

Documentation for the OTtablx Package

Nathan Sanders

nathan.sanders@utoronto.ca

July 18, 2026

Contents

1	Purpose	2
2	The Package	3
2.1	Getting Started	3
2.2	Package Options	3
2.3	Known Package Conflicts	4
3	Basic Tableaux	5
3.1	The OTtableau Environment	5
3.2	Tableau-Specific Options	5
3.3	Vertical Separators	5
4	Cell Contents	7
4.1	The Input/Constraint Row	7
4.2	The Candidate/Violation Rows	8
4.3	Complex Cell Entries	9
5	Advanced Formatting with PSTricks	10
6	Comparative Tableaux	11
7	Miscellaneous Macros	12
8	The Future	12

What's New in v0.4.1

- Deprecated and updated documentation in preparation for PGF/TikZ support in version 0.5

This manual describes OTtblx version 0.4.1, a deprecated version.¹ Feel free to use the package for your own serious work, but do not expect current functionality to be replicated in later versions, since commands, options, and default settings may be renamed, revised, or eliminated entirely. The most recent version of OTtblx can be found at:

<http://sanders.phonologist.org/OTtblx/>

If you find a bug, want to request a new feature, or even have advice on improving the package code, please note that this version is no longer supported. Please check out version 0.5, which completely rebuilds the OTtblx architecture with PDF/TikZ.

1 Purpose

The OTtblx package is designed to give linguists the ability to create and customize Optimality Theoretic (OT; Prince and Smolensky 1993/2004) tableaux in L^AT_EX. A sample OT tableau as generated by OTtblx is given below:

	dap	MAX	DEP	*CODA
a.	da	*!		
b.	dapa		*!	
c.	 dap			*

OTtblx only generates the basic layout and formatting of OT tableaux; it does *not* do any sort of automated computation of constraint violations, fatal violations, or winning candidates. While OTtblx was designed primarily with OT phonology in mind, it should be suitable for other uses of OT, such as in syntax.

A number of design criteria were considered when deciding how tableaux should be formatted. While some tableau properties were left as options for the user, many properties had to be held constant and are not customizable with OTtblx at this time. If there is sufficient demand for a particular customization that I have not yet implemented—and if it’s within my power to do so—I’ll eventually get around to updating the package. But for now, if you want to change something that doesn’t have an explicit option, you’ll have to tweak the guts of the package on your own.

A few particular layout decisions warrant brief discussion. Row heights and depths are a bit more spacious than in the standard L^AT_EX tabular environment, to allow IPA diacritics to fit more comfortably. In addition, because OT tableaux tend to make heavier use of tall characters and upper diacritics than descenders and lower diacritics (especially in constraint names, violation marks, candidate labels, and even phonological strings), there is slightly more space at the top of a row than at the bottom, for overall better visual balance.

Further, the double line separators in OTtblx intersect with their middles merged, rather than with the (uglier) asymmetrically layered look of L^AT_EX’s regular tabular environment:

ugly	layered
double	lines

¹Thanks to Michael Becker for his help in testing previous versions of this package and to Alan Munn for writing and testing compatibility code for X_YL^AT_EX.

The tabular double lines are also inconsistent with respect to whether or not they break the outside border: the horizontal double line does break the border, but the vertical double line does not. In `OTtblx`, the double lines behave consistently, either both breaking the outside border or both leaving the border intact (user's choice).

Finally, there is no cell shading in this version of `OTtblx`, but it has been implemented in version 0.5.

2 The Package

2.1 Getting Started

To enable use of the package and its commands, make sure that the `OTtblx.sty` file is in your $\text{\LaTeX}$ path, and add the following command to your document preamble:

■ `\usepackage [options] {OTtblx}`

Note that `OTtblx` loads the packages `pstricks`, `pst-node`, `pst-coil`, `calc`, `ifthen`, `pifont`, `wasysym` so you do not need to load them yourself.

Use with $\text{\LaTeX}$ When used with `latex+dvips`, `OTtblx` also loads the `tipa` package. If you need a particular option from that package (such as TIPA's `[vowel]` or `[extra]` options), you must load the package with its options *before* loading `OTtblx`.

Use with $\text{Xe}\text{\LaTeX}$ When used with `xelatex`, `OTtblx` also loads the `fontspec` package, using the Doulos SIL font by default. This font is free and available from [SIL](#). This provides two commands: `\ipafont` and `\textipa{}`. To change the IPA font, use the following command, which behaves just like the `\newfontfamily` command from `fontspec`.

■ `\DefineIPAfont [fontspec options] {fontname}`

2.2 Package Options

`OTtblx` comes with multiple options for customizing the global format of tableaux in a document. The following options are currently available (defaults are italicized):

- *square* tableau corners are squared (standard)
- *round* tableau corners are rounded (funky yet sleekly modern)

- *closed* outside border is not broken by double lines (Kager 1999)
- *open* outside border is broken by double lines (McCarthy 2002)
- *noborder* outside border is not drawn (Prince and Smolensky 1993/2004)

- *alpha* candidates are labeled with lowercase letters: a, b, c (standard)
- *arabic* candidates are labeled with Arabic numerals: 1, 2, 3
- *roman* candidates are labeled with lowercase Roman numerals: i, ii, iii
- *nolabel* candidates aren't labeled

- *light* line thickness is — (0.5pt, suitable for thinner fonts, like Computer Modern)
- *medium* line thickness is — (0.75pt, suitable for medium fonts, like Times)
- *heavy* line thickness is — (1.0pt, suitable for thicker fonts, like Bookman)

- *ipa* input and candidates in left column are automatically enclosed inside an IPA font (the TIPA fonts for L^AT_EX, Doulos SIL for X_YL^AT_EX)
- *noipa* input and candidates in left column are unformatted (suitable for syntax tableau or other non-phonetic tableaux)

If no options are specified, as is the case for this document, the default style is used, which is equivalent to calling `OTtblx` as follows:

`\usepackage [square,closed,alpha,light,ipa] {OTtblx}`

dap	MAX	*CODA
a. da	*!	
b.  dap		*

The formatting options may be used in any combination and in any order. Further, if conflicting options are listed, the last one declared on the list will take effect. For example, the option list `[square,round]` is functionally equivalent to just using `[round]`. If you want to use any default options, you do not need to list them, but you may choose to do so for clarity. The following tableaux show various combinations of formatting options. Note how default options are loaded automatically when they do not conflict with any of the declared options:

`\usepackage [arabic,round,medium] {OTtblx}`

dap	MAX	*CODA
1. da	*!	
2.  dap		*

`\usepackage [open,nolabel,heavy] {OTtblx}`

dap	MAX	*CODA
da	*!	
 dap		*

`\usepackage [noborder,roman] {OTtblx}`

dap	MAX	*CODA
i. da	*!	
ii.  dap		*

`\usepackage [noipa] {OTtblx}`

DAP	MAX	*CODA
a. DA	*!	
b.  DAP		*

2.3 Known Package Conflicts

`OTtblx` makes heavy use of `PSTricks`, which precludes direct use of `pdfLATEX` to generate PDFs. There are some solutions to the PS/PDF conflict that will hopefully allow you to use `OTtblx` with only minor tweaking to your usual `LATEX` routine. Version 0.5 uses `PGF/TikZ`, so it can be used with `pdfLATEX` directly.

3 Basic Tableaux

3.1 The OTtableau Environment

The core feature of the OTtblx package is the relatively simple and customizable OTtableau environment, which takes one optional argument for listing formatting options (see §3.2) and one obligatory argument that specifies the number of constraints, which must be a positive integer:

```
■ \begin{OTtableau} [options] {num_constraints}
    ⋮
\end{OTtableau}
```

In addition, various commands for drawing vertical constraint separators are enabled within the OTtableau environment. These are described in §3.3.

3.2 Tableau-Specific Options

The optional argument in the OTtableau environment allows you to temporarily override the global tableau format for one specific tableau. The possible options are the same as the global options available for the package: [square], [round], [nolabel], etc.

To use any of these options, simply include them in a comma-separated list inside square brackets [] when calling the OTtableau environment. The tableaux in the previous section were typeset in this manner, with tableau-specific options for each one:

```
\begin{OTtableau} [arabic,round,medium] {2}
```

	dap	MAX	*CODA
1.	da	*!	
2.	⌈ ₃ dap		*

You should get no error if you happen to use a tableau-specific option that matches the global setting for the current document; OTtblx will just silently format the tableau with the logical result.

3.3 Vertical Separators

Inside the OTtableau environment, you can issue commands to determine what type of vertical separators—solid, dashed, or jagged—are drawn between the constraint columns. There are three commands, one for each type, all with the same syntax:

```
■ \OTsolids [PSTricks_options] {constraint_number_list}
■ \OTdashes [PSTricks_options] {constraint_number_list}
■ \OTjagged [PSTricks_options] {constraint_number_list}
```

Usage of the optional argument for these commands is described in §5. The obligatory argument for each is a comma-separated list of numbers that determine which constraint columns are to be followed by that type of separator. For example, the command \OTdashes{2,4,1} will put dashed separators after the first, second, and fourth constraint columns. Notice that order of the numbers doesn't matter. Also, repeated numbers and numbers larger than the current number of constraints should produce no noticeable effect.

The commands for vertical separators can be put anywhere inside the `OTtableau` environment, though for ease of reading your code, it's recommended that they all be put in the same place, either at the beginning of the environment or at the end. The following tableau shows how to use the `\OTsolids` command to get two solid separators:

```
\begin{OTtableau}{3}
  \OTsolids{1,2}
```

dap	MAX	DEP	*CODA
a. da	*!		
b. dapa		*!	
c.  dap			*

This tableau shows the use of both dashed and solid separators at the same time:

```
\begin{OTtableau}{3}
  \OTdashes{1}
  \OTsolids{2}
```

dap	MAX	DEP	*CODA
a. da	*!		
b. dapa		*!	
c.  dap			*

The vertical separator commands can be issued in any order, and they can even be put on the same line. If you do not want any constraint separators, just issue no vertical separator commands at all:

```
\begin{OTtableau}{3}
```

dap	MAX	DEP	*CODA
a. da	*!		
b. dapa		*!	
c.  dap			*

The third separator command, `\OTjagged`, is specially designed for constructing a tableau that shows multiple separate constraint rankings at the same time. Jagged separators always break the outside border, even when the `[closed]` option is set, and they always cross over the double line rather than merging it with.

```
\begin{OTtableau}{4}
  \OTsolids{1,3}
  \OTjagged{2}
```

dap	MAX	*CODA	IDENT	*VOIOBS
a. da	*!			*
b. tap		*	*!	
c.  dap		*		*

Currently, issuing multiple instances of the same vertical separator command in the same tableau, such as `\OTsolids{1}` followed by `\OTsolids{3}`, will result in only the last usage being obeyed. I'm working on a fix for this, so that vertical separators of the same type but with different options can be used within the same tableau.

4 Cell Contents

4.1 The Input/Constraint Row

The OTtableau environment also enables special commands for specifying the contents of the cells. The command for cell contents that should be issued first is `\OTtoprow` or its starred version `\OTtoprow*`, both of which have one optional argument for the input and one obligatory argument for the list of constraints:

- `\OTtoprow` [*input*] {*constraint_list*}
- `\OTtoprow*` [*input*] {*constraint_list*}

Because the input for these commands is optional, you can leave out the square brackets [] when you wish to leave the input cell blank.

Because most OT tableaux are used for phonology, the input is automatically embedded within a phonetic font command (using the TIPA package for L^AT_EX, and the Doulos SIL font for X_YL^AT_EX), so you can enter phonetic characters in the input without having to put it inside a phonetic font command every time. For example, to get *ðaʔ* to appear as your input in L^AT_EX, you can simply use the TIPA shortcut `DAP`, rather than having to write out `\textipa{DAP}`. To temporarily get non-phonetic text in the input when using L^AT_EX, use either the `\normalfont` command (the general recommendation, especially if your input is large or complex) or TIPA's `\*` command (which works for most, but not all, cases). If you want to suppress phonetic font embedding entirely, either for a given tableaux or for the entire document, use the `[noipa]` option. If you use `[noipa]` globally, but want a specific tableaux to have phonetic font embedding, you can use the local `[ipa]` option to switch it back on for that tableau.

The constraints in the obligatory argument must appear as a comma-separated list, as in the following (note that spacing within the constraint list and between the input and constraint list is ignored, which allows you to space out your code for clarity):

```
\begin{OTtableau}{3}
  \OTsolids{2}\OTdashes{1}
  \OTtoprow [dap] {\textsc{Max},
                  \textsc{Dep},
                  *\textsc{Coda}}
```

	dap	MAX	DEP	*CODA
a.	da	*!		
b.	dapa		*!	
c.	 dap			*

`\OTtoprow` and `\OTtoprow*` have nearly identical effects, except for the placement of the input within the top left cell. `\OTtoprow` aligns the input with the candidate list, while `\OTtoprow*` centers the input in the top left cell, across all three of the label, marker, and candidate. Centering is useful when the input is very wide in comparison to the candidates, as is often the case when the input contains multiple morpheme boundary symbols, like in the following tableau:

```
\begin{OTtableau}{3}
  \OTsolids{2}\OTdashes{1}
  \OTtoprow* [d+a+p] {\textsc{Max},
                     \textsc{Dep},
                     *\textsc{Coda}}
```

	d+a+p	MAX	DEP	*CODA
a.	da	*!		
b.	dapa		*!	
c.	 dap			*

4.2 The Candidate/Violation Rows

The rest of a tableau’s contents are entered with `\OTcandrow` and/or `\OTcandrow*`, one of which must be used for every candidate row. These commands have one optional argument for marking the candidate in a special way (such as using ☞ to indicate the winner), one obligatory argument for the candidate itself, and a second obligatory argument containing a list of the candidate’s constraint violations:

- `\OTcandrow` [*marker*] {*candidate*} {*violation_list*}
- `\OTcandrow*` [*marker*] {*candidate*} {*violation_list*}

The special candidate marker is optional, and in most cases, you won’t need one, so you can usually leave the square brackets [] off. The `OTtblx` package defines the command `\OThand` to produce the traditional OT pointing hand ☞ used for marking the winner, but you are free to use any other symbol or command you have defined as a candidate marker. (See §7 for other common markers defined in `OTtblx`.)

By default, the candidate is embedded inside a phonetic font, just like the input for `\OTtoprow` and `\OTtoprow*`, so TIPA shortcuts can be used to enter phonetic text in $\text{\LaTeX}$, and non-phonetic text needs to be protected by either the `\normalfont` or `\*` commands. As with the input, the options `[ipa]` and `[noipa]` control the general behavior of phonetic font embedding, turning it on or off (respectively) for both the input and the candidates.

The violations must be a comma-separated list that matches the order of the constraints listed in `\OTtoprow` or `\OTtoprow*`, as in the following complete code for a full tableau:

```
\begin{OTtableau}{3}
  \OTsolids{2}\OTdashes{1}
  \OTtoprow [dap] {\textsc{Max},\textsc{Dep},*\textsc{Coda}}
  \OTcandrow {da} {*, , }
  \OTcandrow {dapa} { ,*, , }
  \OTcandrow [\OThand]
    {dap} { , , *, }
\end{OTtableau}
```

	dap	MAX	DEP	*CODA
a.	da	*!		
b.	dapa		*!	
c.	☞ dap			*

`\OTcandrow` and `\OTcandrow*` have nearly identical effects, except for the relative placement of the candidate marker and the candidate label. Unstarred `\OTcandrow` puts the candidate label to the left of the candidate marker, while `\OTcandrow*` puts them in the opposite order, with the marker on the outside:

```
\begin{OTtableau}{3}
  \OTsolids{2}\OTdashes{1}
  \OTtoprow [dap] {\textsc{Max},
    \textsc{Dep},
    *\textsc{Coda}}
  \OTcandrow* {da} {*, , , }
  \OTcandrow* {dapa} { ,*, , }
  \OTcandrow* [\OThand]
    {dap} { , , *, }
\end{OTtableau}
```

	dap	MAX	DEP	*CODA
a.	da	*!		
b.	dapa		*!	
☞ c.	dap			*

4.3 Complex Cell Entries

For most normal uses, directly entering the various arguments for the cell content commands will work fine. If you need to do something special, such as including a comma as part of a constraint name or violation, putting curly braces { } around the special argument will usually suffice:

```
\begin{OTtableau}{2}
  \OTsolids{2}\OTdashes{1}
  \OTtoprow [dap] {{\textsc{Max}},\textsc{Dep}},*\textsc{Coda}}
  \OTcandrow {da} {*, }
  \OTcandrow {dapa} {*, }
  \OTcandrow [\OTthand]
    {dap} { ,*}
\end{OTtableau}
```

	dap	MAX,DEP	*CODA
a.	da	*!	
b.	dapa	*!	
c.	 dap		*

However, sometimes the arguments you want to use may be so complex that curly braces are not sufficient protection, in which case, you need to use a macro instead. You may wish to use a macro anyway for long or complicated arguments, whether they need to be defined that way or not, since judicious use of macros can improve the legibility of your source code.

For example, to use a table as a candidate, you could simply enclose the table with curly braces, but it will be easier to read the underlying code if you define a macro instead, as in the following:

```
\begin{OTtableau}{2}
  \OTsolids{1}
  \OTtoprow [\dapHH] {OCP,\textsc{Ident}}
  \OTcandrow {\dapHH} {*, }
  \OTcandrow [\OTthand]
    {\dapHL} { ,*}
\end{OTtableau}
```

	$\begin{array}{c} H\ H \\ \ \\ \hline \text{dap} \end{array}$	OCP	IDENT
a.	$\begin{array}{c} H\ H \\ \ \\ \hline \text{dap} \end{array}$	*!	
b.	$\begin{array}{c} H\ L \\ \ \\ \hline \text{dap} \end{array}$ 		*

where \dapHH and \dapHL were defined before the tableau with the following:

```
\def\dapHH
{\begin{tabular}[b]{@{}c@{}c@{}c@{}c@{}}
  & \makebox[0pt]{\*H}& & \makebox[0pt]{\*H}\\
  & \vline& & \vline\\
  d&a&p&a\\
\end{tabular}}

\def\dapHL
{\begin{tabular}[b]{@{}c@{}c@{}c@{}c@{}}
  & \makebox[0pt]{\*H}& & \makebox[0pt]{\*L}\\
  & \vline& & \vline\\
  d&a&p&a\\
\end{tabular}}
```

5 Advanced Formatting with PSTricks

Various customizations to the lines drawn in OTtblx tableaux can be made with knowledge of PSTricks. Specifically, the commands `\OTsolids`, `\OTdashes`, and `\OTjagged` have the ability to take any PSTricks style commands in their optional argument, as a comma-separated list. Some useful PSTricks options for separators include:

- `linewidth=dim` changes the thickness of the line
- `linecolor=color` changes the color of the line (default is black; other available built-in colors are darkgray, gray, lightgray, white, red, green, blue, yellow, magenta, and cyan; new colors can be defined with psticks commands such as `\newgray`, `\newrgbcolor`, etc.)
- `dash=dim1 dim2` changes the length (*dim1*) and spacing (*dim2*) of dashes (default is a 0.75ex dash with 0.75ex spacing)

The following tableau demonstrates changing linewidth and linecolor of the separators:

```
\begin{OTtableau}{3}
  \OTsolids[linewidth=2pt, linecolor=cyan]{2}
  \OTdashes[linecolor=red, linewidth=1pt]{1}
  \OTtoprow [dap] {\textsc{Max},\textsc{Dep},*\textsc{Coda}}
  \OTcandrow {da} {*, , }
  \OTcandrow {dapa} { ,*, , }
  \OTcandrow [\OTthand]
    {dap} { , ,*}
\end{OTtableau}
```

	dap	MAX	DEP	*CODA
a.	da	*!		
b.	dapa		*!	
c.	$\mathbb{S}$ dap			*

Even more useful is the ability to change the measurements for dashed separators, so that you can make them look exactly how you want them. The following tableau shows a dashed separators with longer dashes:

```
\begin{OTtableau}{3}
  \OTsolids{2}
  \OTdashes[dash=5pt 2pt]{1}
  \OTtoprow [dap] {\textsc{Max},
    \textsc{Dep},
    *\textsc{Coda}}
  \OTcandrow {da} {*, , }
  \OTcandrow {dapa} { ,*, , }
  \OTcandrow [\OTthand]
    {dap} { , ,*}
\end{OTtableau}
```

	dap	MAX	DEP	*CODA
a.	da	*!		
b.	dapa		*!	
c.	$\mathbb{S}$ dap			*

Shortening the dash results in a more dotted look:

```

\begin{OTtableau}{3}
  \OTSolids{2}
  \OTdashes[dash=1pt 2pt]{1}
  \OTtoprow [dap] {\textsc{Max},
                  \textsc{Dep},
                  *\textsc{Coda}}
  \OTcandrow {da}  {*, , }
  \OTcandrow {dapa} { ,*, }
  \OTcandrow [\OTthand]
              {dap} { , ,*}
\end{OTtableau}

```

	dap	MAX	DEP	*CODA
a.	da	*!		
b.	dapa		*!	
c.	$\mathcal{W}$ dap			*

Both `\OTSolids` and `\OTdashes` are drawn with the PSTricks `\psline` command and can take any style option suitable for `\psline`. `\OTjagged` is drawn in a special way with two instances of the `\pszigzag` command, one of which is white and extends beyond the outside border of the tableau, to ensure that it always breaks. Because of this, not every PSTricks option suitable for `\pszigzag` can be used with `\OTjagged`. At minimum, the following options will likely *not* work as expected: `doublesepcolor`, `doublesepwidth`, `coilwidth`, `coilheight`, `coilarm`, `coilaspect`, `coilinc`, and `linearc`. Adjusting these parameters requires deeper manipulation of `OTtblx`'s code. It can be done, but not easily, and currently, not with any user-friendly options provided by `OTtblx`.

6 Comparative Tableaux

`OTtblx` provides a few extra commands that allow the creation of comparative tableaux like the following:

dap	MAX	DEP	*CODA
$\mathcal{W}$ dap			1
a. $\sim$ da	₁ W		₀ L
b. $\sim$ dapa		₁ W	₀ L

Comparative tableaux are constructed using the `OTcomparative` environment, which has exactly the same syntax and arguments as the `OTtableau` environment:

```

■ \begin{OTcomparative} [options] {num_constraints}
  :
\end{OTcomparative}

```

Within this environment, and only within this environment, two new commands are enabled. The first command is `\OTwinrow`, which is used for the row immediately following `\OTtoprow` or `\OTtoprow*`. This command has the same syntax and behavior as `\OTcandrow`, except that the default candidate marker is $\mathcal{W}$, since `\OTwinrow` is usually only used for the winning candidate:

```

■ \OTwinrow[marker]{candidate}{violation_list}

```

The third and final command needed for comparative tableau is `\OTcompviol`, which can be used to create a two-part comparative violation mark, with the first optional argument in the lower

left corner of the cell (usually used for actual number of violations incurred) and the second obligatory argument in the center of the cell (usually filled with W or L to indicate which candidate is comparatively better):

■ `\OTcompviol[lower_left_mark]{centered_mark}`

The `\OTcandrow` command is slightly modified within the `OTcomparative` environment so that the default candidate marker is `~` rather than empty. This can be overridden for individual candidates by specifying the marker directly in the optional argument in square brackets.

The comparative tableau at the beginning of this section was created with the following code:

```
\begin{OTcomparative}{3}
  \OTsolids{2}\OTdashes{1}
  \OTtoprow [dap] {\textsc{Max},\textsc{Dep},*\textsc{Coda}}
  \OTwinrow {dap} { , ,1}
  \OTcandrow {da} {\OTcompviol[1]{W}, ,\OTcompviol[0]{L}}
  \OTcandrow {dapa} { ,\OTcompviol[1]{W},\OTcompviol[0]{L}}
\end{OTcomparative}
```

7 Miscellaneous Macros

The `OTtblx` package contains a few macros for quick typesetting of some terms, constraint names, and symbols commonly used in OT:

<code>\Align</code>	ALIGN	<code>\Gen</code>	GEN	<code>\OThand</code>	
<code>\Anchor</code>	ANCHOR	<code>\Ident</code>	IDENT	<code>\OThand*</code>	
<code>\Con</code>	CON	<code>\Lin</code>	LIN	<code>\OTface</code>	
<code>\Contig</code>	CONTIG	<code>\Max</code>	MAX	<code>\OTface*</code>	
<code>\Dep</code>	DEP	<code>\Onset</code>	ONSET	<code>\OTtharm</code>	
<code>\Eval</code>	EVAL	<code>\Parse</code>	PARSE	<code>\OTtharm*</code>	
<code>\Fill</code>	FILL	<code>\Unif</code>	UNIF	<code>\OTdom</code>	
<code>\Faith</code>	FAITH			<code>\OTdom*</code>	

All `OTtblx` commands that begin with `\OT` commands are defined strongly, so that any previous definition is lost. This is usually because the `OTtblx` definition is necessary for proper functioning of the package. The non-`\OT` commands are only “provided”, using the command `providecommand`, which means that they will *not* supersede previously defined macros with the same name.

8 The Future

There is no future for version 0.4.x of `OTtblx`. As of version 0.5, it has been completely rewritten with PGF/TikZ, and that’s where all future active development will now occur. Please check it out, and if have any questions, suggestions, or feature requests, please do let me know. Thanks for using `OTtblx`!

*Nathan Sanders
Toronto, Canada*

References

- Kager, René. 1999. *Optimality Theory*. Cambridge University Press.
- McCarthy, John J. 2002. *A thematic guide to Optimality Theory*. Cambridge University Press.
- Prince, Alan, and Paul Smolensky. 1993/2004. *Optimality Theory: Constraint interaction in generative grammar*. Malden, MA: Blackwell Publishers.