{#3}}}}
1071 \let\adl@noalign\relax
1072 \def\nodashgapcolor{\adl@noalign{\gdef\adl@dashgapcolor{\relax}}}
\end{verbatim}

This aims to do \noalign{\gdef\arrayrulecolor{\gdef...}} in \array/tabular and do \{\gdef...\} outside but has two problems: First, if they are in >\{\> construct, they are expanded with \noalign inappropriately when the argument of > is expanded. Second, they may appear at a place where \baselineskip is 0 but is outside of \array/tabular and will cause the misplaced \noalign error. To solve the second problem, we introduced \adl@noalign which is set to \noalign in the environment by our own \@array, and \relax outside. We also introduced \adl@idefcolor\(\langle\mathrm{cs}\rangle\langle\mathrm{opt}\rangle\) for the common job to define \(\langle\mathrm{cs}\rangle\) as \color with \langle\mathrm{opt}\rangle, in \noalign if necessary, by \adl@idefcolor. Thus \arrayrulecolor and \doublerulesepcolor are modified to define \CT@arc@ and \CT@drsc@ using \adl@idefcolor, and our own \dashgapcolor is defined similarly to define \adl@dashgapcolor. Another macro \nodashgapcolor to nullify \dashgapcolor is also defined with \adl@noalign to reset \adl@dashgapcolor to \relax.
The tougher bug of \texttt{colortbl} is the expansion timing of \texttt{arrayrulecolor} and \texttt{doublerulesepcolor} in a \texttt{-}-argument. We have to modify \texttt{@classz} to extract them from \texttt{\toks@tempcnta} as its original version does for \texttt{\columncolor}. Thus we inserted the invocation of \texttt{\adl@extract@arc} for \texttt{arrayrulecolor}, \texttt{\adl@extract@drsc} for \texttt{doublerulesepcolor}, and \texttt{\adl@extract@dgc} for \texttt{dashgapcolor} just after the invocation of \texttt{\CT@extract}. Note that the other part of \texttt{@classz} is not modified logically, but done for author’s preference of indentation. Also note that both \texttt{\adl@act@classz} and \texttt{\adl@org@classz} are \texttt{\let}-equal to the modified \texttt{@classz} because we have to be bug free even if \texttt{\ADLinactive} is in effect.

\begin{verbatim}
\def\@classz{\@classx \@tempcnta\count@ \prepnext@tok
\expandafter\CT@extract\the\toks\@tempcnta\columncolor!\@nil
\expandafter\adl@extract@arc\the\toks\@tempcnta\arrayrulecolor!\@nil
\expandafter\adl@extract@drsc\the\toks\@tempcnta\doublerulesepcolor!\@nil
\expandafter\adl@extract@dgc\the\toks\@tempcnta\dashgapcolor!\@nil
\@addtopreamble{%
\setbox\z@\hbox\bgroup\bgroup
\ifcase \@chnum
\hskip\stretch{.5}\kern\z@
\dollarbegin
\insert\column
\dollarend\hskip\stretch{.5}%
\or \dollarbegin \insert\column \dollarend \hfill
\or \hfill \kern\z@ \dollarbegin \insert\column \dollarend
\or \$\vcenter
\insert\column \@startpbox{\@nextchar}\insert\column \@endpbox \$
\or \vtop \@startpbox{\@nextchar}\insert\column \@endpbox
\or \vbox \@startpbox{\@nextchar}\insert\column \@endpbox
\fi
\egroup\egroup
\begingroup
\CT@setup
\CT@columncolor
\CT@rowcolor
\CT@cellcolor
\CT@do@color
\endgroup
\@tempdima\ht\z@
\advance\@tempdima\minrowclearance
\vrule\@height\@tempdima\@width\z@
\unhbox\z@)%
\prepnext@tok}
\let\adl@act@classz\@classz
\let\adl@org@classz\@classz
\adl@def@extract
\adl@extract@arc
\adl@extract@arc@b
\CT@arc@
\adl@extract@drsc
\adl@extract@drsc@b
\CT@drsc@
\adl@extract@dgc
\adl@extract@dgc@b
\adl@dashgapcolor
\end{verbatim}

The definitions of \texttt{\adl@extract@x \(x \in \{\text{arc, drsc, dgc}\)} are quite similar to each other.
For example \adl@extract@arc is defined as follows.

\begin{verbatim}
def\adl@extract@arc#1\arrayrulecolor#2#3\@nil{% 
  \if!#2\toks\@tempcnta(#1)\let\@tempa\relax%
  \else\if#2%
    \def\@tempa{\adl@extract@arc@b{#1}#3\@nil}%
  \else \def\CT@arc@{\color{#2}}%
    \def\@tempa{\adl@extract@arc#1#3\@nil}%
  \fi\fi \@tempa}
def\adl@extract@arc@b#1#2\]#3{%
  \def\CT@arc@{\color[#2]{#3}}%
  \adl@extract@arc#1}
\end{verbatim}

This code extracts all the occurrences of \arrayrulecolor\{(m)\}\{(c)\} from the token register and \def-ines \CT@arc@ as \color\{(m)\}\{(c)\}. Note that \CT@extract does a similar job for \columncolor but it mistakenly ignores the possibility that the token register has two or more \columncolor. Anyway, if we copy the code above and replace `@arc' with `@drsc', \arrayrulecolor with \doublerulesepcolor, and \CT@arc@ with \CT@drsc@, we will have \adl@extract@drsc@ for \doublerulesepcolor. The code for \adl@extract@dgc@ will be also obtained similarly. However, having three relatives for a almost common job is too awful. Thus we introduce;

\begin{verbatim}
def\adl@def@extract#1#2#3{%
  \expandafter\def\csname adl@extract@#1\endcsname##1#2##2##3\@nil{% 
    \if!##2\toks\@tempcnta(#1)\let\@tempa\relax%
    \else\if[##2% 
      \def\@tempa{\@nameuse{adl@extract@#1@b}{##1}##3\@nil}%
    \else \def##3{\color{##2}}%
      \def\@tempa{\@nameuse{adl@extract@#1}##1##3\@nil}%
    \fi\fi \@tempa}
def\adl@extract@#1@b\endcsname##1##2\]##3{%
  \def##3{\color[##2]{##3}}%
  \@nameuse{adl@extract@#1}##1}}
de\adl@def@extract{arc}\arrayrulecolor\CT@arc@
de\adl@def@extract{drsc}\doublerulesepcolor\CT@drsc@
de\adl@def@extract{dgc}\dashgapcolor\adl@dashgapcolor
\end{verbatim}

to define the macros \adl@extract@key@ and \adl@extract@key@b for the user interface macro \(mac@) in which a color macro \(mac@) is defined with \color. For example, we will obtain \adl@extract@arc@b shown above by;

\begin{verbatim}
de\adl@def@extract{arc}\arrayrulecolor\CT@arc@
de\adl@def@extract{drsc}\doublerulesepcolor\CT@drsc@
de\adl@def@extract{dgc}\dashgapcolor\adl@dashgapcolor
\end{verbatim}

Note that \color is made \relax in the preamble construction phase by colorbl@’s \@mkpream and regain its proper meaning after the phase.

1111 \def\adl@def@extract#1#2#3{%
1112 \expandafter\def\csname adl@extract@#1\endcsname##1#2##2##3\@nil{%
1113 \if!##2\toks\@tempcnta(#1)\let\@tempa\relax%
1114 \else\if[##2% 
1115 \def\@tempa{\@nameuse{adl@extract@#1@b}{##1}##3\@nil}%
1116 \else \def##3{\color{##2}}%
1117 \def\@tempa{\@nameuse{adl@extract@#1}##1##3\@nil}%
1118 \fi\fi \@tempa}
1119 \expandafter\def\csname adl@extract@#1@b\endcsname##1##2\]##3{%
1120 \def##3{\color[##2]{##3}}%
1121 \@nameuse{adl@extract@#1@b}{##1}}
1122 \adl@def@extract{arc}\arrayrulecolor\CT@arc@
1123 \adl@def@extract{drsc}\doublerulesepcolor\CT@drsc@
1124 \adl@def@extract{dgc}\dashgapcolor\adl@dashgapcolor
1125

29Fixing this bug is not our business.
4.16.4 Compatibility with \texttt{longtable}

Yet another compatibility issue is to cope with both \texttt{longtable} and \texttt{colortbl}. We redefine \texttt{LT@hline} and \texttt{LT@inactivehd1} in order to put \texttt{\CT@arc@} before line drawing and to push them in a group. Modified \texttt{\adl@LTidashline} first invokes \texttt{\adl@hclinesetup} and open \texttt{\noalign} because it is closed by \texttt{\adl@hclinesetup}. The contents of \texttt{\adl@LTidashline} for \texttt{\adl@LTidashline} is simply \texttt{\adl@hcline} because it does \texttt{\multispan} now. The macro \texttt{\adl@LTixhline} is modified to paint the \texttt{\doublerulesep} gap by \texttt{\leaders\hrule} with color of \texttt{\CT@drsc@} if it is not \texttt{\relax}.

1126 \texttt{\ifx\longtable\undefined\else} \\
1127 \texttt{\def\LT@hline{\noalign{\ifnum0='}\fi} \\
1128 \gdef\adl@LThdlrow{\multispan{\LT@cols}\unskip{\CT@arc@} \\
1129 \leaders\hrule@height\arrayrulewidth\hfill}\cr}% \\
1130 \adl@LThdline} \\
1131 \texttt{\def\adl@LTidashline[#1/#2]{\adl@hclinesetup\@ne\adl@columns} \\
1132 \noalign{\ifnum0='}\fi} \\
1133 \gdef\adl@LThdlrow{\adl@hcline\z@[#1/#2]}% \\
1134 \adl@LThdline} \\
1135 \texttt{\def\adl@LTinactivehd1[#1/#2]{%} \\
1136 \gdef\adl@LThdlrow{\multispan{\LT@cols}\unskip{\CT@arc@} \\
1137 \leaders\hrule@height\arrayrulewidth\hfill\cr}% \\
1138 \adl@LThdline} \\
1139 \texttt{\def\adl@LTixhline{\%} \\
1140 \texttt{\ifx\CT@drsc@\relax \gdef\adl@LThdlrow{\noalign{ \\
1141 \penalty-\@medpenalty \vskip\doublerulesep}}} \\
1142 \texttt{\else \gdef\adl@LThdlrow{\noalign{\penalty0\%} \\
1143 \multispan{\LT@cols}\unskip{\CT@drsc@} \\
1144 \leaders\hrule@height\doublerulesep\hfill\cr}\fi} \\
1145 \texttt{\ifnum0='\fi}\adl@LThdlrow \noalign{\ifnum0='}\fi} \\
1146 \texttt{\adl@hline\relax\doublerulesep \global\let\adl@LThdlrow\@empty} \\
1147 \texttt{\fi} \\
1148 \texttt{\fi}

Acknowledgments

The author thanks to Monty Hayes who gave the author the opportunity to make this style, and Weimin Zhang and Takahiro Kubota who pointed out bugs in early versions. He also thanks to the following people; Sebastian Rahtz and Graham Williams who kindly invited the style to \TeX\ CTAN and online catalogue compiled by Graham; Peter Ehrbar who showed the style was incompatible with \texttt{array} and kindly accepted the offer to be an alpha-user of v1.4 alone; Zsuzsanna Nagy who reported another incompatibility problem with \texttt{colortab}; Ralf Heydenreich who reported the bug causing that glues in a column have no effect; Yaxin Liu who reported the incompatibility bug of \texttt{array} and \texttt{\ADLinactivate}; Craig Leech who reported the incompatibility problem with \texttt{longtable}, which was also reported by Uwe Jehnlich, Torge Thielemann and Florian Weig, and had waited for two years and a half (!) for the solution; Klaus Dalinghaus who reported yet another incompatibility with \texttt{colortbl}; Morten Hogholm who reported the bug of \texttt{m-type} columns of \texttt{array} which...
had not manifested in five (!!) years since the author released the first \texttt{array}-compatible version; Maïeul Rouquette who reported another bug of \texttt{m} -type columns of \texttt{longtable} with \texttt{array} which had peacefully hidden in the package for eleven years and a half (!!!) since the author made the bug fix shown above carelessly, yet another bug related to \texttt{longtable}, and most surprisingly a problem on intersections of horizontal and vertical (dash-) lines which has hidden for 23 years (!!!!) since the very first version of the package; and Hironobu Yamashita who pointed out bugs hidden for 19 years (!!!!!) by which \texttt{delarray} did not work, and compatibility problems with \texttt{array} v2.4i and \texttt{longtable} in \texttt{latex-tools} 2019-01-05.

The base implementation of \texttt{array} and \texttt{tabular} environments, part of which the author gives new definitions referring original ones, are written by Leslie Lamport as a part of \LaTeX\ 2e (1997/12/01) to which Johannes Braams and other authors also contributed. The author also refers \texttt{array} package (v2.4j) written by Frank Mittelbach and David Carlisle; \texttt{colortab} package (v0.9) written by Timothy van Zandt; and \texttt{longtable} (v4.11) and \texttt{colortbl} (v0.1j) packages written by David Carlisle; to make the style compatible with those packages.
Index

Italicized number refers to the page where the specification and usage of corresponding entry are described, while underlined is for the implementation of the entry. To find a control sequence, remove prefixes \@, \adl@ and \ifadl@ from its name if it has one of them.

Symbols

| \ADLdrawingmode | 6, 45 |
| \ADLinactivate   | 6, 15 |
| \ADLnohanded     | 7, 47 |
| \adl@LTth\LT@firsthead | 52, 58 |
| \adl@LTth\LT@foot   | 52, 58 |
| \adl@LTth\LT@head   | 52, 58 |
| \adl@LTth\LT@lastfoot | 52, 58 |
| \adl@rows\LT@firsthead | 52, 58 |
| \adl@rows\LT@foot   | 52, 58 |
| \adl@rows\LT@head   | 52, 58 |
| \adl@rows\LT@lastfoot | 52, 58 |

A

| \AC | 7 |
| \adl@act@endbox | 21, 47 |
| \adl@act@startbox | 21, 47 |
| \adl@act@vlineL | 21, 47 |
| \adl@act@vlineR | 21, 47 |
| \adl@act@argcr | 21, 47 |
| \adl@act@arrayclassz | 21, 47 |
| \adl@act@classz | 21, 47 |
| \adl@act@cdline | 21, 47 |
| \adl@act@clearpage | 21, 47 |
| \adl@act@clapline | 21, 47 |
| \adl@act@closebox | 21, 47 |
| \adl@act@cr | 21, 47 |
| \adl@act@CT@setup | 21, 47 |
| \adl@act@endarray | 21, 47 |
| \adl@act@endbox | 21, 47 |
| \adl@act@hline | 21, 47 |
| \adl@act@ihashline | 21, 47 |
| \adl@activate | 21, 47 |
| \adl@activatembox | 21, 47 |
| \adl@act@acclassz | 21, 47 |
| \adl@activate | 20, 42 |
| \adl@activatembox | 20, 42 |
| \adl@addr1 | 37, 40, 41, 58 |
| \adl@addr1L | 42, 58 |
| \adl@addr1R | 42, 58 |
| \ADLactivate | 7, 15 |

B

| \c@LT@chunks | 58 |
| \CC | 48 |
| \adl@CC | 48 |
| \dashline | 4, 5, 35 |
| \adl@cdline | 35, 64 |
| \adl@cdline | 20, 21, 35, 47, 64 |
| \adl@cdline | 35, 64 |
environments:
  longtable  .  19, 48
  Array  .  7
  array  .  4
  Longtable  .  8
  longtable  .  8
  Tabular  .  7
  tabular  .  4
\everyvbox  .  18
\adl@everyvbox  .  17, 18
\adl@extract@arc0b  .  66
\adl@extract@arc  .  66
\adl@extract@dgcoB  .  66
\adl@extract@dgco  .  66
\adl@extract@drsc0b  .  66
\adl@extract@drsc  .  66
\extrarowheight  .  7
\adl@finaldepth  .  16, 32, 33, 36, 37
@finalstrut  .  29
\firsthdashline  .  4, 36
\adl@firsthdashline  .  36
\firstline  .  7, 36
\adl@gap  .  17, 38, 42
\adl@gapcolor  .  28
\adl@hcline  .  34, 35, 36, 53, 64
\adl@hclinesetup  .  64
\dashline  .  4, 5, 34
\adl@dashline  .  34, 36, 49, 53
\dashlinengap  .  14
\dashlinewidth  .  14
\adl@height  .  16, 19, 22, 32, 37, 38, 51
\adl@heightsave  .  16, 19, 38
\line  .  4, 33, 53, 63
\adl@line  .  20, 21, 33, 35, 47, 54
\adl@mark  .  44
\adl@idefflhd1  .  36
\@ifdefinable  .  46
\@ifpackageloaded  .  62
\adl@hdashline  .  20, 21, 34, 36, 47, 49, 53, 64
\adl@iimakevlr  .  41
\adl@imakevlr  .  41
\@ifadl@inactive  .  20, 49, 62
\adl@inactivecdl  .  20, 34, 53, 63
\@ifadl@inactivehd  .  15, 49
\adl@inactivevl  .  20, 29
\adl@ivline  .  28
\adl@ivmakevlr  .  41
\adl@ixhline  .  63
\@lastchclass  .  26
\@lastconn  .  38, 40
\lastdashline  .  4, 36
\adl@lastdashline  .  36
\lastline  .  7, 36
\LCC  .  7, 48
\@ifadl@leftrule  .  14, 23, 26
\Longtable  .  46
\longtable  .  8
\longtable (environment)  .  45
\longtable (environment)  .  8, 19, 48
\longtable (package)  .  3, 8, 48
\adl@Longtable  .  45
\@lowpenalty  .  54
\LT@array  .  49
\adl@LT@array  .  49
\LT@bchunk  .  58
\LT@echunk  .  49, 51
\LT@empty  .  49, 52
\LT@firsthead  .  52, 55, 56, 58
\LT@foot  .  52, 56, 58
\LT@head  .  52, 56, 58
\LT@hline  .  53, 68
\LT@kill  .  49, 52
\LT@lastfoot  .  52, 56, 58
\LT@make@row  .  49, 50
\LT@output  .  49, 56
\LT@rows  .  58
\LT@save@row  .  58
\LT@start  .  55
\LT@starpbox  .  50, 55
\LT@chunksize (counter)  .  8
\adl@LTdraw  .  57
\adl@LTendbbox  .  50, 55
\adl@LTendbbox  .  50, 55
\ifadl@LTfirstpage  .  49, 57
\adl@LTfootL  .  57
\adl@LTfootR  .  57
Change History

v1.0
  General: The style was born on a good day ... (1993/04/01) ................. 1
v1.05
  General: Cope with \ with negative optional vertical space. (1993/06/18) .... 1
v1.1
  General: Save and restore the \catcode for ‘\’. (1993/06/24) ............... 1
v1.2
  General: Various changes shown below. (1998/07/16) ......................... 1
v1.2-1
  General: Add this document. .............................................. 1
v1.2-2
  General: Cope with \texttt{\LaTeX}. ..................................... 1
v1.2-3
  General: Allow mixture of vertical solid- and dash-lines. .................... 1
v1.2-4
  General: Add the feature of explicit dash/gap specification. ................. 1
v1.2-5
  General: Fix some bugs and change codes. .................................. 1
v1.3
  General: Fix one bug shown below. (1998/10/08) ............................. 1
  \adl@activatepbox: \def-s for \adl@mcarrayrule etc. are enclosed in a group. ... 30
v1.4
  General: Make compatible with array package and add new features. (1999/06/25) ... 1
v1.4-1
  General: The following are changes of this document. ........................ 1
  General: The history on the compatibility with \texttt{array} package. ........... 3
  General: Explanation of package loading is added. ................................ 3
  General: Description of \texttt{\first/lastdashline} is added. ................... 4
  General: Description of the real width of vertical lines is added. ............. 5
  General: Description of drawing mode is added. .................................. 5
  General: Description of (in)activation is added. .................................. 6
  General: Description of characters and commands of \texttt{array} package is added... 7
  General: Description about ‘!’ of array package is added. ....................... 9
  General: Reference to the section for drawing mode is added. ................... 9
  General: Description on minimum length is added. .................................. 9
  General: Reference to the performance tuning section is added. .................. 9
  General: The title of section 4.1 is changed. .................................... 10
  General: \texttt{\hfil} is replaced with \texttt{\hss} taking the possibility of negative wide columns into account. .......................................................... 10
  General: Section 4.12 is added. ................................................ 44
  General: Section 4.13 is added. ................................................ 45
  General: Thank to more people. .................................................. 68
v1.4-2-1
  General: The following are for the general compatibility with \texttt{array}. .......... 1
  \ifadl@usingarraypkg: Introduced to know if \texttt{array} is loaded. ........... 15
  \adl@ncol: Introduced for new column counting in preamble construction. .......... 16
  \adl@everybox: Introduced for a tricky modification of \texttt{@array}. ............ 17
  \adl@array: Introduced to save original definition of \texttt{@array}. ............ 17

75
\@array: Drastically modified to avoid copy-and-modify. ........................................ 17
\@array: Introduced because \array uses it. ............................................................ 18
\adl@arrayinit: Modified for new column counting in preamble construction. ........ 19
\@mkpream: Modified for new column counting and control sequence redefinition. .... 23
\@addamp: Modified for new column counting in preamble construction. .............. 23
\@testpac: The version for array is introduced. .................................................... 24
\@class: Introduced because \array uses it. ............................................................ 24
\adl@class@start: Introduced for class number identification. .............................. 25
\adl@class@iiiorvii: Introduced for class number identification. ........................ 25
\adl@class@start: Introduced for class number identification. ............................... 26
\adl@class@iiiorvii: Introduced for class number identification. ........................ 26
\adl@arrayrule: Modified to replace \@columns with \adl@ncol. .............................. 26
\adl@arraydashrule: Modified to replace \@columns with \adl@ncol. ........................ 26
\adl@arraydashrule: Modified to replace \@columns with \adl@ncol. ........................ 26
\adl@arraydashrule: Modified to replace \@columns with \adl@ncol. ........................ 26
\adl@arraydashrule: Modified to refer \adl@preaminit rather than \LaTeX{}'s 6. ..... 26
\adl@colhtdp: Initialized by calling \adl@preaminit. ........................................ 28
\adl@vlineL: Initialized by calling \adl@preaminit. .......................................... 28
\adl@vlineR: Initialized by calling \adl@preaminit. .......................................... 28
\@endpbox: Introduced because array uses it. .................................................... 29
\multicolumn: Modified for several reason. ...................................................... 30
\adl@mcaddamp: Introduced for the complaint on multiple columns if with array. .... 30
\@activatepbox: Introduced to do nothing if with array. .................................. 30
\adl@mcaddamp: Introduced for the complaint on multiple columns if with array. .... 30
\@arraycr: The version for array is introduced. .................................................. 32
\v1.4-2-2
General: The following are to control the effective width of vertical lines. ............ 1
\ifadl@zwvrule: Introduced to indicate vertical lines have null width. ................. 14
\ADLnullwidehline: Introduced to make vertical lines null wide. ........................ 15
\ADLsomewidehline: Introduced to make vertical lines \arraydashline wide. ............... 15
\adl@arraydashrule: Modified to add invisible rule of \arrayrulewidth wide if \ADLsome wide. .............................................................. 26
\adl@nullwide: Modified to make vertical line null wide only if \ADLnullwide. .......... 43
\v1.4-2-3
General: The following are for inactivation of dash-line functions. ....................... 1
\ifadl@one: Introduced to indicate dash-line functions are inactive. .................... 15
\adl@org@arrayclassz: Introduced to restore \@arrayclassz. ........................... 17
\adl@org@tabclassz: Introduced to restore \@tabclassz. ................................ 17
\adl@org@classz: Introduced to restore \@classz. ........................................ 17
\adl@org@startpbox: Introduced to restore \@startpbox. .................................. 17
\adl@org@endpbox: Introduced to restore \@endpbox. .................................... 17
\adl@org@cline: Introduced to restore \cline. ................................................. 17
\adl@arrayinit: Modified to call \adl@inactivate. ........................................ 19
\adl@inactivate: Introduced to inactivate \@arrayclassz etc. ............................ 20
\adl@inactivevl: Introduced to emulate \hline and \vline. ............................... 29
\adl@inactivevd: Introduced to emulate \hline by \line. .................................. 34
\adl@inactivevd1: Introduced to emulate \vline by \line. .................................. 35
\adl@Array: Introduced as the body of \Array. ............................................... 43
\adl@Tabular: Introduced as the body of \Tabular. .......................................... 45

76
General: The following are to fix the bug by which the depth of `array` was always zero.

- `\adl@finaldepth`: Introduced to measure the depth of the last row. 16
- `\adl@org@cline`: Introduced to refer original version in modified `ccline`. 17
- `\adl@cr`: Modified to set `\adl@finaldepth`. 32
- `\hline`: Modified to set `\adl@finaldepth` to zero. 33
- `\cline`: Modified to set `\adl@finaldepth` to zero. 33
- `\adl@endarr`: Modified to set the depth of `array/tabular` to `\adl@finaldepth`. 37

General: The following are to rename macros for `\dashline`. 1
- `\dashline`: Modified to call renamed `\adl@cdline`. 35
- `\adl@cdline`: Renamed and modified to call renamed `\adl@cdline/a`. 35
- `\adl@cdlines`: Renamed. 35
- `\adl@cdlineb`: Renamed. 35

General: The following are to cope with very narrow or negative wide columns. 1
\adl@makevlrL: Modified to replace \hfil with \hss to prevent drawing vertical lines widen columns. ......................................................... 38
\adl@makevlrR: Modified to replace \hfil with \hss to prevent drawing vertical lines widen columns. ......................................................... 38
v1.4-2-9
\adl@arrayinit: The bug of saving \adl@colsR is fixed. .............................. 19
v1.4-3
General: Released to CTAN on 2000/07/04. ......................................... 1
v1.5
General: Make compatible with colortab, and fix bugs. (2000/07/12) .......... 1
v1.5-1
General: The following are for the compatibility with colortab. ................. 1
General: The history on the compatibility with colortab package. ............... 3
General: Caution about loading order of colortab is added. ...................... 7
General: Section 2.7 is added. ....................................................... 3
General: Description of colortab commands is added. ................................ 7
General: Caution about \AC/\EAC pair for vertical line coloring is added. .......... 9
\adl@arrayinit: Use new macro \adl@arraysave to save registers/structures. ...... 19
\adl@arraysave: Introduced to use in modified \CC@ of colortab. ................. 19
\CC@: Modified to save/restore globals before/after height measurement. ........ 48
v1.5-2
General: The following are for bug fix of \adl@putlrc. ............................ 1
\adl@colhtdp: The pseudo-formal description of (put-lrc) is modified. ............ 22
\adl@putlrc: \adl@putlrc must do \unhbox\adl@box to make glues effective. ....... 26
v1.5-3
General: The following are for bug fix of \adl@inactivate. .......................... 1
\adl@noalign: Move \adl@inactivate to @array from \adl@arrayinit. ............... 18
\adl@arrayinit: Move \adl@inactivate from \adl@arrayinit to \array. ............... 19
\adl@inactivate: Change \adl@inactivate caller to \array. ............................ 20
General: Thank to Yaxin Liu. ......................................................... 68
v1.54
General: Bug fixes. (2003/08/25) ...................................................... 1
v1.54-1
General: The following are for bug fix of \adl@vvl. .................................. 1
\adl@vrow: Rows for vertical lines are replaced by \adl@drawvl. ....................... 37
\adl@drawvl: Introduced to draw vertical lines correctly if \ADLsomewide. ........ 43
\adl@vvl: Insert a negative skip to left/right of the line if \ADLsomewide. ........ 43
v1.54-2
General: The following are for bug fix of activation. ................................ 1
\adl@noalign: Invoke \adl@activate if not \ifadl@inactive. ......................... 18
\adl@inactivate: Add \adl@argcr to inactivation. ..................................... 20
\adl@activate: Introduced to activate \arrayclassz etc. again. ..................... 20
\adl@act@arrayclassz: Introduced to activate \arrayclassz etc. again. ............. 47
v1.54-3
General: The following are miscellaneous modifications. .......................... 1
\adl@hcline: Omit \vskip if the space is 0. ......................................... 36
v1.6
General: The following are for the compatibility with longtable. (2003/08/25) .... 1
General: The history on the compatibility with longtable package. .................. 3
General: Caution about loading order of longtable is added. ....................... 3

78
General: Description of \longtable is added. ........................................ 8
General: Description of \discard is added. ....................................... 11
\adl@discard: Add initialization of \adl@discard. ............................... 19
General: Add a summary of activation/inactivation. ......................... 21
\adl@discrc: Modified to insert \adl@discard. ................................. 32
\adl@LLongtable: Introduced as the body of \Longtable. ..................... 45
\Longtable: Introduced as the always-active \longtable. .................... 46
\endLongtable: Introduced to \end the environment \Longtable. ............... 46
\ADLnoshorthandd: \Longtable and \endLongtable are added. ................. 47
General: §4.15 is added. ............................................................. 48
General: Thank to people for \longtable. ...................................... 68

v1.7

General: The following are for the compatibility with \colortbl. (2004/05/21) ....... 1
General: The history on the compatibility with \colortbl package is added. ........... 3
General: Caution about loading order of \colortbl is added. ........................ 3
General: Description of \colortbl and related commands is added. ................. 7
General: Comment on vertical line coloring with \colortbl is added. ............... 9
General: Add notes for dash line coloring. ...................................... 10
General: A dash/gap specification $d_j/g_j$ now has color. ......................... 11
\endtabular: Modified to refer proper \endarray depending on the existence of \colortbl. 37
General: Codes for \longtable is surrounded by \ifx/\fi .......................... 49
General: §4.16 is added. ............................................................. 60
General: Thank to Klaus Dalinghaus and refer original \colortbl. .................. 68

v1.7-1

General: The following are for null-wide horizontal lines. ........................ 1
\if\adl@zhrule: Introduced to indicate horizontal lines have null width. ........ 15
\ADLnullwide: Introduced to make horizontal lines null wide. .................. 15
\ADLsomewide: Introduced to make horizontal lines \arraydashline wide. ........ 15
\adl@inactivate: Remove \cline because our own version is needed for null-wide. 20
\cline: Modified to shift up if null-wide. ...................................... 33
\adl@hline: Modified to shift up if null-wide. .................................. 33
\adl@hdashline: Modified for null-wide horizontal lines. ....................... 34
\adl@hdashline: Modified not to shift null-wide \cdashline down. .............. 35
\adl@hdashline: Modified to shift up if null-wide. ................................ 35
\adl@hdashlined: Modified to invoke \cline rather than \adl@orgcline for null-wide. 35
\adl@hdashline: Modified to shift up null-wide \cdashline down. ............... 36
\ADLThdashline: Keep original without shift up because it is done by \adl@Thdashline. 53
\ADLThdashline: Modified to shift up if null-wide. ................................ 54

v1.7-2

General: The following are to fix the bug of \arrayrulecolor etc. in \colortbl. .... 1
\adl@noalign: Introduced to fix a bug of \colortbl. .............................. 17
\adl@noalign: Make \adl@noalign \let-equal to \noalign. .......................... 18

v1.7-3

General: The following are for vertical line coloring. ............................ 1
\adl@vline: Modified to add color arguments to \adl@vlineL/R ................... 26
\adl@vlineL: Color arguments are added. ........................................... 28
\adl@vlineR: Color arguments are added. .......................................... 28
\adl@ivline: Invocations of \adl@setcolor are added. ............................ 28
\texttt{\adl@setcolor}: Introduced to color vertical lines. .......................... 28
\texttt{\adl@nocolor}: Introduced to examine if coloring is specified. ............... 28
\texttt{\adl@dashcolor}: Introduced as the temporary variable of color specification of dashes. 28
\texttt{\adl@gapcolor}: Introduced as the temporary variable of color specification of gaps. .... 28
\texttt{\adl@inactivevl}: Modified to color the \texttt{\line} by the first argument. ............... 29
\texttt{\adl@makevlr}: Modified to initialize \texttt{\adl@dashcolor} and \texttt{\adl@gapcolor}. .......... 40
\texttt{\adl@iimakevlr}: Modified to check color indentity. ................................. 41
\texttt{\adl@ivmakevlr}: Modified not to see \texttt{d} and \texttt{g} which now have colors. .......... 41
\texttt{\adl@addvlL}: Modified to add colors to \texttt{\delta} and \texttt{\xi}. ............................ 42
\texttt{\adl@addvlR}: Modified to add colors to \texttt{\delta} and \texttt{\xi}. ............................ 42
\texttt{\adl@setcolor}: Introduced to the bug fix of \texttt{\array}'s \texttt{m}-columns. ............. 43

\texttt{v1.71}

General: The following are for bug fix for \texttt{array}'s \texttt{m}-columns. (2004/7/31) ........... 1
\texttt{\@mkpream}: Modified to nullify \texttt{\adl@startmbox} and \texttt{\adl@endmbox} for \texttt{array}'s \texttt{m}-columns. 23
\texttt{\@classz}: Modified to call \texttt{\adl@startmbox} and \texttt{\adl@endmbox} for \texttt{array}'s \texttt{m}-columns. .... 24
\texttt{\adl@startmbox}: Introduced to the bug fix of \texttt{array}'s \texttt{m}-columns. ............. 29
\texttt{\adl@endmbox}: Introduced to the bug fix of \texttt{array}'s \texttt{m}-columns. ............. 29
General: Thank to Morten Høgholm. ................................................................. 68

\texttt{v1.72}

General: Bug fix and revision of §2.4. (2016/03/19)................................. 1
\texttt{\adl@LT@make@row}: Added to process footnotes in \texttt{longtable}'s \texttt{m}-columns. .......... 1
\texttt{\LT@make@row}: Modified to add \texttt{\let}-assignments to \texttt{\adl@beginmbox} and \texttt{\adl@endmbox} so that footnotes are correctly processed at the closing of a \texttt{m}-type column. .... 50
\texttt{\adl@LTendmbox}: Added to process footnotes in \texttt{m}-type columns appropriately. ........ 55
General: Thank to Maëul Rouquette. .............................................................. 68

\texttt{v1.72-2}

General: Revise §2.4 reflecting the fix of \texttt{\xleaders}. ........................... 5
General: Remove the caution about the dash segment dropping. ..................... 9
General: Change the title of §4.2 and rephrase sentences according to the fix of \texttt{\xleader}'s problem. ................................................................. 13

\texttt{v1.73}

General: Bug fix. (2016/04/28) ................................................................. 1
General: Thank to Maëul Rouquette again. .................................................... 68

\texttt{v1.73-1}

General: The following are to fix the problem that the top edge a vertical (dash-)line is at the bottom of a horizontal line rather than it top. ................................. 1
General: Add a paragraph describing the perfect contacts of vertical and horizontal lines. 4
General: Add the definition of \texttt{\eta}, and addition/subtraction of it for \texttt{\tau} and \texttt{\beta}. ........ 12
General: Add \texttt{\eta} = \texttt{\adl@lastconn}, its initialization and updates, and the addition to \texttt{\tau}. 38
\texttt{\adl@makevlr}: Add \texttt{\eta} = \texttt{\adl@lastconn} ← 0. ................................ 40
\texttt{\adl@imakevlr}: Modify the definition of \texttt{\adl@connect} to pass \texttt{h} to \texttt{\adl@connect}. ........ 41
\texttt{\adl@iimakevlr}: Replace two occurences of \texttt{\tau} ← \texttt{\beta} with \texttt{\tau} ← \texttt{\beta + \eta} and add \texttt{\eta} ← 0, where \texttt{\eta} = \texttt{\adl@lastconn}. ........................................ 41
\texttt{\adl@endmakelvlrcut}: Add \texttt{\eta} = \texttt{\adl@lastconn} ← 0. ................................ 41
\texttt{\adl@connect}: Add \texttt{\eta} = \texttt{\adl@lastconn} ← \texttt{h} with the added argument \texttt{h}. ........ 41

\texttt{v1.73-2}

General: The following are to fix the bug that \texttt{\adashline} is not properly processed in a \texttt{array/tabular} environment if \texttt{longtable} is loaded. ................................. 1

80
\LTarray: Add \let-assignment of \adl@LTdashedline to \adl@hdashline so that the longtable version of \adl@hdashline is effective only in longtable environment rather than globally. 49
\adl@LTdashedline: Renamed from \adl@hdashline to make it effective only in longtable environments. 53

v1.74
General: The following are to fix the bug in the array-compatible mechanism by which delarray did not work well. 1
General: Comment on \plextarydshln is added. 9
\@@array: Make \@@array \let-equal to \array only when it is made so by array and the equality is kept. 18
\endarray: Add conditional invocation of \arrayright. 37
\endarray: Add conditional invocation of \arrayright. 63
General: Thank to Hironobu Yamashita. 68
General: Thank to Hironobu Yamashita for coloring problem. 68

v1.75
General: The following are to cope with the change in array v2.4i or later in which \@startpbox and \@endpbox have \color@begingroup and \color@endgroup, respectively. 1
\adl@org@startpbox: Introduced to restore \@startpbox. 17
\adl@inactivate: Add \@startpbox to inactivation. 20
\adl@activate: Add \@startpbox to activation. 20
\@startpbox: Introduced to cope with the \color@begingroup/\color@endgroup problem. 29
\@endpbox: Modified to ensure that the macro has \color@endgroup irrespective of array's version. 29
\adl@startpbox: Replace \@startpbox with \adl@org@startpbox to avoid the color-grouping problem. 29
\adl@act@startpbox: Introduced because \@startpbox may be different from the original. 47
\LT@make@row: Add description that \adl@LTendpbox is common for \@endpbox and \@endpbox. 50
\adl@LTendpbox: Add description that the macro is used for both of \@endpbox and \@endpbox. 55
\adl@activate: Add inactivation of \@startpbox. 62

v1.76
General: The following are to cope with the change in longtable bundled in latex-tools 2019-01-05 or later in which \LT@startpbox and \LT@endpbox have \color@begingroup and \color@endgroup, respectively. 1
\adl@startpbox: Replace \adl@org@startpbox with \@startpbox to do color-grouping always and to invoke \adl@LTstartpbox through \@startpbox when longtable is in use. 29
\adl@endpbox: Replace \adl@org@endpbox with \@endpbox of array has to perform color-grouping always. 29
\LT@make@row: Modified to add \let-assignment \@startpbox = \adl@LTstartpbox if array is in use, and to replace the RHS of \let-assignments of \@startpbox and \@startpbox for the case without array, which was \adl@LTstartpbox, with newly introduced \adl@LTstartpbox. 50
\adl@LTstartpbox: Added for p columns without \array. 55
\adl@LTendpbox: Modified to invoke \color@begingroup always. 55
\adl@LTendpbox: Modified to invoke \color@endgroup always. 55