Contents

1 Purpose ........................................ 2

2 The Package .................................... 3
  2.1 Getting Started ............................... 3
  2.2 Package Options .............................. 3
  2.3 Known Package Conflicts ................... 5

3 Basic Tableaux .................................. 5
  3.1 The \O Ttableau 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.x

• Added support for \Xe\LaTeX
• Added \ipa and \noipa options
This manual describes OTtablx version 0.4, a beta version designed primarily for testing purposes. Feel free to use the package for your own serious work, but do not expect current functionality to extend into later versions, since commands, options, and default settings may be renamed, revised, or eliminated entirely. The most recent version of OTtablx can be found at:

http://sanders.phonologist.org/OTtablx/

If you find a bug, want to request a new feature, or even have advice on improving the package code, please let me know. I make no guarantees that OTtablx will ever be designed to do exactly what you want or that it will even work for you at all, but I will do my best to make the package functional and useful for as many users as possible, given my own constraints on time and ability.

1 Purpose

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

<table>
<thead>
<tr>
<th></th>
<th>MAX</th>
<th>DEP</th>
<th>*CODA</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.</td>
<td>da</td>
<td>*!</td>
<td></td>
</tr>
<tr>
<td>b.</td>
<td>dapa</td>
<td>!</td>
<td>!</td>
</tr>
<tr>
<td>c.</td>
<td>*</td>
<td>!</td>
<td></td>
</tr>
</tbody>
</table>

OTtablx 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 OTtablx 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 OTtablx 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 LATEX \texttt{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 OTtablx intersect with their middles merged, rather than with the (uglier) asymmetrically layered look of LATEX’s regular \texttt{tabular} environment:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ugly</td>
<td>layered</td>
</tr>
<tr>
<td>double</td>
<td>lines</td>
</tr>
</tbody>
</table>

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 XƎLATEX.
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 OTtablx, the double lines behave consistently, either both breaking the outside border or both leaving the border intact (user's choice).

Finally, there is currently no cell shading in OTtablx, primarily because it's too difficult to program an intuitive interface for it. And since cell shading is an unnecessary decoration that often does not photocopy very well, it has diminished in popularity, so the lack of cell shading in OTtablx isn’t really a huge loss. There is a remote chance that cell shading capabilities may someday be added to OTtablx, but don’t count on it.

2 The Package

2.1 Getting Started

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

\usepackage [options] {OTtablx}

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

Use with \LaTeX When used with latex+dvips, OTtablx 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 OTtablx.

Use with Xe\LaTeX When used with xelatex, OTtablx also loads the fontspec package, using the Doulos SIL font by default. This font is free and available from SIL. This provides two commands: \textipa{} 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

OTtablx 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 2004)
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 LATEX, Doulos SIL for XE\LaTeX)
- 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 \OTtablx as follows:

\usepackage [square,closed,alpha,light,ipa] {OTtablx}
\begin{tabular}{|c|c|c|}
\hline
\texttt{dap} & \texttt{MAX} & \texttt{*CODA} \\
\hline
a. da & & *! \\
b. \texttt{dap} & & * \\
\hline
\end{tabular}

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 \[\texttt{[square,round]}\] is functionally equivalent to just using \[\texttt{[round]}\]. 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] {OTtablx}
\begin{tabular}{|c|c|c|}
\hline
\texttt{dap} & \texttt{MAX} & \texttt{*CODA} \\
\hline
1. da & & *! \\
2. \texttt{dap} & & * \\
\hline
\end{tabular}

\usepackage [open,nolabel,heavy] {OTtablx}
\begin{tabular}{|c|c|c|}
\hline
\texttt{dap} & \texttt{MAX} & \texttt{*CODA} \\
\hline
da & & *! \\
\texttt{dap} & & * \\
\hline
\end{tabular}

\usepackage [noborder,roman] {OTtablx}
\begin{tabular}{|c|c|c|}
\hline
\texttt{dap} & \texttt{MAX} & \texttt{*CODA} \\
\hline
i. da & & *! \\
ii. \texttt{dap} & & * \\
\hline
\end{tabular}

\usepackage [noipa] {OTtablx}
\begin{tabular}{|c|c|c|}
\hline
\texttt{DAP} & \texttt{MAX} & \texttt{*CODA} \\
\hline
a. DA & & *! \\
b. \texttt{dap} & & * \\
\hline
\end{tabular}
2.3 Known Package Conflicts

\texttt{OTtablx} makes heavy use of PSTricks, which precludes direct use of \texttt{pdf\LaTeX} to generate PDFs. Unfortunately, I don’t really know how to rewrite the package to avoid this inconvenience, but there are some solutions to the PS/PDF conflict that will hopefully allow you to use \texttt{OTtablx} with only minor tweaking to your usual \LaTeX routine.

3 Basic Tableaux

3.1 The \texttt{OTtableau} Environment

The core feature of the \texttt{OTtablx} package is the relatively simple and customizable \texttt{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{verbatim}
\begin{OTtableau} [options] {num_constraints}
    \end{OTtableau}
\end{verbatim}

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

3.2 Tableau-Specific Options

The optional argument in the \texttt{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: \texttt{[square]}, \texttt{[round]}, \texttt{[nolabel]}, etc.

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

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

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

3.3 Vertical Separators

Inside the \texttt{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:

\begin{verbatim}
\OTsolids \{PSTricks_options\} \{constraint_number_list\}
\OTdashes \{PSTricks_options\} \{constraint_number_list\}
\OTjagged \{PSTricks_options\} \{constraint_number_list\}
\end{verbatim}

<table>
<thead>
<tr>
<th>\texttt{dap}</th>
<th>\texttt{MAX}</th>
<th>\texttt{*CODA}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. da</td>
<td></td>
<td>*!</td>
</tr>
<tr>
<td>2. *dap</td>
<td></td>
<td>*</td>
</tr>
</tbody>
</table>

You should get no error if you happen to use a tableau-specific option that matches the global setting for the current document; \texttt{OTtablx} will just silently format the tableau with the logical result.
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 \Tdashes{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 \Tsolids command to get two solid separators:

\begin{OTtableau}{3}
\Tsolids{1,2}
dap & MAX & DEP & *CODA \\
ap & *!
\bap & *!
\cap & *
\end{OTtableau}

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

\begin{OTtableau}{3}
\Tdashes{1}
\Tsolids{2}
dap & MAX & DEP & *CODA \\
ap & *!
\bap & *!
\cap & *
\end{OTtableau}

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 \\
ap & *!
\bap & *!
\cap & *
\end{OTtableau}

The third separator command, \Tjagged, 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}
\Tsolids{1,3}
\Tjagged{2}
dap & MAX & *CODA & IDENT & *VOI\text{OBS} \\
ap & *!
\bap & * & *!
\cap & *
\end{OTtableau}

Currently, issuing multiple instances of the same vertical separator command in the same tableau, such as \Tsolids{1} followed by \Tsolids{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 \texttt{\textbackslash OTtoprow} or its starred version \texttt{\textbackslash OTtoprow*}, both of which have one optional argument for the input and one obligatory argument for the list of constraints:

\begin{itemize}
  \item \texttt{\textbackslash OTtoprow [input] \{constraint\_list\}}
  \item \texttt{\textbackslash OTtoprow* [input] \{constraint\_list\}}
\end{itemize}

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 \texttt{TIPA} package for \LaTeX, and the Doulos SIL font for \TeX), 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 \texttt{\textipa{DAP}} to appear as your input in \LaTeX, you can simply use the \texttt{TIPA} shortcut \texttt{\textipa{DAP}}, rather than having to write out \texttt{\textipa{DAP}}. To temporarily get non-phonetic text in the input when using \LaTeX, use either the \texttt{\normalfont} command (the general recommendation, especially if your input is large or complex) or \texttt{TIPA}'s \texttt{\*} 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 \texttt{[noipa]} option. If you use \texttt{[noipa]} globally, but want a specific tableaux to have phonetic font embedding, you can use the local \texttt{[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{verbatim}
\begin{OTtableau}{3}
  \OTsolids{2}\OTdashes{1}
  \OTtoprow [dap] \{\texttt{textsc{Max}},
                \texttt{textsc{Dep}},
                *\texttt{textsc{Coda}}\}
\end{OTtableau}
\end{verbatim}

\begin{center}
\begin{tabular}{l|l|l|l}
  & \texttt{\textsc{Max}} & \texttt{\textsc{Dep}} & \texttt{\*\textsc{Coda}} \\
  \hline
  a. & \texttt{da} & \*! & \\
  b. & \texttt{dapa} & \*! & \\
  c. & \texttt{\&\,dap} & \* & \\
\end{tabular}
\end{center}

\texttt{\textbackslash OTtoprow} and \texttt{\textbackslash OTtoprow*} have nearly identical effects, except for the placement of the input within the top left cell. \texttt{\textbackslash OTtoprow} aligns the input with the candidate list, while \texttt{\textbackslash 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{verbatim}
\begin{OTtableau}{3}
  \OTsolids{2}\OTdashes{1}
  \OTtoprow* [d+a+p] \{\texttt{textsc{Max}},
                \texttt{textsc{Dep}},
                *\texttt{textsc{Coda}}\}
\end{OTtableau}
\end{verbatim}

\begin{center}
\begin{tabular}{l|l|l|l}
  & \texttt{\textsc{Max}} & \texttt{\textsc{Dep}} & \texttt{\*\textsc{Coda}} \\
  \hline
  a. & \texttt{da} & \*! & \\
  b. & \texttt{dapa} & \*! & \\
  c. & \texttt{\&\,dap} & \* & \\
\end{tabular}
\end{center}
## 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 \OTtablx 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 \OTtablx.)

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 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}

\begin{tabular}{|c|c|c|c|}
  \hline
  dap & MAX & DEP & *CODA \\
  \hline
  a. da & *! & & \\
  b. dapa & & *! & \\
  c. ☟ dap & & & *
  \hline
\end{tabular}

\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}

\begin{tabular}{|c|c|c|c|}
  \hline
  dap & MAX & DEP & *CODA \\
  \hline
  a. da & *! & & \\
  b. dapa & & *! & \\
  ☟ c. dap & & & *
  \hline
\end{tabular}
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}
\begin{tabular}{@{}c@{}c@{}c@{}c@{}}
H & H & &
dapa
\end{tabular}
\end{OTtableau}

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}
\begin{tabular}{@{}c@{}c@{}c@{}c@{}}
H & H & &
dapa
\end{tabular}
\end{OTtableau}

where \dapaHH and \dapaHL were defined before the tableau with the following:

\def\dapaHH
{\begin{tabular}{|@{}c@{}c@{}c@{}c|}
 & &
dapa
\end{tabular}}

\def\dapaHL
{\begin{tabular}{|@{}c@{}c@{}c@{}c|}
 & &
dapa
\end{tabular}}
5 Advanced Formatting with PSTricks

Various customizations to the lines drawn in OTtableaux 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 \ 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 \{\OThand\}
\{dap\} \{ , ,*
\end{OTtableau}

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 \{\OThand\}
\{dap\} \{ , ,*
\end{OTtableau}

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 [\OThand]
\{dap\} \{ , ,*\}
\end{OTtableau}

Both \OTsolids and \OTdashes are drawn with the PSTricks \texttt{psline} command and can take any style option suitable for \texttt{psline}. \OTjagged is drawn in a special way with two instances of the \texttt{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 \texttt{pszigzag} can be used with \OTjagged. At minimum, the following options will likely not work as expected: \texttt{doublesepcolor}, \texttt{doublesepwidth}, \texttt{coilwidth}, \texttt{coilheight}, \texttt{coilmargin}, \texttt{coilscale}, \texttt{coilline}, and \texttt{linearc}. Adjusting these parameters requires deeper manipulation of OTtablx's code. It can be done, but not easily, and currently, not with any user-friendly options provided by OTtablx.

6 Comparative Tableaux

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

\begin{tabular}{|c|c|c|}
\hline
  dap & MAX & DEP & *CODA \\
\hline
  \texttt{☞ dap} & 1 & W & 0 L \\
\hline
\end{tabular}

Comparative tableaux are constructed using the \texttt{OTcomparative} environment, which has exactly the same syntax and arguments as the \texttt{OTtableau} environment:

\begin{itemize}
  \item \texttt{\begin{OTcomparative} [options] \{num_constraints\}}
  \item \texttt{\end{OTcomparative}}
\end{itemize}

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

\begin{itemize}
  \item \texttt{\OTwinrow[marker]{candidate}{violation_list}}
\end{itemize}

The third and final command needed for comparative tableau is \texttt{\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 \(\sim\) 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}\}
\OTtwinrow {dap} \{ , ,1\}
\OTcandrow {da} \{,\OTcompviol[1]{W},,\OTcompviol[0]{L}\}
\OTcandrow {dapa} \{ ,\OTcompviol[1]{W},\OTcompviol[0]{L}\}
\end{OTcomparative}

\section{Miscellaneous Macros}

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

\begin{align*}
\texttt{\Align} & \quad \textit{ALIGN} & \texttt{\Gen} & \quad \textit{GEN} & \texttt{\OThand} & \quad \textit{\OThand}\texttt{*} \\
\texttt{\Anchor} & \quad \textit{ANCHOR} & \texttt{\Ident} & \quad \textit{IDENT} & \texttt{\OThand*} & \quad \textit{\OThand}\texttt{*} \\
\texttt{\Con} & \quad \textit{CON} & \texttt{\Lin} & \quad \textit{LIN} & \texttt{\OTharm} & \quad \textit{\OTharm}\texttt{*} \\
\texttt{\Contig} & \quad \textit{CONTIG} & \texttt{\Max} & \quad \textit{MAX} & \texttt{\OTharm*} & \quad \textit{\OTharm}\texttt{*} \\
\texttt{\Dep} & \quad \textit{DEP} & \texttt{\Onset} & \quad \textit{ONSET} & \texttt{\OTharm} & \quad \textit{\OTharm}\texttt{*} \\
\texttt{\Eval} & \quad \textit{EVAL} & \texttt{\ Parse} & \quad \textit{PARSE} & \texttt{\OTharm}\texttt{*} & \quad \textit{\OTharm}\texttt{*} \\
\texttt{\Fill} & \quad \textit{FILL} & \texttt{\Unif} & \quad \textit{UNIF} & \texttt{\OTharm} & \quad \textit{\OTharm}\texttt{*} \\
\texttt{\Faith} & \quad \textit{FAITH} & \texttt{\OTharm}\texttt{*} & \quad \textit{\OTharm}\texttt{*} \\
\end{align*}

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

\section{The Future}

These are some of the modifications and plans that I have in mind for future versions of OTtablx:

- add more formatting options, including control over column widths
- add an optional column for inputs to allow for multiple inputs in the same tableau (as in Pavel Iosad’s OTtablx-extras)
- allow multiple instances of the same vertical separator command within a single tableau to function as expected
• maybe add commands for shading cells, though I have yet to hear anyone argue for their inclusion...

If you have strong opinions on whether any of these should be added, or if you have ideas of your own that aren’t listed above, let me know what you think. And if you can figure out how to modify the \texttt{\textbackslash opttablex} package to incorporate any of these modifications, definitely get in touch!

Also, feel free to ask questions and offer suggestions about the documentation. If you find something that is worded in a particularly unclear way, tell me what it is, so that I can rewrite it. Meanwhile, enjoy the package, and thanks for testing it!

\textit{Nathan Sanders}
\textit{Swarthmore PA}

\section*{References}

