

# The `ytableau` package\*

Ryan Reich  
ryan.reich@gmail.com

February 1, 2011

## Contents

<b>1</b>	<b>Y tableau?</b>	<b>2</b>
<b>2</b>	<b>User commands</b>	<b>3</b>
2.1	The <code>ytableau</code> environment . . . . .	3
2.2	The <code>ytableaushort</code> command . . . . .	4
2.3	The <code>ydiagram</code> command . . . . .	4
2.4	Chaining . . . . .	4
<b>3</b>	<b>Package options</b>	<b>5</b>
<b>4</b>	<b>Samples</b>	<b>5</b>
4.1	Standard Young tableaux . . . . .	5
4.2	Skew tableaux . . . . .	6
4.3	Color and chaining . . . . .	6
<b>5</b>	<b>The Code</b>	<b>7</b>
5.1	Global defintions . . . . .	7
5.1.1	Box registers . . . . .	7
5.1.2	Token registers . . . . .	8
5.1.3	Dimension registers . . . . .	8
5.1.4	Count registers . . . . .	8
5.1.5	Macros . . . . .	9
5.2	Options . . . . .	10
5.3	<code>ytableau</code> environment . . . . .	12
5.4	<code>ytableaushort</code> command . . . . .	15
5.5	<code>ydiagram</code> commmand . . . . .	17
<b>6</b>	<b>Change History</b>	<b>19</b>
<b>7</b>	<b>Index</b>	<b>20</b>

---

\*This document describes `ytableau` v1.1, dated 2011/01/31.

*For Greta*

## 1 Y tableau?

At present there exist two packages with which one can draw Young tableaux: `young` and `youngtab`. As the latter is explicitly an alternative to the former, they do not overlap very much except in what they eventually produce. Between them, they define the following three basic constructions of Young tableaux:

- An environment with array-style syntax;
- A short-form macro;
- An even shorter-form macro for drawing Young diagrams (having nothing inside the boxes).

In this package we also implement these methods. However, we aim to take them as far as possible so that the conceivable needs of a mathematician making serious use of Young tableaux can be met with as little effort as possible on their part (and, thus, great effort on the part of this author). In writing this package we pursued the following major goals:

- The syntax should be as convenient as possible. The `young` package makes unfortunate use of the `\cr` delimiter for lines in an array, which is nowhere to be seen in modern  $\text{\LaTeX}$  and is likely unfamiliar to casual, young writers. The `youngtab` package requires the author to define individual, separate macros to draw items requiring more than one token to represent in  $\text{\TeX}$ . That is, if a cell of a tableau is to contain the expression  $n + 1$ , then one must place it into an auxiliary macro. Also, the same command, `\young`, delimits its contents with parentheses `(...)` rather than braces `{...}`.
- The package should make no assumptions about the intentions of the user. In particular, esoteric constructions such as skew tableaux and disconnected tableaux are not in principle any more difficult to draw, and should be no more difficult to write.
- Tableaux should support totally arbitrary decoration. We took this to mean that they should be easily colored; this possibility allows the depiction of tableaux within tableaux, an application which was specifically requested of the author (and was the original reason for writing this package).
- Configuration should be easy and plentiful. The `young` package has none, while `youngtab` package uses a strange syntax. Now that `keyval` is available there is no excuse for not providing a keyword-driven user interface to options controlling all aspects of the appearance of tableaux.

- Interoperability with all the common environments. In particular, since this is a mathematics package, it should work properly in the AMS environments, and since it is an array-based package it should work properly in array environments.

We believe the package achieves all these things. There should be nothing that one would want to do with Young tableaux that cannot be accomplished in the obvious way using the commands given below.

## 2 User commands

We provide three commands for drawing Young tableaux and diagrams. Each one is convenient for slightly different purposes and each supports various operations more or less easily than the others.

### 2.1 The `ytableau` environment

`ytableau` The `ytableau` environment is the core drawing engine for this package. It may be called as follows (similarly to `young`):

```
\begin{ytableau}[\langle general formatting \rangle]
  \langle entry \rangle & *(\langle color name \rangle) \langle entry \rangle & \dots \\\
  \dots \\
\end{ytableau}
```

The result is an array of boxes separated by lines of width 0.4pt (not tunable, and not affected by outside influences), each containing the entries specified in the environment.

Each  $\langle entry \rangle$  is typeset in math mode (by default, but text mode is possible) and the entries are horizontally and (mostly) vertically centered in their box. In fact, the entries are treated as though they consist of a single line of text, and the baselines of all the entries in a row are aligned with each other for a consistent appearance. The environment may appear in or out of math mode without any ill effect (and without any effect on the contents).

`\none` An entry may be omitted by writing `\none`, which prevents the drawing of a frame but places an invisible box of the correct dimension inside the entry. Thus, one may create a tableau “starting” at an offset or even a “tableau” consisting of several disconnected regions. One can actually get things into these “empty” boxes by passing an optional argument to `\none`.

The  $\langle color name \rangle$  can be any color name familiar to the package `xcolor`, or (of course) user-defined. The background of this box will be drawn in that color; by default, if no color is given the background is *transparent*, which probably means white, unless the tableau is somehow overlaid on something else (see 2.4).

The  $\langle general formatting \rangle$  is simply  $\TeX$  material which is placed in front of each  $\langle entry \rangle$ . It can also contain a  $\langle color name \rangle$ , which is overridden by those specified individually.

## 2.2 The `ytableaushort` command

`\ytableaushort` This command (however ironically named) allows inline specification of a tableau:

```
\ytableaushort [general formatting] {line,line,...}
```

where *general formatting* is as before, and each *line* is a sequence of tokens representing entries in the tableau, similarly to `youngtab`. However, it is possible to include complex entries by surrounding them in `{...}`. This command internally reduces its functioning to `ytableau`, so the entries may contain colors and in general behave exactly as described above.

## 2.3 The `ydiagram` command

`\ydiagram` This command draws Young diagrams somewhat in the manner of `youngtab`:

```
\ydiagram [general formatting] {[offset +]number,...}
```

producing an array of *identical* boxes (empty by default), each row having *number* in it with *offset* blank boxes preceding (*offset* is optional but, when provided, is not written with `[...]`). Thus, a typical invocation might be

```
\ydiagram{2 + 1, 3, 1}.
```

Both *offset* and *number* may be any `TeX` expressions evaluating to the textual representation of a number (e.g. 6 or `\thecountername`, but not just `\countername`). The boxes can be colored or filled with a single expression by means of *general formatting*.

## 2.4 Chaining

The coloring facility for `\ydiagram` is not very interesting as-is. Thus, the package allows for the augmentation of several diagrams in the following manner:

```
\ydiagram arguments * arguments * ...
```

produces a *single* Young diagram obtained by layering the ones specified by the various *arguments* from left (on top) to right (at the bottom). In fact, one can even write

```
\ytableaushort tableau arguments * diagram arguments * ...
```

where first a Young tableau is constructed according to the initial set of arguments, then all subsequent arguments are passed to `\ydiagram`, with the result layered from left to right. This allows the construction of arbitrary color patterns with arbitrary contents.

This operation is not possible with `\begin{ytableau}...\end{ytableau}` since the `\end` command obscures the following text from the internally-called

`\endytableau` command. If you want to chain a `ytableau`, instead write it in the TeX style `\ytableau... \endytableau`. Unfortunately it is not possible to work around this.

### 3 Package options

The package accepts the following options:

- `boxsize`
  - `boxsize=<dimension>`. This manually sets the height (and width, which is the same) of boxes in all tableaux to  $\langle dimension \rangle$ . If you change the size and want to get back to the default (1.5em), just say `boxsize = normal`.
  
- `smalltableaux`  
`nosmalltableaux`
  - `smalltableaux/nosmalltableaux`. The first option makes the box size quite small; indeed, small enough to fit a  $\$f\$$  precisely, as computed by careful eyeballing tests. It also passes `\scriptstyle` to each box, which as usual can be overridden if you wish (but you don't). The second option returns things to how they were.
  
- `aligntableaux`  
`centertableaux`  
`nocentertableaux`
  - `aligntableaux=<alignment>/centertableaux/nocentertableaux`. The first argument allows any of `top`, `center`, or `bottom`. With `top`, the tableaux are all aligned on the baseline of their top row, with `bottom` they are aligned on the baseline of their bottom row, and `center` centers them (they correspond to `\vtop`, `\vbox`, and `\vcenter`). The other two arguments are semantically pleasing shorthand for `aligntableaux = center` and `aligntableaux = top`.
  
- `textmode`  
`mathmode`
  - `textmode/mathmode`. The former sets all the boxes in text mode, and the latter returns to math mode (the default).

It may not be useful to set these options globally, so we provide a macro for changing each of these parameters “on the fly”:

`\ytableausetup` Takes all of the above options and acts on them, setting parameters for all subsequent tableaux. The assignments are global with respect to TeX nestings.

One can also pass options to `xcolor` when calling `ytableau`, but these options are not further configurable through `\ytableausetup`.

### 4 Samples

Note that the option settings are persistent.

#### 4.1 Standard Young tableaux

<i>a</i>	<i>d</i>	<i>f</i>
<i>b</i>	<i>e</i>	<i>g</i>
<i>c</i>		

```
\ytableausetup{centertableaux}
\begin{ytableau}
a & d & f \\
b & e & g \\
c
\end{ytableau}
```

a	d	f
b	e	g
c		

```
\ytableausetup{textmode}
\begin{ytableau}
a & d & f \\
b & e & g \\
c
\end{ytableau}
```

1	2	3	...	$2n-1$	$2n$
2	3	4	...	$2n$	
$\vdots$	$\vdots$	$\vdots$			
$2n-1$	$2n$				
$2n$					

```
\ytableausetup
{mathmode, boxsize=2em}
\begin{ytableau}
1 & 2 & 3 & \none[\dots]
& \scriptstyle 2n - 1 & 2n \\
2 & 3 & 4 & \none[\dots]
& 2n \\
\none[\vdots] & \none[\vdots]
& \none[\vdots] \\
\scriptstyle 2n - 1 & 2n \\
2n
\end{ytableau}
```

## 4.2 Skew tableaux

		1	2
	1	2	
1	2		
2			

```
\ytableausetup{boxsize=normal}
\begin{ytableau}
\none & \none & 1 & 2 \\
\none & 1 & 2 \\
1 & 2 \\
2
\end{ytableau}
```

		1	2
	1	2	
1	2		
2			

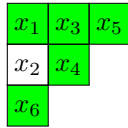
```
\ytableausetup{smalltableaux}
\ytableaushort{
\none\none 12,
\none 12, 12, 2}
```


```
\ydiagram{2+2,1+2,2,1}
```

## 4.3 Color and chaining

1	3	5
2	4	
6		

```
\ytableausetup{nosmalltableaux}
\begin{ytableau}
*(red) 1 & *(red) 3 & *(red) 5 \\
*(blue) 2 & *(blue) 4 \\
*(blue) 6
\end{ytableau}
```

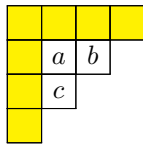


```
\ytableaushort[* (green) x_]
{135, {*(white)2}4, 6}
```

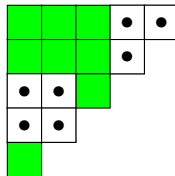
$$\begin{array}{c} x_1 & x_3 & x_5 \\ x_2 & x_4 \\ x_6 \end{array} + \begin{array}{c} \color{red}{\square} \\ \color{red}{\square} \\ \color{red}{\square} \end{array} = \begin{array}{c} \color{red}{x_1} & \color{red}{x_3} & \color{red}{x_5} \\ \color{blue}{x_2} & \color{blue}{x_4} \\ \color{blue}{x_6} \end{array} \quad (1)$$

```
\begin{multline}
\ytableaushort[* (green) x_]
{135, {*(white)2}4, 6}

\begin{multline}
\ytableaushort[* (red) ]{3} \\
+ \ytableaushort[* (blue) ]{3, 2, 1}
\end{multline}
= \ytableaushort[x_] {135, 24, 6}
* [* (red) ]{3} * [* (blue) ]{3, 2, 1}
\end{multline}
```



```
\ytableaushort[* (yellow) ]{4, 1, 1, 1}
\ytableaushort
{\none, \none ab, \none c}
* {4, 3, 2, 1}
* [* (yellow) ]{4, 1, 1, 1}
```



```
\ydiagram[* (white) \bullet]
{3+2, 3+1, 2, 2}
* [* (green) ]{5, 4, 3, 2, 1}
```

## 5 The Code

### 5.1 Global definitions

Here are all the registers set and “variables” used.

#### 5.1.1 Box registers

- `\tableaux@YT` When chaining, collects the successive tableaux.
- `\thistableau@YT` When chaining, stores the current tableaux in the chain. Used in `\endytableau`.
  - 1 `\newbox\tableaux@YT`
  - 2 `\newbox\thistableau@YT`
- `\thisbox@YT` Stores the box currently being constructed. We define it as an alias because it is local to the construction of `\thistableau@YT` and, afterwards, irrelevant. Used in `\startbox@YT` and `\endbox@YT`.
  - 3 `\let\thisbox@YT=\thistableau@YT`

`\refhtdp@YT` Boxes to hold our reference-height and reference-depth letters. Is `$b$` the tallest and `$g$` the deepest? Used in `\endbox@YT`.

```
4 \newbox\refhtdp@YT
5 \setbox\refhtdp@YT=\hbox{$b$}
```

### 5.1.2 Token registers

`\toks@YT` Accumulates what will be put in the `ytableau` environment. Used in `\diagram`, `\getnumbers@YT`, `\getentries@YT`, `\loop@YT`, and `\ytableaushort`.

```
6 \newtoks\toks@YT
```

`\opttoksa@YT` Store the optional arguments (minus color specifications) when processing each entry of the tableau. Also used as temporary token registers. Used in `\startbox@YT`, `\getline@YT`.

```
7 \newtoks\opttoksa@YT
8 \newtoks\opttoksb@YT
```

### 5.1.3 Dimension registers

`\boxdim@normal@YT` Respectively, these record the normal size of a box in a tableau for use in option `boxsize`, and the previously used size when using the option pair `smalltableaux`, `nosmalltableaux`. Used in options `boxsize`, `smalltableaux`, `nosmalltableaux`.

```
9 \newdimen\boxdim@normal@YT
10 \boxdim@normal@YT=1.5em
11 \newdimen\boxdim@save@YT
12 \boxdim@save@YT=\boxdim@normal@YT
```

`\boxdim@YT` The size of the boxes in a tableau. Used in `\none`, `\start@YT`, `\end@YT`.

```
13 \newdimen\boxdim@YT
14 \boxdim@YT=\boxdim@normal@YT
```

`\tableauwd@YT` Save the total width of a tableau for supporting the “chaining” operation. Used in `\endytableau`.

```
15 \newdimen\tableauwd@YT
```

`\boxframe@YT` The width of the frame in a tableau. Used in `\ytableau` and `\endytableau`.

```
16 \newdimen\boxframe@YT \boxframe@YT=0.4pt
```

### 5.1.4 Count registers

`\count@YT` Just a counter for looping. Used in `\loop@YT`, `\fullexpand@YT`.

```
17 \newcount\count@YT
```



### 5.1.5 Macros

<code>\ifstar@YT</code>	Fix a bug with <code>amsmath</code> where <code>\@ifnextchar</code> (used in <code>\@ifstar</code> ) doesn't ignore spaces. 18 <code>\def\ifstar@YT#1{\kernel@ifnextchar *{\@firstoftwo{#1}}}</code>
<code>\skipin@YT</code> <code>\skipout@YT</code>	Stores the delimiter for text mode or math mode which absorbs spaces around the contents of a box. 19 <code>\def\skipin@YT{}</code> 20 <code>\def\skipout@YT{}</code>
<code>\thisboxcolor@YT</code>	Stores the color of the current box in a tableau. The color <code>clear</code> is not recognized by <code>xcolor</code> but denotes for us a transparent box. Used in <code>\getcolor@@YT</code> and <code>\endbox@YT</code> . 21 <code>\def\thisboxcolor@YT{clear}</code>
<code>\centering@YT</code>	What kind of vertical alignment our tableaux will have. Used in <code>\ytableau</code> and options <code>aligntableaux</code> , <code>centertableaux</code> , <code>nocentertableaux</code> . 22 <code>\def\centering@YT{top}</code>
<code>\defarg@YT</code>	The “default optional argument” which is applied to every tableau before all the others passed by the user. At the moment all it does is support option <code>smalltableaux</code> , but perhaps it can be put to greater use? Used in option <code>smalltableaux</code> . 23 <code>\def\defarg@YT{}</code>
<code>\compare@YT</code>	Compares two strings. Neither of them should be hidden in macros; i.e. it compares exactly what is given. Used in options <code>boxsize</code> and <code>aligntableaux</code> as well as in <code>\getline@@YT</code> , <code>\getentries@@YT</code> , <code>\loop@YT</code> , and <code>\getnumbers@YT</code> .
<code>\compare@@YT</code>	Compares two strings, where the first is hidden in one layer of macros. Used in <code>\endytableau</code> , <code>\fcolorbox@YT</code> .
<code>\ifeq@YT</code>	Tests the result of <code>\compare@YT(0)</code> . 24 <code>\def\compare@YT#1#2{%</code> 25 <code>\def\tmpa@YT{#1}\def\tmpb@YT{#2}%</code> 26 <code>\ifx\tmpa@YT\tmpb@YT%</code> 27 <code>\global\eq@YTtrue%</code> 28 <code>\else%</code> 29 <code>\global\eq@YTfalse%</code> 30 <code>\fi%</code> 31 <code>}</code> 32 <code>\def\compare@@YT#1#2{%</code> 33 <code>\def\tmpb@YT{#2}%</code> 34 <code>\ifx#1\tmpb@YT%</code> 35 <code>\global\eq@YTtrue%</code> 36 <code>\else%</code> 37 <code>\global\eq@YTfalse%</code> 38 <code>\fi%</code> 39 <code>}</code> 40 <code>\newif\ifeq@YT</code>

## 5.2 Options

We include `xkeyval` to support various options.

```
41 \RequirePackage{xkeyval}
```

`\ytableausetup` The user interface to options once the document is in progress.

```
42 \newcommand{\ytableausetup}[1]{\setkeys[ytableau]{setup}{#1}}
```

`boxsize` Box size. Takes a dimension or `normal`.

```
43 \define@key[ytableau]{setup}{boxsize}{%
```

Make tableaux un-small before changing the box size, even if the user wants to go smaller, because there is also the issue of `\defarg@YT` being set, and it is only ever changed in that option.

```
44 \setkeys[ytableau]{setup}{nosmalltableaux}%
```

```
45 \compare@YT{#1}{normal}%
```

```
46 \ifeq@YT%
```

```
47 \global\boxdim@YT=\boxdim@normal@YT%
```

```
48 \else%
```

```
49 \global\boxdim@YT=#1%
```

```
50 \fi%
```

```
51 }
```

`aligntableaux` Most general alignment option, can be any of `top`, `center`, or `bottom`.

`centertableaux` `centertableaux` is `aligntableaux = center`,

`nocentertableaux` `nocentertableaux` is `aligntableaux = top`.

```
52 \define@choicekey*[ytableau]{setup}{aligntableaux}
```

```
53 {top,center,bottom}[true]{%
```

```
54 \gdef\centering@YT{#1}
```

```
55 }
```

```
56 \define@choicekey[ytableau]{setup}{centertableaux}{true}[true]{%
```

```
57 \gdef\centering@YT{center}%
```

```
58 }
```

```
59 \define@choicekey[ytableau]{setup}{nocentertableaux}{true}[true]{%
```

```
60 \gdef\centering@YT{top}%
```

```
61 }
```

`smalltableaux` Small tableaux: reduce the box size and the text size. `nosmalltableaux` resets everything to the way it was before `smalltableaux` was passed.

```
62 \define@boolkey[ytableau]{setup}{smalltableaux}[true]{%
```

```
63 \ifytableau@setup@smalltableaux%
```

We use the sign of `\boxdim@save@YT` to indicate whether we are “in” small tableaux; this prevents double-calling the option. Even if the user is not so malicious, this can (does) happen in the `amsmath` display environments.

```
64 \ifnum\boxdim@save@YT>0%
```

```
65 \gdef\defarg@YT{\scriptstyle}%
```

```
66 \global\boxdim@save@YT=-\boxdim@YT%
```

Arrived at by eyeballing. Exactly fits an  $\$f\$$ .

```
67 \global\boxdim@YT=.81em%
68 \fi%
69 \else%
70 \ifnum\boxdim@save@YT<0%
71 \gdef\defarg@YT{}%
72 \global\boxdim@YT=-\boxdim@save@YT%
73 \global\boxdim@save@YT=\boxdim@normal@YT%
74 \fi%
75 \fi%
76 }
77 \define@boolkey[ytableau]{setup}{nosmalltableaux}[true]{%
78 \ifytableau@setup@nosmalltableaux%
79 \setkeys[ytableau]{setup}{smalltableaux=false}%
80 \else%
81 \setkeys[ytableau]{setup}{smalltableaux=true}%
82 \fi%
83 }
```

**textmode** Requests that the boxes in tableaux be typeset in text mode rather than the default  
**mathmode** math mode. In text mode, the `skipout` macro has to `\unskip` prior spaces, while the `skipin` macro has to ignore following ones.

```
84 \define@boolkey[ytableau]{setup}{textmode}[true]{%
85 \ifytableau@setup@textmode%
86 \global\def\skipin@YT{\ignorespaces}%
87 \global\def\skipout@YT{\unskip}%
88 \else%
89 \global\def\skipin@YT{${}%
90 \global\def\skipout@YT{${}%
91 \fi%
92 }
93 \define@boolkey[ytableau]{setup}{mathmode}[true]{%
94 \ifytableau@setup@mathmode%
95 \setkeys[ytableau]{setup}{textmode=false}%
96 \else%
97 \setkeys[ytableau]{setup}{textmode=true}%
98 \fi%
99 }
```

Now the default option, where one can ask `xcolor` for things.

```
100 \DeclareOptionX*{\PassOptionsToPackage{\CurrentOption}{xcolor}}
```

Process the options now. We don't really need to be so specific but it doesn't hurt. Then we load `xcolor` with the options we may have collected.

```
101 \ProcessOptionsX[ytableau]<setup>[]
102 \RequirePackage{xcolor}
```

### 5.3 ytableau environment

`ytableau` The core tableau-drawing environment. The first argument, which is optional, is just “formatting” pasted on to each entry. The contents are an `\halign`-style array; if an entry begins with `*(\color)`, then the background of that box is colored.

```
103 \newenvironment{ytableau}[1] []
104 {%
```

Despite the alignment requirements, we set the tableau top-aligned so that it can be easily chained. This will get fixed before we print it, though.

The point of the mysterious `\iffalse` is to produce a syntactically balanced pair of braces `{}` which semantically is equivalent to just an open brace `{`. This is required to support tableaux nested inside other alignments because `\halign` does not recognize `\bgroup... \egroup` as designating a nesting! (We will use this fact later, actually.) But we can’t just write `{` and (in `\endytableau`) `}` either.

```
105 \global\setbox\thistableau@YT=\vtop{\iffalse}\fi%
106 \setlength{\fboxrule}{\boxframe@YT}%
107 \setlength{\fboxsep}{0pt}%
```

I hate `\cr`, let’s use the L<sup>A</sup>T<sub>E</sub>X convention.

```
108 \let\=\cr@YT%
```

Lines and columns should abut, accounting for the fact that each entry is framed.

```
109 \offinterlineskip%
110 \openup-\fboxrule%
111 \tabskip=-\fboxrule%
```

We have to make sure `\everycr` is empty or else strange things could happen (like in the `amsmath` environment `gather`). Thanks to Harald Hanche-Olsen for telling me about this.

```
112 \everycr={}%
```

Now we begin the `\halign`. Each entry is passed as an argument to our box-building function, but we can’t just write something like `\box@YT{##}` because of the following complication:

When TeX sees `\box@YT{`, it absorbs tokens up until the next unmatched `}` without interpreting them and then feeds that to the macro as `#1`. Unfortunately, we would like it to be possible to omit `\` on the last line (as people are used to this, and Knuth provided for it with `\crr`). But since `ytableau` is an environment, the ending of `\halign` is hidden in the macro `\endytableau` (or `\end{ytableau}`) which is *not expanded by* `\halign` while reading for `##` in the proposed code.

The workaround is to pretend that `##` is not an argument to a macro until we get deep inside `\startbox@YT`, where (after some processing) it is fed to an `\hbox` inside math mode. `\hbox` is not really a macro (it’s a builtin) and it does interpret its contents as it reads them, and since we have finally set up the desired typesetting environment we can let it read `##` properly. Since we are *still* inside an `\halign`, eventually it will expand `\endytableau` and `##` will terminate properly. Whew.

```
113 \halign\bgroup&\startbox@YT{\defarg@YT}{#1}##%
```

```

114                                     \endbox@YT\cr%
115 }
116 {%

```

The `\crr` supports the omission of `\\` in the last row. That's a pretty modest goal for all the work that went into thinking up this crazy scheme.

```

117   \crr\egroup%
118   \iffalse{\fi}%

```

Support for chaining. We allow `\endytableau` to be followed by `*[...]{...}`, which is fed to `\ydiagram` as-is. This only works in the short forms `\ytableaushort` and `\ydiagram`, since in `\end{ytableau}` there is extra code intervening before the following characters and no way to insert things in it.

```

119 \ifnum\wd\thistableau@YT>\wd\tableaux@YT%
120   \tableauwd@YT=\wd\thistableau@YT%
121   \advance\tableauwd@YT by -\wd\tableaux@YT%
122 \else%
123   \tableauwd@YT = Opt%
124 \fi%

```

We have saved the larger width, but now `\thistableau@YT` must have width zero so that it can be overlaid with the existing tableaux.

```

125 \wd\thistableau@YT=Opt%
126 \setbox\tableaux@YT%
127   =\hbox{\box\thistableau@YT\unhbox\tableaux@YT%
128     \kern\tableauwd@YT}%
129 \ifstar@YT%
130   {\ydiagram}%
131   {%

```

We adjust the vertical alignment finally and print the boxes. `\leavevmode` ensures that the tableau is treated in horizontal mode. You are on your own if you put this inside of another box.

```

132   \leavevmode%
133   \compare@YT{\centering@YT}{center}%
134   \ifeq@YT%
135     \hbox{\$ \vcenter{\box\tableaux@YT}$}%
136   \else\compare@YT{\centering@YT}{bottom}%
137   \ifeq@YT%
138     \hbox{\raise\dp\tableaux@YT\box\tableaux@YT}%
139   \fi%
140   \box\tableaux@YT%
141   \fi%
142 }%
143 }

```

`\cr@YT` Annoying to have to do this, but nested `halign` chokes when `\cr` appears inside the definition.

```

144 \def\cr@YT{\cr}

```

`\none` This one is for omitting entries but leaving their space. We also allow something to be placed in the empty space (e.g. `\dots`), but don't allow color (that would defeat the purpose of omitting the box). To support the optional argument without screwing up the `\omit`, we have to go in two steps.

```
145 \def\none{\omit\none@YT}
```

`\none@YT` This finds the optional argument to `\none` and makes the box itself. We draw an invisible frame by replacing the actual frame with the frame separation. We also use `\ignorespaces` to eat up any spaces that occur after `\none`; in a normal box, they would be chopped in math mode.

```
146 \newcommand{\none@YT}[1] [] {%
147 \def\thisboxcolor@YT{clear}%
148 \setlength{\fboxsep}{\boxframe@YT}%
149 \setlength{\fboxrule}{0pt}%
150 \startbox@YT#1\endbox@YT%
151 \ignorespaces%
152 }
```

`\startbox@YT` #1 = general formatting, #2 = specific formatting. We want to extract the colors from each and then pass the whole thing on to `\startbox@@YT`.

```
153 \def\startbox@YT#1#2{%
```

We get the colors and then put the rest into temporary token registers.

```
154 \getcolor@YT{\save@YT{\opttoksa@YT}}#1\@nil%
155 \getcolor@YT{\save@YT{\opttoksb@YT}}#2\@nil%
```

Now we get the color from the entry and proceed.

```
156 \getcolor@YT%
157 {\startbox@@YT\the\opttoksa@YT\the\opttoksb@YT}%
158 }
```

`\save@YT` Stick the following text into the token register in #1. Note that we use `\@nil` as an end-marker; it is not actually defined, so hopefully we never expand it!

```
159 \def\save@YT#1#2\@nil{#1={#2}}
```

`\getcolor@YT` #1 is pasted in front of what remains after removing the color. Basically, it's a "do next".

```
160 \def\getcolor@YT#1{\ifstar@YT{\getcolor@@YT{#1}}{#1}}
```

`\getcolor@@YT` Save the (optional) color argument and pass the rest to `\startbox@@YT`.

```
161 \def\getcolor@@YT#1(#2){%
162 \def\thisboxcolor@YT{#2}%
163 #1%
164 }
```

`\startbox@@YT` Start collecting the current entry into a horizontally-centered hbox, but save the result.

```
165 \def\startbox@@YT{%
```

Use a `\bgroup... \egroup` so as not to introduce nesting that would block `&` or `\cr`.

```
166 \setbox\thisbox@YT=\hbox to \boxdim@YT\bgroup%
167   \hss%
168   \skipin@YT%
169 }
```

Since we are now in the intended typesetting context (i.e. an `hbox` with math mode on) we can let `\halign` expand tokens in the rest of the entry until it finds a `&` or `\cr` (`= \`)

`\endbox@YT` Now we can finish the box and set it.

```
170 \def\endbox@YT{%
171   \skipout@YT%
172   \hss%
173   \egroup%
```

We want all the boxes to have a consistent baseline, so we normalize them to the same size. Multiple text lines will be aligned with the baseline of the last line at the center, so this really only works well for single lines of text.

```
174 \ht\thisbox@YT=\ht\refhtdp@YT%
175 \dp\thisbox@YT=\dp\refhtdp@YT%
176 \fcolorbox@YT{\thisboxcolor@YT}{%
177   \vbox to \boxdim@YT{\vss\box\thisbox@YT\vss}%
178   }%
179 }
```

`\fcolorbox@YT` We need a wrapper around `\fcolorbox` since it produces an opaque box, and sometimes, we want `clear`.

#1 = color, #2 = contents

```
180 \def\fcolorbox@YT#1#2{%
181   \compare@YT{#1}{clear}%
182   \ifeq@YT%
```

Clear background; don't draw anything.

```
183   \fbox{#2}%
184   \else%
```

Colored background; pass it to `\fcolorbox`.

```
185   \fcolorbox{.}{#1}{#2}%
186   \fi%
187 }
```

## 5.4 `ytableaushort` command

`\ytableaushort` The short form of `ytableau`. It takes a comma-separated list of lines, each one a string of entries given as individual tokens. `{...}` is allowed (and encouraged) for complex entries, and color is possible. All sorts of redundancies in the syntax are allowed.

```
188 \newcommand{\ytableaushort}[2] [] {%
```

\endytableau has to be right at the end, so we can't use scope to reset \toks@YT.

```
189 \toks@YT={}%
190 \getentries@YT{\getentries@YT}{#2,\@nil%
191 \ytableau[#1]\the\toks@YT\endytableau%
192 }
```

\getentries@YT Split the CSV into rows. This is really a job for etoolbox:\docsvlist but whatever. We put a . in front of the string so that a line may be enclosed entirely in {...} Otherwise, \def\cs#1,{etc} would make #1 = ... and not #1 = {...} as we want.

```
193 \def\getentries@YT#1#2{\getline@YT{#1}{#2}.}
```

\getline@YT Grab the first *line*, in the string and remove the initial .

```
194 \def\getline@YT#1#2#3,{%
195 \opttoksa@YT=\expandafter{\@gobble#3}%
196 \opttoksb@YT={\getline@YT{#1}{#2}}%
```

We pass #3 back as an *argument* to \getline@YT, thus avoiding the braces issue.

```
197 \edef\next@YT{\the\opttoksb@YT{\the\opttoksa@YT}}%
198 \next@YT%
199 }
```

\getline@YT #1 = the macro to process each row, #2 = the junk to put after each row (followed by \@nil), #3 = everything before the first comma, #4 = the token after the first comma (possibly another comma).

```
200 \def\getline@YT#1#2#3#4{%
```

Handle double commas or trailing commas.

```
201 \compare@YT{#4}{,}%
202 \ifeq@YT%
```

Try again.

```
203 \def\next@YT{\getline@YT{#1}{#2}{#3}}%
204 \else%
```

If this is not the last row, we have to recurse down the list. Otherwise, just process the current row.

```
205 \compare@YT{#4}{\@nil}%
206 \ifeq@YT%
207 \def\next@YT{#1#3#2\@nil}%
208 \else%
```

#4 is a single token, so it should be replaced as one. Note that this adds braces; we will have to correct for this all the way down in \fullexpand@YT when we don't want them.

```
209 \def\next@YT{#1#3#2\@nil\getentries@YT{#1}{#2}{#4}}%
210 \fi%
211 \fi%
212 \next@YT%
213 }
```



`\getentries@YT` Separates the entries in a line of `\ytableaushort` and reformats them for `\ytableau`. Takes two tokens and checks if the second is `\@nil`, which means the first is the last entry.

```

214 \def\getentries@YT#1#2{%
    If this is not the last entry, we have to recurse down the line. Otherwise, we just
    print \\.
215 \compare@YT{#2}{\@nil}%
216 \ifeq@YT%
217 \toks@YT=\expandafter{\the\toks@YT#1\}%
218 \def\next@YT{}%
219 \else%
220 \toks@YT=\expandafter{\the\toks@YT#1&}%
221 \def\next@YT{\getentries@YT{#2}}%
222 \fi%
223 \next@YT%
224 }

```

## 5.5 `ydiagram` command

`\ydiagram` Takes the same optional argument as the other macros. Its main argument `#2` is of the form

$[\langle offset \rangle + ]\langle number \rangle, \dots$

where both  $\langle offset \rangle$  and  $\langle number \rangle$  may be any expression evaluating to a textual number (e.g. `\the\count<n>` rather than `\count<n>`).

```

225 \newcommand\ydiagram[2] []{%

```

We need `\endytableau` to be right at the end, so we can't use scope to reset `\toks@YT`.

```

226 \toks@YT={}%
227 \getentries@YT{\getnumbers@YT}{+}#2,\@nil%
228 \ytableau[#1]\the\toks@YT\endytableau%
229 }

```

`\getnumbers@YT` Separates the entries in a line of `\ydiagram` and reformats them for `\ytableau`.

```

230 \def\getnumbers@YT#1+#2\@nil{%

```

If `#2 = {}`, then there is no offset and `#1` is the row shape.

```

231 \compare@YT{#2}{}%
232 \ifeq@YT%
233 \def\next@YT{}%
234 \loop@YT{#1}{}%
235 }%

```

Else `#1` is the offset and `#2` is the shape.

```

236 \else%
237 \def\next@YT{}%
238 \loop@YT{#1}{\none}%

```

Now #2 looks like  $\langle number \rangle +$ , so we feed it back in.

```
239 \getnumbers@YT#2\@nil%
240 }%
241 \fi%
242 \next@YT%
243 }
```

`\loop@YT` Loops on the first argument, building a `\ytableau` line whose entries are the second argument. The results go in `\toks@YT`.

```
244 \def\loop@YT#1#2{%
Fills \count@YT.
245 \fullexpand@YT{#1}%
246 \loop\ifnum\count@YT>1%
247 \toks@YT=\expandafter{\the\toks@YT#2&}%
248 \advance\count@YT by -1%
249 \repeat%
```

The last entry in the list may not be the last entry in the line. If it's empty, it is (according to our usage), otherwise not.

```
250 \ifnum\count@YT=1%
251 \compare@YT{#2}{}%
252 \ifeq@YT%
253 \toks@YT=\expandafter{\the\toks@YT#2\}%
254 \else
255 \toks@YT=\expandafter{\the\toks@YT#2&}%
256 \fi%
257 \fi%
258 }
```

`\fullexpand@YT` Fully expand its argument. Hopefully you passed an argument for which that makes sense. This also strips braces.

```
259 \def\fullexpand@YT#1{%
TeX's brace parser will strip the braces for us.
260 \def\tmpa@YT##1{##1}%
```

Turns e.g. both  $\#1 = \{23\}$  and  $\#1 = 23$  into 23. As a side effect, e.g. if  $\#1 = \{2\}3$ , the result is 23. However,  $\#1 = 2\{3\}$  stays that way, so is probably an error.

```
261 \edef\tmpb@YT{\tmpa@YT#1}%
262 \edef\tmpc@YT{\tmpb@YT}%
263 \count@YT=\tmpc@YT%
264 }
```

## 6 Change History

v1.0		
General: Initial version	.....	1
v1.1		
General: Changed the namespace convention from <code>ytableau@...</code> to <code>...@YT</code> to make the index useful and the macros shorter.	.....	7
Removed null initializations of temp macros <code>\tmp(a,b,c)@YT</code> .	.....	9
<code>nosmalltableaux</code> : Correctly reassign a positive value to <code>\boxdim@save@YT</code> when exiting small tableaux.	.....	10
<code>\refhtdp@YT</code> : Put the reference letters into a single box and renamed it more semantically.	.....	8
	<code>\skipout@YT</code> : Split up the “skip in” and “skip out” macros.	9
	<code>ytableau</code> : Added the <code>everycr</code> fix.	12
	Moved the <code>leavevmode</code> down to the output routine and simplified the alignment computations now that the <code>everycr</code> fix is in.	13
	Removed an unnecessary layer of boxes and moved the fake braces into the outer layer.	12
	Replaced explicit setting of the chained box width by a kern.	13
	Save the box width difference rather than the larger width.	13
	<code>\thisbox@YT</code> : Made <code>\thisbox@YT</code> an alias for <code>\thistableau@YT</code> .	7

## 7 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.

<b>Symbols</b>		
<code>\@nil</code> . . . . .	154, 155, 159, 190, 205, 207, 209, 215, 227, 230, 239	
<code>\\</code> . . . . .	108, 217, 253	
<b>A</b>		
<code>aligntableaux</code> (option) . . . . .	5, <u>52</u>	
<b>B</b>		
<code>\boxdim@normal@YT</code> . . . . .	9, 14, 47, 73	
<code>\boxdim@save@YT</code> <u>9</u> , 64, 66, 70, 72, 73		
<code>\boxdim@YT</code> . . . . .	<u>13</u> , 47, 49, 66, 67, 72, 166, 177	
<code>\boxframe@YT</code> . . . . .	<u>16</u> , 106, 148	
<code>boxsize</code> (option) . . . . .	5, <u>43</u>	
<b>C</b>		
<code>\centering@YT</code> <u>22</u> , 54, 57, 60, 133, 136		
<code>centertableaux</code> (option) . . . . .	5, <u>52</u>	
<code>\compare@@YT</code> . . . . .	<u>24</u> , 133, 136, 181	
<code>\compare@YT</code> . . . . .	<u>24</u> , 45, 201, 205, 215, 231, 251	
<code>\count@YT</code> . . . . .	<u>17</u> , 246, 248, 250, 263	
<code>\cr</code> . . . . .	114, 144	
<code>\cr@YT</code> . . . . .	108, <u>144</u>	
<b>D</b>		
<code>\defarg@YT</code> . . . . .	<u>23</u> , 65, 71, 113	
<b>E</b>		
<code>\endbox@YT</code> . . . . .	114, 150, <u>170</u>	
<code>\endytableau</code> . . . . .	191, 228	
environments:		
<code>ytableau</code> . . . . .	3, <u>103</u>	
<code>\everycr</code> . . . . .	112	
<b>F</b>		
<code>\fbox</code> . . . . .	183	
<code>\fboxrule</code> . . . . .	106, 110, 111, 149	
<code>\fboxsep</code> . . . . .	107, 148	
<code>\fcolorbox</code> . . . . .	185	
<code>\fcolorbox@YT</code> . . . . .	176, <u>180</u>	
<code>\fullexpand@YT</code> . . . . .	245, <u>259</u>	
<b>G</b>		
<code>\getcolor@YT</code> . . . . .	160, <u>161</u>	
<code>\getcolor@YT</code> . . . . .	154, 155, 156, <u>160</u>	<code>nosmalltableaux</code> . . . . . 5, <u>62</u>
<code>\getentries@@YT</code> . . . . .	190, <u>214</u>	<code>smalltableaux</code> . . . . . 5, <u>62</u>
<code>\getentries@YT</code> . . . . .	190, <u>193</u> , 209, 227	<code>textmode</code> . . . . . 5, <u>84</u>
<code>\getline@@YT</code> . . . . .	196, <u>200</u>	<code>\opttoksa@YT</code> . . . . . <u>7</u> , 154, 157, 195, 197
<code>\getline@YT</code> . . . . .	193, <u>194</u>	<code>\opttoksb@YT</code> . . . . . <u>7</u> , 155, 157, 196, 197
<code>\getnumbers@YT</code> . . . . .	227, <u>230</u>	
<b>H</b>		
<code>\halign</code> . . . . .	113	
<b>I</b>		
<code>\ifeq@YT</code> . . . . .	<u>24</u> , 46, 134, 137, 182, 202, 206, 216, 232, 252	
<code>\iffalse</code> . . . . .	105, 118	
<code>\ifstar@YT</code> . . . . .	18, 129, 160	
<code>\ignorespaces</code> . . . . .	86, 151	
<b>K</b>		
<code>\kern</code> . . . . .	128	
<code>\kernel@ifnextchar</code> . . . . .	18	
<b>L</b>		
<code>\loop@YT</code> . . . . .	234, 238, <u>244</u>	
<b>M</b>		
<code>mathmode</code> (option) . . . . .	5, <u>84</u>	
<b>N</b>		
<code>\next@YT</code> 197, 198, 203, 207, 209, 212, 218, 221, 223, 233, 237, 242		
<code>nocentertableaux</code> (option) . . . . .	5, <u>52</u>	
<code>\none</code> . . . . .	3, <u>145</u> , 238	
<code>\none@YT</code> . . . . .	145, <u>146</u>	
<code>nosmalltableaux</code> (option) . . . . .	5, <u>62</u>	
<b>O</b>		
<code>\offinterlineskip</code> . . . . .	109	
<code>\omit</code> . . . . .	145	
<code>\openup</code> . . . . .	110	
options:		
<code>aligntableaux</code> . . . . .	5, <u>52</u>	
<code>boxsize</code> . . . . .	5, <u>43</u>	
<code>centertableaux</code> . . . . .	5, <u>52</u>	
<code>mathmode</code> . . . . .	5, <u>84</u>	
<code>nocentertableaux</code> . . . . .	5, <u>52</u>	
<b>R</b>		
<code>\raise</code> . . . . .	138	
<code>\refhtdp@YT</code> . . . . .	<u>4</u> , 174, 175	
<code>\RequirePackage</code> . . . . .	41, 102	
<b>S</b>		
<code>\save@YT</code> . . . . .	154, 155, <u>159</u>	
<code>\skipin@YT</code> . . . . .	<u>19</u> , 86, 89, 168	
<code>\skipout@YT</code> . . . . .	<u>19</u> , 87, 90, 171	
<code>smalltableaux</code> (option) . . . . .	5, <u>62</u>	
<code>\startbox@@YT</code> . . . . .	150, 157, <u>165</u>	
<code>\startbox@YT</code> . . . . .	113, <u>153</u>	
<b>T</b>		
<code>\tableauwd@YT</code> <u>15</u> , 120, 121, 123, 128		
<code>\tableaux@YT</code> . . . . .	<u>1</u> , 119, 121, 126, 127, 135, 138, 140	
<code>\tabskip</code> . . . . .	111	
<code>textmode</code> (option) . . . . .	5, <u>84</u>	
<code>\thisbox@YT</code> . . . . .	<u>3</u> , 166, 174, 175, 177	
<code>\thisboxcolor@YT</code> <u>21</u> , 147, 162, 176		
<code>\thistableau@YT</code> . . . . .	. . . . .	
. . . . .	<u>1</u> , 3, 105, 119, 120, 125, 127	
<code>\tmpa@YT</code> . . . . .	25, 26, 260, 261	
<code>\tmpb@YT</code> . . . . .	25, 26, 33, 34, 261, 262	
<code>\tmpc@YT</code> . . . . .	262, 263	
<code>\toks@YT</code> . . . . .	<u>6</u> , 189, 191, 217, 220, 226, 228, 247, 253, 255	
<b>U</b>		
<code>\unskip</code> . . . . .	87	
<b>Y</b>		
<code>\ydiagram</code> . . . . .	4, 130, <u>225</u>	
<code>\ytableau</code> . . . . .	191, 228	
<code>ytableau</code> (environment) . . . . .	3, <u>103</u>	
<code>\ytableausetup</code> . . . . .	5, <u>42</u>	
<code>\ytableaushort</code> . . . . .	4, <u>188</u>	