The multirow, bigstrut and bigdelim packages

Pieter van Oostrum*
Øystein Bache
Jerry Leichter†

Released 2021/03/15, version v2.8

Contents

*catalogued "active author"
†Documentation originally put together by Robin Fairbairns

1 Introduction

These packages offer a series of extensions to the standard LaTeX `tabular` environment. Their respective functions are:

multirow which provides a construction for table cells that span more than one row of the table;

bigstrut which creates struts which (slightly) stretch the table row in which they sit.

bigdelim which creates an appropriately-sized delimiter (for example, brace, parenthesis or bracket) to fit in a single multirow, to indicate a relationship between other rows; and

2 Changes in version 2

version 2.8

- Optional argument ⟨*vmove*⟩ for `\ldelim` and `\rdelim`.

version 2.7

- Make `\@xmultirow` and `\multirow@setcolwidth` `\long` to allow multiparagraph text. See https://github.com/pietvo/multirow/issues/1

version 2.6

- Make the `supertabular` option compatible with newer versions of the supertabular package
- Initialize `\@arstrutbox` when not defined, to enable some uses of the big delims outside of an `array` or `tabular`.

version 2.5a

- Changed contact information

version 2.5

- Solve a clash with the colortbl package, when multirow uses the `longtable` option. There was a clash with both packages redefining `\@cline`.

version 2.4

- Add a `\leavevmode` in bigstrut to force horizontal mode
- Make ⟨*width*⟩ and ⟨*vmove*⟩ in `\multirow` calc compatible

version 2.3

- Replaced `\textrm` with `\textnormal` in text beside big braces in bigdelim.sty.

version 2.2

- Support for fractional values of ⟨*nrows*⟩.

version 2.0

- `\multirow` now has an first optional parameter `[`⟨*vpos*⟩`]`.
- The ⟨*width*⟩ parameter can be specified as `=` to use the defined width of the column in which the `\multirow` appears.
- Optional prefix letters (`t`, `b`) for the ⟨*bigstruts*⟩ parameter (see section 3.5).
- Package option `debug`.
- Package option `longtable` to work around a bug in longtable. See section 3.6.
- Package option `supertabular` to better support supertabular. See section 3.7.
- Better positioning in some cases.
- Lots of documentation.
- The distribution is now based on a `.dtx` file.
- Backwards compatible with v1.6.

3 Using multirow

`\multirow` `\multirow` sets a piece of text in a `tabular` or similar environment, spanning multiple rows. We will call the block of rows and columns that the text spans the multirow block. Usually this covers one column, but by combining it with `\multicolumn` more columns can be covered.

The basic syntax is:

`\multirow[`⟨*vpos*⟩`]{`⟨*nrows*⟩`}[`⟨*bigstruts*⟩`]{`⟨*width*⟩`}[`⟨*vmove*⟩`]{`⟨*text*⟩`}`

where

⟨***vpos***⟩ defines the vertical positioning of the text in the multirow block. The default is `[c]` which means the text will be vertically centered. Other options are `[t]` for top alignment and `[b]` for bottom alignment.

⟨***nrows***⟩ is the number of rows to span. You should leave the other rows empty at this column, otherwise the stuff created by `\multirow` will over-write it. With a positive value of ⟨*nrows*⟩ the spanned columns are this row and (⟨*nrows*⟩-1) rows below it. With a negative value of ⟨*nrows*⟩ they are this row and (1-⟨*nrows*⟩) above it. Fractional values are permitted for ⟨*nrows*⟩; this allows for some fine-tuning.

⟨***bigstruts***⟩ is mainly used if you've used the bigstrut package. In that case it is the total number of uses of `\bigstrut` within the rows being spanned. Count 2 uses for each `\bigstrut`, 1 for each `\bigstrut[`⟨*x*⟩`]` where ⟨*x*⟩ is either `t` or `b`. The default is 0.

The ⟨*bigstruts*⟩ parameter can optionally be preceded by a prefix letter `t`, `b` or `tb` for finer control. See section 3.5 for details. The letter may be separated from the number by a space character.

⟨***width***⟩ is the width to which the text is to be set. Special values are `*` to indicate that the text parameter's natural width is to be used, and `=` to indicate that the specified width of the column in which the `\multirow` entry is set should be used.

⟨***vmove***⟩ is a length used for fine-tuning: the text will be raised (or lowered, if ⟨*vmove*⟩ is negative) by that length above (below) wherever it would otherwise have gone.

⟨***text***⟩ is the actual text of the construct. If the width was set explicitly, the text will be set in a `\parbox` of that width; you can use `\\` to force linebreaks where you like.

If the width was given as `*` the text will be set in LR mode. If you want a multiline entry in this case you could use a `tabular` or `array` environment in the text parameter. See for example the `minitab` below.

The width can also be given as `=` when the `\multirow` entry is given in a column that has a defined width, for example in a `p{}` column, an `X` column in `tabularx` or a `L`, `C`, `R` or `J` column in a `tabulary` environment. The text will be set in a `\parbox` of that width. If you give "=" in other situations, you will get strange results (usually a too wide column).

In multirow version 2.4 and later, the ⟨*width*⟩ and ⟨*vmove*⟩ arguments can be given as calc expressions if the calc package is loaded. It is the responsibility of the document writer to include the calc package; multirow does not do this.

N.B. `\multirow` can be used in the `tabular` environment and most derivatives of it, for example `tabularx`, `tabulary`, `supertabular`, `ltablex`, `xtab`, `longtable`, `tabu`, `booktabs` and `ctable`. For some of these you have to pay special attention to certain cases, see below.

`\multirowsetup` Just before ⟨*text*⟩ is expanded, the `\multirowsetup` macro is expanded to set up any special environment. Initially, `\multirowsetup` contains just `\raggedright`. It may be redefined with `\renewcommand`.

If you want to use both `\multirow` and `\multicolumn` on the same entry, you must put the `\multirow` inside the `\multicolumn`. The other way around will not work. For example:

```
\multicolumn{2}{c}{\multirow{3}{*}{Multi-multi}}
```

3.1 Package Options

`multirowdebugtrue` `multirowdebugfalse` The following options are defined:

debug This option causes information about multirow boxes to be written to the log file. This is done by the TEX `\showbox` command. Note: this will cause the LATEX compilation to be considered failed, even if there is no real error. This option can also be activated anywhere in the document with the command `\multirowdebugtrue` and deactivated with `\multirowdebugfalse`. When such a command is placed just before a `\multirow`, it applies only to that specific `\multirow` entry.

longtable The `longtable` option redefines the `\cline` macro to work around a bug in the longtable package. See section 3.6.

3.2 Examples

An example with both multirow and bigstrut):

```
\newcommand{\minitab}[2][l]{\begin{tabular}{#1}#2\end{tabular}}
\begin{tabular}{|c|c|}
\hline
\multirow{4}{1in}{Common g text} & Column g2a\\
      & Column g2b \\
      & Column g2c \\
      & Column g2d \\
\hline
\multirow{3}[6]*{Common g text} & Column g2a\bigstrut\\\cline{2-2}
      & Column g2b \bigstrut\\\cline{2-2}
      & Column g2c \bigstrut\\
\hline
\multirow{4}[8]{1in}{Common g text, but a bit longer.} &
                                    Column g2a\bigstrut\\\cline{2-2}
      & Column g2b \bigstrut\\\cline{2-2}
      & Column g2c \bigstrut\\\cline{2-2}
      & Column g2d \bigstrut\\
\hline
\multirow{4}*{\minitab[c]{Common \\ g text}} & Column g2a\\
      & Column g2b \\
      & Column g2c \\
      & Column g2d \\
\hline
\end{tabular}
```

which will appear as:

Common g text	Column g2a Column g2b Column g2c Column g2d	Normal case
Common g text	Column g2a	With `\bigstruts` and `*` as ⟨*width*⟩
	Column g2b	
	Column g2c	
Common g text, but a bit longer.	Column g2a	With `\bigstruts` and normal ⟨*width*⟩
	Column g2b	
	Column g2c	
	Column g2d	
Common g text	Column g2a Column g2b Column g2c Column g2d	Multiline text in `\multirow`

An example with the "`=`" ⟨*width*⟩ specifier in a `tabulary` (Note: The braces around the `=` may be omitted):

```
\setlength{\extrarowheight}{2pt}
\begin{tabulary}{11cm}{|L|L|L|}
  \hline
  All human beings are born free and equal in dignity and rights. &
     \multirow{2}{=}{Everyone is entitled to all the rights and
     freedoms set forth in this Declaration, without distinction of
     any kind, such as race, colour, sex, language, religion,
     political or other opinion, national or social origin, property,
     birth or other status.}
  & Everyone has the right to life, liberty and security of person. \\
\cline{1-1}\cline{3-3}
     No one shall be held in slavery or servitude; slavery and the
     slave trade shall be prohibited in all their forms. &
  & No one shall be subjected to torture or to cruel, inhuman or
  degrading treatment or punishment. \\
\hline
\end{tabulary}
```

All human beings are born free and equal in dignity and rights.	Everyone is entitled to all the rights and freedoms set forth in this Declaration, without distinction of any kind, such as race, colour, sex, language, religion, political or other opinion, national or social origin, property, birth or other status.	Everyone has the right to life, liberty and security of person.
No one shall be held in slavery or servitude; slavery and the slave trade shall be prohibited in all their forms.		No one shall be subjected to torture or to cruel, inhuman or degrading treatment or punishment.

A few observations about this example:

1. The middle column is the `\multirow`. You would expect it to be vertically centered, but it isn't. This is because `\multirow` doesn't know the height of the box. The only estimate `\multirow` can make about the height is the number of rows × the normal height of a row. It tries to center the text in that space, but that space is too low in this example. Therefore the text is at the top of the box. If you want it to be centered, you have to supply a ⟨*vmove*⟩ argument to shift it down.

2. We have used an `\extrarowheight`[1] of 2pt, to make a bit room between the `\hline` and the following text. However, this is not applied to the `\multirow`, because this is thought to be centered. In this case you can give the ⟨*vpos*⟩ argument as `[t]`, in which case `\multirow` will do the proper positioning.

Now with a negative ⟨*nrows*⟩.

```
\setlength{\extrarowheight}{2pt}
\begin{tabulary}{11cm}{|L|L|L|}
  \hline
  All human beings are born free and equal in dignity and rights. &
  & Everyone has the right to life, liberty and security of person. \\
\cline{1-1}\cline{3-3}
     No one shall be held in slavery or servitude; slavery and the
     slave trade shall be prohibited in all their forms. &
     \multirow{-2}{=}[12mm]{Everyone is entitled to all the rights and
     freedoms set forth in this Declaration, without distinction of
     any kind, such as race, colour, sex, language, religion,
     political or other opinion, national or social origin, property,
     birth or other status.}
  & No one shall be subjected to torture or to cruel, inhuman or
  degrading treatment or punishment. \\
\hline
\end{tabulary}
```

All human beings are born free and equal in dignity and rights.	Everyone is entitled to all the rights and freedoms set forth in this Declaration, without distinction of any kind, such as race, colour, sex, language, religion, political or other opinion, national or social origin, property, birth or other status.	Everyone has the right to life, liberty and security of person.
No one shall be held in slavery or servitude; slavery and the slave trade shall be prohibited in all their forms.		No one shall be subjected to torture or to cruel, inhuman or degrading treatment or punishment.

In this case the text would be centered somewhere in the bottom row, which would make it stick out of the bottom. Therefore we applied a ⟨*vmove*⟩ of 12mm. The ⟨*vmove*⟩ usually requires some experimentation.

[1] This is only available with the array package, which tabulary includes automatically.

3.3 Fine-Tuning

If any of the spanned rows are unusually large, or if you're using the bigstrut package and `\bigstruts` are used asymetrically about the centerline of the spanned rows, the vertical centering may not come out right. Use the ⟨*vmove*⟩ parameter in this case. Sometimes it may be more helpful to just use a larger value for ⟨*nrows*⟩, including fractional values. See an example in section 3.8.

It's just about impossible to deal correctly with descenders. The text will be set up centered, but it may then have a baseline that doesn't match the baseline of the stuff beside it, in particular if the stuff beside it has descenders and ⟨*text*⟩ does not. This may result in a small misalignment. About all that can be done is to do a final touchup on ⟨*text*⟩, using the ⟨*vmove*⟩ optional parameter. (Hint: If you use a measure like `.1ex`, there's a reasonable chance that the ⟨*vmove*⟩ will still be correct if you change the point size.)

`\multirow` is mainly designed for use with `tabular`, as opposed to `array`, environments. It might not work well in an array environment if there are big formulas in some rows; in that case you can use the ⟨*vmove*⟩ parameter to refine the result.

In some cases you might want to align the multirow entry with the top of the other row cells, for example if you have a large capital in it. when you use ⟨*vpos*⟩ = `[t]`, the baselines will be aligned, which is the wrong thing in this case. You can then do the positioning with the ⟨*vmove*⟩ parameter and let LaTeX calculate the amount. For example:

```
\usepackage{calc}
\newlength{\shiftdown}
\setlength{\shiftdown}{\heightof{\Huge\bfseries B}-\heightof{f}}
. . .
\begin{tabular}{cll}
  \toprule
  \multirow[t]{5}{*}[-\shiftdown]{\Huge\bfseries B}
                    & foo  & Lorem ipsum dolor sit    \\
                    & bar  & Maecenas sed purus       \\
                    & baz  & Nullam luctus id tellus  \\
                    & qux  & Aenean consequat commodo \\
  \bottomrule
\end{tabular}
```

B	foo	Lorem ipsum dolor sit
	bar	Maecenas sed purus
	baz	Nullam luctus id tellus
	qux	Aenean consequat commodo

In multirow version 2.4 and later you can also directly use the expression `-\heightof{\Huge\bfseries B}-\heightof{f}` instead of `-\shiftdown` for the ⟨*vmove*⟩ argument.

3.4 Multirow and colored cells

If you use `\multirow` with the colortbl package (or the xcolor package with the `[table]` option) you have to take precautions if you want to color the column

that has the \multirow in it. The colortbl package works by coloring each cell separately. So if you use \multirow with a positive ⟨*nrows*⟩ value, colortbl will first color the top cell, then \multirow will typeset ⟨*nrows*⟩ cells starting with this cell, and later colortbl will color the other cells, effectively hiding the text in that area. This can be solved by putting the \multirow in the last row with a negative ⟨*nrows*⟩ value. See, for example:

```
\begin{tabular}{l>{\columncolor{yellow}}l}
  aaaa & \\
  cccc & \\
  dddd & \multirow{-3}*{bbbb}\\
\end{tabular}
```

which will produce:

aaaa	
cccc	bbbb
dddd	

When you use colored multirow cells together with the hhline package you may find some white stripes in your colored multirow cell. For example:

```
  \begin{tabular}{|>{\columncolor{red}}c|c|}
  \hline
  \bfseries ColumnOne & \bfseries ColumnTwo\\ \hline
  First data & 932\\ \hline
   & 239\\ \hhline{|~|-|}
   & 137\\ \hhline{|~|-|}
  \multirow{-3}{*}{More data} & 319\\ \hline
  Last data & 132\\ \hline
\end{tabular}
```

ColumnOne	**ColumnTwo**
First data	932
	239
More data	137
	319
Last data	132

This can be solved by putting colored horizontal rules with the same color in the colored multirow cell.

```
  \begin{tabular}{|>{\columncolor{red}}c|c|}
  \hline
  \bfseries ColumnOne & \bfseries ColumnTwo\\ \hline
  First data & 932\\ \hline
   & 239\\ \hhline{|>{\arrayrulecolor{red}}->{\arrayrulecolor{black}}|-|}
   & 137\\ \hhline{|>{\arrayrulecolor{red}}->{\arrayrulecolor{black}}|-|}
  \multirow{-3}{*}{More data} & 319\\ \hline
  Last data & 132\\ \hline
\end{tabular}
```

ColumnOne	ColumnTwo
First data	932
More data	239
	137
	319
Last data	132

3.5 Fine-tuning the ⟨*bigstruts*⟩ argument

`\multirow` can calculate the height of the required multirow box from ⟨*nrows*⟩ and ⟨*bigstruts*⟩, supposed that all the rows don't have "unusual" heights. However, there are cases when this is not enough to properly position the box, especially when there is a `\bigstrut` on top of the first row and/or one on the bottom of the last row. In that case `\multirow` should be given additional information. This is done by prefixing the ⟨*bigstruts*⟩ argument with a letter (or two) indicating which of these two are present.

See the following examples:
(in these examples we have `\setlength{\bigstrutjot}{10pt}` to make the effect clearly visible)

Multirow	T
	X
	B

Multirow	T
	X
	B

```
\begin{tabular}{|c|c|}
  \hline
  \multirow{3}[1]{*}{Multirow} & T \bigstrut[t] \\
  \cline{2-2}
  &  X \\
  \cline{2-2}
  &  B \\
  \hline
\end{tabular}
```

In the top box in the above example the text "Multirow" should be centered, but it is a bit below the center, because of the `\bigstrut[t]` in the top row. We can correct this by giving the ⟨*bigstruts*⟩ parameter as "t 1", indicating a bigstrut in the top. This is done in the bottom box, where `\multirow{3}[t 1]{*}{Multirow}` is used.

A second example:

Multirow

	T
	X
	B

Multirow	T
	X
	B

```
\begin{tabular}{|c|c|}
  \hline
  & T \\
  \cline{2-2}
  &  X \\
  \cline{2-2}
  \multirow[t]{-3}[1]{*}{Multirow} &  B \bigstrut[b] \\
  \hline
\end{tabular}
```

In the top box the `\multirow[t]` should be positioned on the same height as the `T`, but it is too high, because there is a `\bigstrut` in the bottom. We

can correct that by specifying the ⟨*bigstruts*⟩ argument as "b 1", i.e. using `\multirow[t]{-3}[b 1]{*}{Multirow}`.

The possibilities for the prefix are:

`t` There is a bigstrut in the top, i.e. a `\bigstrut` or `\bigstrut[t]` in the top row.

`b` There is a bigstrut in the bottom, i.e. a `\bigstrut` or `\bigstrut[b]` in the bottom row.

`tb` They are both present. Note: this cannot be given as `bt`.

The space between the letter(s) and the number is optional. Please note that the prefix does not depend on whether the `\multirow` is in the top or the bottom row.

3.6 Use with longtable

It is possible to use `\multirow` in a `longtable` environment (as well as in its descendent `longtabu`). However, care must be taken that the longtable doesn't break the multirow entry when it is near the bottom of the page. For example:

...	...	...
Sept. 21	09:00	event 1
Sept. 22	10:00	event 2
Sept. 23	10:00	event 3
	12:00	event 4
	15:00	event 5
Sept. 24	09:00	event 6
...	...	...

```
\begin{longtable}{|l|l|l|}
\ldots & \ldots & \ldots \\
\hline
Sept. 21 & 09:00 & event 1 \\
\hline
Sept. 22 & 10:00 & event 2 \\
\hline
\multirow{3}*{Sept. 23} & 10:00 & event 3 \\
                        & 12:00 & event 4 \\
                        & 15:00 & event 5 \\
\hline
Sept. 24 & 09:00 & event 6 \\
\hline
\ldots & \ldots & \ldots \\
\end{longtable}
```

In this case if the "Sept. 23" entry comes close to the bottom of the page, you want to prevent the pagebreak to occur in the middle of this entry. You can do this by ending the intermediate rows with `\\*` instead of `\\`.

```
\multirow{3}*{Sept. 23} & 10:00 & event 3 \\*
                        & 12:00 & event 4 \\*
                        & 15:00 & event 5 \\
```

There is, however, a bug in longtable, that causes the `\\*` not to work if it is followed by a `\cline`, like in the following example:

```
\multirow{3}*{Sept. 23} & 10:00 & event 3 \\*
\cline{2-3}
                        & 12:00 & event 4 \\*
\cline{2-3}
                        & 15:00 & event 5 \\
```

multirow has a package option `longtable` that redefines `\cline` so that the `\\*` will also work when followed by `\cline`. The code comes from David Carlisle.

3.7 Use with supertabular

With the package supertabular (or the augmented version xtab) there is the same requirement to keep the rows of a multirow together when a pagebreak occurs. Unfortunately, supertabular does not have a way to specify that a pagebreak should be suppressed. I.e. `\\*` does not suppress a pagebreak. Therefore multirow provides a package option `supertabular` that redefines `\\*` inside a `supertabular` to suppress the pagebreak. You should use this to end the intermediate rows in a multirow block. However, this does not cause `supertabular` to consider breaking the page before the `\multirow`, contrary to `longtable`. Thus the table may become too long.

`\STneed` Therefore when the `supertabular` option is given, multirow also provides a command `\STneed` to be used in a `supertabular` that specifies how much space we need on the page. Then if there is not enough space, a pagebreak will occur at that place. For example:

```
\tabletail{\hline}
\begin{supertabular}{|l|l|l|}
\ldots & \ldots & \ldots \\
\hline
Sept. 20 & 10:00 & event 1\\
\hline
Sept. 21 & 10:00 & event 2\\
\hline
Sept. 22 & 10:00 & event 3\\
\hline
\STneed {2cm}
\multirow{3}*{Sept. 23} & 10:00 & event 4\\*
\cline{2-3}
                        & 12:00 & event 5 \\*
\cline{2-3}
                        & 15:00 & event 6 \\
\hline
 Sept. 24 & 09:00 & event 7 \\
\hline
\ldots & \ldots & \ldots \\
```

You will have to experiment a bit with the value to see what works. Sometimes it is better to exaggerate the required space a bit.

3.8 Dealing with tall entries

Sometimes there are rows that are taller than what is expected. This section gives some hints how to deal with these situations. There are two cases:

1. When there is an exceptionally tall row outside of the multirow block the positioning of the `\multirow` might be wrong. This is because `\multirow` does not have information about the heights of the rows. This can happen for

example when a large formula is entered in a cell, or a multi-line paragraph in a `[{}]` column. An example:

```
\begin{tabular}{| l | l | p{4cm} |}
  \hline
  \multirow{3}*{Week 38} & Monday & Rain most of the day\\
  \cline{2-3}
                    & Tuesday & Sunny with some clouds\\
  \cline{2-3}
                    & Wednesday & A clear day with a lot of sunshine.
                    However, the strong wind will bring down the
                    temperature. \\
  \hline
\end{tabular}
```

Week 38	Monday	Rain most of the day
	Tuesday	Sunny with some clouds
	Wednesday	A clear day with a lot of sunshine. However, the strong wind will bring down the temperature.

The `\multirow` is positioned on the second row, because it specifies that it should cover 3 rows. However, the second row is not the vertical center in this case because the third row is much taller.

To remedy this, the ⟨*vmove*⟩ parameter could be used. However, in this case it would be easier to pretend that `\multirow` spans 6 rows (the total number of lines in the last column). So use `\multirow{6}`... and we get:

Week 38	Monday	Rain most of the day
	Tuesday	Sunny with some clouds
	Wednesday	A clear day with a lot of sunshine. However, the strong wind will bring down the temperature.

2. The second case is when the `\multirow` entry is taller than the surrounding normal rows. In that case the multirow text will stick out of its block. We must now enlarge the other rows, and that is something `\multirow` cannot do.

   An example: (Don't take this as a medical advice. The names are fake anyway.)

```
\begin{tabular}{| p{2mm} l | p{5cm} |}
 \hline
 \multicolumn{2}{|l|}{\textbf{Medicine \& dose}}
                              & \textbf{Possible Side effects} \\
 \hline
 \multicolumn{2}{|l|}{Spirino}
                     & \multirow{3}={Confusion,
                       hallucinations, rapid breathing,
```

```
                      seizure (convulsions);
                      upset stomach, heartburn; severe nausea,
                      vomiting, or stomach pain or mild headache.} \\
 \cline{1-2}
  & initial: 200 mg/day & \\
 \cline{1-2}
   & maintenance: 100-400 mg/day & \\
 \hline
 \multicolumn{2}{|l|}{Conzac}
                     & \multirow{3}={Anxiety; nervousness;
                      insomnia; anorexia; mild bradycardia;
                      SA node slowing; weight loss;
                      solar photosensitivity; hyponatremia;
                      sexual dysfunction (both genders); may
                      alter glycemic control in diabetic patients.} \\
 \cline{1-2}
  & initial: 10 mg/day & \\
 \cline{1-2}
   & maintenance: 10-40 mg/day & \\
 \hline
\end{tabular}
```

Medicine & dose	**Possible Side effects**
Spirino	Confusion, hallucinations, rapid breathing, seizure (convulsions); upset stomach, heartburn; Anxiety; nervousness; insomnia; anorexia; mild bradycardia; SA node slowing; weight loss; solar photosensitivity; hyponatremia; sexual dysfunction (both genders); may alter glycemic control in diabetic patients.
initial: 200 mg/day	
maintenance: 100-400 mg/day	
Conzac	
initial: 10 mg/day	
maintenance: 10-40 mg/day	

Both `\multirow` entries are too high; the first sticks out into the second entry, and the second one sticks out of the table.

There are two ways we can correct this: The simplest would be to add extra empty rows to cover the overlapping space. For the first entry that would be 2 extra rows; for the second 4. So we add twice `& & \\` before the third `\hline`, and four of these before the last `\hline`. This gives us just the correct table:

Medicine & dose	**Possible Side effects**
Spirino	Confusion, hallucinations, rapid breathing, seizure (convulsions); upset stomach, heartburn; severe nausea, vomiting, or stomach pain or mild headache.
initial: 200 mg/day	
maintenance: 100-400 mg/day	
Conzac	Anxiety; nervousness; insomnia; anorexia; mild bradycardia; SA node slowing; weight loss; solar photosensitivity; hyponatremia; sexual dysfunction (both genders); may alter glycemic control in diabetic patients.
initial: 10 mg/day	
maintenance: 10-40 mg/day	

The second way is to stretch the normal rows vertically, such that they fit with the multirow entry. In this table, where the font size is 10pt, each row has a total height of 12pt. For the first entry we need 24pt extra (2 rows), Because this space must be divided over 3 rows that is 8pt per row, making the total height of the row 20pt. The normal row has a height of 8.4pt and a depth of 3.6pt (total 12pt). We can add 4pt on the top and 4pt on the bottom, or any other combination that adds up to 8pt. In this case I have chosen to make the height 12pt and the depth 8pt. We do this with a `\rule` with 0 width. `\newcommand{\mystrut}{\rule[-8pt]{0pt}{20pt}}` and put `\mystrut` in each of the first 3 rows. By defining your own struts you have complete control over the layout. You can choose to give some rows more space than others, or to put all the space is the last row, for example.

`\mystrut`

For the second entry we need 48pt extra (4 rows). We will use `\bigstrut` in each row, that is 16pt per row, and as a `\bigstrut` is 2`\bigstrutjot`s, we set `\bigstrutjot` to 8pt. The booktabs package adds some extra vertical space around the rules, therefore when using the normal `tabular` environment, it is probably better to make the struts a little bit bigger, or a bit smaller with booktabs. After some experimentation it appeared that a `\bigstrutjot` of `7pt` was enough. Of course we added the ⟨*bigstruts*⟩ argument of `[tb6]` to the second multirow. Please note that this is not possible with our own struts, unless we cheat.

Now with booktabs the code becomes:

```
\newcommand{\mystrut}{\rule[-8pt]{0pt}{20pt}}
\setlength{\bigstrutjot}{7pt}
\begin{tabular}{ p{2mm} l  p{5cm} }
 \toprule
 \multicolumn{2}{l}{\textbf{Medicine \& dose}}
                              & \textbf{Possible Side effects} \\
 \cmidrule(r){1-2} \cmidrule(l){3-3}
 \multicolumn{2}{l}{Spirino} \mystrut
                       & \multirow{3}={Confusion,
                         hallucinations, rapid breathing,
                         seizure (convulsions);
                         upset stomach, heartburn; severe nausea,
                         vomiting, or stomach pain or mild headache.} \\
 \cmidrule(r){1-2}
```

```
   & initial: 200 mg/day \mystrut & \\
\cmidrule(r){1-2}
    & maintenance: 100-400 mg/day \mystrut & \\
\midrule
\multicolumn{2}{l}{Conzac} \bigstrut
                    & \multirow{3}[tb6]={Anxiety; nervousness;
                     insomnia; anorexia; mild bradycardia;
                     SA node slowing; weight loss;
                     solar photosensitivity; hyponatremia;
                     sexual dysfunction (both genders); may
                     alter glycemic control in diabetic patients.} \\
\cmidrule(r){1-2}
  & initial: 10 mg/day \bigstrut & \\
\cmidrule(r){1-2}
  & maintenance: 10-40 mg/day \bigstrut  & \\
\bottomrule
\end{tabular}
```

Medicine & dose	**Possible Side effects**
Spirino	Confusion, hallucinations, rapid breathing, seizure (convulsions); upset stomach, heartburn; severe nausea, vomiting, or stomach pain or mild headache.
initial: 200 mg/day	
maintenance: 100-400 mg/day	
Conzac	Anxiety; nervousness; insomnia; anorexia; mild bradycardia; SA node slowing; weight loss; solar photosensitivity; hyponatremia; sexual dysfunction (both genders); may alter glycemic control in diabetic patients.
initial: 10 mg/day	
maintenance: 10-40 mg/day	

4 Using bigstrut

\bigstrut
\bigstrutjot

`\bigstrut[`⟨*x*⟩`]` produces a strut (a rule with width 0) which is `\bigstrutjot` (`2pt` by default) higher, lower, or both than the standard `array`/`tabular` strut. Use it in table entries that are adjacent to `\hline`s to leave an extra bit of space — according to the TeXbook (page 246), "This is a little touch that improves the appearance of boxed tables; look for it as a mark of quality."

Although you could use `\bigstrut` in an array, there isn't normally much point since arrays are 'opened up' by `\jot` anyway.

`\bigstrut[t]` adds height; `\bigstrut[b]` adds depth. Just `\bigstrut` adds both. So: Use `\bigstrut[t]` in the row just *after* an `\hline`; `\bigstrut[b]` in the row just *before*; and `\bigstrut` if there are `\hline`s both before and after.

Spaces after the `\bigstrut` are ignored, even if it has an optional argument. Spaces before the `\bigstrut` are generally ignored (by a single `\unskip`).

\bigstrutjot

Note: The multirow package makes use of `\bigstrutjot`. If both packages are used, they can be used in either order, as each checks to see if the other has

already defined `\bigstrutjot`. However, the default values they set are different: if only multirow is used, `\bigstrutjot` will be set to `3pt`. If bigstrut is used, with or without multirow, `\bigstrutjot` will be `2pt`.

5 Using bigdelim

The package is for working in a `tabular` or `array` environment, in which the multirow package is also used.

`\ldelim` `\rdelim`

Syntax of use is

`\ldelim` (`{`⟨*n*⟩`}[`⟨*vmove*⟩`]{`⟨*width*⟩`}[`⟨*text*⟩`]`
`\rdelim`) `{`⟨*n*⟩`}[`⟨*vmove*⟩`]{`⟨*width*⟩`}[`⟨*text*⟩`]`

The commands are used in a column of a `tabular` or `array`; they create a big parenthesis, brace or whatever delimiter that extends over the ⟨*n*⟩ rows starting at the one containing the command. Corresponding cells in the following rows must be explicitly given as empty cells.

The first parameter is a delimiter to be used, e.g., `\{ \} [ ] ( )` — in fact, anything that can be used with `\left` or `\right`, as appropriate.

Here is an example:

```
\begin{equation}
  \begin{array}{ccccccc}
    \ldelim({4}{4mm} & x & x & x & x & \rdelim){4}{4mm}  \\
    & x & x & x & x &  & i \\
    & x & x & x & x &  & j \\
    & x & x & x & x &  \\
    &   & u & v &   &
  \end{array}
\end{equation}
```

$$\left(\begin{array}{ccccc} x & x & x & x & \\ x & x & x & x & \\ x & x & x & x & \\ x & x & x & x & \\ & & u & v & \end{array}\right)\begin{array}{c} \\ i \\ j \\ \\ \\ \end{array} \tag{1}$$

The optional parameter ⟨*vmove*⟩ is a length used for fine-tuning: the delimiter (with the optional ⟨*text*⟩) will be raised (or lowered, if ⟨*vmove*⟩ is negative) by that length above (below) wherever it would otherwise have gone. This is just like with `\multirow`, but note that here the ⟨*vmove*⟩ goes before the ⟨*width*⟩.

When `\ldelim` is used, the optional ⟨*text*⟩ is set centred to the left of `\ldelim`. If `\rdelim` is used it is set to the right of `\rdelim`. The ⟨*width*⟩ parameter is the space that is reserved for the delimiter and its text; as with the multirow package, the ⟨*width*⟩ may be given as `*`. Compare for example these:

```
\begin{tabular}{p{2em}l}
  \ldelim\{{3}{*}[type] & dvi \\
                        & ps \\
                        & pdf \\
\end{tabular}
```

$$\text{type}\left\{\begin{array}{l}\text{dvi}\\ \text{ps}\\ \text{pdf}\end{array}\right.$$

```
\begin{tabular}{l@{\,}l}
  \ldelim\{{3}{*}[type] & dvi \\
                        & ps \\
                        & pdf \\
\end{tabular}
```

type { dvi
ps
pdf

In the first example we cheated: by using a column width that is too small, we swallowed up some of the intercolumn space, at the cost of an "Overfull `\hbox`" message. In the second example we did it the proper way by inserting `@{\,}` to replace the default intercolumn space with a narrow space.

Also the commands may be used in the last row of the extension with a negative $\langle n \rangle$ argument. This is useful in combination with the colortbl or xcolor packages (see the discussion in section 3 on multirow). If there are unusually tall rows you may have to enlarge $\langle n \rangle$ (you can use fractional values). If you have horizontal lines that interact with the braces you are advised to use the hhline package to make the lines.

If you decrease or eliminate the intercolumn space with `@{}` and use colored backgrounds with the colortbl or xcolor packages (commands `\columncolor`, `\rowcolor`, `\rowcolors` or `\cellcolor`), you will notice that part of the brace will be cut off. In reality it will be overwritten with the color of the next cell. See this example:

```
\rowcolors{2}{green!25}{green!75}
\begin{tabular}{ c @{}c c c }
                  & 1 & 2 & 3 \\
                  & 4 & 5 & 6 \\
\ldelim\{{-3}{*} & 7 & 8 & 9 \\
\end{tabular}
```

{ 1 2 3
4 5 6
7 8 9

This is not a problem of multirow or bigdelim; it will also happen if there is normal text in the column before the `@{}`. The reason is that these color commands extend the color to cover the intercolumn spaces on both sides to prevent gaps in the color. The size of these so-called *overhangs* is `\tabcolsep` (or `\arraycolsep` when an `array` is used) on each side. However, when `@{}` is used there is no such intercolumn space and the extension covers parts of the previous column. This can be cured by setting the left *overhang* explicitly to 0pt with a `\columncolor` command in the tabular header, like `>{\columncolor{white}[0pt][\tabcolsep]}`. Unfortunately the explicit color `white`, removes the transparency of the column, so if there are cells in that column that have no explicit color, these cells are affected. If the background of the `tabular` is white, this normally will not be noticed, but if the background color is changed, for example with the `\pagecolor` command, then that color should be used rather than `white`. Unfortunately, there is no command to specify the *overhangs* without also specifying a color.

In the following example we have done this. In order to keep the table header compact, we put the deinition in a `\newcolumn` command (using the array package).

```
\newcolumntype{z}{@{}>{\columncolor{white}[0pt][\tabcolsep]}}
\rowcolors{2}{green!25}{green!75}
\begin{tabular}{ c zc c c }
                & 1 & 2 & 3 \\
                & 4 & 5 & 6 \\
\ldelim\{{-3}{*} & 7 & 8 & 9 \\
\end{tabular}
```

$$\left\{\begin{array}{ccc} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{array}\right.$$

In case you want to have a paragraph type text as optional parameter you could put it in a `\parbox`. Alternatively you could add an extra column with the text in a `\multirow`, like in

```
\begin{tabular}{l@{}l@{}l}
   dvi   & \rdelim\}{3}{1em} & \multirow{3}{4cm}{These are the output types,
        that are commonly used for \TeX.} \\
   ps    & & \\
   pdf   & & \\
\end{tabular}
```

dvi
ps
pdf } These are the output types, that are commonly used for TEX.

Note that we used `@{}` to eliminate the intercolumn space to get the text tight to the brace.

6 Contact Information

Pieter van Oostrum
email: pieter@vanoostrum.org
www: http://pieter.vanoostrum.org

The source code can be found on Github:
https://github.com/pietvo/multirow
Bugs can be reported at
https://github.com/pietvo/multirow/issues

7 Implementation

7.1 The multirow package

`\ifmultirowdebug` `\multirowdebugtrue` `\multirowdebugfalse`

This is a boolean to [de]activate debugging (showing the generated box contents). It is activated by the `debug` package option. The `\newif` initializes it to false.

```
\newif\ifmultirowdebug
\DeclareOption{debug}{\multirowdebugtrue}
```

`\cline`

The package option `longtable` redefines the `\cline` macro to work around a bug in longtable. See section 3.6[2]. First we check if the macro `\CT@arc` is defined. If

[2] Thanks to David Carlisle. See his answer on stackexchange.

so, this indocates that the colortbl package is loaded. As colortbl also redefines `\@cline`, we must take this into account with our own redefinition of `\@cline`.

```
\DeclareOption{longtable}{%
\AtBeginDocument{%
\@ifundefined{CT@arc}
{\def\@cline#1-#2\@nil{%
  \omit
  \@multicnt#1%
  \advance\@multispan\m@ne
  \ifnum\@multicnt=\@ne\@firstofone{&\omit}\fi
  \@multicnt#2%
  \advance\@multicnt-#1%
  \advance\@multispan\@ne
  \leaders\hrule\@height\arrayrulewidth\hfill
  \cr
  \noalign{\nobreak\vskip-\arrayrulewidth}}}
{\def\@cline#1-#2\@nil{%
  \omit
  \@multicnt#1%
  \advance\@multispan\m@ne
  \ifnum\@multicnt=\@ne\@firstofone{&\omit}\fi
  \@multicnt#2%
  \advance\@multicnt-#1%
  \advance\@multispan\@ne
  {\CT@arc@\leaders\hrule\@height\arrayrulewidth\hfill}%
  \cr
  \noalign{\nobreak\vskip-\arrayrulewidth}}}
}}
```

The package option `supertabular` redefines `\\*` inside a `supertabular`. The redefinition is delayed until the `\begin{document}`.

`Supertabular` version 4.1f and later need a call to `\ST@save@lineno` to function properly, but earlier versions cannot use this as it doesn't exist in these versions. So the definitions of both `\ST@tabularcr` and `\MRST@cr` are different depending on whether `\ST@save@lineno` is defined. There are also some other differences, so some intermediate versions of `supertabular` might need more subtle adaptations, but for now we leave it at that.

```
\DeclareOption{supertabular}{%
\AtBeginDocument{%
```

`\ST@tabularcr` This macro is the definition of `\\` inside a `supertabular`. We check for a `*`, and if it is present we call our own version, otherwise the `supertabular` version. First we get the older version for pre-4.1f `supertabular`, then the newer version.

```
\ifx\ST@save@lineno\undefined
\def\ST@tabularcr{%
  {\ifnum0=`}\fi
  \@ifstar{\MRST@xtabularcr}{\ST@xtabularcr}}
```

`\MRST@cr` `\MRST@cr` is a truncated copy of `\ST@cr`. It does all the bookkeeping about the space the `longtable` occupies, but it doesn't do the pagebreaking part.

```
\def\MRST@cr{%
```

```
  \noalign{%
    \ifnum\ST@pboxht<\ST@lineht
      \global\advance\ST@pageleft -\ST@lineht
      \global\ST@prevht\ST@lineht
    \else
      \global\advance\ST@pageleft -\ST@pboxht
      \global\advance\ST@pageleft -0.1\ST@pboxht
      \global\advance\ST@pageleft -\ST@stretchht
      \global\ST@prevht\ST@pboxht
      \global\ST@pboxht\z@
    \fi
    \global\advance\ST@pageleft -\ST@toadd
    \global\ST@toadd=\z@}}
\else
```

These are the newer versions.

```
\def\ST@tabularcr{%
  {\ifnum0=`}\fi
  \ST@save@lineno
  \@ifstar{\MRST@xtabularcr}{\ST@xtabularcr}}
\def\MRST@cr{%
  \noalign{%
    \ifnum\ST@pboxht<\ST@lineht
      \global\advance\ST@pageleft -\ST@lineht
      \global\ST@prevht\ST@lineht
    \else
     \ST@trace@cr\thr@@{Added par box with height \the\ST@pboxht}%
      \global\advance\ST@pageleft -\ST@pboxht
      \global\advance\ST@pageleft -0.1\ST@pboxht
      \global\ST@prevht\ST@pboxht
      \global\ST@pboxht\z@
    \fi
    \global\advance\ST@pageleft -\ST@toadd
    \global\ST@toadd=\z@
    \ST@trace@cr\thr@@{Space left for tabular: \the\ST@pageleft}}}
\fi
```

\MRST@xtabularc
\MRST@argtabularc
\MRST@xargtabularc
\MRST@yargtabularc

These are copies of the corresponding macros from supertabular, but instead of \ST@cr they call \MRST@cr.

```
\def\MRST@xtabularcr{%
  \@ifnextchar[%]
    {\MRST@argtabularcr}%
    {\ifnum0=`{\fi}\cr\MRST@cr}}
\def\MRST@argtabularcr[#1]{%
  \ifnum0=`{\fi}%
  \ifdim #1>\z@
    \unskip\MRST@xargarraycr{#1}
  \else
    \MRST@yargarraycr{#1}%
  \fi}
\def\MRST@xargarraycr#1{%
  \@tempdima #1\advance\@tempdima \dp \@arstrutbox
  \vrule \@height\z@ \@depth\@tempdima \@width\z@ \cr
  \noalign{\global\ST@toadd=#1}\MRST@cr}
```

```
\def\MRST@yargarraycr#1{%
  \cr\noalign{\vskip #1\global\MRST@toadd=#1}\MRST@cr}
}
```

`\STneed` This macro can be used in a `supertabular` to indicate how much space a multirow entry needs. See section 3.7.

```
\def\STneed#1{\ifdim\ST@pageleft<#1\ST@newpage\ST@next\fi}
}
```

```
\ProcessOptions
```

`\multirow@colwidth` is a length that is used to implement the "=" variant of ⟨*width*⟩.

`\multirow@colwidth` `\multirow@colwidth` is a length that is used to implement the "=" variant of ⟨*width*⟩.

```
\newlength{\multirow@colwidth}
```

`\multirow@cntb` `\multirow@dima` Define two counters and a length for internal use in `\multirow`.

```
\newcount\multirow@cntb
\newlength\multirow@dima
```

`\multirow@setcolwidth` This macro calculates `\multirow@colwidth` for an entry that has the ⟨*width*⟩ given as "=". We check if we are inside a `tabulary` environment, by checking if `\TY@final` is defined. If not, then `\multirow@colwidth` = `\hsize`. The `tabulary` environment will make two passes. On the first pass, we set `\multirow@colwidth` to the size that the text would have in LR mode (with newlines replaced by spaces), so that `tabulary` will gives us enough space. On the second pass (characterized by `\TY@box` = `\TY@box@v`) we use the value that `tabulary` has given us in `\hsize`. This algorithm is not perfect, but good enough in most cases.

```
\long\def\multirow@setcolwidth#1{%
    \ifx\TY@final\multirow@undefined \multirow@colwidth=\hsize
    \else
      \ifx\TY@box\TY@box@v\multirow@colwidth=\hsize
      \else \setbox0\hbox
        {\let\\\space\let\newline\space #1}\multirow@colwidth=\wd0
      \fi
    \fi}
```

`\multirowsetup` `\multirowsetup` is executed at the beginning of each `\multirow`.

```
\newcommand\multirowsetup{\raggedright}
```

`\multirow@vbox` This creates the `\vbox`. Parameters:
`#1` = ⟨*vpos*⟩, `#2` = initialization code (for example to set the width of the `\parbox`), `#3` = box contents.
Depending on the ⟨*vpos*⟩ parameter, it will be top-aligned, vertically centered, or bottom-aligned. This is done by inserting `\vfill` in the proper places.
Note: the `\relax` is to protect against an empty ⟨*vpos*⟩ argument.

```
\long\def\multirow@vbox#1#2#3{\setbox0\vtop to \multirow@dima{#2%
    \if #1t\relax\else\vfill\fi
    \multirowsetup #3\if #1b\relax\else\vfill\fi}}
```

\multirow Make an entry that will span multiple rows of a table.
First collect all the arguments and replace missing optional arguments by their default values.

```
%% \multirow [vpos] {nrows} [bigstruts] {width} [vmove] {text}
\newcommand\multirow[2][c]{\@multirow[#1]{#2}}
\def\@multirow[#1]#2{\@ifnextchar[{\@@multirow[#1]#2}{\@@multirow[#1]#2[0]}}
\def\@@multirow[#1]#2[#3]#4{\@ifnextchar[{\@xmultirow[#1]{#2}[#3]{#4}}%
                                  {\@xmultirow[#1]{#2}[#3]{#4}[0pt]}}
```

\multirow@piii
\ifmultirow@prefixt
\multirow@prefixttrue
\multirow@prefixtfalse
\ifmultirow@prefixb
\multirow@prefixbtrue
\multirow@prefixbfalse

This macro splits off a `t`, `b`, or `tb` prefix of the ⟨*bigstruts*⟩ argument, and sets `\multirow@cntb` to the numerical value. The prefix is remembered in two booleans: `\ifmultirow@prefixt` and `\ifmultirow@prefixb`.

```
\newif\ifmultirow@prefixt
\newif\ifmultirow@prefixb
\def\multirow@piii#1#2#3\end{\multirow@prefixtfalse\multirow@prefixbfalse
  \if t#1\multirow@prefixttrue
    \if b#2\multirow@prefixbtrue \multirow@cntb=#3%
    \else \multirow@cntb=#2#3%
    \fi
  \else
    \if b#1\multirow@prefixbtrue \multirow@cntb=#2#3%
    \else \multirow@cntb=#1#2#3%
    \fi
  \fi}
```

This is the real workhorse. It starts with splitting the ⟨*bigstruts*⟩ argument, and then calculating the height of the multirow box. Because ⟨*nrows*⟩ (`#2`) can be fractional, we cannot use `\ifnum` to test for positive or negative. Therefore we use `\ifdim` by putting a unit (`pt`) after the number.

```
\long\def\@xmultirow[#1]#2[#3]#4[#5]#6{%
  \expandafter\multirow@piii#3\relax\end%
  \setlength\multirow@dima{#2\ht\@arstrutbox}%
  \addtolength\multirow@dima{#2\dp\@arstrutbox}%
  \ifdim#2pt<\z@\setlength\multirow@dima{-\multirow@dima}\fi
  \addtolength\multirow@dima{\multirow@cntb\bigstrutjot}%
```

The text is set in a `\vbox` by calling `\multirow@vbox`.
If the ⟨*width*⟩ argument is `*` set just the text in the `\vbox`.

```
  \if*#4\multirow@vbox{#1}{}{\hbox{\strut#6\strut}}%
```

Otherwise set it in a `\parbox` inside a `\vbox`.
If the ⟨*width*⟩ argument is given as "`=`", we calculate `\multirow@colwidth` and use that as width of the `\parbox`.

```
  \else \if=#4\multirow@setcolwidth{#6}%
    \multirow@vbox{#1}{\setlength\hsize{\multirow@colwidth}\@parboxrestore}{\strut#6\strut\par
```

Otherwise the given argument is used as the width of the `\parbox`.

```
  \else \multirow@vbox{#1}{\setlength\hsize{#4}\@parboxrestore}{\strut#6\strut\par}%
  \fi \fi
```

Now position the `\vbox` properly. More details are given in the appendix. The overview of the calculation of the shift amount can be found in section A.3.

If ⟨*nrows*⟩ > 0:
If ⟨*vpos*⟩ = `[t]`, then the box is already positioned correctly (the baseline is on

the baseline of the row). However, later the top of the box will be taken as the reference point (instead of the baseline), therefore we take the height of the box (h) as the shift amount. See fig. 1.
If ⟨*vpos*⟩ = `[c]` we shift it up h1 (see fig. 2), where h1 = `\ht\@arstrutbox` + (`\bigstrutjot \ifmultirow@prefixt`).
If ⟨*vpos*⟩ = `[b]` we shift it up h1 + h2 (see fig. 3), where h2 = `\dp\@arstrutbox` + (`\bigstrutjot \ifmultirow@prefixb`).
We calculate the required shift in `\multirow@dima`.

```
\ifdim#2pt>\z@
  \if#1t\relax\setlength\multirow@dima{\ht0}\else
    \setlength\multirow@dima{\ht\@arstrutbox}%
    \ifmultirow@prefixt \addtolength\multirow@dima{\bigstrutjot}\fi
    \if#1b\relax \addtolength\multirow@dima{\dp\@arstrutbox}%
      \ifmultirow@prefixb \addtolength\multirow@dima{\bigstrutjot}\fi
    \fi
  \fi
```

If ⟨*nrows*⟩ < 0:
If ⟨*vpos*⟩ = `[t]`, shift the box up H − h1 − h2 + h. See fig. 4.
If ⟨*vpos*⟩ = `[c]`, shift the box up H − h2. See fig. 5.
If ⟨*vpos*⟩ = `[b]`, shift the box up H. See fig. 6.
H is the current value of `\multirow@dima`.

```
\else
  \if#1b\relax\else
    \addtolength\multirow@dima{-\dp\@arstrutbox}%
    \ifmultirow@prefixb \addtolength\multirow@dima{-\bigstrutjot}\fi
    \if#1t\relax\addtolength\multirow@dima{-\ht\@arstrutbox}%
      \ifmultirow@prefixt \addtolength\multirow@dima{-\bigstrutjot}\fi
      \addtolength\multirow@dima{\ht0}%
    \fi
  \fi
\fi
```

Finally, we add the ⟨*vmove*⟩ argument (`#5`), and go into horizontal mode. Then we shift the box up by putting a `\vskip` above it, and add it to the output. Because of the `\vskip` the resulting box will have a height 0. We set the depth of the `\vbox` to 0, so that it will not influence the depth of the current row.
If `\multirowdebug` is true, we show the box.

```
  \addtolength\multirow@dima{#5}%
  \leavevmode
  \setbox0\vtop{\vskip-\multirow@dima\box0\vss}\dp0=\z@
  \ifmultirowdebug{\showboxdepth=5 \showboxbreadth=10 \showbox0}\fi
  \box0
}
```

`\bigstrutjot` Define `\bigstrutjot` if not already defined.

```
\@ifundefined{bigstrutjot}{\newdimen\bigstrutjot \bigstrutjot=\jot}{}
```

7.2 The bigstrut package

`\bigstrutjot` This is a length. By default it is set to 2pt. You can change it with the `\setlength` command.

```
\@ifundefined{bigstrutjot}{\newdimen\bigstrutjot}{}\bigstrutjot=2pt
```

\bigstrut This macro inserts a strut. Depending on the optional parameter it extends above and/or below the standard `array`/`tabular` strut.

```
\newcommand\bigstrut[1][x]{%
  \leavevmode\unskip\@tempdima=\ht\@arstrutbox \@tempdimb=\dp\@arstrutbox
  \ifx #1b\relax \else \advance\@tempdima by \bigstrutjot\fi
  \ifx #1t\relax \else \advance\@tempdimb by \bigstrutjot\fi
  \hbox{\vrule \@height\@tempdima \@depth\@tempdimb \@width\z@}\ignorespaces}
```

7.3 The bigdelim package

```
\RequirePackage{multirow}
```

\ldelim This macro typesets a left delimiter. It calls `\multirow` with the proper arguments. The size of the delimiter is determined by putting a `\vbox` with the proper height and zero width next to it. The height is the one that `\multirow` already has calculated in `\multirow@dima`. That calculation uses the size of `\@arstrutbox`, which is set by `tabular` or `array` environments. In case it is not set, we initialize it to a default value.

```
\newcommand\ldelim[2]{\@ifnextchar[{\@ldelim{#1}{#2}}{\@ldelim{#1}{#2}[0pt]}}
\def\@ldelim#1#2[#3]#4{\@ifnextchar[{\@@ldelim{#1}{#2}{#3}{#4}}{\@@ldelim{#1}{#2}{#3}{#4}[\nul
\def\@@ldelim#1#2#3#4[#5]%
 {\ifvoid\@arstrutbox\setbox\@arstrutbox\hbox{\strut}\fi
  \multirow{#2}{#4}[#3]{%
   \ensuremath
     {\left.\vcenter{\hsize=0pt\vrule height \multirow@dima width 0pt}%
     \textnormal{#5}\right#1}}}
```

\rdelim This macro typesets a right delimiter. It calls `\multirow` with the proper arguments, similar to `\ldelim`.

```
\newcommand\rdelim[2]{\@ifnextchar[{\@rdelim{#1}{#2}}{\@rdelim{#1}{#2}[0pt]}}
\def\@rdelim#1#2[#3]#4{\@ifnextchar[{\@@rdelim{#1}{#2}{#3}{#4}}{\@@rdelim{#1}{#2}{#3}{#4}[\nul
\def\@@rdelim#1#2#3#4[#5]%
 {\ifvoid\@arstrutbox\setbox\@arstrutbox\hbox{\strut}\fi
  \multirow{#2}{#4}[#3]{%
   \ensuremath
     {\left#1\vcenter{\hsize=0pt\vrule height \multirow@dima width 0pt}%
       \textnormal{#5}\right.}}}
```

A Appendix

In this section we explain the `\vbox` positioning in `\multirow`. The positioning depends on the ⟨*nrows*⟩, ⟨*vpos*⟩, ⟨*bigstruts*⟩ and ⟨*vmove*⟩ arguments. The box is constructed with `\vtop`. The algorithm of `\vtop` is described in *The TEXbook*, p. 81.

Each case is described by a figure. In the figure the lefthand column indicates the context of the tabular in which the multirow appears, i.e ⟨*nrows*⟩ rows. The righthand column is the multirow box that is to be inserted. The baseline is the natural position where the material will be positioned in the first place. Later it will be shifted up to the desired location.

H is the calculated height of the box: ⟨*nrows*⟩ × the natural height of a row + ⟨*bigstruts*⟩ × `\bigstrutjot`.

topstrut = `\bigstrutjot` if there is a `\bigstrut` on the top of the first row (as indicated by the `t` prefix in the ⟨*bigstruts*⟩ argument), otherwise 0.

botstrut = `\bigstrutjot` if there is a `\bigstrut` on bottom of the last row (as indicated by the `b` prefix in the ⟨*bigstruts*⟩ argument), otherwise 0.

h1 = height of a tabular row + topstrut

h2 = depth of a tabular row + botstrut

Note: the following descriptions describe the vertical shift of the box without taking the ⟨*vmove*⟩ into account. In all cases ⟨*vmove*⟩ has to be added if it is given.

A.1 Case ⟨*nrows*⟩ > 0

⟨*vpos*⟩ = [t]

In this case the `\vbox` contains the text followed by a `\vfill`. Such a `\vbox` has a height that is the height of the top line of the text (h). H = height + depth of the box. This means that the box is already positioned correctly. However, later we will put the box inside another `\vbox`, with a `\vskip` on to of it, and this will make the top of the box its reference point. Therefore we will have to shift it up again over a distance h (which probably will be different from the height of the tabular row). So the total shift becomes h. See fig. 1.

Alternatively, we could have omitted the `\vskip` in this case, thereby leaving the baseline undisturbed, but this would make the code unsymmetrical. Moreover, this would not work when a non-zero ⟨*vmove*⟩ is present. Therefore we choose to set the shift amount to h here.

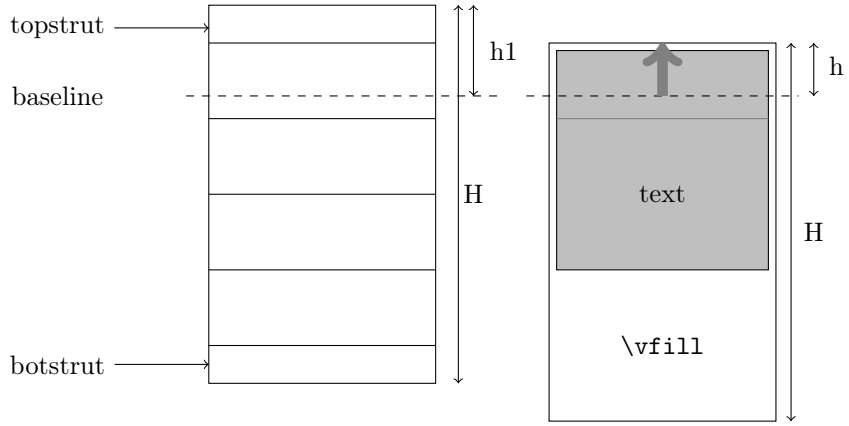

Figure 1: Case ⟨*nrows*⟩ > 0, ⟨*vpos*⟩ = [t]

⟨*vpos*⟩ = [c]

In this case the `\vbox` contains a `\vfill`, the text, and another `\vfill`. Such a `\vbox` has a height 0, i.e. the top of the box is on the baseline. Because both boxes have the same size (H), they can be aligned by shifting the `\vbox` up over h1. See fig. 2.

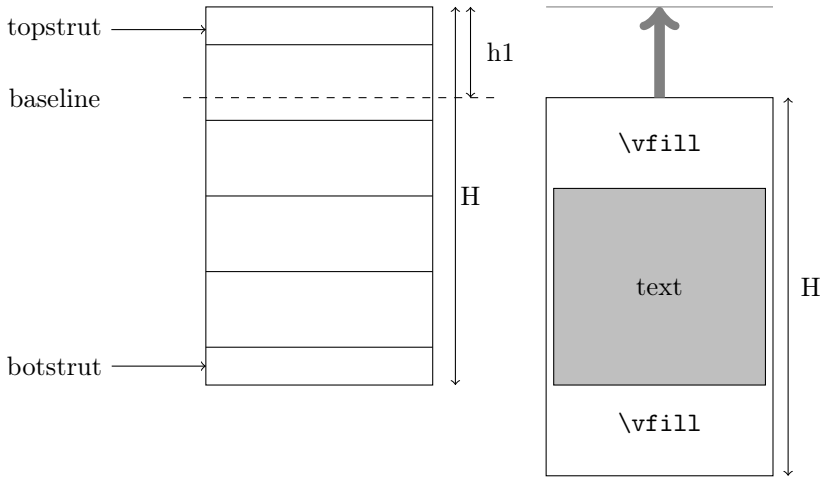

Figure 2: Case $\langle nrows \rangle > 0$, $\langle vpos \rangle$ = [c]

$\langle$ ***vpos*** $\rangle$ = **[b]**

Now the `\vbox` contains a `\vfill`, followed by the text. Because it ends with the text, it gets an additional depth equal to the depth of the last line of the text. Such a `\vbox` has a height 0, i.e. the top of the box is on the baseline, but its depth is H + that depth. In other words the baseline of the last text line is H below the top.

Because $\langle vpos \rangle$ = `[b]` we want the baseline of the last textline to shift to the baseline of the last tabular row. The amount of the shift is h1 + h2. See fig. 3.

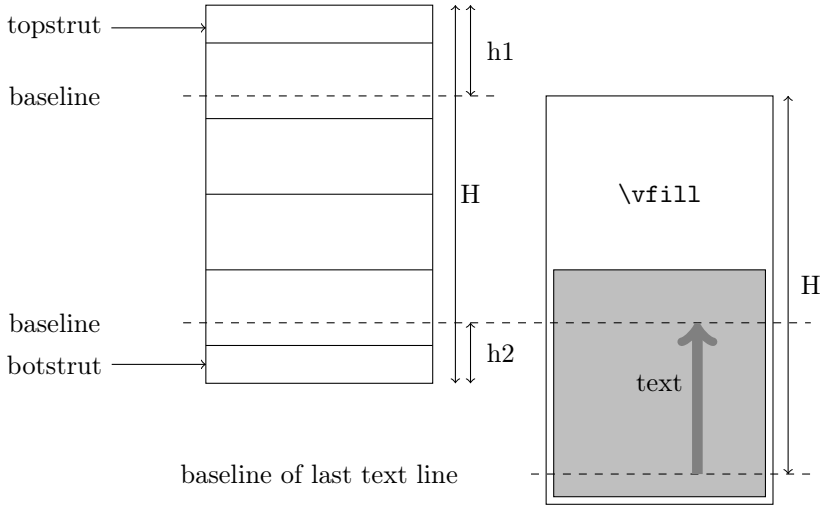

Figure 3: Case $\langle nrows \rangle > 0$, $\langle vpos \rangle$ = [b]

A.2 Case $\langle nrows \rangle < 0$

$\langle nrows \rangle < 0$ when the multirow is positioned in the last row of the multirow block.

$\langle vpos \rangle$ = [t]

In this case the `\vbox` contains the text followed by a `\vfill`. Such a `\vbox` has a height that is the height of the top line of the text. The baseline is aligned with the baseline of the last row. Because $\langle vpos \rangle$ = [t], we want it to be aligned with the baseline of the first row. Therefore it has to be shifted up H − h1 − h2. But because later the height of the box will be set to 0, we must also add the current height h. Therefore the total shift becomes H − h1 − h2 + h. See fig. 4.

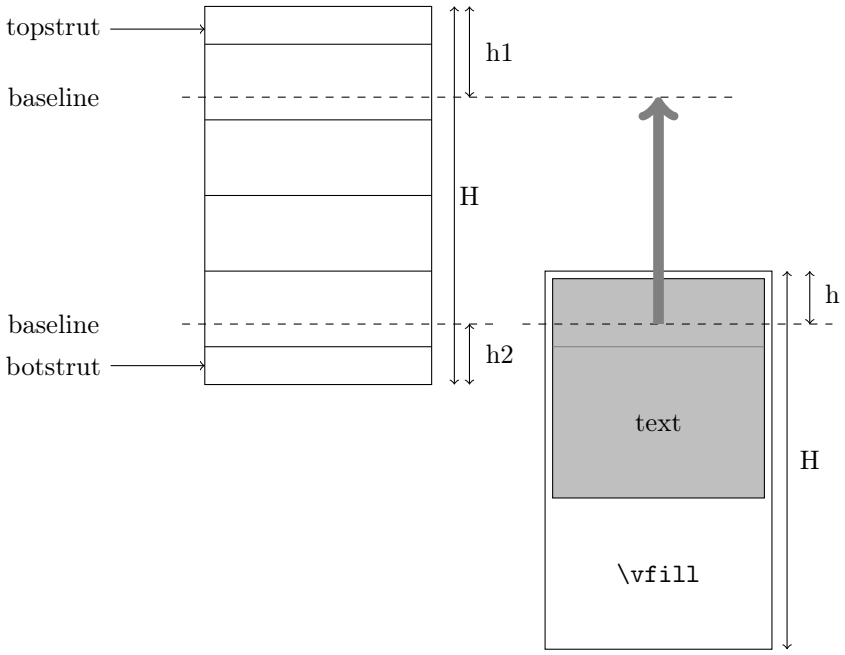

Figure 4: Case $\langle nrows \rangle < 0$, $\langle vpos \rangle$ = [t]

$\langle vpos \rangle$ = [c]

In this case the `\vbox` contains a `\vfill`, the text, and another `\vfill`. Such a `\vbox` has a height 0, i.e. the top of the box is on the baseline. Because both boxes have the same size (H), they can be aligned by shifting the `\vbox` up over H − h2. See fig. 5.

$\langle vpos \rangle$ = [b]

The `\vbox` contains a `\vfill`, followed by the text. Because it ends with the text, it gets an additional depth equal to the depth of the last line of the text. Such a `\vbox` has a height 0, i.e. the top of the box is on the baseline, but its depth is H + that depth. In other words the baseline of the last text line is H below the top.

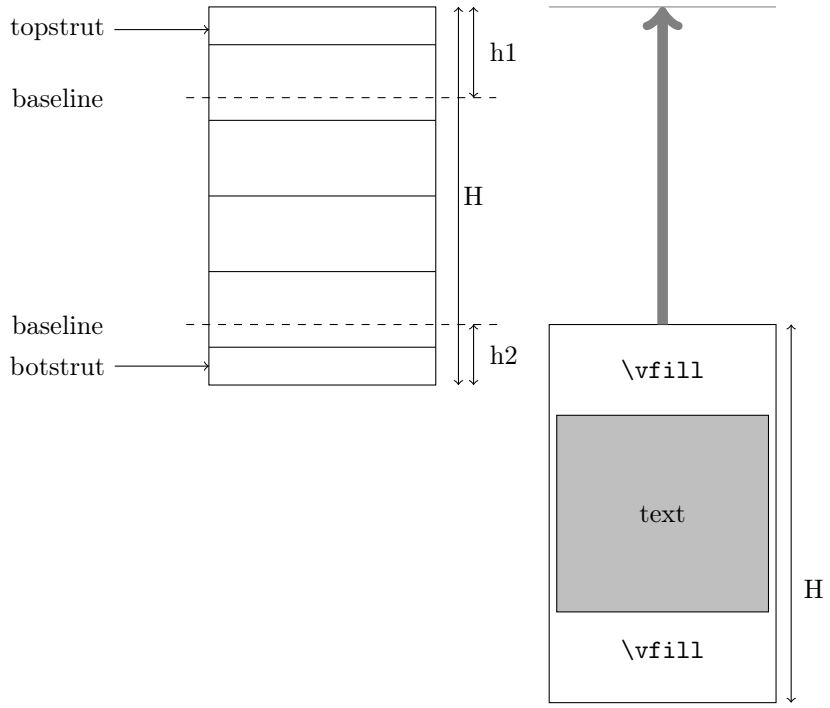

Figure 5: Case $\langle nrows \rangle < 0$, $\langle vpos \rangle$ = [c]

Because $\langle vpos \rangle$ = `[b]` we want the baseline of the last textline to shift to the baseline of the last tabular row. The amount of the shift is H. See fig. 6.

A.3 Overview

$\langle vpos \rangle$	$\langle nrows \rangle > 0$	$\langle nrows \rangle < 0$
`[t]`	h	H − h1 − h2 + h
`[c]`	h1	H − h2
`[b]`	h1 + h2	H
	x	H − h1 − h2 + x

Change History

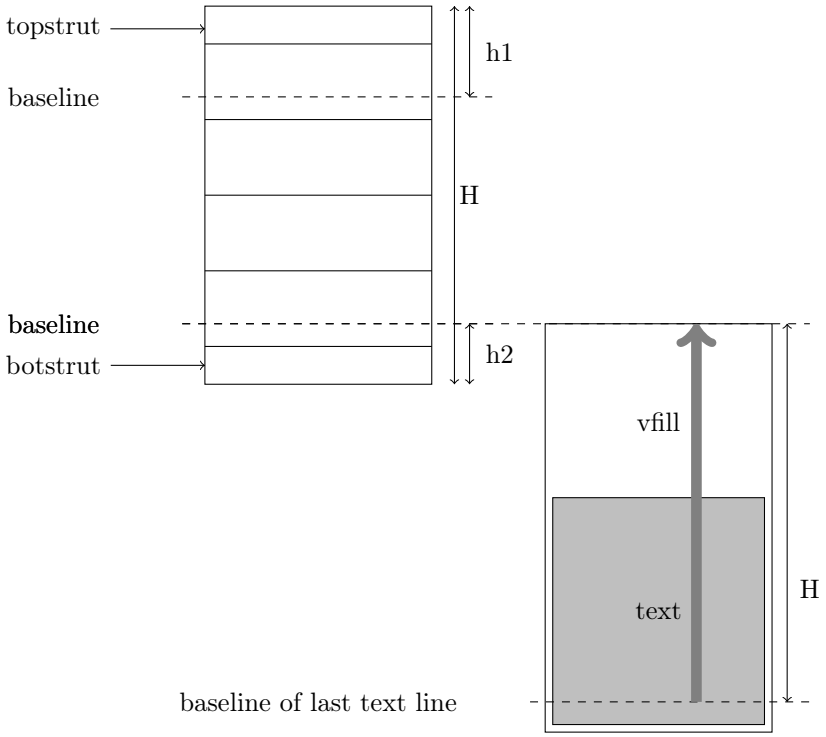

Figure 6: Case $\langle nrows \rangle < 0$, $\langle vpos \rangle$ = [b]

Index

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