The **youngtab** package[*]

Volker Börchers[†]
Stefan Gieseke
Universität Bremen

1999/05/19

Abstract

This package provides two macros for typesetting Young-Tableaux with TEX or LATEX. We use 2 macros, the first for empty boxes (`\yng`), and the second for labelled boxes (`\young`). The syntax of the macros is simple to avoid spelling-mistakes.

Contents

1 Introduction

The Young-Tableau-formalism is a strong algrebraic tool in group theory. Of course we will deal here only with the problems of *typesetting* Young-Tableaux. They are mathematical objects, so they usually occur in mathematical environments. On the surface Young-Tableaux consist of boxes like this: $\boxed{}$ — or optionally labelled (usually with letters): $\boxed{a}$.

There is already one LATEX-package for typesetting Young-Tableaux, the `young.sty` by Jörg Knappen (based on `young.tex` by P. E. S. Wormer). It provides a `Young`-environment with a syntax similar to the `tabular`-environment.

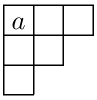
An Example for the usage of `Young`:

---

[*] This file has version number v1.1;, dated 1999/05/19.

[†] email: boercher@physik.uni-bremen.de

```
\begin{Young}
  $a$&&\cr
  &\cr
  \cr
\end{Young}
```



This way of implementing Young-Tableaux has one big advantage: It can deal with an indefinite number of rows and columns (of course this is needed!). A second argument for using a `tabular`-like environment might be it's simple encoding.

Nevertheless we wanted a simpler user-interface, for the use of `&` and `\cr` is usually a very good source of typing mistakes (e.g. you have to type *three* `&`'s to get *four* boxes in one row). In addition the TEX-code for mathematical relations with more than two or three Young-Tableaux becomes unreadable.

The cure would be of course a *macro* with a compact, intuitive syntax. But there is the above mentioned problem: TEX is not made to handle an indefinite number of arguments. For Young-Tableaux composed of empty boxes we would like to have a macro with one argument for each row (counting the number of boxes per row) for example this way: `\thisdoesnotwork{5}{3}{1}`.... For Young-Tableaux composed of labelled boxes the problem is even worse since we have one letter *per box*!

So we had to hide the fact of having an indefinite number of arguments from TEX. The macros in this package have — as TEX sees it — only one argument; this is parsed for the necessary information by the macros themselves. Whereas, for the user, the number of rows and number of boxes per row are plain to see.

2 Using the youngtab package

Invoke the `youngtab` package by requesting it in the preamble:[1]

`\usepackage[...]{youngtab}`

`Options`
`vcentermath`
`noautoscale`
`stdtext`

The package has four options (increasing). The *first option*, `vcentermath`, *turns on* the vertical centering of the tableaux in math mode. The *second option*, `noautoscale`, *turns off* the automatic scaling of the boxes according to the actual font. The *third one*, `stdtext`, switches to text mode in the inside of labelled boxes. Choosing one of this options does not mean that you have, for example, to stick to `noautoscale` till the end of your days. You may change this settings as often as you like.

`enableskew`
New feature
1998/05/05

The *fourth option*, `enableskew` enables typesetting of skew Young-Tableaux (tableaux that are not left-aligned). This option is different from the other in the way that there is no command to enable this feature after the package is loaded.

`vcentermath`

Young-Tableaux usually have more than one row, so the vertical alignment is a problem. You can turn on vertical centering (only) in mathmode either by choosing the option `vcentermath` with `\usepackage` or later with the command `\Yvcentermath`⟨*num*⟩. Say `\Yvcentermath1` if you want vertical centering and `\Yvcentermath0` to switch it off. Without vcentering the tableaux are standing on the baseline.

---

[1] Of course this does not work if you use the `youngtab`-package with TEX instead of LATEX. For compatibility-questions see section 2.3

`\Yautoscale`

Since the boxes are quadratic (if you do not want want this, complain!), there is only one dimension to determine: the length and height of the boxes. Usually the macros determine this for themselves,[2] but you can change this behavior by calling the macro `\Yautoscale`⟨*num*⟩. Call it with an argument of `1` if you want autoscaling after this, or with the argument `0` if you want to adjust the box dimensions yourself.

`\Yboxdim`

If you switch off autoscaling and want to change the box dimensions, you have to call the macro `\Yboxdim`⟨*dim*⟩ with a usual TEX–dimension. The argument has to be appended to the macro: `\Yboxdim13pt`. This dimension is the *total* height and width of the boxes – *including the lines.* (For small linethicknesses this makes of course only a small difference.) Note that the use of `\Yboxdim` *implies* `noautoscale`! So to switch from `autoscale` to `noautoscale` one call to `\Yboxdim` is sufficient.

`\Ylinethick`

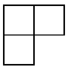
There is only one more parameter which determines the appearance of the Young-Tableau: the lines' thickness. This is set by `\Ylinethick`⟨*dim*⟩. (The default is $0.3pt$).

`\Yinterspace`

You may add space between a tableau and surrounding stuff by invoking the macro `\Yinterspace`⟨*skip*⟩. The advantage of using a *skip* is that this helps TEX to avoid `Over-` or `Underful \hboxes`. The default value is `0ex plus 0.3ex`. (Better take `ex` or `em` for the tableaux are scaleable.)

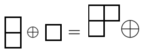
2.1 Empty boxes: \yng

`\yng`

The most used variant of Young-Tableaux are those consisting of empty boxes. For the tableau  you have to type `\yng(2,1)`. If you want it to appear smaller, change the fontsize: `{\tiny\yng(2,1)}` gives ; but remember, this only works if `autoscale` is turned on (default). (If you are in `noautoscale`-mode, restore autoscaling with `\Yautoscale1`.)

`math or text?`

Some notes on the usage of `\yng` and `\young` in math or text mode: You can generate Young-Tableaux in both modes, but differences arise from *vertical centering.* This only works in math mode (if you have chosen it with `\Yvcentermath1`) or `usepackage` option `vcentermath`.

The difference is to be seen here: ⊕ . . . , which has been generated by

```
{\tiny\Yvcentermath1 $\yng(1,1)\oplus\yng(1)=$
  \yng(2,1)}$\oplus$ \dots
```

`sizing`

Here the tableau in math are centered with respect to the axis of the formula while the one in text mode isn't. Note the dimensions of the tableau on the one hand and the `\oplus` on the other: The dimensions of the Young-Tableaux are determined by the actual text's fontsize (while autoscaling!). Try to understand what happens in the following example. — Do you understand the sizes of the math symbols? (Compare to the example above and have look up the `.log` file. Don't let TEX fool you by messages about invalid `\normalsize`!)

---

[2] So that a Young-Tableau e.g. `\young(ab)` $\boxed{a}\boxed{b}$ appears smaller in a footnote than in normal text!

```
{\tiny\begin{equation}
    \Yvcentermath1
    \yng(1,1)\oplus\yng(1)= \Yboxdim{12pt}
    \yng(2,1)\normalsize\oplus\dots
\end{equation}}
```

$$\begin{array}{|c|}\hline \\ \hline \\ \hline\end{array}\oplus\square=\begin{array}{|c|c|}\hline &\\ \hline \\ \cline{1-1}\end{array}\oplus\dots \tag{1}$$

2.2 Labelled boxes: `\young`

`\young`

While the sizing of the boxes and the behavior in math/text mode are exactly the same as with `\yng`, we have of course a different syntax:

`{\scriptsize\young(aa,b)}` results in $\begin{array}{|c|c|}\hline a&a\\ \hline b\\ \cline{1-1}\end{array}$. Each line of the tableau is represented by a row of single letters (exactly: "tokens"). The lines are separated by commas.

As you see from the example above the letters `a` and `b` come out in math italics. This does *not* depend on the Young-Tableau appearing in math or text surround. *The label within a tableau-box is math – no matter if the tableau is used in text or math mode!* But this is only the default setting. Use either the option `stdtext` to the `\usepackage` command to switch to roman letters or use the macro `\Ystdtext` with an argument of `1`: `\Ystdtext1`. To switch to math again use `\Ystdtext0`.

What to do if you don't want to use single letters, but symbols or perhaps some white boxes? There are no limits but it's a bit complicated. The `\young` algorithm for parsing his argument depends strongly on that every label of *one* box,

1. consists of only *one token*. Assume first you had *not* said `\Ystdtext1`. To get a single roman 'a' then you must not say `\mbox{a}`, because this makes *four* tokens (1st is `\mbox`, 2nd is `{`, 3rd is `a`, 4th is `}`). The cure is simply making one token out of `\mbox{a}` by saying `\newcommand{\rma}{\mbox{a}}` and everything is OK:[3] `\young(\rma lright\bullet)` = $\begin{array}{|c|c|c|c|c|c|c|c|}\hline \mathrm{a}&l&r&i&g&h&t&\bullet\\ \hline\end{array}$
   After `\Ystdtext1` it is an error to say `$\bullet$`, since the two math delimiters are single tokens. Do it as described above:
   `\newcommand{\ybullet}{$\bullet$}`
   and everything is OK: `\young(\ybullet works)` = $\begin{array}{|c|c|c|c|c|c|}\hline \bullet&\mathrm{w}&\mathrm{o}&\mathrm{r}&\mathrm{k}&\mathrm{s}\\ \hline\end{array}$.
   To use a symbol or special letter in both modes (`\Ystdtext 0 or 1`) put it in a box (here with an extra tuning):
   `\renewcommand{\ybullet}{\raise1.5pt \hbox{$\bullet$}}`

2. Some characters and macros are not allowed: commas (guess why) and the other punctuation marks, spaces, the macro `\space` and also skips like `\,` and `\:`.[4] If you want to have single empty boxes use (TEX primitive) `\hfil` (For understanding this limitations see also `\y@lastargtest` and `\y@nelettertest` in macro section).

---

[3] Note that the space after `\rma` belongs to the token itself.

[4] Write to us if you find more.

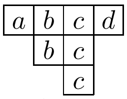
2.2.1 Skew Young-Tableaux with `\young`

New feature
1998/05/05

enableskew

As far as described above, the rows of a Young-Tableau are left-aligned. Since some authors also assigned a meaning to *"skew"* tableaux, this package also supports such along with `\young`. If you want to typeset skew tableaux, you have to enable this via the package option `enableskew`. The default behavior of this package is to *dis*able skew boxes in order to save TEX's time, however, it's save to include this option even if you do not use it.

Here is an example: `\young(abcd,:bc,::c)`:



As you see, the Option `enableskew` makes the colon a special character (only inside the argument of `\young`). [5] In fact all other than the leading colons in a row will simply be ignored, while the first will move the row to the right.

{ This is a LITTLE TEST of skew tableaux. }

2.3 The `youngtab` package and TEX

Compatibility

Because we do not use PLAIN TEX, the compatibility of this package and TEX is not sufficiently tested. However there are some changes necessary in `youngtab.sty` (see file `youngtab.tex`, part of the distribution):

- The macros themselves only use two LATEX-commands, `\vspace{`⟨*dim*⟩`}` and `\hspace{`⟨*dim*⟩`}`. One has to replace them with '`\vskip` ⟨*dim*⟩' resp. '`\hskip` ⟨*dim*⟩'. (The braces also have to get removed) The LATEX-counterparts may be better but no problems should arise from this replacement (we found none).

- One has to remove all lines containing LATEX 2_ε-commands in the preamble of the `.sty` file, e.g. comment out every line that contains the following commands: `\DeclareOption`, `\ProcessOptions`, `\NeedsTeXFormat` and `\ProvidesPackage`.

- Since TEX-input files must not contain a `@`, `\catcode`\@11\relax` has to be added to the begin of the `.sty` file to make `@` a `letter` and `\catcode`\@12\relax` at the end right before the `\endinput`, to make it `other` again afterwards.

- **Skew:** TEX knows nothing about style-options, we have to find a way to enable skew Young-Tableaux. (`enableskew` is the only package option that can *not* be accessed by a macro *after loading* the package since almost all according actions will be done at loading time).
  Therefore, we will check if a macro `\enableskew` is defined at loading time:
  ```
  \expandafter\ifx\csname enableskew\endcsname\relax
   \y@enable@skewfalse \else \y@enable@skewtrue\fi
  ```
  (excerpt from `youngtab.tex`) Then this will enable printing of skew tableaux: `\def\enableskew{1} \input youngtab`

---

[5] If you want a colon to appear as it is (do you??), you have to do the same trick as above with an unpleasant extra: define a macro for the colon using the TEX-primitive `\char` like this: `\newcommand\mycolon\char58`.

- The autoscaling doesn't work as nice as with LaTeX since if you lower the text's fontsize (which determines the boxsize), you have to lower also the math fontsize. Otherwise the letters in labelled boxes (math!) are too big for the box.

2.4 To do

We have a **wish list**, things we think the package should be capable of:

- A vertical alignment, in the way that the *top*line of the Young-Tableaux and the mathsymbols are on the same height (perhaps using `\vtop`?).
- The macros should take notice of math fontsize changes e.g. with `\displaystyle`. (This puts a focus on the autoscale mechanism – see `\y@setdim` in the macro section.)

If You have any problems, suggestions, critical remarks – or whatever according this package write to Volker Börchers (email-adress see title).

3 The Macros

```
⟨*package⟩
```

`\Yautoscale`
`\ify@autoscale`

If autoscaling is active, the size of the boxes (`\y@boxdim`) is determined by the package itselve. The default setting is to allow autoscale. `\Yautoscale`⟨*num*⟩ serves as the user interface.[6] We made an option to `\usepackage` out of that (see below).

```
\newif\ify@autoscale \y@autoscaletrue \def\Yautoscale#1{\ifnum #1=0
  \y@autoscalefalse\else\y@autoscaletrue\fi}
```

`\Yboxdim`
`\y@b@xdim`
`\y@boxdim`

The parameters determining the size of the single boxes and the macro to set them. `\Yboxdim`⟨*dim*⟩ sets `\y@boxdim` (only used, when autoscaling is turned off) and sets `y@autoscale` to `false`. `\y@boxdim` is the *total* size of the box (including the delimiting lines) while `\y@b@xdim` (to be set in `\y@setdim`) is the height and width of the inner of the box.

```
\newdimen\y@b@xdim
\newdimen\y@boxdim \y@boxdim=13pt
\def\Yboxdim#1{\y@autoscalefalse\y@boxdim=#1}
```

`\Ylinethick`
`\y@linethick`

The line thickness and `\Ylinethick`⟨*dim*⟩ as user interface to it.

```
\newdimen\y@linethick    \y@linethick=.3pt
\def\Ylinethick#1{\y@linethick=#1}
```

`\Yinterspace`
`\y@interspace`

To give the Young-Tableaux extra-space (before and after) `\y@interspace` can be set to more than `0pt` by calling `\Yinterspace`. (See `\yng` and `\young`.) Note that this is a *skip* and no *dimemsion*.

```
\newskip\y@interspace \y@interspace=0ex plus 0.3ex
\def\Yinterspace#1{\y@interspace=#1}
```

---

[6] I know - it's against the LaTeX 2_εnaming conventions, but it is nicer. . .

`\Yvcentermath`
`\ify@vcenter`

The switch to turn vertical centering in math on or off. The default is `\y@vcenterfalse`. The second option to `\usepackage`.

```
\newif\ify@vcenter   \y@vcenterfalse
\def\Yvcentermath#1{\ifnum #1=0 \y@vcenterfalse\else\y@vcentertrue\fi}
```

`\Ystdtext`
`\ify@stdtext`

Normally the inside of the boxes (`\young`) is in math mode. This code allows to switch to text mode.

```
\newif\ify@stdtext   \y@stdtextfalse
\def\Ystdtext#1{\ifnum #1=0 \y@stdtextfalse\else\y@stdtexttrue\fi}
```

`\ify@enable@skew`

In contrast to the other this `if` may be set only via the package option `enableskew`.

```
\newif\ify@enable@skew   \y@enable@skewfalse
```

`noautoscale`
`vcentermath`
`stdtext`
`enableskew`

Declaration and processing of the options for use with LaTeX 2$_\varepsilon$. (For TeX-users: Comment the following 8 lines out).

```
\DeclareOption{noautoscale}{\y@autoscalefalse}
\DeclareOption{vcentermath}{\y@vcentertrue}
\DeclareOption{stdtext}{\y@vcentertrue}
\DeclareOption{enableskew}{\y@enable@skewtrue}
\DeclareOption*{\PackageWarning{youngtab}{%
    Unknown option `\CurrentOption' (Known:\MessageBreak
    `vcentermath', `noautoscale', `stdtext', `enableskew'.)}}
\ProcessOptions\relax
```

`\y@vr`

A single vertical line to build the boxes. The reason for the depht of the line should be clear. (The values for the depth maybe something you want to tune)

```
\def\y@vr{\vrule height0.8\y@b@xdim width\y@linethick depth 0.2\y@b@xdim}
```

`\y@emptybox`

This macro does less than its name pretend: It only makes one vertical line followed by a empty box (width: `\y@boxdim` − 2`\y@linethick`).

```
\def\y@emptybox{\y@vr\hbox to \y@b@xdim{\hfil}}
```

`\y@abcbox`
`\y@mathabcbox`

If `enableskew` is not choosen (the package options have been parsed now), a `\y@abcbox` or a `\y@mathabcbox` is the simple analogue to `\y@emptybox` but it has a *text-font* letter centered in it.

For `\y@mathabcbox` the inside is math. We rather want an easier access to math italics and symbols than to normal text; so we usually use `\y@mathabcbox` instead of `\y@abcbox`.

Skew Young-Tableaux can only be used with `\young`. Then a colon is used to *shift* the row to the *right* – instead of "inserting a box without border" – so the colon boxes must be ignored.

```
\ify@enable@skew
 \def\y@abcbox#1{\if :#1\else
   \y@vr\hbox to \y@b@xdim{\hfil#1\hfil}\fi}
 \def\y@mathabcbox#1{\if :#1\else
   \y@vr\hbox to \y@b@xdim{\hfil$#1$\hfil}\fi}
\else
 \def\y@abcbox#1{\y@vr\hbox to \y@b@xdim{\hfil#1\hfil}}
 \def\y@mathabcbox#1{\y@vr\hbox to \y@b@xdim{\hfil$#1$\hfil}}
\fi
```

`\y@setdim` This macro is called at the beginning of the macros `\yng` and `\young`. If `autoscale` is deactivated the box height and width is `\y@boxdim` - 2`\y@linethick`. If not we save a box for temporary usage, use the height of it to find a good size of the box, fill the boxregister with a empty box, and clear the box again by using it.

That's not elegant but I didn't know better. . . How can I determine the height of the font in an other way? (I tried it with the TEX-measure `ex` – but this did not seem to work in any case.) Or – if I have to use setbox – which boxregister should I use? Is the `\ifvoid` test obsolete? (First I used `\box0` – till I found me competing with AMS-TEX for this box.)

```
\def\y@setdim{%
  \ify@autoscale%
   \ifvoid1\else\typeout{Package youngtab: box1 not free! Expect an
     error!}\fi%
   \setbox1=\hbox{A}\y@b@xdim=1.6\ht1 \setbox1=\hbox{}\box1%
  \else\y@b@xdim=\y@boxdim \advance\y@b@xdim by -2\y@linethick
  \fi}
```

`\y@counter` A counter for loops.

```
\newcount\y@counter
```

`\y@lastargtest` This macro is a hack to get TEX to accept a indefinite number of arguments (here: the number of *lines* separated by commas). It does nothing with its arguments but to see if the second is a space, and, if it is to set `\y@islastargtrue`.[7]

```
\newif\ify@islastarg
\def\y@lastargtest#1,#2 {\if\space #2 \y@islastargtrue
  \else\y@islastargfalse\fi}
```

How does it work? An Example:

`\y@lastargtest␣first,second␣`

Here `#1` is `first` and `#2` is `second`. The `\if` construction tests if the (expanded) *first* token of `#2` and `\space` (here: `s`) are equal. – They are not. In the next example the condition is `true`:

`\y@lastargtest␣first,␣`

Here `#1` is again `first` but `\y@lastargtest` catches the blank after the comma to be `#2`. The `\if` condition now is true and `\y@islastargtrue` is set. Remember: the blank is essential!

`\y@emptyboxes` This macro draws `#1` (this is a number) empty boxes (`\y@emptybox`) – without a top or bottom line:

`:\y@emptyboxes3:` is :| | | :

```
\def\y@emptyboxes#1{\y@counter=#1\loop\ifnum\y@counter>0
  \advance\y@counter by -1 \y@emptybox\repeat}
```

`\y@nelineemptyboxes` This calls `\y@emptyboxes` and adds the top and bottom lines (I think this is faster than draw single, complete boxes) and a right closing vertical line. Because of the `\vspace` command the top line of the eventually following next box-row will be drawn over the bottom line of this box-row.

```
\def\y@nelineemptyboxes#1{%
  \vbox{%
    \hrule height\y@linethick%
```

---

[7] The `islastarg` conditions are also used if the end of one line of labelled boxes is reached.

```
    \hbox{\y@emptyboxes{#1}\y@vr}
    \hrule height\y@linethick}\vspace{-\y@linethick}}
```

`\yng` The user-macro for empty Young-Tableaux. As mentioned above it has only one argument, enclosed in brackets. It first calls `\y@setdim` to determine the size of the boxes and opens a `\vcenter` if in math mode and `vcentermath` is set. Otherwise only a (then obsolet) `{` is opened. `\y@lastargtest` determines if the tableau should have only one line, setting `\y@islastargtrue` or `-false`. If there is only one line, `\yng` does the whole job in calling `\y@nelineemptyboxes` once. Otherwise it calls `\y@ungempty` to do the real thing. Note the spacing before and after the tableau using `\hspace`.

```
\def\yng(#1){%
  \y@setdim%
  \hspace{\y@interspace}%
  \ifmmode\ify@vcenter\vcenter\fi\fi{%
  \y@lastargtest#1,
  \vbox{\offinterlineskip
    \ify@islastarg
     \y@nelineemptyboxes{#1}
    \else
     \y@ungempty(#1)
    \fi}}\hspace{\y@interspace}}
```

`\y@ungempty` It is called with `\y@ungempty(#1)` from `\yng`. Now the parsing of the arguments starts! As we know from `\y@lastargtest` in `\yng`, there really are two arguments left. Explicitely: Let the the initial command (given by the user) be `\yng(3,2,1)`; then `\yng` calls `\y@ungempty(3,2,1)`. But `\y@ungempty` sees two arguments: `#1` is `3` and `#2` is `2,1`! (Note the importance of the brackets). `\y@ungempty` then processes the first argument immediately, calling `\y@nelineemptyboxes`. (The braces protect numbers with more than one digit.)

The next step is to check if `#2` is the last line. If it is, it calls `\y@nelineemptyboxes` again. Otherwise it calls *recursively itselve*, till all lines are processed.

```
\def\y@ungempty(#1,#2){%
  \y@nelineemptyboxes{#1}
  \y@lastargtest#2,
  \ify@islastarg
   \y@nelineemptyboxes{#2}
  \else
   \y@ungempty(#2)
  \fi}
```

`\y@nelettertest` The first macro especially for labelled boxes, that is, for `\young`. Like `\y@lastargtest` it is used for handling a indefinite number of arguments. (As mentioned before Young-Tableaux with labelled boxes have problem double.) `\y@nelettertest` determines if the last box of the line has been reached.

Again we will examine it's working method by examples:

`\y@nelettertest ab.`

The period after the argument `ab` (and in the definition of `\y@nelettertest`) is merely a mark for the end of the arguments. (Otherwise TeX would complain about missing arguments.) Here `a` is `#1` and `#2` is `b` – the `\if`-condition is `false` and so `\y@islastargfalse` is set. Now a little miracle:

```
\y@nelettertest a.
```

In the opposite to the case of `\y@lastargtest` it's not understandable for me, why `#2` now is a space – but it is! (Do *you* know the reason why?) The rest is known from `\y@lastargtest`.

```
\def\y@nelettertest#1#2. {\if\space #2 \y@islastargtrue
  \else\y@islastargfalse\fi}
```

`\y@abcboxes`

Again we use a period to mark the end of arguments. Compare to `\y@emptyboxes` – Now it is more strenuous! The `\y@nelettertest` macro makes sure that `\y@abcboxes` is called with at least two boxes to process. If there are only two boxes `\y@abcboxes` calls `\y@abcbox` two times, otherwise it calls itselve recursively again. (The argument-splitting works as in `\y@ungempty`.)

```
\def\y@abcboxes#1#2.{%
  \ify@stdtext\y@abcbox#1\else\y@mathabcbox#1\fi%
  \y@nelettertest #2.
```

There must be a space or (equivalently) a carriage return after this line!

```
  \ify@islastarg\unskip%
   \ify@stdtext\y@abcbox{#2}\else\y@mathabcbox{#2}\fi%
  \else\y@abcboxes#2.\fi}
```

The `\unskip` is needed for the case that `#2` is only one token. Then a space appears between two boxes. (I do not understand really what happens – see `\y@nelettertest`.) The `\unskip` resolves this problem.

`\ify@enable@skew`
`\y@full@b@xdim`
`\y@m@veright@cnt`

Most of the skew boxes stuff can be found here along the definition of `\y@nelineabcboxes`. We need a further dimension, for the effective lenght of one box in a row, `\y@full@b@xdim` = `\y@b@xdim` + `1\y@linethick` and one counter for the number of boxes to be left out on the left side of the tableau (determined by the number of colons at the beginning of a row), `\y@m@veright@cnt`.

```
\ify@enable@skew
 \newdimen\y@full@b@xdim
 \newcount\y@m@veright@cnt
```

`\y@get@m@veright@cnt`

To find out the number of leading colons in a row we have the macro `\y@get@m@veright@cnt`. Here we use exactly the same trick as in `\y@nelettertest`.

```
 \def\y@get@m@veright@cnt#1#2.{%
   \if :#1 \advance\y@m@veright@cnt by 1\y@get@m@veright@cnt#2.\fi}
```

`\y@setdim`

Since we now have another dimension that depends on `\y@b@xdim`, the dimension of the boxes, we have to extend `\y@setdim` to calculate `\y@full@b@xdim` too. (The dummy `\y@setdim@` is used to avoid a recursion.)

```
 \let\y@setdim@=\y@setdim
 \def\y@setdim{%
   \y@setdim@ \y@full@b@xdim=\y@b@xdim
   \advance\y@full@b@xdim by 1\y@linethick}
```

`\y@m@veright@ifskew`

Now we put it all into the following macro that computes the offset and moves the tableau to the right if necessary. If skew Young-Tableaux are not enabled, `\y@m@veright@ifskew` will be a null-command (but catches the argument).

```
 \def\y@m@veright@ifskew#1{
   \y@m@veright@cnt=0 \y@get@m@veright@cnt#1.
   \moveright \y@m@veright@cnt\y@full@b@xdim}
```

```
\else
 \def\y@m@veright@ifskew#1{}
\fi
```

`\y@nelineabcboxes` This is the counterpart of `\y@nelineemptyboxes`. The difference is the treatment of the case of a row of only one box – Then only `\y@abcbox` is called and not `\y@abcboxes`. If this row must be moved to the right, this is done by `\y@m@veright@ifskew`.

```
\def\y@nelineabcboxes#1{%
  \y@nelettertest #1.
  \ify@islastarg
   \y@m@veright@ifskew{#1}
    \vbox{
      \hrule height\y@linethick%
      \hbox{\ify@stdtext\y@abcbox#1\else\y@mathabcbox#1\fi\y@vr}
      \hrule height\y@linethick}\vspace{-\y@linethick}
  \else
   \y@m@veright@ifskew{#1}
    \vbox{
      \hrule height\y@linethick%
      \hbox{\y@abcboxes #1.\y@vr}%
      \hrule height\y@linethick}\vspace{-\y@linethick}
  \fi}
```

`\young` The user-macro for labelled Young-Tableaux. It has one argument as `\yng`, enclosed in brackets. Everything is identical as in the emptybox case (sizing, vcentering, treatment of multiple rows, . . .).

```
\def\young(#1){%
  \y@setdim%
  \hspace{\y@interspace}%
  \y@lastargtest#1,
  \ifmmode\ify@vcenter\vcenter\fi\fi{%
  \vbox{\offinterlineskip
    \ify@islastarg\y@nelineabcboxes{#1}%
    \else\y@ungabc(#1)%
    \fi}}\hspace{\y@interspace}}
```

`\y@ungabc` Again nothing new – see `\y@ungempty`

```
\def\y@ungabc(#1,#2){%
  \y@nelineabcboxes{#1}%
  \y@lastargtest#2,
  \ify@islastarg\y@nelineabcboxes{#2}%
  \else\y@ungabc(#2)%
  \fi}
```

```
⟨/package⟩
```

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.