The \texttt{arydshln} package

Hiroshi Nakashima
(Kyoto University)

2018/09/26

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 .................................................. 21
4.6 Building Columns ................................................ 27
4.7 Multi-columns ..................................................... 29
4.8 End of Rows ....................................................... 31
4.9 Horizontal Lines .................................................. 33
4.10 End of Environment .............................................. 36
4.11 Drawing Vertical Lines .......................................... 38

*This file has version number v1.74, last revised 2018/09/26.
4.12 Drawing Dash-lines ......................................................... 43
4.13 Shorthand Activation ....................................................... 45
4.14 Compatibility with \texttt{colortab} ....................................... 47
4.15 Compatibility with \texttt{longtable} ....................................... 48
  4.15.1 Initialization ......................................................... 48
  4.15.2 Ending Chunks ......................................................... 50
  4.15.3 Horizontal Lines and \texttt{p}-Boxes ................................. 52
  4.15.4 First Chunk .......................................................... 54
  4.15.5 Output Routine ...................................................... 55
4.16 Compatibility with \texttt{colortbl} ..................................... 60
  4.16.1 Initialization, Cell Coloring and Finalization .................. 61
  4.16.2 Horizontal Line Coloring ........................................... 63
  4.16.3 Vertical Line Coloring .............................................. 64
  4.16.4 Compatibility with \texttt{longtable} ................................. 67
1 Introduction

In January 1993, Weimin Zhang kindly posted a style \texttt{hvdashln} written by the author, which draws horizontal/vertical dash-lines in \LaTeX'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{arydshln} 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{arydshln} 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\textsuperscript{1} until three other complaints were made. Then the author attacked the problem hoping it is the last compatibility issue\textsuperscript{2}.

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\textsuperscript{3}.

2 Usage

2.1 Loading Package

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

\begin{verbatim}
\usepackage{arydshln}
\end{verbatim}

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

\begin{verbatim}
\documentstyle[\ldots,arydshln,\ldots]{⟨style⟩}
\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 \LaTeX Tools Bundle, and \texttt{colortab} and \texttt{colortbl} package is that \texttt{arydshln} 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.

\textsuperscript{1}Two years and a half! Sorry Craig.

\textsuperscript{2}But his hope was dashed as described below.

\textsuperscript{3}Without dreaming it is the last compatibility issue.
\usepackage{array} % and/or
\usepackage{longtable} % and/or
\usepackage{colortab} % or
\usepackage{colortbl}
\usepackage{arydshln}

2.2 Basic Usage

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

dashline Drawing a vertical dash-line is quite simple. Use ‘;’ in the preamble as the separator of columns separated by the dash-line, just like using ‘|’ to draw a vertical solid-line. The \emph{preamble} means not only that of the environment, but also the first argument of \texttt{multicolumn}.

cdashline It is also simple to draw a horizontal dash-line. Use \texttt{hdashline} and \texttt{cdashline} as the counterparts of \texttt{\hline} and \texttt{\cline}.

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

will produce the following result.

A B C
AAA BBB CCC
AB C

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 \emph{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\textsuperscript{4}.

\texttt{\firsthdashline} and \texttt{\lasthdashline} named \texttt{\first} hdashline and \texttt{\lasthdashline} are available.

2.3 Style Parameters

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

\textsuperscript{4}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.
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;

\[\{\text{\texttt{dash}}/\text{\texttt{gap}}\}\]

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{\textbackslash hdashline} and \texttt{\textbackslash cdashline} as follows.

\begin{quote}
\texttt{\textbackslash hdashline\{\texttt{dash}}/\texttt{(gap)}\}
\texttt{\textbackslash cdashline\{col1\}–\{col2\}\{\texttt{\textbackslash cdashline\{\texttt{col1}–\{col2\}\}}/\texttt{(gap)}\}}
\end{quote}

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}
\begin{tabular}{|l::c;{2pt/2pt}r|} \hline A&B&C \\
AAA&BBB&CCC \\
AB & & C \\
\hline
\end{tabular}
\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{\textbackslash ADLnullwide} if you love \LaTeX standard, or \texttt{\textbackslash 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{\textbackslash 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{\textbackslash 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 \textit{drawing modes} which you may specify by \texttt{\textbackslash ADLdrawingmode\{\textit{m}\}} as follows.
Figure 1: Drawing mode controlled by \texttt{ADL\textbackslash drawingmode}

<table>
<thead>
<tr>
<th>Figure</th>
<th>Drawing Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>$m = 1$</td>
</tr>
<tr>
<td>(2)</td>
<td>$m = 2$</td>
</tr>
<tr>
<td>(3)</td>
<td>$m = 3$</td>
</tr>
</tbody>
</table>

- **$m = 1$**
  As shown in Figure 1(1), it gives most beautiful result by \texttt{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 \texttt{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 \texttt{array/tabular} with dash-lines is poorer than that of ordinary ones. In fact, according to author’s small performance evaluation with a \texttt{tabular} having nine vertical and ten horizontal dash-lines, typesetting the \texttt{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 \texttt{arydshln} makes typesetting \texttt{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 \texttt{array/tabular} with solid lines, \LaTeX{} will run slowly even with quite few (or no) \texttt{array/tabular} with dash-lines.

To cope with this problem, you may inactivate dash-line functions by the command \texttt{ADL\textbackslash inactivate} 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 \texttt{xleaders}, the second bottom/rightmost segments of right/lower lines were dropped.}

6
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 \ADLnactivate in the preamble, switch on by \ADLactivate
before you really need dash-lines, and switch off again afterward. A wiser way could be
surrounding 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 \Array and 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
\Array and/or \Tabular have already been defined when \arydshln is loaded, you will get a
warning to show you have to define new environments, say \dlarray and \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 \arydshln is loaded, their definitions are
silently replaced or \LaTeXX complains of multiple definitions. The error in the latter case
will be avoided by putting \ADLnoshorthand just after \usepackage{arydshln}.

2.7 Compatibility with Other Packages

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

Also users of \colortab package may use \LCC/ECC construct to color columns. A hori-
zontal solid/dash line may be colored by, e.g. \NAC\hdashline\ENAC. The pair of \AC and
\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 \ADLinactivate.

Another (and more convenient) table coloring tool \colortbl may be also used simply by
loading it before \arydshln. Not only the painting commands \rowcolor, \columncolor and
\cellcolor work well, but both solid and dash lines are also colored by the command
\arrayrulecolor of \colortbl\footnote{The \colortbl manual says \arrayrulecolor and \doublerulesepcolor may be in \(\ldots\) in a preamble but they cause an error with the original implementation. This bug is fixed in \arydshln and they are now usable to specify the color of the vertical (dash) lines whose specifications occur after the commands.}. One caution is that \arrayrulecolor defines the color of the dash-part of dash lines and thus gap-part has no color (i.e. color of the paper on which
the line drawn). Therefore, if you have a \tabular like;

\begin{tabular}{|>{\columncolor{red}}l:>{\colomncolor{green}}r|}
... \\
\end{tabular}
you will find the vertical dash line is a sequence of black (or the color of \arrayrulecolor) and white segments. This problem is partly solved by declaring \ADLnullwide to conjunct the red and blue columns and to draw the dash line on their border.

Unfortunately, however, \ADLnullwide does not affect the real width of horizontal (dash) lines and thus you will still see white gaps in \hdashline and \cdashline. A solution is to put \ADLnullwide before you start a \array/tabular. With this command, a horizontal (dash) line is drawn adjusting its bottom edge to that of the row above. The command \ADLsomewide 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 \dashgapcolor with arguments same as \color. For example;
\arrayrulecolor{green}\dashgapcolor[rgb]{1,1,0}
makes colorful dash lines with green dashes and yellow gaps. The command can be placed outside of \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 \dashgapcolor (no arguments) nullifies the effect of \dashgapcolor. Note that \dashgapcolor is different from \dashgapcolorwhite because the former makes gaps transparent while the later whiten them.

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

\begin{longtable}{|c|c|c|c|}
\hline
1 & 2 & 3 & \\
\hline
wide multicolumn spanning 1–3 & \\
\hline
multicolumn 1–2 & 3 & \\
\hline
wide 1 & 2 & 3 & \\
\hline
\end{longtable}

Since 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 arydshln.

\begin{longtable}{|c|c|c|c|}
\hline
1 & 2 & 3 & \\
\hline
wide multicolumn spanning 1–3 & \\
\hline
multicolumn 1–2 & 3 & \\
\hline
wide 1 & 2 & 3 & \\
\hline
\end{longtable}

As you see, the vertical lines are drawn at the column edges of the last row because 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.

\footnote{Since colortbl automatically loads array, the default is \ADLsomewide}
\footnote{This command also makes \cline and \cdashline visible even if the row below is painted.}
\footnote{More precisely, drawn according to the column widths established by all the chunks preceding page output.}
3 Known Problems

There are following known problems.

1. The new preamble specifiers ‘:’ and ‘;{⟨dash⟩/⟨gap⟩}’ cannot be followed or preceded by ‘φ{⟨text⟩}’, 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 ‘φ{⟨text⟩}’, while that to draw at the right edge cannot be followed by ‘φ{⟨text⟩}’.

2. If you use array package, the restriction of ‘φ’ shown above is also applied to ‘!' .

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 \dashlinedash. In mode 2, the minimum length to avoid the protrusion is $1.5 \times \text{\normalfont\dashlinedash} + \text{\normalfont\dashlinegap}$.

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

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

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

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>wide multicolumn spanning 1–3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>multicolumn 1–2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>wide 1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
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}{cc}
A & B \\
A & B
\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}{cc}
\omit & \omit \\
\langle dash-line of 36 pt high\rangle & \omit
\end{tabular}

Note that \langle dash-line of 36 pt high\rangle have to be \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 `colortbl` 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 \[ R = (r_1, r_2, \ldots, r_N). \]

2. The \(i^{th}\) element of \(R\), \(r_i\), is one of the following\(^{\text{11}}\).
   
   (a) 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.
   
   (b) `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.
   
   (c) 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 broken at the row.
   
   (d) `disconnect(h_i)` for a vertical gap generated by a sequence of `\h(dash)line` meaning that \(r_i\) is an empty pseudo row of \(h_i\) high and dash-lines are broken at the row.

3. \(C^L_i = \langle e^L_1, e^L_2, \ldots, e^L_m \rangle\) where \(e^L_j\) 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 \(C^R_i\), \(e^L_j\), 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.

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

\(^{\text{11}}\)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 \( e_j \sim e_j' \).

\[
e_j \sim e_j' \iff i < i' \land
e_j^c = e_j'^c \land d_j^c = d_j'^c \land g_j^c = g_j'^c \land p_j^c = p_j'^c \land
\forall k(i < k < i' \rightarrow r_k \in \{\text{connect}(h_k), \text{discard}(h_k)\})
\]

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

- \( k \neq k' \leftrightarrow S_k \cap S_{k'} = \emptyset \)
- \( e_j \sim e_j' \leftrightarrow \exists k: e_j, e_j' \in S_k \land S_k = \{\ldots, e_j, e_j', \ldots\} \)
- \( k < k' \leftrightarrow \forall e_j \in S_k, \forall e_j' \in S_{k'}: (e_j^c < e_j'^c) \lor (e_j^c = e_j'^c \land p_j^c < p_j'^c) \lor (e_j^c = e_j'^c \land p_j^c = p_j'^c \land i < i') \).

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 the \( \pi_k^{th} \) line in the \( \gamma_k^{th} \) 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 tabular, where:

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

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;

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

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

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

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 $C_i^R$ with the following modifications.

\[k < k' \Leftrightarrow \forall c_i' \in S_k, \forall c_j' \in S_{k'} : (c_i' < c_j') \vee (c_i' = c_j' \wedge p_i' > p_j') \vee (c_i' = c_j' \wedge p_j' = p_i') \wedge i < i')\]

\[\sigma_1 = \texttt{\textbackslash omit\textbackslash hss[\textbackslash omit\textbackslash hss]}^{\gamma_1 - 1}\]

\[\sigma_{k>1} = \begin{cases} \texttt{\textbackslash hskip\textbackslash doublerulesep} & \text{if } \gamma_{k-1} = \gamma_k \wedge \pi_{k-1} = \pi_k \\ \texttt{\textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbacklash}^{\gamma_k - \gamma_{k-1}} & \text{if } \gamma_{k-1} = \gamma_k \wedge \pi_{k-1} \neq \pi_k \\ \texttt{\textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbacklash}^{\gamma_k - \gamma_{k-1}} & \text{if } \gamma_{k-1} \neq \gamma_k \end{cases}\]

\[\sigma_{n+1} = [\texttt{\textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbackslash \textbacklash}^{\gamma_n - 1}\]

### 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 $10\text{sp}$, 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 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;

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

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

\[D_0(d/2)G_0(\varepsilon')D_1(d)G_1(\varepsilon')\texttt{\textbackslash xleaders} G_0^n(\varepsilon')D_{n+1}(d/2)\]

\[= D_0(d/2)G_0(\varepsilon')G_0^n(\varepsilon')G_1^n(\varepsilon)G_2^n(\varepsilon)\ldots\]

\[G_{n-1}^n(\varepsilon)(G_{n-1}^n(\varepsilon)D_n(d)G_n(\varepsilon')D_{n+1}(d/2)\]

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

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

\[
D_0(d/2)G_0^t(g/2)D_1^t(d)G_1^t(g/2)G(-d - g)
\]

That is, \( n^{th} \) \textbackslash box 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:

\[
D_0^t(d/2)G_0'(g/2)D_1'(d)(g/2)G(-d - g)G_1'(g/2)D_n'(g/2)G_n'(g)D_n+1(d/2)
\]

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

\section{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.

Next, the following six switches are declared.

\texttt{\ifadl@leftrule} is used in the preamble analysis macro \texttt{\@mkpream} and is true during it 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.
\texttt{\ifadl@zwrule} • \texttt{\ifadl@zwrule} 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-\arrayulewidth} 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{\arrayulewidth} contribute to the height of \texttt{array/tabular} as usual.

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

\texttt{\ifadl@inactive} • \texttt{\ifadl@inactive} inactivates dash-line functions if it is true. Its default value is false.

We also use a working switch \texttt{\@tempswa}.

8 \texttt{\newif\ifadl@leftrule}
9 \texttt{\newif\ifadl@connected}
10 \texttt{\newif\ifadl@doublerule}
11 \texttt{\newif\ifadl@zwvrule}
12 \texttt{\newif\ifadl@zwhrule}
13 \texttt{\newif\ifadl@usingarypkg}
14 \texttt{\ifx\extrarowheight\undefined \adl@usingarypkgfalse}
15 \texttt{\else \adl@usingarypkgtrue \fi}
16 \texttt{\newif\ifadl@inactive \adl@inactivefalse}

\texttt{\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}.

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

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

18 \texttt{\def\ADLnullwide{\adl@zwvruletrue}}
19 \texttt{\def\ADLsomewide{\adl@zwvrulefalse}}
20 \texttt{\ifadl@usingarypkg \ADLsomewide \else \ADLnullwide \fi}
21 \texttt{\def\ADLnullwidehline{\adl@zwvruletrue}}
22 \texttt{\def\ADLsomewidehline{\adl@zwvrulefalse}}
23 \texttt{\ADLsomewidehline}
24
25 \texttt{\def\ADLactivate{\adl@inactivefalse}}
26 \texttt{\def\ADLinactivate{\adl@inactivetrue}}
27

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

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

15
The \texttt{dimen} registers \texttt{adl@height} and \texttt{adl@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{adl@box} and are updated if they are less.

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{adl@heightsave/adl@depthsave} at the beginning of the environment and are restored at its end.

The other \texttt{dimen} register \texttt{adl@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}.

The \texttt{adl@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{adl@columns} directly but increment \texttt{adl@ncol} when each column of preamble is built and move its value to \texttt{adl@columns} after the preamble is constructed.

To process \texttt{multicolumn}, we have to know the column number where it appears. Thus we have a column counter \texttt{adl@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{adl@currentcolumnsave}.

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

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

\texttt{adl@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 \texttt{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{adl@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}.

- In order to check \( e_{ij}' \sim e_{ij}'' \), the size attributes of \( d_{ij} \) and \( g_{ij} \) are kept in the registers \texttt{\adl@dash} and \texttt{\adl@gap} when we process \( e_{ij}' \). As explained above, \( d_{ij} \) and \( g_{ij} \) are integers and thus \texttt{\adl@dash} and \texttt{\adl@gap} are \texttt{\count} registers.

- The coding of \texttt{\cadashline} is similar to that of \texttt{\cline} in \LaTeX{}-2.09 which uses two global \texttt{\count} registers \texttt{\@cla} and \texttt{\@clb}. These registers are omitted from \LaTeX{}-2ε because its \texttt{\cline} is completely recoded. We could adopt new coding but it requires some other macro definitions that \LaTeX{}-2.09 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 \LaTeX{}-2.09 and \LaTeX{}-2ε.

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

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 \LaTeX{} and \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}.

The other declarative stuff consists of the sequence of \texttt{\let} to capture the original definitions of macros that we will modify afterword. 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.

\section{Initialization}

\LaTeX{}'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}/\texttt{tabular}. This saving and initializing are performed by \texttt{\adl@arrayinit} as explained below. The problem in the
The main difference is that \LaTeX builds \texttt{@preamble} locally, while \texttt{array} does globally exploiting the fact that the lifetime of \texttt{@preamble} ends before another \texttt{array/tabular} appears in a column. The latter implementation will work well unless the building process in \texttt{@mkpream} produces something referred to after \texttt{@preamble} is thrown into \TeX's stomach. In our implementation, unfortunately, the number of columns has to be counted in \texttt{@mkpream} and will be referred to by \texttt{\textbackslash hdashline} and the vertical line drawing procedure.

Thus we have to change the column counting mechanism depending on whether or not \texttt{array} is in use. The simplest way could be to copy the codes of \LaTeX and \texttt{array} and modify them appropriately examining the value of \texttt{\textbackslash 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 \texttt{\textbackslash everyvbox} in which \texttt{\textbackslash adl@arrayinit} is temporarily included to initialize registers and locally set \texttt{\textbackslash adl@columns} to the number of columns \texttt{\textbackslash global-ly} counted by \texttt{\textbackslash adl@ncol}. This trick works well so far because:

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

and will hopefully work in future.

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

\texttt{@@array} Another stuff for the compatibility with \texttt{array} is to \texttt{\textbackslash let} a control sequence \texttt{@@array} be equal to \texttt{@array} if it is made so by \texttt{array} and the equality is kept. That is, with \texttt{array} \texttt{@@array} is invoked by \texttt{@tabarray} and it is \texttt{\textbackslash let}-equal to \texttt{@array} by default, while \texttt{@@array} can be made different from \texttt{@array} by some other package, e.g., \texttt{delarray}, to do some special operations defined in the package. Therefore by the conditional equalization with \texttt{\textbackslash ifx}, our own \texttt{@array} is directly invoked through \texttt{@@array} if the default equality is kept, while otherwise the package-dependent definition of \texttt{@@array} is respected.
As described in §4.3, registers updated \texttt{\global-ly}, which are \texttt{\global@height}, \texttt{\global@depth}, \texttt{\global@currentcolumn} and \texttt{\global@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.

\begin{itemize}
\item 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}.

\item 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}.
\end{itemize}

In the real implementation, \( e_i^j \) is represented by a control sequence \texttt{\@elt}, and \texttt{\adl@connect} is made \texttt{\let}-equal to \texttt{\relax} to keep them from expansion during \( R \) is constructed. In \texttt{longtable} environment, \texttt{\adl@connect} for negative vertical space inserted by \texttt{\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 \texttt{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}
59 \def\adl@arrayinit{%
60   \adl@arraysave
61   \global\adl@height\z@ \global\adl@depth\z@ \\
62   \global\adl@currentcolumn\one \global\adl@totalheight\z@ \\
63   \gdef\adl@rowsL\{}\gdef\adl@rowsR\{}\gdef\adl@colsL\{}\gdef\adl@colsR\{}\%
64   \let\@elt\relax \let\adl@connect\relax \def\adl@discard\{\adl@connect\%
65   \adl@columns\adl@ncol\}
66 \def\adl@arraysave{%
67   \adl@heightsave\adl@height \\
68   \adl@depthsave\adl@depth \\
69   \adl@currentcolumnsave\adl@currentcolumn \\
70   \adl@totalheightsave\adl@totalheight \\
71   \let\adl@rowsLsave\adl@rowsL \\
72   \let\adl@rowsRsave\adl@rowsR \\
73   \let\adl@colsLsave\adl@colsL \\
74   \let\adl@colsRsave\adl@colsR}
75\end{verbatim}
\adl@inactivate If \ADLinactivate has effect and thus \ifadl@inactive is true, the macro \adl@ inactivate is called from \Array\footnote{Before v1.53, \adl@inactivate was called from \adl@arrayinit and thus invoked after the preamble of \array is built. This was incorrect of course and made inactivating any \array macro inactive again in order to cope with the case in which an inactivate \array/\tabular has active children in it\footnote{Before v1.54, an active \array/\tabular in an inactive parent was not activated.}. To do that, \adl@activate makes}. This \let\-s the following control sequences be equal to their counterparts in \LaTeX\ and/or \array package.

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

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

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

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

We also \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 \adl@vlineL and \adl@vlineR, because the latter is referred to when \array is in use while the former is done otherwise. Their \R relatives are also inactivated by the same reason.

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

\adl@activate On the other hand, if \ifadl@inactive is false, the macro \adl@activate is called from \Array to make inactivated macros active again in order to cope with the case in which an inactive \array/\tabular has active children in it\footnote{Before v1.54, an active \array/\tabular in an inactive parent was not activated.}.
### Table 1: Active and Inactive Operations

<table>
<thead>
<tr>
<th>command</th>
<th>active</th>
<th>inactive</th>
</tr>
</thead>
<tbody>
<tr>
<td>l c r</td>
<td>\texttt{\textbackslash adl\texttt{\textbackslash act@classz}}</td>
<td>\texttt{\textbackslash adl\texttt{\textbackslash org@classz}}</td>
</tr>
<tr>
<td></td>
<td>\texttt{\textbackslash adl\texttt{\textbackslash act@tabclassz}}</td>
<td>\texttt{\textbackslash adl\texttt{\textbackslash org@tabclassz}}</td>
</tr>
<tr>
<td></td>
<td>\texttt{\textbackslash adl\texttt{\textbackslash act@arrayclassz}}</td>
<td>\texttt{\textbackslash adl\texttt{\textbackslash org@arrayclassz}}</td>
</tr>
</tbody>
</table>

| p m b (open) | \texttt{\textbackslash adl\texttt{\textbackslash act@classz}} | \texttt{\textbackslash adl\texttt{\textbackslash org@classz}} |
|              | \texttt{\textbackslash adl\texttt{\textbackslash act@startpbox}} | \texttt{\textbackslash adl\texttt{\textbackslash org@startpbox}} |

| p m b (close) | \texttt{\textbackslash adl\texttt{\textbackslash act@endpbox}} | \texttt{\textbackslash adl\texttt{\textbackslash org@endpbox}} |

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

### 4.5 Making Preamble

Each preamble character is converted to a part of \texttt{\textbackslash halign}'s preamble as follows.
\texttt{\adl@colhtdp} • ‘l’, ‘r’ and ‘c’ are converted to the following \texttt{(lrc)}.

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

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{\adl@vlineL} \texttt{\adl@vlineR} • ‘|’, ‘:’ and \texttt{{\{dash\}}/{\{gap\}}} are converted to the following \texttt{(vline)}.

\begin{equation*}
\texttt{(vline)} := \texttt{[\hskip\doublerulesep\vline-LR]} \end{equation*}
\begin{equation*}
\texttt{(vline-LR)} := \texttt{\adl@vlineL{\{I_d\}}{\{I_g\}}{\{c\}}{\{d\}}{\{g\}}} \end{equation*}
\begin{equation*}
\texttt{\adl@vlineR{\{I_d\}}{\{I_g\}}{\{c\}}{\{d\}}{\{g\}}} \end{equation*}
\begin{equation*}
\texttt{(d)} := 0 \quad \text{... for ‘|’} \end{equation*}
\begin{equation*}
\texttt{\dashlinedash} \quad \text{... for ‘;’} \end{equation*}
\begin{equation*}
\texttt{\dashlinegap} \quad \text{... for ‘::’} \end{equation*}
\begin{equation*}
\texttt{(g)} := 0 \quad \text{... for ‘|’} \end{equation*}
\begin{equation*}
\texttt{\dashlinegap} \quad \text{... for ‘;’} \end{equation*}
\begin{equation*}
\texttt{\dashlinegap} \quad \text{... for ‘::’} \end{equation*}
\begin{equation*}
\texttt{(c)} \end{equation*}

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

Additionally, each column except for the last one has;

\begin{equation*}
\texttt{\global\advance\adl@currentcolumn\@ne} \end{equation*}

just before \& to increment \texttt{\adl@currentcolumn}. Other features, such as inserting spaces of \texttt{\arraycolsep/\tabcolsep}, are as same as original scheme. This means that \texttt{@{(text)}} and \texttt{!{(text)}} of \texttt{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{\LaTeX} original is done by \texttt{(modified) \@startpbox} and \texttt{\@endpbox} and thus the preamble for ‘p’ is not modified. In the case with \texttt{array}, however, the preambles for ‘p’ and its relatives ‘m’ and ‘b’ are modified to set \texttt{\adl@box} to the box for them.

\texttt{\@mkpream} To make the preamble shown above, \texttt{\@mkpream} is modified to \texttt{\let} control sequences \texttt{\adl@colhtdp, \adl@vlineL} and \texttt{\adl@vlineR} be \texttt{\relax} in order to keep them from being expanded by \texttt{\edef/\xdef} for the preamble construction. The control sequences
\texttt{\adl@startmbox} and \texttt{\adl@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 after this final \texttt{\afterassignment} to make \texttt{\@preamble}.

With \texttt{array}, on the other hand, calling \texttt{\adl@preaminit} is safe because \texttt{\@mkpream} is followed by \texttt{\xdef} 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.

112 \% Making Preamble
113 \let\adl@mkpream\@mkpream
114 \def\@mkpream#1{\let\adl@colhtdp\relax
115 \let\adl@vlineL\relax \let\adl@vlineR\relax
116 \let\adl@startmbox\relax \let\adl@endmbox\relax
117 \global\adl@ncol\@ne \adl@leftruletrue
118 \adl@mkpream{#1}\adl@columns\adl@ncol \afterassignment\adl@preaminit}
119 \@addamp

The macro \texttt{\@addamp} is also modified to add the code for incrementing the counter \texttt{\adl@currentcolumn} to \texttt{\@preamble} with \texttt{&}. The counter \texttt{\adl@ncol} is also incremented by \texttt{\global} \texttt{\@addamp} so that we can refer to its value as \texttt{⟨\texttt{c}⟩} of \texttt{\adl@vlineL/R}. 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}.

122 \def\@addamp{\if\@firstamp\@firstampfalse \else
123 \@addtopreamble{\global\advance\adl@currentcolumn\@ne &}\%}
124 \global\advance\adl@ncol\@ne \fi}
125

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 \texttt{ET\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.

\@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 \add@putlrc, and to set \add@box to the box for ‘p’ and its relatives. Note that a m-type preamble (\chnum = 3) has to be generated to have \add@startmbox and \add@endmbox in it because a \vcenter construct cannot be assigned to \add@box by \setbox directly.
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 `;' as explained later. In the case with \texttt{array}, the former is class-4 and we use `G' (class-7) for the latter.

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

\textbf{Without array}

\begin{verbatim}
\def\testpach#1{\ifcase \@lastchclass \def\@preamble{$\relax\@sharp$}\or \let\@arrayrule\adl@arrayrule \or \let\@arrayrule\adl@arraydashrule \or \let\@arrayrule\adl@argarraydashrule \or \gdef\preamble{\fi} \fi \fi \fi \fi \fi \fi \fi \fi \fi \fi \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\arrayclass{\ifcase \@lastchclass \or \def\@preamble{\ifcase \@chnum \hfil}$
\fi}\or \fi \fi \fi \fi \fi \fi \fi \fi \fi \fi \fi \fi}
\end{verbatim}
In \LaTeX, the beginning of preamble is class-6 and we use `p` (class-3) to get the argument of `;`.

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@arraydashrule{(cL)}{(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 = \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({(cR)}){(d)}/{(g)}} if \texttt{\ifadl@leftrule} is true indicating the vertical line will appear at the left edge of the column (\texttt{cL}), or \texttt{\adl@vlineR({(cR)}){(d)}/{(g)}} 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 \S 4.7. Also note that \texttt{(cR)} and \texttt{(d)} 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.

The argument of `;` is not provided by \texttt{\adl@argarraydashrule} but is directly passed from the preamble text through \texttt{\@nextchar}. This direct passing is implemented by the following trick. The macro \texttt{\adl@argarraydashrule} set \texttt{\@chclass} to \texttt{\adl@class@iiiorvii} to
pretend it is for ‘p’ if array is not in use, or ‘∅’ otherwise. Then it temporally changes the
definition of \classv, which is incidentally for the argument of ‘p’ and ‘∅’ in the case
without/with array respectively, to \adl@classvfordash to process the argument of ‘;’
rather than that of ‘p’ or ‘∅’. Then \adl@classvfordash is invoked by \@mkpream and it
adds the argument to \@preamble. Finally, it restores the definition of \classv and sets
\chclass to 1 to indicate that the last item is a vertical line specification.

207 \def\adl@arrayrule{%
208 \adl@xarraydashrule
209 \@ne{\adl@ncol}{\z@/\z@}}
210 \def\adl@arraydashrule{%
211 \adl@xarraydashrule
212 {\dashlinedash/\dashlinegap}}
213 \def\adl@argarraydashrule{%
214 \adl@xarraydashrule
215 {\@ne}{\adl@ncol}{}
216 \@chclass\adl@class@iiiorvii \let\@classv\adl@classvfordash}
217 \def\adl@xarraydashrule#1#2#3{%
218 \ifnum\@lastchclass=\adl@class@start\else
219 \ifnum\@lastchclass=\@ne\else
220 \adl@leftrulefalse \fi\fi
221 \ifadl@zwvrule\else \ifadl@inactive\else
222 \@addtopreamble{\vrule\@width\arrayrulewidth
223 \@height\z@ \@depth\z@}\fi \fi
224 \ifadl@leftrule
225 \@addtopreamble{\adl@vlineL{\CT@arc@}{\adl@dashgapcolor}\
226 \number#1#3}\
227 \else \@addtopreamble{\adl@vlineR{\CT@arc@}{\adl@dashgapcolor}\
228 {\number#2#3}}\fi}
229 \let\adl@classv\@classv
230 \def\adl@classvfordash{\@addtopreamble{\@nextchar}}\let\@classv\adl@classv
231 \@chclass\@ne}
232 \adl@preaminit
233 \adl@colhtdp
234 \adl@vlineL
235 \adl@vlineR

4.6 Building Columns

\adl@preaminit\adl@colhtdp\adl@vlineL\adl@vlineR

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

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

\begin{verbatim}
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}
\end{verbatim}

The macro \texttt{\adl@@vlineL} \((\gamma_d)\langle \Gamma_g \rangle \langle c \rangle \{ \langle d \rangle \langle g \rangle \langle \gamma_d \rangle \langle \gamma_g \rangle \}) adds the element \(e = \langle c, d, g \rangle = \texttt{\elt}\{\langle \rangle\}\{\langle d \rangle\}\{\langle g \rangle\}\{\langle \gamma_d \rangle\}\{\langle \gamma_g \rangle\}\) to the tail of the list \texttt{\adl@colsL} to construct \(C^L_i\), where \(\gamma_d\) and \(\gamma_g\) are the color specifications given by \texttt{\color} macros in \(\Gamma_d\) and \(\Gamma_g\). The macro \texttt{\add@@vlineR} performs similar operation but the element is added to the head of \texttt{\adl@colsR} for \(C^R_i\) because it is processed right-to-left manner. The argument \((d)\) and \((g)\) are extracted by the macro \texttt{\adl@ivline} which converts given dimensional values of them to integers. It also sets \((d)\) and \((g)\) 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 \texttt{\adl@setcolor} to define \texttt{\adl@dashcolor} and \texttt{\adl@gapcolor} with the color specification of \(\Gamma_d\) and \(\Gamma_g\). Since \texttt{\adl@setcolor} locally expands \texttt{\color} macro in \(\Gamma_d\) and \(\Gamma_g\) to define \texttt{\current@color} that becomes the body of \texttt{\adl@dashcolor} \((\gamma_d)\) and \texttt{\adl@gapcolor} \((\gamma_g)\) with expansion, different \texttt{\color} specifications of a color, such as \texttt{\color{red}} and \texttt{\color[rgb]{1,0,0}}, will produce a unified result such as \texttt{\color[rgb]{1 0 0}}. If \(\Gamma_d\) or \(\Gamma_g\) is \texttt{\relax} which is the body of \texttt{\adl@nocolor}, \(\gamma_d\) or \(\gamma_g\) is also \texttt{\relax} to indicate dashes are colored (or not colored) as done in outer world and gaps are transparent.

\begin{verbatim}
def\adl@@vlineL#1#2#3#4{\adl@ivline#4\@nil{#1}{#2}{\number\@tempcnta}{\number\@tempcntb}{\adl@dashcolor}{\adl@gapcolor}}
\end{verbatim}

\begin{verbatim}
def\adl@@vlineR#1#2#3#4{\adl@ivline#4\@nil{#1}{#2}{\number\@tempcnta}{\number\@tempcntb}{\adl@dashcolor}{\adl@gapcolor}}
\end{verbatim}

After \texttt{\adl@@colhtdp}, \texttt{\adl@@vlineL} and \texttt{\adl@@vlineR} are defined, we call \texttt{\adl@preaminit} to let their single @ counterparts be equal to them. Therefore, in the case with array, \texttt{\adl@@colhtdp} etc. are temporarily \texttt{\relax} when \texttt{\@preamble} is being generated.
in the group of \@mkpream, and regain their own definitions outside the group where the completed \@preamble is referred to.

263 \adl@preaminit

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

265 \def\adl@inactivevl#1#2#3#4{\ifadl@zwvrule 
266 {#1\vline}\ifadl@zwvrule 
267 \fi}

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

The macros to make \parbox for ‘p’ (and ‘b’ of array), \@@startpbox and \@@endpbox, are modified for height/depth measurement. The code for \@@endpbox is based on that of \LaTeX{} 2e to fix the bug of \texttt{strut}-ing in \LaTeX{}-2.09, but \@finalstrut is manually expanded because it is not available in \LaTeX{}-2.09. In array, \@@endpbox is not used but \@endpbox is. Therefore, we let them be equal. As for \@@startpbox, however, we may not worry about it because we have modified \@classz in §4.5 for the measurement. However, we have to take care of m-type columns specially because its body \texttt{vcenter} cannot be assigned directly to \adl@box by \setbox. Thus we enclose a \texttt{vcenter(...)}$\$ construct in a \texttt{hbox} and assign it to \adl@box. The macro \adl@startmbox opens the construct with array’s \@@startpbox, while \adl@endmbox closes it calling \adl@org@endpbox which is the unmodified \@endpbox of array and measures the height and depth of the \texttt{hbox} by \adl@colhtdp.

268 \def\@@startpbox#1{\setbox\adl@box\vtop\bgroup 
269 \hspace{1\arrayparboxrestore}
270 \unskip \iftex \nobreak
271 \vrule\@width\zero@\@height\zero@\@depth\dp\@arstrutbox \fi
272 \par \egroup \setbox\adl@colhtdp \box\adl@box \hfil}
273 \let\@endpbox\@@endpbox
274 \def\adl@endmbox{\bgroup \$\vcenter\@startpbox}
275 \adl@org@endpbox \$\egroup \adl@colhtdp \box\adl@box \hfil}
276 \vfill

4.7 Multi-columns

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

\begin{itemize}
\item The macros to construct the parts of \@preamble for vertical lines, \adl@arrayrule, \adl@arraydashrule and \adl@argarraydashrule, 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 array or tabular which may occur in the third argument of \texttt{\multicolumn}. In order to make \@preamble work well outside of the group
\end{itemize}

\footnote{The author had forgotten this fact until Morten Høgholm pointed out it. Thanks Morten.}
containing \makepreamble, \adl@preamble is \global-ly \let-equal to \@preamble just after \makepreamble in the group and then reverse \let-assignment is performed just after the group is closed. These global assignment is unnecessary with array because \@preamble is constructed \global-ly, but safe.

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

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

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

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

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

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

Note that \adl@columns is modified by \makepreamble, but it is not referred to by \adl@mcarrayrule etc., and its value is restored before referred to by \hdashline, etc.

```
277 \% Multi-Columns
278 \def\multicolumn#1#2#3{\multispan{#1}\begingroup \begingroup
279 \def\adl@arrayrule{\adl@mcarrayrule{#1}}% 280 \def\adl@arraydashrule{\adl@mcarraydashrule{#1}}% 281 \def\adl@argarraydashrule{\adl@mcargarraydashrule{#1}}% 282 \let\@addamp\adl@mcaddamp 283 \@mkpream{#2}\@addtopreamble\@empty 284 \global\let\adl@preamble\@preamble\endgroup 285 \let\@preamble\adl@preamble 286 \def\@sharp{#3}\let\protect\relax 287 \adl@activatepbox 288 \@arstrut\@preamble\hbox{}\endgroup 289 \global\advance\adl@currentcolumn#1\ignorespaces}
290 \ifadl@usingarypkg
291 \def\adl@mcaddamp{\if@firstamp\@firstampfalse \else\preambleerror5\fi}
292 \let\adl@activatepbox\relax
293 \else
294 \@arstrut\@preamble\hbox{}\endgroup
295 \global\advance\adl@currentcolumn#1\ignorespaces}
296 \ifadl@usingarypkg
297 \def\adl@mcaddamp{\if@firstamp\@firstampfalse \else\preamerror5\fi}
298 \let\adl@activatepbox\relax
299 \else
300 \@arstrut\@preamble\hbox{}\endgroup
301 \global\advance\adl@currentcolumn#1\ignorespaces}
```
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 \span-ned. 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@xarraydashrule are:

\[
\begin{align*}
    c_L &= c, \\
    c_R &= c + n - 1
\end{align*}
\]

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

---

\% End of Rows

\begin{verbatim}
\def\adl@mcarrayrule#1\{\adl@tempcnta#1\advance\adl@tempcnta\adl@currentcolumn
\advance\adl@tempcnta\m@ne
\adl@xarraydashrule
\adl@mcarraydashrule\adl@mcargarraydashrule
\end{verbatim}

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, \\cr-s in the macros \adl@xarraycr, \adl@xtabularcr, \adl@xarraycr and \adl@xarraycr are replaced with our own \adl@cr. The macro \adl@xarraycr\langle dimen \rangle is also modified but its \cr is replaced with \adl@argcr\langle dimen \rangle to add (negative) \dimen to \( h_i \). Note that \adl@xarraycr\langle dimen \rangle uses ordinary \adl@cr because the extra vertical space of \langle dimen \rangle is inserted to the last column.

Note that the implementation of \adl@xarraycr is slightly different between \LaTeX{} and \array, we have to have two versions and choose one.
The macro \adl@cr and \adl@argcr perform \cr and then invoke the common macro \adl@@cr. The argument \langle x \rangle is the extra (negative) vertical space for \adl@argcr, while it is 0 for \adl@cr.

\adl@cr The macro \adl@cr and \adl@argcr perform \cr and then invoke the common macro \adl@@cr. The argument \langle x \rangle is the extra (negative) vertical space for \adl@argcr, while it is 0 for \adl@cr.

\adl@argcr The macro \adl@argcr(x) at first calculate \( h_i \) as follows. The registers \adl@height = \( \eta \) and \adl@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 \@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 \langle x \rangle is negative, a vertical space of \( x \) is inserted below the row. Thus the integer value of \( h_i + x \) is \global-ly added to \adl@totalheight, and the elements \( \langle C^L_i = \adl@colsL, h_i \rangle \) and \( \langle C^R_i = \adl@colsR, h_i \rangle \) are added to the tail of \( R^L = \adl@rowsL \) and \( R^C = \adl@rowsR \). If \( x \) is not 0 (negative), discard(\( h_i \)) or connect(\( h_i \)) is also added after \( \langle C^L_i, h_i \rangle \) or \( \langle C^R_i, h_i \rangle \) according to the current environment (\longtable or not). In the real implementation, \( R^L \) and \( R^C \) has the following format of \langle rows \rangle.

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

Since \adl@discard is \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 \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.

\footnote{Before v1.54, negative \langle x \rangle 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 array/tabular.}
The macro \hline is modified to insert \vskip-\arrayrulewidth before drawing if \ADLnullwidehline is in effect, or to add the element \textit{connect}(w) = (\adl@connect/ \number\arrayrulewidth) to the end of \textit{R}^L and \textit{R}^R 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.

\% Horizontal Lines

The macro \hline is modified to insert \vskip-\arrayrulewidth before drawing if \ADLnullwidehline is in effect, or to add the element connect(w) = (\adl@connect/ \number\arrayrulewidth) to the end of \textit{R}^L and \textit{R}^R 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.
\dashline
\adl@dashline
\adl@dashline

The macro \dashline calls \adl@dashline to open the \noalign construct by the well-known trick {\ifnum0='}\fi and then to invoke \adl@dashline checking the existence of its optional argument \[(\text{dash})/(\text{gap})\]. Before the invocation, it inserts \vskip-\arrayrulewidth if \ADLnullwide is in effect, or adds connect\(w\) to the end of \(R^L\) and \(R^R\). Then \adl@dashline closes the \noalign by \ifnum0='\fi to start the pseudo row for the horizontal dash-line. Before the dash-line is drawn by \adl@dashline 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.

\adl@inactivehdl

If \ADLinactive is in effect, \adl@dashline is \let equal to \adl@inactivehdl. This macro simply puts a \hrule discarding its arguments after inserting \vskip -\arrayrulewidth if \ADLnullwide is in effect.

\adl@xhline

The macro \adl@xhline is the counterpart of the original \@xhline. This is introduced to check the mixed sequence of \hline and \dashline, and to add the element \(\text{disconnect} = (\text{relax}/\text{doublerulesep})\) to the end of \(R^L\) and \(R^R\) by \adl@dashline if a pair of \h(dash)line is found.

\adl@dashline

The macro \adl@dashline\(\langle cs\rangle\langle dimen\rangle\) \global ly adds the integer value of \langle dimen\rangle to \adl@totalheight and adds the element \((\langle cs\rangle/\number\langle dimen\rangle)\) to the tail of \(R^L\) and \(R^R\). The arguments \langle cs\rangle\langle dimen\rangle are \adl@connect\arrayrulewidth for connect\(w\) or \relax\doublerulesep for disconnect\(s\).
The macro \cdashline at first opens \noalign and then invokes \adl@cdline checking the existence of its optional argument \[(dash)/(gap)]. The macro \adl@cdline first inserts \vskip-\arrayrulewidth if \ADLnullwidehline is in effect. Then it performs column \span-ing by the code based on that of \@cline in \LaTeX-2.09 because \LaTeX 2ε’s version will not work with \LaTeX-2.09. The main job is done by \adl@hcline after the target columns are \spanned by \adl@cdlinea or \adl@cdlineb.

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

\adl@hcline The macro \adl@hcline{\[d\]/\(g\)]} 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 \(w\), \(\neg -\arrayrulewidth\) for \cdashline, is inserted if it is not 0 (for \hdashline) and \ADLnullwidehline is not in effect.

\firstdashline If \array is in use, we wish to have dashed counterparts of \first/lasthline named \firstdashline/lastdashline, which simply call \adl@hdashline with an argument to call \adl@first/lastdashline after closing \noalign opened by \adl@hdashline.
The macros \texttt{\adl@first}/\texttt{lastdashline}, however, are defined in a tricky manner to replace \texttt{\hline} in \texttt{\first}/\texttt{lasthline} with:

\begin{verbatim}
\adl@dashline\adl@ihdashline[⟨dash]/⟨gap⟩]
\end{verbatim}

in order to avoid copy-and-replace. To do that, we define \texttt{\adl@defflhdl} and \texttt{\adl@idefflhdl} in which the body of \texttt{\first/lasthline} is expanded by \texttt{\expandafter} and the parts preceding and following \texttt{\hline} are extracted. Then the preceding part \((p)\), the calling sequence of \texttt{\adl@dashline}, and the following part \((f)\) are connected to be the body of \texttt{\adl@first/lastdashline}. Thus we define \texttt{\adl@firstdashline} as follows.

\begin{verbatim}
def\adl@firstdashline[#1/#2]{%
  ⟨p⟩
adl@dashline\adl@ihdashline[#1/#2]
  ⟨f⟩}%
\end{verbatim}

\begin{verbatim}
\if\adl@usingarypkg\def\firstdashline{\adl@dashline{\ifnum0=′{\fi}\adl@firstdashline}}\def\lastdashline{\adl@dashline{\ifnum0=′{\fi}\adl@lastdashline}}\fi
\end{verbatim}

\begin{verbatim}
def\adl@defflhdl#1\hline#2\@nil{\@namedef{#1}[##1/##2]{#1\adl@dashline\adl@ihdashline[#1/##2]#2}}
\adl@defflhdl{\adl@firstdashline}\firsthline\@nil
\adl@defflhdl{\adl@lastdashline}\lasthline\@nil
\fi
\end{verbatim}

%% End of Environment

\subsection*{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}
\def\endarray{\adl@endarray \egroup \adl@arrayrestore \egroup
  \csname @arrayright\endcsname}
\expandafter\let\csname endtabular\endcsname\endtabular
\end{verbatim}
The macro `\endarray` at first closes the last row by `\crcr`. If this `\crcr` has real effect, we have to invoke `\@cr` to perform our own end-of-row operations. We assume that the `\crcr` is effective if either `\@height` or `\@depth` has a non-zero value\(^{17}\).

Then 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 `\@vlrowL` and `\@vlrowR` by `\@makevlrL` and `\@makevlrR`. In the real implementation, \(L_k = (\gamma_k, \pi_k, \delta_k, \tau_k, \beta_k)\) is represented as:

\[
\text{\@vl}{\beta_k} - \delta_k \gamma_k \{\xi_k\}.
\]

Thus `\@vl` is made \let-equal to `\relax` when the rows are constructed and to `\@vl` when the rows are put.

Since `\@makevlrL` and `\@makevlrR` shares common macros, they conceptually have the following interface.

\[
\text{\@vlrow} = \text{\@makevlrL/R} (\text{\@rows}, \text{\@currentcolumn}, \text{\@addvl})
\]

Thus they are invoked as;

\[
\text{\@vlrowL} = \text{\@makevlrL} (\text{\@rowsL}, 1, \text{\@addvlL})
\]
\[
\text{\@vlrowR} = \text{\@makevlrR} (\text{\@rowsR}, \text{\@columns}, \text{\@addvlR})
\]

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

\(^{17}\)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}.

\begin{verbatim}
\def\adl@arrayrestore{\%}
  \global\adl@height\adl@heightsave
  \global\adl@depth\adl@depthsave
  \global\adl@currentcolumn\adl@currentcolumnsave
  \global\adl@totalheight\adl@totalheightsave
  \global\let\adl@rowsL\adl@rowsLsave
  \global\let\adl@rowsR\adl@rowsRsave
  \global\let\adl@colsL\adl@colsLsave
  \global\let\adl@colsR\adl@colsRsave}
\end{verbatim}

\section*{4.11 Drawing Vertical Lines}

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

\begin{verbatim}
\adl@makevlrL \adl@makevlrR \adl@makevlrL \adl@makevlrR
\end{verbatim}

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:

\begin{itemize}
  \item (1) \texttt{A} \leftarrow \langle \rangle; \texttt{R} \leftarrow \texttt{R}; \texttt{\gamma} \leftarrow \texttt{\Gamma};
  \item (2) \textbf{while} \texttt{\gamma} > 0 \textbf{do begin}
\end{itemize}

and (31)–(36) are:

\begin{itemize}
  \item (31) \textbf{if} \texttt{double} \textbf{then} \texttt{A} \leftarrow \langle \texttt{\hskip\doublerulesep, A} \rangle;
  \item (32) \textbf{else begin}
  \item (33) \texttt{\gamma} \leftarrow \texttt{\gamma} - 1;
  \item (34) \textbf{if} \texttt{\gamma} = 0 \textbf{then} \texttt{A} \leftarrow \langle \texttt{\hss, A} \rangle;
  \item (35) \textbf{else} \texttt{A} \leftarrow \langle \texttt{\omit\hss, A} \rangle;
  \item (36) \textbf{end};
\end{itemize}
Λ ← \{\}; R ← R^L; γ ← 1;

while γ ≤ Γ do begin
    τ ← H; β ← H; η ← 0; δ ← \langle -1, \bot \rangle; ξ ← \langle -1, \bot \rangle;
    conn ← false; double ← false; R' ← \{}
    while R ≠ \{} do begin
        ⟨r, R⟩ ← R;
        ⟨C, h⟩ ← r;
        if C = \{} then begin add(τ, β, δ, ξ); η ← 0; end;
        elseif C = \langle connect \rangle then η ← h;
        else begin
            ⟨e, C'⟩ = C; ⟨c, d, g⟩ = e;
            if c = γ then begin
                if d = δ ∧ g = ξ then begin
                    τ ← β + η; conn ← true;
                end;
            end;
            else begin
                add(τ, β, δ, ξ);
                δ ← d; ξ ← g; τ ← β + η; conn ← true;
            end;
            if C' = \langle (γ, ?, ?), ? \rangle then double ← true;
            C ← C';
        end;
        else add(τ, β, δ, ξ);
        η ← 0;
    end;
    β ← β - h; R' ← \{R', C, h\} \}
end;
add(τ, β, δ, ξ); R ← R';
if double then Λ ← \{Λ, \hskip\doublerulesep\}
else begin
    γ ← γ + 1;
    if γ > Γ then Λ ← \{Λ, \hfil\}
    else Λ ← \{Λ, \hfil\}
end;
procedure add(τ, β, δ, ξ) begin
    if conn then begin
        Λ ← \{Λ, \langle β, τ - β, δ, ξ \rangle\}; conn ← false;
    end;
end;

Figure 2: Conceptual Code of adl@makevlrL
The macro \texttt{\adl@makevlr} corresponds to the lines (3)–(4) and (30).

\begin{enumerate}
\item \texttt{\adl@makevlr} expands $C$ attaching the sentinel \texttt{\adl@endmakevlr}. Thus the lines (10)–(21) and (25)–(26) are performed by \texttt{\adl@iimakevlr}.
\end{enumerate}
Then:

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

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

2. If \( C = \langle \text{connect} \rangle \), 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 \( \texttt{adl@connect}(h) \). The macro performs (9) and implicit \( C \leftarrow C (= \langle \text{connect} \rangle) \) eating the sentinel.

3. If \( C = \emptyset \), \( \texttt{adl@endmakevlrconn} \) that is \( \texttt{let} \)-equal to the sentinel \( \texttt{adl@endmakevlr} \) is invoked to perform (8) and implicit \( C \leftarrow C (= \emptyset) \).

509 \texttt{def \textbackslash adl@iiimakevlr#1#2#3#4#5{\let \textbackslash elt \textbackslash adl@ivmakevlr \let \textbackslash next/relax}
510 \texttt{\ifnum#1=\textbackslash adl@currentcolumn/relax}
511 \texttt{\let\textbackslash adl@endmakevlr\textbackslash adl@endmakevlrconn}
512 \texttt{\@tempswafalse}
513 \texttt{\ifnum#2=\textbackslash adl@dash/relax}
514 \texttt{\ifnum#3=\textbackslash adl@gap/relax}
515 \texttt{\def\@tempa{#4}\ifx\@tempa\textbackslash adl@dashcolor}
516 \texttt{\def\@tempa{#5}\ifx\@tempa\textbackslash adl@gapcolor}
517 \texttt{\@tempswatrue}
518 \texttt{\fi\fi/relax}
519 \texttt{\if@tempswa}
520 \texttt{\if\texttt{adl@connected/else}}
521 \texttt{\@tempcnta@tempcntb}
522 \texttt{\advance@tempcnta\textbackslash adl@lastconn/relax}
523 \texttt{\textbackslash adl@connectedtrue}
524 \texttt{\else}
525 \texttt{\fi}
526 \texttt{\else}
527 \texttt{\textbackslash adl@addvl}
528 \texttt{\textbackslash adl@dash\mc/relax \textbackslash adl@gap\mc/relax}
529 \texttt{\def\textbackslash adl@dashcolor\mc/\textbackslash adl@gapcolor\mc/}
530 \texttt{\@tempcnta@tempcntb}
531 \texttt{\advance\@tempcnta\textbackslash adl@lastconn/relax}
532 \texttt{\textbackslash adl@connectedtrue}
533 \texttt{\else}
534 \texttt{\fi}
535 \texttt{\textbackslash adl@addvl}
536 \texttt{\textbackslash adl@vmakevlr\mc/\textbackslash elt\gb{#1}{#2}{#3}{#4}{#5}}
537 \texttt{\if\textbackslash adl@lastconn\gb{z}{next}}
538 \texttt{\textbackslash adl@vmakevlr\mc/\textbackslash elt\gb{#1}}
539 \texttt{\if\textbackslash adl@currentcolumn \textbackslash adl@doubleruletrue \textbackslash fi}
540 \texttt{\textbackslash adl@vmakevlr\mc/\textbackslash elt\gb{#1}}}
The macro \adl@addvlL corresponds to the lines (38)–(42), i.e. the procedure \verb|add|. The macro \adl@addvlR performs similar operations, but its conceptual code is the following.

\begin{verbatim}
procedure add(\tau,\beta,\delta,\xi) begin
  if conn then begin
    \Lambda \leftarrow \langle\beta,\tau-\beta,\delta,\xi\rangle,\Lambda
    conn \leftarrow false;
  end;
end;
\end{verbatim}

After the macros \adl@vlrowL and \adl@vlrowR are constructed, they are expanded to draw vertical lines by \adl@drawvl. Prior to the expansion, the macro \adl@drawvl globally defines \adl@vl@leftskip and \adl@vl@rightskip, which are the amount of negative spaces inserted to the left/right of a vertical line, as follows.

\begin{verbatim}
\adl@vl@leftskip = \begin{cases} \arrayrulewidth/2 & \text{if } \text{\adl@zrule} \\ 0 & \text{otherwise} \end{cases}
\end{verbatim}

\begin{verbatim}
\adl@vl@rightskip = \begin{cases} \arrayrulewidth/2 & \text{if } \text{\adl@zrule} \\ 0 & \text{otherwise} \end{cases}
\end{verbatim}

That is, if \ADLnulwide is in effect, a vertical line is surrounded by horizontal spaces of \verb-\arrayrulewidth/2- to adjust the center of the line to the left or right edge of its column. Otherwise, a horizontal space \verb-\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 \ADLsomewide and thus disconnected lines were not aligned vertically.}.
Then the macros \texttt{\textbackslash\textbackslash vlrow\textbackslash l} and \texttt{\textbackslash\textbackslash vlrow\textbackslash r} are expanded. These macros will have \texttt{\textbackslash vl}, which is made \texttt{\textbackslash let\textbackslash -\textbackslash equal\textbackslash to\textbackslash \textbackslash@@ vl} prior to the expansion, to draw a vertical line. The macro \texttt{\textbackslash vl (β\textbackslash (λ) (δ\textbackslash l) (γ\textbackslash l) (δ\textbackslash c) (γ\textbackslash c) \langle x\textbackslash l \textbackslash and\textbackslash x\textbackslash c \textbackslash are\textbackslash length\textbackslash and\textbackslash color\textbackslash ) draws\textbackslash a\textbackslash solid\textbackslash line\textbackslash if\textbackslash γ\textbackslash l = 0 or a dash\textbackslash line\textbackslash otherwise\textbackslash in\textbackslash a\textbackslash \textbackslash vbox\textbackslash of\textbackslash λ = τ − β \textbackslash high\textbackslash and\textbackslash \textbackslash raise\textbackslash s\textbackslash it\textbackslash by\textbackslash β. The method to draw a dash line in the \texttt{\textbackslash vbox\textbackslash is\textbackslash analogous\textbackslash to\textbackslash that\textbackslash for\textbackslash horizontal\textbackslash line shown\textbackslash in\textbackslash §4.9, except\textbackslash that\textbackslash a\textbackslash line\textbackslash is\textbackslash surrounded\textbackslash by\textbackslash horizontal\textbackslash spaces\textbackslash of\textbackslash \texttt{\textbackslash vl\textbackslash leftskip\textbackslash and\textbackslash \texttt{\textbackslash vl\textbackslash rightskip\textbackslash . Coloring\textbackslash gaps\textbackslash is\textbackslash done\textbackslash by\textbackslash drawing\textbackslash a\textbackslash vertical\textbackslash rule\textbackslash setting\textbackslash γ\textbackslash c by\textbackslash \texttt{\textbackslash set\textbackslash color\textbackslash prior\textbackslash to\textbackslash dash\textbackslash line\textbackslash drawing\textbackslash if\textbackslash γ\textbackslash c is\textbackslash not\textbackslash \texttt{\textbackslash relax\textbackslash To\textbackslash color\textbackslash dashes\textbackslash or\textbackslash solid\textbackslash line, \texttt{\textbackslash set\textbackslash color\textbackslash with\textbackslash δ\textbackslash c is\textbackslash done\textbackslash if\textbackslash it\textbackslash is\textbackslash not\textbackslash \texttt{\textbackslash relax\textbackslash before\textbackslash line\textbackslash drawing.}

\begin{verbatim}
561 \def\adl@drawvl{\
562 \omit \relax \if\adl@zwvrule
563 \gdef\adl@vl@leftskip{.5\arrayrulewidth}\
564 \else \global\let\adl@vl@leftskip\adl@vl@rightskip
565 \global\let\adl@vl@rightskip\arrayrulewidth
566 \fi \adl@vlrowL \cr
567 \omit \relax \if\adl@zwvrule
568 \gdef\adl@vl@leftskip{.5\arrayrulewidth}\
569 \else \global\let\adl@vl@leftskip\adl@vl@rightskip
570 \global\let\adl@vl@rightskip\arrayrulewidth
571 \fi \adl@vlrowR \cr
572 \def\adl@@vl#1#2#3#4#5#6{\vbox to\z@{\vss\hbox{\
573 \hskip-\adl@vl@leftskip
574 \ifnum#3=\z@\else \def\@tempa{#6}\ifx\@tempa\adl@nocolor\else
575 \raise#1sp\hbox{\let\current@color\@tempa \set@color
576 \vrule height#2sp width\arrayrulewidth}\
577 \hskip-\arrayrulewidth \fi \fi
578 \raise#1sp\vbox to#2sp{\
579 \def\@tempa{#5}\ifx\@tempa\adl@nocolor\else
580 \hskip-\arrayrulewidth \fi \fi
581 \hrule height#2sp depth\z@ width\arrayrulewidth
582 \else \vbox{\def\@tempdima{#3}\let\tempdima\@tempdima
583 \hrule height#2sp depth#3\space width\arrayrulewidth
584 \else \vbox{\def\@tempdima{#3}\let\tempdima\@tempdima
585 \def\@tempdimb{#4}\let\tempdimb\@tempdimb
586 \adl@draw\adl@hrule\vskip\vbox{\
587 \hskip-\adl@vl@rightskip}}}}

4.12 Drawing Dash-lines
\texttt{\textbackslash vl\textbackslash rule\textbackslash \textbackslash hrule} As explained later, horizontal and vertical lines are drawn by a common macro \texttt{\textbackslash vl\textbackslash to\textbackslash which\textbackslash the\textbackslash length\textbackslash of\textbackslash a\textbackslash dash\textbackslash segment,\textbackslash d,\textbackslash is\textbackslash passed\textbackslash through\textbackslash \texttt{\textbackslash set\textbackslash dima. The\textbackslash macro\textbackslash also\textbackslash has\textbackslash an\textbackslash argument\textbackslash that\textbackslash is\textbackslash either\texttt{\textbackslash vl\textbackslash rule\textbackslash to\textbackslash draw\textbackslash a\textbackslash dash\textbackslash for\textbackslash horizontal\textbackslash lines\textbackslash or\texttt{\textbackslash vl\textbackslash hrule\textbackslash for\textbackslash vertical.\textbackslash These\textbackslash two\textbackslash macros\textbackslash commonly\textbackslash have\textbackslash one\textbackslash argument\textbackslash \langle f\textbackslash \rangle\textbackslash to\textbackslash draw\textbackslash a\textbackslash dash\textbackslash of\textbackslash f × d\textbackslash long\textbackslash and\textbackslash of\texttt{\textbackslash arrayrulewidth\textbackslash wide.\end{verbatim}}

43
The macro \adl@draw is to draw a horizontal or vertical line. It is \let-equal to one of \adl@drawi, \adl@drawii and \adl@drawiii according to the drawing mode specified by \ADLdrawingmode. These three macros have common interface, \@tempdimb for the length of dash and gap, \(d\) and \(g\), and three arguments ⟨rule⟩, ⟨skip⟩ and ⟨box⟩ with which \adl@draw is called in the following manner.

\adl@draw\adl@vrule\hskip\vbox ... horizontal 
\adl@draw\adl@hrule\vskip\vbox ... vertical

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

\[
\langle \text{rule} \rangle \{1/2\} \ (\text{skip}) \{g/2\} \langle \text{box} \rangle \{ \langle \text{skip} \rangle \{g/2\} \langle \text{rule} \rangle \{1\} \ (\text{skip}) \{g/2\} \langle \text{skip} \rangle \{0\plus\text{fil}\minus\text{fil}\} \langle \text{skip} \rangle \{g/2\} \langle \text{rule} \rangle \{1/2\} \langle \text{skip} \rangle \{g/2\} \langle \text{box} \rangle \{ \langle \text{skip} \rangle \{g/2\} \langle \text{rule} \rangle \{1\} \ (\text{skip}) \{g/2\} \langle \text{skip} \rangle \{0\plus\text{fil}\minus\text{fil}\} \langle \text{skip} \rangle \{g/2\} \langle \text{rule} \rangle \{1/2\}\rangle
\]

The conceptual operations of \adl@drawii for mode 2 are as follows.

\[
\langle \text{rule} \rangle \{1/2\} \ (\text{skip}) \{g/2\} \langle \text{box} \rangle \{ \langle \text{skip} \rangle \{g/2\} \langle \text{rule} \rangle \{1\} \ (\text{skip}) \{g/2\} \langle \text{skip} \rangle \{0\plus\text{fil}\minus\text{fil}\} \langle \text{skip} \rangle \{-d-g\} \langle \text{box} \rangle \{ \langle \text{skip} \rangle \{g/2\} \langle \text{rule} \rangle \{1\} \ (\text{skip}) \{g/2\} \langle \text{skip} \rangle \{-d-g\} \langle \text{box} \rangle \{ \langle \text{skip} \rangle \{g/2\} \langle \text{rule} \rangle \{1\} \ (\text{skip}) \{g/2\} \langle \text{skip} \rangle \{-d-g\} \langle \text{box} \rangle \{ \langle \text{skip} \rangle \{g/2\} \langle \text{rule} \rangle \{1\} \ (\text{skip}) \{g/2\} \langle \text{skip} \rangle \{-d-g\} \langle \text{box} \rangle \{ \langle \text{skip} \rangle \{g/2\} \langle \text{rule} \rangle \{1\} \ (\text{skip}) \{g/2\} \langle \text{skip} \rangle \{-d-g\}\rangle
\]

The macro \adl@drawiii for mode 3 is quite similar to \adl@drawi except that \xleaders is replaced by \cleaders. This replacement is done by temporarily \let-ing \xleaders be equal to \cleaders.
The macro \ADLdrawingmode{⟨m⟩} defines the drawing mode by letting \adl@draw be equal to \adl@drawi if \( m = 1 \), and so on. If \( ⟨m⟩ \) is neither 1, 2 nor 3, it is assumed as 1.

\begin{verbatim}
612 \def\ADLdrawingmode#1{\ifcase #1%
613 \let\adl@draw\adl@drawi \or
614 \let\adl@draw\adl@drawi \or
615 \let\adl@draw\adl@drawii \or
616 \let\adl@draw\adl@drawiii \else
617 \let\adl@draw\adl@drawi \fi}
618
619 \end{verbatim}

### 4.13 Shorthand Activation

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

\begin{verbatim}
620 \def\adl@Array\{\adl@inactivefalse \array\}
621 \def\adl@Tabular\{\adl@inactivefalse \tabular\}
622 \def\adl@Tabularstar\{\adl@inactivefalse \@nameuse{tabular*}\}
623 \def\adl@Longtable\{\adl@inactivefalse \longtable\}
624
\end{verbatim}

\begin{verbatim}
630 \ifx\longtable\undefined\else \@ifdefinable\Longtable\relax \fi
631 \endgroup
632
633 \end{verbatim}

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

\begin{verbatim}
628 \begin{group}
629 \def\@notdefinable{\gdef\adl@notdefinable{}\}
630 \@ifdefinable\Array\relax
631 \@ifdefinable\Tabular\relax
632 \ expandsafter\@ifdefinable\csname Tabular\endcsname\relax
633 \@ifdefinable\longtable\undefined\else \@ifdefinable\longtable\relax \fi
634 \endgroup
635
636 \end{verbatim}

45
If \texttt{\textbackslash adl@notdefinable} is \texttt{\textbackslash undefined} indicating that all \texttt{\textbackslash Array} etc. are definable, we \texttt{\textbackslash let} them be equal to \texttt{\textbackslash adl\textbackslash Array} etc. We also \texttt{\textbackslash let} ending macros \texttt{\textbackslash endArray} etc. be equal to \texttt{\textbackslash endarray} etc. Note that \texttt{\textbackslash Longtable} and \texttt{\textbackslash endLongtable} are defined only when \texttt{longtable} is loaded, and \texttt{\textbackslash endLongtable} is \texttt{\textbackslash def}-ined as (not being \texttt{\textbackslash let}-equal to) \texttt{\textbackslash endlongtable} because its definition of our own is not given yet.

Otherwise, we complain with a warning message put by \texttt{\PackageWarning} if it is defined (i.e. \texttt{\LaTeX} 2ε) or \texttt{\@warning} otherwise (i.e. \texttt{\LaTeX}-2.09).

\begin{verbatim}
\ifx\adl@notdefinable\undefined
    \let\Array\adl@Array
    \let\Tabular\adl@Tabular
    \expandafter\let\csname Tabular\textbackslash end\endcsname\adl@Tabularstar
    \let\endArray\endarray
    \let\endTabular\endtabular
    \expandafter\let\csname endTabular\textbackslash end\endcsname\endtabular
    \ifx\longtable\undefined
        \let\Longtable\adl@Longtable
        \def\endLongtable\endlongtable
    \else
        \begingroup
            \ifx\longtable\undefined
                \def\@tempa{Array and Tabular are not defined because one of them has been defined}
            \else
                \def\@tempa{Array/Tabular/Longtable are not defined because one of them has been defined}
            \fi
            \ifx\PackageWarning\undefined
                \def\MessageBreak{\textbackslash newpage}
                \@warning\@tempa
            \else
                \let\on@line\empty
                \PackageWarning[arydshln]\@tempa
            \fi
        \endgroup
    \fi
\fi
\end{verbatim}

\texttt{\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{\textbackslash 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\textbackslash end\endcsname\relax
    \let\endArray\relax
\end{verbatim}

\texttt{\ADLnoshorthanded}
Finally here we define *active* version of \@arrayclassz named \adl@act@arrayclassz etc. for \adl@activate (see §4.4). The definitions are simply done by \let-ing \adl@act@arrayclassz arrayclassz equal to \@arrayclassz etc\textsuperscript{19}.

\begin{verbatim}
\adl@act@arrayclassz \adl@act@tabclassz \adl@act@classz \adl@act@@startpbox \adl@act@@endpbox \adl@act@endpbox \adl@act@cr \adl@act@argcr \adl@act@cline \adl@act@endarray \adl@act@hline \adl@act@ihdashline \adl@act@cdline \adl@act@@vlineL \adl@act@@vlineR
\end{verbatim}

\section{Compatibility with colortab}

The package colortab has a macro;

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

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

Thus we modify \texttt{\LCC}, if the package 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.

\textsuperscript{19}Alternatively, we may define \texttt{\adl@act@arrayclassz} in place of \texttt{\@arrayclassz} but the author chose this way to minimize the possibility of \texttt{enbug}.  

\begin{verbatim}
\adl@arraysave \adl@arrayrestore
\end{verbatim}

%% Compatibility with colortab
Making arydshln compatible with longtable is a hard job because a longtable consists of multiple chunks and each chunk is a distinct \halign. We could draw vertical lines in each chunks as we do with ordinary array/table. However this straightforward solution should break dash-lines at invisible borders of chunks and produce awful results.

Therefore, this implementation draws dash-lines in \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 \output. This problem is solved by extracting the leading part of $R_L$ (\adl@rowsL) and $R^R$ (\adl@rowsR) by the height/depth of the table fraction to be put and removing the part from $R^{L/R}$.

### 4.15.1 Initialization

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

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

\texttt{\textbackslash \textless \textit{LTarray}}
\texttt{\textgreater \textit{LTarray}}
\texttt{\textbackslash \textless \textit{discard}}

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

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

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

\texttt{\textless adl@LTinactivate}

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

\begin{verbatim}
\texttt{\textless endlongtable \textless LT@make@row \textless LT@echunk \textless LT@end@hd@ft \textless LT@kill \textless LT@output}
\end{verbatim}

It also make \texttt{\textless adl@idashline} \texttt{\textless let}-equal to its inactive version because we need the macro to find mixed \texttt{\hline} and \texttt{\hdashline} sequence.

\begin{verbatim}
\texttt{\textless adl@LTarray/\textless LT@array}
\texttt{\textless def/\textless LT@array{\textless adl@arrayinit \textless adl@LTfirstpagetrue}
\texttt{\textless let/\textless adl@discard/\textless relax \textless let/\textless adl@idashline/\textless adl@LTidashline}
\texttt{\textless let/\textless adl@idashline/\textless adl@LTidashline}
\texttt{\textless ifadl@inactive \textless adl@LTinactivate \textless fi}
\texttt{\textless adl@LTarray}}
\texttt{\textless def/\textless adl@LTinactivate{\textless adl@inactivate}
\texttt{\textless let/\textless endlongtable/\textless adl@org/endlongtable}
\texttt{\textless let/\textless LT@make@row/\textless adl@org@LT@make@row}
\texttt{\textless let/\textless LT@echunk/\textless adl@org@LT@echunk}
\texttt{\textless let/\textless LT@end@hd@ft/\textless adl@org@LT@end@hd@ft}
\texttt{\textless let/\textless LT@kill/\textless adl@org@LT@kill}
\texttt{\textless let/\textless LT@output/\textless adl@org@LT@output}
\texttt{\textless let/\textless adl@idashline/\textless adl@LTinactive@hdl}}
\end{verbatim}

\texttt{\textless adl@org@LT@make@row}
\texttt{\textless LT@make@row}

The macro \texttt{\textless LT@make@row} is redefined for additional initialization which must be done after the original \texttt{\textless LT@array} performs its own initialization. First, \texttt{\textless LT@make@row} itself is reset to its original version \texttt{\textless adl@org@LT@make@row} to initialize stuff only once, since \texttt{\textless LT@make@row}
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-equal to our own \LT@array if array is not in use because with array opening a p-box is not done by \@startpbox but is embedded in \@preamble. Also note that we need \adl@LT@endmbox 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.

```latex
\let\adl@org@LT@make@row\LT@make@row
\def\LT@make@row{\let\LT@make@row\adl@org@LT@make@row\adl@columns\adl@ncol \ifadl@usingarypkg\else \let\@@startpbox\adl@LTstartpbox \let\@startpbox\adl@LTstartpbox \fi \let\@@endpbox\adl@LTendpbox \let\@endpbox\adl@LTendpbox \let\adl@@endmbox\adl@endmbox \let\adl@endmbox\adl@LTendmbox \adl@org@LT@make@row}
```

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 \crcr, we have to add the information of the last row to $R^{L/R} = \adl@rowsL/R$ 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@LT@echunk and \adl@org@LT@echunk respectively. Note that \adl@LTlastrow only has \crcr and \noalign and thus another \crcr 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.

```latex
\let\adl@org@endlongtable\endlongtable
\def\endlongtable{\adl@LTlastrow \adl@org@endlongtable}
\let\adl@org@LT@echunk\LT@echunk
\def\LT@echunk{\adl@LTlastrow \adl@org@LT@echunk}
\def\LT@echunk{\adl@LTlastrow \adl@noalign{\noalign}}
```
Table 2: Active and Inactive longtable Operations

<table>
<thead>
<tr>
<th>command</th>
<th>active</th>
<th>inactive</th>
</tr>
</thead>
<tbody>
<tr>
<td>p b (open) with array</td>
<td>\adl@act@classz \rightarrow \LT@startpbox</td>
<td>\adl@org@classz \rightarrow \LT@startpbox</td>
</tr>
<tr>
<td></td>
<td>\adl@LTstartpbox</td>
<td>\LT@startpbox</td>
</tr>
<tr>
<td>p b (close)</td>
<td>\adl@LTendpbox</td>
<td>\LT@endpbox</td>
</tr>
<tr>
<td>m (open)</td>
<td>\adl@act@classz \rightarrow \adl@LTstartpbox</td>
<td>\adl@org@classz \rightarrow \LT@startpbox</td>
</tr>
<tr>
<td></td>
<td>\adl@LTstartpbox \rightarrow \LT@startpbox</td>
<td></td>
</tr>
<tr>
<td>m (close)</td>
<td>\adl@LTendmbx</td>
<td>\LT@endpbox</td>
</tr>
<tr>
<td>\hline</td>
<td>\adl@act@hline \rightarrow @gobbletwo</td>
<td>\adl@LTihdashline \rightarrow \adl@LTinactivehd1 \rightarrow @gobbletwo</td>
</tr>
<tr>
<td>\hdashline</td>
<td>\adl@LTihdashline \rightarrow \adl@LTinactivehd1 \rightarrow @gobbletwo</td>
<td></td>
</tr>
<tr>
<td>\endlongtable</td>
<td>modified version</td>
<td>\adl@org@endlongtable</td>
</tr>
<tr>
<td>\LT@make@row</td>
<td>\adl@org@LT@make@row</td>
<td></td>
</tr>
<tr>
<td>\LT@echunk</td>
<td>\adl@org@LT@echunk</td>
<td></td>
</tr>
<tr>
<td>\LT@send@hd@oft</td>
<td>\adl@org@LT@send@hd@oft</td>
<td></td>
</tr>
<tr>
<td>\LT@kill</td>
<td>\adl@org@LT@kill</td>
<td></td>
</tr>
<tr>
<td>\LT@output</td>
<td>\adl@org@LT@output</td>
<td></td>
</tr>
</tbody>
</table>

Another chunk ending macro is \LT@send@hd@oft(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@send@hd@oft saves them together with \adl@totalheight to weirdly named macros; after closing the last row by \adl@LTlastrow. The \string representation of the macros looks like;

\adl@org@LT@send@hd@oft
\LT@send@hd@oft
\adl@LTfsave
\adl@LTth
\adl@LTth\LT@firsthead
\adl@LTth\LT@head
\adl@LTth\LT@lastfoot
\adl@LTth\LT@foot
\adl@rowsL\LT@firsthead
\adl@rowsL\LT@head
\adl@rowsL\LT@lastfoot
\adl@rowsL\LT@foot
\adl@rowsR\LT@firsthead
\adl@rowsR\LT@head
\adl@rowsR\LT@lastfoot
\adl@rowsR\LT@foot

\adl@LTth\LT@firsthead
\adl@LTth\LT@head
\adl@LTth\LT@lastfoot
\adl@LTth\LT@foot
\adl@rowsL\LT@firsthead
\adl@rowsL\LT@head
\adl@rowsL\LT@lastfoot
\adl@rowsL\LT@foot
\adl@rowsR\LT@firsthead
\adl@rowsR\LT@head
\adl@rowsR\LT@lastfoot
\adl@rowsR\LT@foot

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.

\begin{verbatim}
\let\adl@org@LT@end@hd@ft\LT@end@hd@ft
\def\LT@end@hd@ft#1{\adl@LTlastrow
  \noalign{\edef\adl@LTth{\number\adl@totalheight}\
    \adl@LThfsave#1\adl@LTth \global\adl@totalheight\z@\
    \adl@LThfsave#1\adl@rowsL\gdef\adl@rowsL{}}\
  \adl@LThfsave#1\adl@rowsR\gdef\adl@rowsR{}}
\adl@org@LT@end@hd@ft#1
\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}
\end{verbatim}

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 $h_i$ 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 $h_i$ of the last element to \@tempcnta, which is subtracted from \adl@totalheight globally. Finally, the original version \adl@org@LT@kill is called.

\begin{verbatim}
\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}
\end{verbatim}

4.15.3 Horizontal Lines and p-Boxes

The macro \LT@hline, longtable version of \hline, is redefined to add pseudo row information to $R^L/R$ and to check mixed sequence of \hline and \dashline. We also define the macro \LT@tdashline as the longtable version of \adl@tdashline and its inactive counterpart \LT@tdashline. These two macros, the main part of \dashline, are redefined to make it possible that \dashline can be broken into two part by \TeX’s page breaker.

\footnote{In the original \longtable, a sequence of three \hline-s are not recognized. This buggy feature is fixed in this implementation.}
These three macros call a common routine \texttt{\texttt{adl@LThdline}} after defining \texttt{\texttt{adl@LThdlrow}} which makes a row of horizontal (dash) line drawn by \texttt{\texttt{multispan}} and \texttt{\texttt{leaders\hrule}} or \texttt{\texttt{adl@hcline}}[(\texttt{dash})/(\texttt{gap})].

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

\begin{verbatim}
\def\LT@hline{\noalign{\ifnum0='}\fi
  \gdef\adl@LThdlrow{\multispan{\LT@cols}\unskip\leaders\hrule\@height\arrayrulewidth\hfill\cr}%
  \adl@LThdline}
\def\adl@LThdashline#1{\noalign{\ifnum0='}\fi
  \@ifnextchar\[%\]{#1}{#1[\dashlinedash/\dashlinegap]}}
\def\adl@LTihdashline[#1/#2]{\gdef\adl@LThdlrow{\multispan{\LT@cols}\unskip\adl@hcline\z@[#1/#2]}\adl@LThdline}
\def\adl@LTinactivehdl[#1/#2]{\gdef\adl@LThdlrow{\multispan{\LT@cols}\unskip\leaders\hrule\@height\arrayrulewidth\hfill\cr}\adl@LThdline}
\end{verbatim}

The macro \texttt{\texttt{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 \texttt{\texttt{\vskip-\arrayrulewidth}} with another break inhibitor if \texttt{ADLnullwidehline} is in effect, or adds the pseudo row information \texttt{connect(\arrayrulewidth)} to \texttt{R/L/R} by \texttt{\adl@hline}. Next, it draw a horizontal (dash) line by \texttt{\adl@LThdlrow} and checks if the following control sequence is \texttt{\hline} or \texttt{\hdashline} by \texttt{\futurelet} and \texttt{\adl@LTxhline}. If \texttt{\hline} or \texttt{\hdashline} is the next token, \texttt{\adl@LTixhline} is called to insert a vertical penalty of \texttt{\@medpenalty} and a vertical space of \texttt{\doublerulesep}. The macro \texttt{\adl@LTixhline} also adds \texttt{disconnect(\doublerulesep)} to \texttt{R/L/R} and makes \texttt{\adl@LThdlrow} void. Otherwise, \texttt{\adl@LThdline} inserts a vertical penalty of \texttt{\@lowpenalty} and a vertical space of \texttt{\arrayrulewidth} and draws the horizontal (dash) line again by \texttt{\adl@LThdlrow}. Thus a page can be broken between two overlaid horizontal (dash) lines. Two pseudo row information, \texttt{discard(\arrayrulewidth)} for the negative vertical space which may be discarded and \texttt{connect(\arrayrulewidth)} for the second horizontal line, are also added to \texttt{R/L/R}.

\begin{verbatim}
\def\adl@LThdline{\penalty\@M
  \@ifnextchar[\{\#1\}
    \{1\}%
    \{1\[\dashlinedash/\dashlinegap]\}}
\def\adl@LTihdashline[#1/#2]{\gdef\adl@LThdlrow{\multispan{\LT@cols}\unskip\adl@hcline\z@[#1/#2]}\adl@LThdline}
\end{verbatim}

\texttt{\adl@LThdline} \texttt{\adl@LTxhline} \texttt{\adl@LTixhline} Or do noting if inactive and thus it is \texttt{\let}-equal to \texttt{\@gobbletwo}.

\footnote{If the page is broken, the horizontal line at the beginning of the succeeding page has a width even if \texttt{ADLnullwidehline} is in effect.}
Macros for opening/closing p-boxes are fairly simple. The macro \texttt{\adl@LTstartpbox{⟨\w⟩}} is \texttt{\let}-assigned to \texttt{@\@startpbox} by \texttt{\LT@make@row} to open a p-box of \texttt{\w} wide by our own \texttt{\adl@act@@startpbox} and performs a footnote related operation introduced by \texttt{longtable}, when \texttt{array} is not in use. Note that if \texttt{array} is in use, a p-box is opened by codes embedded in \texttt{@preamble} and its initialization is done by \texttt{\startpbox = \LT@startpbox}, unnecessitating our own version of opening macros.

On the other hand, the closing macro \texttt{\adl@LTendpbox} for p(or d)-boxes is \texttt{\let}-equal to \texttt{\endpbox} and \texttt{\@@endpbox} for the cases with/without \texttt{array}, and performs the footnote operations after doing our own ones by \texttt{\adl@act@@endpbox}. Similarly, \texttt{\adl@LTendmbox} for m-boxes is \texttt{\let}-equal to \texttt{\endmbox} and performs our own operations by \texttt{\adl@@endmbox} in which the original definition of \texttt{\endmbox} is kept.

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

4.15.4 First Chunk

\texttt{\LT@start} The macro \texttt{\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 \texttt{\LTpre}, \texttt{\endgraf} is performed so that the skip contributes to \texttt{\pagetotal}.
- 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 underfull message, and restored from \texttt{\@tempcnta}.

\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.}

\footnote{This is also necessary for the original version.}

54
• The \texttt{dimen} register \texttt{\adl@LTpagetotal} 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 \texttt{\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 inactive counterpart because the modification shown above is desirable (first two) or not-harmful\footnote{Logically, at least.} (last two) to the original version.

\begin{verbatim}
\def\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
  \advance\dimen@\dp
  \ifdim\dp\@arstrutbox>\dp\tw@\@arstrutbox\else\tw@\fi
  \ifdim\maxdimen<\ht\pagegoal
    \ifdim\ht\pagegoal\advance\ht\LT@foot\pagegoal\dimen@\ht
    \ifdim\ht\LT@foot\else
      \advance\ht\LT@foot
      \global\adl@LTpagetotal\pagetotal
      \global\@colroom\@colht
      \ifvoid\LT@foot\else
        \advance\vsze\ht\LT@foot
        \global\advance\@colroom\ht\LT@foot
        \dimen@\pagegoal\advance\dimen@\ht\LT@foot\pagegoal\dimen@\ht
        \maxdepth\z@
      \fi
      \copy\ifvoid\LT@firsthead \LT@head \else \LT@firsthead \fi
      \output{\LT@output}
    \fi
  \fi
\endverbatim

4.15.5 Output Routine

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

- Three fractions of the original version to compile the final output image of the table portion into \box255 or the main vertical list are modified to set the image into \box255 unconditionally and to call \adl@LTdraw(foot)(tail) which is the real heart of the compatible implementation. The argument (foot) is \LT@foot or \LT@lastfoot according to the portion of the \longtable to be output. The argument (tail) is \vss if the last item is it which is not included in \box255 yet, or \empty otherwise. Since \adl@LTdraw builds final output image drawing vertical (dash) lines in \box255, it is put to the main vertical list if the \longtable portion is the last one.

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

865 \% Compatibility with \longtable: output routine
866 \let\adl@org@LT@output\LT@output
867 \def\LT@output{%
868 \ifnum\outputpenalty <-@Mi
869 \ifnum\outputpenalty > -\LT@end@pen
870 \LT@err{floats and marginpars not allowed in a \longtable}@ehc
871 \else
872 \setbox\z@\vbox{\unvbox\@cclv}%
873 \ifdim\ht\LT@lastfoot\ht\LT@foot
874 \dimen@\pagegoal
875 \advance\dimen@-\ht\LT@lastfoot
876 \ifdim\dimen@<\ht\z@
877 \setbox\@cclv\vbox{\unvbox\z@\copy\LT@foot}%
878 \adl@LTdraw\LT@foot\vss
879 \@makecol
880 \@outputpage
881 \setbox\z@\vbox{\copy\LT@head}%
882 \fi
883 \fi
884 \global@colroom@colht
885 \global@vsize@colht
886 \setbox\@cclv\vbox{\unvbox\z@}
887 \copy\ifvoid\LT@lastfoot\LT@foot\else\LT@lastfoot\fi%
888 \adl@LTdraw\LT@lastfoot\empty@box@\@cclv
889 \fi
890 \else
891 \setbox\@cclv\vbox{\unvbox\@cclv\copy\LT@foot}%
892 \adl@LTdraw\LT@foot\vss
893 \@makecol
894 \@outputpage
895 \global@vsize@colroom
896 \copy\LT@head
897 \fi}
The macro \LTdraw\langle foot\rangle\langle tail\rangle draws vertical (dash) lines onto the image in \box255. First it measures the total height \( H \) (\adltotalheight) of \longtable rows in \box255 and the total height \( H_b \) (\@tempdima) of its body which consists of the rows without the header and footer, as follows where \( H_{255} \), \( H_b \) and \( H_t \) are the height plus depth of \box255 and the effective header and footer of the page respectively.

\[
T = \begin{cases} 
\text{\adltotalheight} & \text{if } \lnot \LTfirstpage \\
0 & \text{otherwise}
\end{cases}
\]

\[
t = \begin{cases} 
\topskip \text{ glue} & \text{if } \longtable \text{ is the first item of the page} \\
0 & \text{otherwise}
\end{cases}
\quad \left( \lnot (\ifadl@firstpage \land T>0) \right)
\]

\[
H = H_{255} - t - T
\]

\[
H_b = H - H_b - H_t
\]

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

Subtraction of \( H_b \) and \( H_t \) is done by the macro \LTinit\langle hf\rangle\langle box\rangle, where \langle hf\rangle is head or foot and \langle box\rangle is one of \LT@firsthead, \LT@head and \langle foot\rangle (\LT@lastfoot or \LT@foot). This macro also copies the contents of weirdly named structure such as \adl@rows\LT@head into \LT@head and so on if \langle box\rangle is not void. Otherwise, \adl@headL etc. is kept to their initial value, \@empty.

Next, we make rows for vertical lines by \LTmakevlL/R after extracting the leading part of \RLR corresponding to the body by the macro \LTsplit\RLR where \RHL and \RJL are \LT@headL and so on. Since the macro defines \adl@rows given to \LTmakevlL/R to the sequence of \RHL, the extracted part of \RLR and \RJL, the rows for vertical lines for all the rows including header and footer are build in \adl@vlrowL and \adl@vlrowR as in the ordinary case without \longtable.

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

\[\def\adl@LTdraw#1#2{%\verbatim
\@tempswafalse
\ifadl@LTfirstpage\ifdim\adl@LTpagetotal>\z@\@tempswafalse \fi\fi
\if@tempswa
\setbox\z@\vbox{\hrule height1sp\unvcopy\@cclv}
\edef\adl@LTdraw\numexpr\ifdim\adl@LTpagetotal>\z@1\else0\fi\relax
\edef\adl@LTdraw\ifdim\adl@LTpagetotal>\z@\@tempswa\fi\relax
\setbox\z@\vbox{\hrule height1sp\unvcopy\@cclv}\fi\fi\]

\footnote{Copying by \edef can be replaced by \let with many \expandafter but it is not comprehensible.}
The macro \texttt{\adl@LTsplit} moves leading elements in $R^\text{L}/R$ into $R'$ (\texttt{\adl@rows}) until total heights of the elements summed in $h$ (\texttt{@tempdimb}) reaches to $H_b$.
by a straightforward loop with the macros \adl@LTisplit to fetch the \(i\)-th element and \adl@LTisplit to get \(h_i\). Before moving, however, we have to remove discardable item(s)\footnote{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.} from the top of \(R^{L/R}\). Since an element for a discardable item is \textit{disconnect} (\relax) or \textit{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 (\(R^{L/R}_b\) is \@empty). 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 \def-ined as \adl@connect in \adl@LTdraw, non-discarded \textit{discard} transforms into \textit{connect} in \(R'\). Also note that since the remaining part of \(R^{L/R}\) is \def-ined as the body of \@tempb which is globally \let-assigned to \(R^{L/R}\) again, \adl@discard survives in the new \(R^{L/R}\).

\begin{verbatim}
\def\adl@LTsplit#1#2#3{\def\adl@rows{\@empty}\@tempdimb\z@ \expandafter\adl@LTxsplit#1\@nil;\% \edef\adl@rows{#2\adl@rows#3}\% \global\let\@tempb\@tempdimb} \def\adl@LTxsplit#1;{\def\@tempa{#1}\% \ifx\@tempa\@nnil \def\@tempb{}\let\next\relax \else\ifx\adl@LTheadL\@empty \def\next{\adl@LTysplit#1}\% \else \def\next{\adl@LTisplit#1;}\% \fi \fi \next} \def\adl@LTrowrelax{\relax} \def\adl@LTrowdiscard{\adl@discard} \def\adl@LTysplit(#1/#2){\def\@tempa{#1}\% \ifx\@tempa\adl@LTrowrelax \let\next\adl@LTxsplit \else\ifx\@tempa\adl@LTrowdiscard \let\next\adl@LTxsplit \else \def\next{\adl@LTisplit(#1/#2);}\% \fi \fi \next} \def\adl@LTisplit#1;{\def\@tempa{#1}\% \ifx\@tempa\@nnil \def\@tempb{}\let\next\relax \else\ifdim\@tempdimb<\@tempdim\ \adl@LTisplit#1;\% \else \def\next{\adl@LTsplitend#1;\@nil;\%} \fi \fi \next} \def\adl@LTsplitend#1;\@nil;\% {\def\@tempb{#1;\%}} \fi \next \def\adl@LTsplit(#1/#2){\def\@tempa{#1}\% \ifx\@tempa\@nnil \def\@tempb{}\let\next\relax \else\ifx\@temp\@tempdim\ \adl@LTisplit#1;\% \else \def\next{\adl@LTsplitend#1;\@nil;\%} \fi \fi \next \def\adl@LTrows{\@empty}\@tempdimb\z@ \expandafter\adl@LTxsplit#1\@nil;\% \edef\adl@rows{#2\adl@rows#3}\% \global\let\@tempb\@tempdimb} \def\adl@LTrowrelax{\relax} \def\adl@LTrowdiscard{\adl@discard} \def\adl@LTysplit(#1/#2){\def\@tempa{#1}\% \ifx\@tempa\adl@LTrowrelax \let\next\adl@LTxsplit \else\ifx\@tempa\adl@LTrowdiscard \let\next\adl@LTxsplit \else \def\next{\adl@LTisplit(#1/#2);}\% \fi \fi \next} \def\adl@LTisplit#1;{\def\@tempa{#1}\% \ifx\@tempa\@nnil \def\@tempb{}\let\next\relax \else\ifdim\@tempdimb<\@tempdim\ \adl@LTisplit#1;\% \else \def\next{\adl@LTsplitend#1;\@nil;\%} \fi \fi \next} \def\adl@LTsplitend#1;\@nil;\% {\def\@tempb{#1;\%}} \fi \next \def\adl@LTrows{\@empty}\@tempdimb\z@ \expandafter\adl@LTxsplit#1\@nil;\% \edef\adl@rows{#2\adl@rows#3}\% \global\let\@tempb\@tempdimb}
\end{verbatim}
4.16 Compatibility with colortbl

The implementation to make arydshln compatible with 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 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 \texttt{arrayrulecolor} modifies in the row building phase. Since the macro \texttt{\CT@drsc\@} defined by \texttt{doublerulesepcolor} is examined if it is \texttt{relax} or not in the preamble construction phase, \texttt{doublerulesepcolor} in the preamble has no effect regardless the existence of the outside definition.

Thus we have to expand and evaluate \texttt{arrayrulecolor} and \texttt{doublerulecolor} in the preamble construction phase to define \texttt{CT@arc\@} and \texttt{\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.

\textbf{4.16.1 Initialization, Cell Coloring and Finalization}

\begin{verbatim}
\CT@arc\@ \adl@dashgapcolor
\end{verbatim}

First of all, we initialize the macro \texttt{\CT@arc\@}, which will be \texttt{\def}ned 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.

\begin{verbatim}
\def\CT@arc@{\relax}
\def\adl@dashgapcolor{\relax}
\end{verbatim}

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.

\begin{verbatim}
\if\@tempswa
\adl@org@inactivate
\adl@org@activate
\adl@inactivate
\adl@activate
\CT@setup
@endpbox
\end{verbatim}

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{\adl@inactivate} is in effect, or to our own version \texttt{\adl@act\CT@setup} which will be defined soon. New \texttt{\adl@activate} also \textit{inactivates} \texttt{\CT@setup} \texttt{\let} because our own one for column height/depth measurement is inappropriate with \texttt{colortbl} as explained soon.

\begin{verbatim}
\let\adl@org@inactivate\adl@inactivate
\let\adl@org@activate\adl@activate
\let\adi@org@inactivate\adi@org@inactivate \let\adi@org@activate\adi@org@activate
\adi@activate\adi@act\adi@CT@setup
\adi@endpbox
\end{verbatim}

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\CT@setup \@classz} \texttt{\box0} 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 \texttt{colortbls}'s \texttt{\@classz} placing it just before the \texttt{\box0} is put where \texttt{\ht0} plus \texttt{\minrowclearance} is calculated, but did not because the author wished to make it clear that \texttt{\@classz} is modified only for the bug fix of \texttt{\arrayrulecolor} and \texttt{\doublerulesepcolor} (and to introduce \texttt{\dashgapcolor}).

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 \texttt{colortbls}'s \texttt{\@classz} placing it just before the \texttt{\box0} is put where \texttt{\ht0} plus \texttt{\minrowclearance} is calculated, but did not because the author wished to make it clear that \texttt{\@classz} is modified only for the bug fix of \texttt{\arrayrulecolor} and \texttt{\doublerulesepcolor} (and to introduce \texttt{\dashgapcolor}).
4.16.2 Horizontal Line Coloring

\hline
\adl@inactivehline
\adl@ixhline

To color \hline and inactivated \hdashline, we modify our own \hline and \adl@inactivehline 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 colorbl's \cline renamed as \adl@org@cline and invoked from our own one.

\adl@ihdashline
\adl@act@ihdashline
\adl@cdline
\adl@act@cdline

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.

\adl@hclinesetup
\adl@cdlinea
\adl@cdlineb

The macro \adl@hclinesetup(f)\{t\} makes the prefix of a \multispan-ned row from the column \textit{f} to \textit{t} and \global-ly defines it as \gtempa. This is done by a code very similar to original \adl@cdline (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 \gtempa.
The modified version of \adl@hcline(\(w\))[\([d]/[g]\)] 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.

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

$$\text{ifdim}\baselineskip=0\text{align} \{\text{gdef}\CT@arc@{\color...}\}$$

This aims to do \noalign{\gdef{\CT@arc@{...}}} 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 and \doublerulesepscolor are modified to define \CT@arc@ and \CT@drc@ 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 colortbl is the expansion timing of \arrayrulecolor and \doublerulesepcolor in a >-argument. We have to modify \@classz to extract them from \toks\@tempcnta as its original version does for \columncolor. Thus we inserted the invocation of \adl@extract@arc for \arrayrulecolor, \adl@extract@drc for \doublerulesepcolor, and \adl@extract@dgc for \dashgapcolor just after the invocation of \CT@extract. Note that the other part of \@classz is not modified logically, but done for author’s preference of indentation. Also note that both \adl@act@classz and \adl@org@classz are \let-equal to the modified \@classz because we have to be bug free even if \ADLinactive is in effect.
The definitions of \adl@extract\(x\) \(x \in \{\text{arc}, \text{drsc}, \text{dgc}\}\) are quite similar to each other. For example \adl@extract\texttt{arc} is defined as follows.

\begin{verbatim}
\def\adl@extract@arc#1\arrayrulecolor#2\@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-in\texttt{CT@arc} as \color\(\{m\}\}\{c\}\). Note that \texttt{CT@extract} does a similar job for \texttt{columncolor} but it mistakenly ignores the possibility that the token register has two or more \texttt{columncolor}\textsuperscript{29}. Anyway, if we copy the code above and replace ‘\texttt{@arc}’ with ‘\texttt{@drsc}’, \arrayrulecolor\texttt{doublerulesepcolor}, and \texttt{CT@arc} with \texttt{CT@drsc}, we will have \adl@extract\texttt{drsc}@B for \texttt{doublerulesepcolor}. The code for \adl@extract\texttt{dgc}@B 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{key}{umac}{cmac}
\end{verbatim}

to define the macros \adl@extract@key{} for the user interface macro \texttt{umac} in which a color macro \texttt{cmac} is defined with \color{} for example, we will obtain \adl@extract@arc@B shown above by;

\begin{verbatim}
\adl@def@extract{arc}{\arrayrulecolor}{\CT@arc@}
\end{verbatim}

Note that \color{} is made \relax{} in the preamble construction phase by colortbl's \texttt{\@mkpream} and regain its proper meaning after the phase.

\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{\adl@extract@#1@b{##1}##3\@nil}%
\else \def\CT@{\color{##2}}%
\def\@tempa{\adl@extract@#1##1##3\@nil}%
\fi\fi \@tempa}
\end{verbatim}

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

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

\begin{verbatim}
1110 \ifx\longtable\undefined\else
1111 \def\LT@hline{%n\noalign\{\ifnum0='\fi
1112 \gdef\adl@LTidashline{\multispan{\LT@cols}\unskip{\CT@arc@
1113 \leaders\hrule\@height\arrayrulewidth\hfill}\cr}%
1114 \adl@LTidashline
1115 \def\adl@LTidashline[1/#2]{\adl@hclinesetup\@ne\adl@columns
1116 \noalign\{\ifnum0='\fi
1117 \gdef\adl@LTidashline{\adl@hcline\z@[1/#2]}%
1118 \adl@LTidashline
1119 \def\adl@LTinactivehd1[1/#2]{%
1120 \gdef\adl@LTidashline{\multispan{\LT@cols}\unskip{\CT@arc@
1121 \leaders\hrule\@height\arrayrulewidth\hfill}\cr}%
1122 \adl@LTidashline
1123 \def\adl@LTidashline{%
1124 \ifx\CT@drsc@\relax \gdef\adl@LTidashline{\noalign{
1125 \penalty-\@medpenalty \vskip\doublerulesep}}
1126 \else \gdef\adl@LTidashline{\noalign{\penalty0}%
1127 \multispan{\LT@cols}\unskip{\CT@drsc@
1128 \leaders\hrule\@height\doublerulesep\hfill}\cr}\fi
1129 \ifnum0='\fi\adl@LTidashline \noalign{\ifnum0='\fi}
1130 \adl@hline\relax\doublerulesep \global\let\adl@LTidashline\@empty}
1131 \fi
1132 \fi
\end{verbatim}

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{\textbackslash ADLinactivate}; Craig Leech who reported the incompatibility problem with \texttt{longtable}, which was also reported by Uwe Johnlich, 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.

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 \TeX 2.09 and \TeX 2ε (1997/12/01) to which Johannes Braams and other authors also contributed. The author also refers \texttt{array} package (v2.4c) written by Frank Mittelbach and David Carlisle; \texttt{colortab} package (v0.9) written by Timothy van Zandt; and \texttt{longtable} (v4.10) 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.

<table>
<thead>
<tr>
<th>Symbols</th>
<th>\ADLinactivate</th>
<th>\ADLnoshorthanded</th>
<th>\ADLnullvide</th>
<th>\ADLnulvidehline</th>
<th>\ADLsomevidehline</th>
<th>\ADLsomevide</th>
<th>\aftersegment</th>
<th>\ADLActivate {package}</th>
<th>\Array {environment}</th>
</tr>
</thead>
<tbody>
<tr>
<td>\AC</td>
<td>7</td>
<td>4, 5</td>
<td>4, 5</td>
<td>4, 5</td>
<td>4, 5</td>
<td>4, 5</td>
<td>4, 5</td>
<td>4, 5</td>
<td>4, 5</td>
</tr>
<tr>
<td>\adl@act@endbox</td>
<td>21, 47, 54</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>21, 47, 54</td>
<td></td>
</tr>
<tr>
<td>\adl@act@startpbox</td>
<td>21, 47, 54</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>21, 47, 54</td>
<td></td>
</tr>
<tr>
<td>\adl@act@vlinel</td>
<td>21, 47</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>21, 47</td>
<td></td>
</tr>
<tr>
<td>\adl@act@vlinek</td>
<td>21, 47</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>21, 47</td>
<td></td>
</tr>
<tr>
<td>\adl@act@argcr</td>
<td>21, 47</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>21, 47</td>
<td></td>
</tr>
<tr>
<td>\adl@act@arrayclassz</td>
<td>21, 47</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>21, 47</td>
<td></td>
</tr>
<tr>
<td>\adl@act@cdline</td>
<td>21, 47, 68</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>21, 47, 68</td>
<td></td>
</tr>
<tr>
<td>\adl@act@cdlinea</td>
<td>21, 47, 69</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>21, 47, 69</td>
<td></td>
</tr>
<tr>
<td>\adl@act@cdlineb</td>
<td>21, 47, 69</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>21, 47, 69</td>
<td></td>
</tr>
<tr>
<td>\adl@act@lineline</td>
<td>21, 47</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>21, 47</td>
<td></td>
</tr>
<tr>
<td>\adl@act@ihdashline</td>
<td>21, 47, 69</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>21, 47, 69</td>
<td></td>
</tr>
<tr>
<td>\adl@act@tableclassz</td>
<td>21, 47</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>21, 47</td>
<td></td>
</tr>
<tr>
<td>\adl@activate</td>
<td>20, 61</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>20, 61</td>
<td></td>
</tr>
<tr>
<td>\adl@activatebbox</td>
<td>29, 62</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>29, 62</td>
<td></td>
</tr>
<tr>
<td>\addamp</td>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>\adl@addrL</td>
<td>42, 57</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>42, 57</td>
<td></td>
</tr>
<tr>
<td>\ADLactivate</td>
<td>7, 15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7, 15</td>
<td></td>
</tr>
<tr>
<td>\ADLdrawingmode</td>
<td>6, 45</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6, 45</td>
<td></td>
</tr>
</tbody>
</table>

\textbf{A}

\textbf{B}

\textbf{C}

\begin{itemize}
\item \c@LT@chunkwidth \textcolor{red}{57}
\item \CC@ \textcolor{red}{47}
\item \adl@CC@ \textcolor{red}{47}
\item \cdashline \quad \textcolor{red}{4, 5, 45}
\item \adl@addrL \quad \textcolor{red}{42, 57}
\item \adl@addrR \quad \textcolor{red}{20, 21, 25, 47, 63}
\item \ADLactivate \quad \textcolor{red}{7, 15}
\item \ADLcdlineb \quad \textcolor{red}{35, 63}
\item \cellcolor \quad \textcolor{red}{7}
\item \cellcolor \quad \textcolor{red}{7}
\end{itemize
\chiclass ........................................ 26
\ccla ........................................ 17
\adl@ccla .................................. 17, 35
\adl@cclass@iiiorvii ....................... 25, 26
\adl@cclass@start .......................... 25, 26
\cclassv ..................................... 26
\adl@cclassv .................................. 26
\adl@cclassvfordash .......................... 26
\cclassz ...................................... 20, 21, 24, 47, 65
\clr .......................................... 17
\adl@clrb .................................... 17, 35
\cleaders ...................................... 44
\cline ........................................ 4, 33, 47, 63
\cline ........................................ 35
\adl@colhtdp ................................ 21, 22, 26, 27, 28, 29
\adl@colhtdp ................................ 28, 61
\color ......................................... 8, 26, 28
color{tab} (package) ...................... 3, 7
color{tabl} (package) ..................... 3, 7
\adl@cola ..................................... 19, 32, 38
\adl@colaLsave ............................... 19, 38
\adl@colaR .................................... 19, 32, 38
\adl@colaRsave ............................... 19, 38
\columncolor ................................ 7
\adl@columns ................................ 16, 23, 38, 49
\adl@connect ................................ 19, 32, 33, 40, 53
\adl@connect ................................ 40
\ifadl@connected ................................ 14, 98, 42

counters:
LTchunksize ................................ 8
\adl@cr ....................................... 20, 21, 32, 47
\adl@cr ....................................... 20, 32, 36, 50
\ct@arc0 ..................................... 22, 26, 61, 63, 64, 66
\ct@cell0\color .................................. 62
\ct@column0\color .................................. 62
\ct@color ..................................... 62
\ct@drsc0 ..................................... 63, 64, 66
\ct@end ......................................... 62
\adl@ct@end ................................ 62
\ct@row0\color .................................. 62
\ct@setup ..................................... 61
\adl@ct@setup ................................ 61
\ct@start ...................................... 62
\adl@ct@start ................................ 62
\current@color ................................ 28
\adl@current@columnsave .................. 16, 19, 38
\adl@current@columnsave .................. 16, 19, 38

D
\adl@dashes .................................. 17, 38, 42
\adl@dashes .................................. 17, 38, 42
dashgapcolor ................................ 8, 18, 64
dashgapcolor ................................ 8, 18, 64
dashgpcolor .................................. 62
\adl@dashed .................................. 22, 26, 61, 64, 66
dashline ..................................... 4, 14, 34
dashline ..................................... 4, 14, 34
dashnegap ..................................... 66
\adl@def@extract ............................. 66
\adl@defcolor .................................. 64
\adl@defhlhd ................................ 36
\adl@depp ..................................... 16, 19, 21, 32, 36, 38, 50
\adl@deppsave ............................... 16, 19, 38
\adl@discard ................................ 19, 32, 49, 53, 58
\documentstyle ................................ 3
\ifadl@doublerule ................................ 14, 38
\doubleruleseps ................................ 7, 18, 64
\adl@draw ................................... 35, 42, 44, 64
\adl@drawvi .................................. 44
\adl@drawv .................................. 44
\adl@drawvi .................................. 44
\adl@drawv .................................. 44
\adl@drawv .................................. 36, 42, 57

E
\EAC ........................................... 7
\@elt ........................................ 19, 40
\ENAC ......................................... 7
\endarr ........................................ 46, 62
\endarray ....................................... 36, 62
\adl@endarray ................................ 20, 21, 36, 47
\endfirsthead .................................. 51
\endfoot ....................................... 51
\endhead ....................................... 51
\endlastfoot .................................. 51
\endlongtable ................................ 46
\endlongtable ................................ 46
\adl@endmakevlr ................................ 40
\adl@endmakevrlconn .......................... 40
\adl@endmakevrlrect ................................ 40
\adl@endmakevlr ................................ 40
\adl@endmakevlr ................................ 40
\adl@endmakevlr ................................ 40
\adl@endmakevlr ................................ 40
\adl@endmakevlr ................................ 40
\adl@endmakevlr ................................ 40
\adl@endmakevlr ................................ 40
\adl@endmakevlr ................................ 40
\@endbox ..................................... 22, 24, 29, 49
\adl@endbox ................................ 49
\@endbox ..................................... 20, 21, 29, 47, 49
\@endbox ..................................... 20, 21, 29, 47, 49, 54
\endtabular .................................. 46
\endtabular .................................. 46
\endtabular* .................................. 46
\endtabular* .................................. 46
\endtabular* .................................. 46

environments:
longtable .................................. 19, 48
Array ......................................... 7

70
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 \textquotesingle. (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 \LaTeXe. ................................. 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
  \texttt{\textbackslash adl@activatepbox} \texttt{\textbackslash def-s for \textbackslash adl@mcarrayrule etc. are enclosed in a group.} .... 29

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{\textbackslash 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 \textbackslash textquotesingle of \texttt{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: \textbackslash hfil is replaced with \textbackslash hss taking the possibility of negative wide columns into account. ......................... 10
  General: Section 4.12 is added. ................................................. 43
  General: Section 4.13 is added. ................................................. 45
  General: Thank to more people. .................................................. 67

v1.4-2-1  
General: The following are for the general compatibility with \texttt{array}. .......... 1
  \texttt{\textbackslash ifadl@usingarraypkg}: Introduced to know if \texttt{array} is loaded. 15
  \texttt{\textbackslash adl@ncol}: Introduced for new column counting in preamble construction. 16
  \texttt{\textbackslash adl@everybbox}: Introduced for a tricky modification of \texttt{@array}. 17
  \texttt{\textbackslash adl@array}: Introduced to save original definition of \texttt{@array}. 17

74
\@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. ............ 22
@addamp: Modified for new column counting in preamble construction. .............................. 23
@testpach: The version for array is introduced. .......................................................... 24
@c\class: Introduced because \array uses it. ................................................................. 24
\adl\@class\@start: Introduced for class number identification. ................................... 25
\adl\@class\@start: Introduced for class number identification. ................................... 25
\adl\@class\@start: Introduced for class number identification. ................................... 26
\adl\@class\@start: Introduced for class number identification. ................................... 26
\adl\@arrayrule: Modified to replace \adl@columns with \adl@ncol. ............................... 26
\adl\@arraydashrule: Modified to replace \adl@columns with \adl@ncol. ......................... 26
\adl\@arraydashrule: Modified to replace \adl@columns with \adl@ncol. ......................... 26
\adl\@arraydashrule: Modified to pretend \p or \@ depending on if \array is in use. .......... 26
\adl\@arraydashrule: Modified to refer \adl@class\@start rather than \LaTeXX'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
@endbox: Introduced because \array uses it. ............................................................... 29
@multicolumn: Modified for several reasons. ............................................................. 29
@\mc@addamp: Introduced for the complaint on multiple columns if with \array. .................. 29
@\mc@activatepbox: Introduced to do nothing if with \array. ....................................... 29
@\mc\@arraydashrule: Modified to pretend \p or \@ depending on if \array is in use. ........... 31
@\mcarraycr: The version for \array is introduced. ..................................................... 31

v1.4-2-2
General: The following are to control the effective width of vertical lines. ..................... 1
@\inactivate: Introduced to indicate vertical lines have null width. ............................. 14
\ADL\nullwidehline: Introduced to make vertical lines null wide. .................................. 15
\ADL\some\nullwidehline: Introduced to make vertical lines \array\nullwidehline wide. .......... 15
\adl\@arraydashrule: Modified to add invisible rule of \array\rulewidth wide if \ADL\some\nullwidehline wide. .................................................. 26
\adl\@ovrule: Modified to make vertical line null wide only if \ADL\nullwide. .................. 42

v1.4-2-3
General: The following are for inactivation of dash-line functions. ................................ 1
@\inactivate: 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
@\inactivate: Modified to call \adl\@inactivate. ..................................................... 19
@\inactivate: Introduced to inactivate \@array\classz etc. ....................................... 20
@\inactivate\@vrule: Introduced to emulate \cdashline by \cline. ................................ 34
@\inactivate\@vrule: Introduced to emulate \vdashline by \cline. ................................ 35
@\array: Introduced as the body of \Array. .............................................................. 45
@\Tabular: Introduced as the body of \Tabular. ......................................................... 45
v1.4-2-4
General: The following are for drawing mode to cope with the bug of \xleaders.
\adl@hcline: Modified to use \adl@draw. ............................... 35
\adl@vrule: Modified to use \adl@draw. ............................... 42
\adl@hrule: Introduced to draw a dash for horizontal lines in \adl@draw. 43
\adl@drawi: Introduced as \adl@draw in mode 1. .......................... 44
\adl@drawii: Introduced as \adl@draw in mode 2. .......................... 44
\adl@drawiii: Introduced as \adl@draw in mode 3. .......................... 44
\adl@draw: Introduced as the mode and axis independent line drawing macro. 44
\ADLdrawingmode: Introduced to specify drawing mode. ...................... 45
v1.4-2-5
General: The following are to implement dashed version of \firstline and \lastline of array. ............................. 1
\adl@dashline: Modified to make \adl@dashline usable for \firstline/\lastline. 34
\adl@dashline: Modified to be usable for \firstline/\lastline. 34
\adl@dashline: Introduced as the substitute of old \adl@dashline. 34
\firstdashline: Introduced as the dashed version of \firstline. 35
\lastdashline: Introduced as the dashed version of \lastline. 35
\adl@deffldashline: Introduced for the tricky definition of \adl@first/\lastdashline. 36
\adl@deffldashline: Introduced for the tricky definition of \adl@first/\lastdashline. 36
\adl@firstdashline: Introduced as the body of \firstdashline. 36
\adl@lastdashline: Introduced as the body of \lastdashline. 36
v1.4-2-6
General: The following are to fix the bug by which the depth of array/tabular was always zero. ................................. 1
\adl@finaldepth: Introduced to measure the depth of the last row. 16
\adl@org@cline: Introduced to refer original version in modified cline. 17
\adl@crr: Modified to set \adl@finaldepth. ................................. 32
\cline: Modified to set \adl@finaldepth to zero. ......................... 33
\cline: Modified to set \adl@finaldepth to zero. ......................... 33
\adl@endarray: Modified to set the depth of array/tabular to \adl@finaldepth. 36
v1.4-2-7
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@cdlinea/b. 35
\adl@cdlines: Renamed. ................................................. 35
\adl@cdlineb: Renamed. ................................................. 35
v1.4-2-8
General: The following are to cope with very narrow or negative wide columns. ........ 1
\texttt{\textbackslash adl@makevlrL}: Modified to replace \texttt{\textbackslash hfil} with \texttt{\textbackslash hss} to prevent drawing vertical lines widen columns. ................................................. 38
\texttt{\textbackslash adl@makevlrR}: Modified to replace \texttt{\textbackslash hfil} with \texttt{\textbackslash hss} to prevent drawing vertical lines widen columns. ................................................. 38

v1.4-2-9
\texttt{\textbackslash adl@arrayinit}: The bug of saving \texttt{\textbackslash adl@colsR} is fixed. ................................. 19

v1.4-3
General: Released to CTAN on 2000/07/04. ................................. 1

v1.5
General: Make compatible with \texttt{colortab}, and fix bugs. (2000/07/12) ....................... 1

v1.5-1
General: The following are for the compatibility with \texttt{colortab}. ................................. 1
General: The history on the compatibility with \texttt{colortab} package. ................................. 3
General: Caution about loading order of \texttt{colortab} is added. ................................. 3
General: Section 2.7 is added. ................................. 7
General: Description of \texttt{colortab} commands is added. ................................. 7
General: Caution about \texttt{\AC}/\texttt{\EAC} pair for vertical line coloring is added. ................................. 9
\texttt{\textbackslash adl@arraysave}: Introduced to use in modified \texttt{\CC@} of \texttt{colortab}. ................................. 19
\texttt{\CC@}: Modified to save/restore globals before/after height measurement. ................................. 47

v1.5-2
General: The following are for bug fix of \texttt{\textbackslash adl@putlrc}. ................................. 1
\texttt{\textbackslash adl@colhtdp}: The pseudo-formal description of \texttt{(put-lrc)} is modified. ................................. 21
\texttt{\textbackslash adl@putlrc}: \texttt{\textbackslash adl@putlrc} must do \texttt{\unhbox}\texttt{\textbackslash adl@box} to make glues effective. ................................. 26

v1.5-3
General: The following are for bug fix of \texttt{\textbackslash adl@inactivate}. ................................. 1
\texttt{\textbackslash adl@noalign}: Move \texttt{\textbackslash adl@inactivate} to \texttt{\array} from \texttt{\textbackslash adl@arrayinit}. ................................. 18
\texttt{\textbackslash adl@arrayinit}: Move \texttt{\textbackslash adl@inactivate} from \texttt{\textbackslash adl@arrayinit} to \texttt{\array}. ................................. 19
\texttt{\textbackslash adl@inactivate}: Change \texttt{\textbackslash adl@inactivate} caller to \texttt{\array}. ................................. 20
General: Thank to Yaxin Liu. ................................. 67

v1.54
General: Bug fixes. (2003/08/25) ................................. 1

v1.54-1
General: The following are for bug fix of \texttt{\textbackslash adl@vrow}. ................................. 1
\texttt{\textbackslash adl@vrow}: Rows for vertical lines are replaced by \texttt{\textbackslash adl@drawvl}. ................................. 37
\texttt{\textbackslash adl@drawvl}: Introduced to draw vertical lines correctly if \texttt{\textbackslash ADL@somewide}. ................................. 42
\texttt{\textbackslash adl@vrow}: Insert a negative skip to left/right of the line if \texttt{\textbackslash ADL@somewide}. ................................. 42

v1.54-2
General: The following are for bug fix of activation. ................................. 1
\texttt{\textbackslash adl@noalign}: Invoke \texttt{\textbackslash adl@activate} if not \texttt{\ifadl@inactive}. ................................. 18
\texttt{\textbackslash adl@inactive}: Add \texttt{\textbackslash adl@argcr} to inactivation. ................................. 20
\texttt{\textbackslash adl@activate}: Introduced to draw vertical lines correctly if \texttt{\textbackslash ADL@arrayclassz}. ................................. 20
\texttt{\textbackslash adl@act@arrayclassz}: Introduced to activate \texttt{\textbackslash arrayclassz} etc. again. ................................. 47

v1.54-3
General: The following are miscellaneous modifications. ................................. 1
\texttt{\textbackslash adl@ncline}: Omit \texttt{\vskip} if the space is 0. ................................. 35

v1.6
General: The following are for the compatibility with \texttt{longtable}. (2003/08/25) ................................. 1
General: The history on the compatibility with \texttt{longtable} package. ................................. 3
General: Caution about loading order of \texttt{longtable} is added. ................................. 3

77
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@cr: Modified to insert `\adl@discard`. ................................. 32
\adl@Longtable: Introduced as the body of `\Longtable`. ................... 45
\Longtable: Introduced as the always-active `\longtable`. ............... 46
\endLongtable: Introduced to `\end` the environment `Longtable`. ........ 46
\ADLnoesoshorthand: `\Longtable` and `\endLongtable` are added. ........ 46
General: §4.15 is added. ..................................................... 48
General: Thank to people for `longtable`. .................................. 67
v1.7
General: The following are for the compatibility with `colortbl`. (2004/05/21) .... 1
General: The history on the compatibility with `colortbl` package. .......... 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`. 36
General: Codes for `longtable` is surrounded by `\ifx`/`\fi` ............... 48
General: §4.16 is added. ..................................................... 60
General: Thank to Klaus Dalinghaus and refer orignal `colortbl`. .......... 67
v1.7-1
General: The following are for null-wide horizontal lines. .................. 1
\ifadl@zwhrule: 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@hdashline: Modified to shift up if null-wide. ......................... 33
\adl@hdashline: `\adl@hdashline` is moved to `\adl@dashline` for null-wide lines. ................. 34
\adl@inactivethdline: Modified to shift up if null-wide. ... 34
\adl@cdashline: Modified to shift up if null-wide. ......................... 34
\adl@inactivetcdashline: Modified to shift up if null-wide. .............. 35
\adl@inactivecd: Modified to invoke `\cline` rather than `\adl@orgcline` for null-wide. ............. 35
\adl@hline: Modified not to shift null-wide `\dashline` down.............. 35
\adl@LTdashline: Keep original without shift up because it is done by `\adl@LTdashline`. .... 52
\adl@LTdashedline: Modified to shift up if null-wide. ................... 53
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`/`\adl@vlineR`. .... 26
\adl@vlineL: Color arguments are added. .................................. 28
\adl@vlineR: Color arguments are added. .................................. 28
\adl@vline: Invocations of `\adl@setcolor` are added. ..................... 28
\textbackslash adl@setcolor: Introduced to color vertical lines. .................................. 28
\textbackslash adl@nocolor: Introduced to examine if coloring is specified. ..................... 28
\textbackslash adl@dashcolor: Introduced as the temporary variable of color specification of dashes. 28
\textbackslash adl@gapcolor: Introduced as the temporary variable of color specification of gaps. 28
\textbackslash adl@inactivevl: Modified to color the \textbackslash vline by the first argument. .... 29
\textbackslash adl@makevlr: Modified to initialize \textbackslash adl@dashcolor and \textbackslash adl@gapcolor. 40
\textbackslash adl@iiimakevlr: Modified to check color indentity. ................................ 40
\textbackslash adl@studentvlr: Modified not to see $d$ and $g$ which now have colors. .......... 40
\textbackslash adl@addvlL: Modified to add colors to $\Delta$ and $\xi$. .............................. 42
\textbackslash adl@addvlR: Modified to add colors to $\Delta$ and $\xi$. .............................. 42
\textbackslash adl@vl: Modified to color dashes and gaps. ........................................ 42

\textit{v1.71}

General: The following are for bug fix for \textbackslash array's m-columns. (2004/7/31) ....... 1
\textbackslash @mkpream: Modified to nullify \textbackslash adl@startmbox and \textbackslash adl@endmbox for array's m-columns. 22
\textbackslash @classz: Modified to call \textbackslash adl@startmbox and \textbackslash adl@endmbox for array's m-columns. .... 24
\textbackslash adl@startmbox: Introduced to the bug fix of array's m-columns. ................... 29
\textbackslash adl@endmbox: Introduced to the bug fix of array's m-columns. ...................... 29
General: Thank to Morten Høgholm. ................................................................. 67

\textit{v1.72}

General: Bug fix and revision of \textbackslash §2.4. (2016/03/19) ........................................ 1
\textbackslash LT@make@row: Modified to add \textbackslash let-assignments to \textbackslash adl@endmbox and \textbackslash adl@endmbox so that footnotes are correctly processed at the closing of a m-type column. .............. 49
\textbackslash adl@LTendmbox: Added to process footnotes in m-type columns appropriately. ...... 54
General: Thank to Maëul Rouquette. ................................................................. 67

\textit{v1.72-2}

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

\textit{v1.73}

General: Bug fix. (2016/04/28) ................................................................. 1
General: Thank to Maëul Rouquette again. .......................................................... 67

\textit{v1.73-1}

General: The following are to fix the problem that the top edge of 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 $\eta$ and addition/subtraction of it for $\tau_k$ and $\beta_k$. ....... 12
General: Add $\eta = \textbackslash adl@lastconn$, its initialization and updates, and the addition to $\tau$. 38
\textbackslash adl@makevlr: Add $\eta = \textbackslash adl@lastconn \leftarrow 0$. ............................. 40
\textbackslash adl@iiimakevlr: Modify the definition of \textbackslash adl@connect to pass $h$ to \textbackslash adl@connect. 40
\textbackslash adl@studentvlr: Replace two occurrences of $\tau \leftarrow \beta$ with $\tau \leftarrow \beta + \eta$ and add $\eta \leftarrow 0$, where $\eta = \textbackslash adl@lastconn$. ............................................................... 40
\textbackslash adl@endmbox: Add $\eta = \textbackslash adl@lastconn \leftarrow 0$. ............................. 40
\textbackslash adl@endmbox: Add $\eta = \textbackslash adl@lastconn \leftarrow h$ with the added argument $h$. 40

\textit{v1.73-2}

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

79
\LT@array: Add \let-assignment of \adl@LThdashline to \adl@hdashline so that the longtable version of \adl@hdashline is effective only in longtable environment rather than globally. ........................................... 49
\adl@LThdashline: Renamed from \adl@hdashline to make it effective only in longtable environments. .................................................. 52

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. .......................... 36
\endarray: Add conditional invocation of \@arrayright. .......................... 62
General: Thank to Hironobu Yamashita. ......................................... 67