LaTeX Syntax Tree Guide

This is a guide to making beautiful syntax trees using the software LaTeX (pronounced "Lah-tek" [lɑ.tɛk]). LaTeX is a typesetting program that is often used in linguistics and other formal disciplines like mathematics. LaTeX has several packages within it that are useful for doing formal linguistics. LaTeX will give you PDFs as its output.

1 Getting LaTeX

There are essentially two ways that I know of to use LaTeX.

- The first is to download a LaTeX compiler onto your computer. Here is a link to some free compilers. The main benefits to using LaTeX locally on your PC are that it typically compiles faster and works without an internet connection. There are also no file size limits.

- The second is to use Overleaf. Overleaf is a browser-based version of LaTeX. The main benefits to using Overleaf are that you can access your projects from any device with internet access and that Overleaf includes packages automatically, meaning you do not have to download or update them. You can also easily share your projects with collaborators.

2 Using LaTeX

A LaTeX document consists of several components. Every document must have code that begins and ends the document within it somewhere. The text in between is the text that will show up to your reader.

```
\begin{document}

blah blah blah

\end{document}
```

Before the commands that begins the document however, is the *preamble*. The preamble defines what type of document you are using. For example, you can set the document to "article" by using the command

```
\documentclass[12pt]{article}
```

in the preamble. It is also possible to include *packages* in the preamble. *Packages* are additions to LaTeX that come with various functionalities. For example, *tIPA* is a package that can be used to make phonetic symbols in LaTeX documents.

```
\includepackage{tipa}
```

Using the *tIPA* package, the following text

```
\textipa{1234567890}
```

will show up like ɨʌɜɥɐɒɤɵɔʁ.

3 Making Syntax Trees

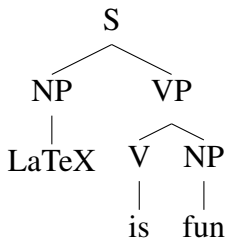
In order to make syntax trees, you will need the following packages in your preamble.[1]

```
\usepackage{tikz}
\usepackage{tikz-qtree}
\usepackage{qtree}
```

The following is an example of the syntax of *qtree*.

```
\Tree [.S [.NP LaTeX ] [.VP [.V is ] [.NP fun ] ] ]
```

This code creates the following tree.

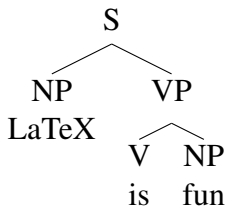


You must mark each node in the tree with a period. This period does not show up in the actual tree but merely tells the tree-drawer that what follows is a node. It is also important to place a space between each of the words *LaTeX*, *is*, and *fun* in a tree like this. If you do not, the document/tree will not compile.

4 Types of Trees

If you, like I usually do, do not wish to have lines between the Head and the word, you can use a double backslash between the node label and the word.

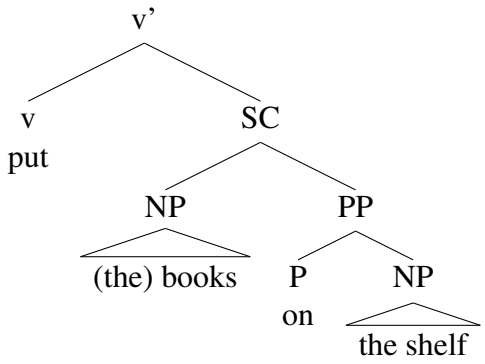
```
\Tree [.S [.NP\\LaTeX ] [.VP [.V\\is ] [.NP\\fun ] ] ]
```



If you want to draw a triangle in a tree, you can use the "qroof" command within the tree. The convention of doing this is to not be explicit about the "triangled" material.

```
\Tree [.v' [.v\\put ][.SC \qroof{(the) books}.NP
[.PP [.P\\on ] \qroof{the shelf}.NP ] ] ]
```

[1] There are other packages that can draw syntax trees, such as "xy-ling" and "forest."



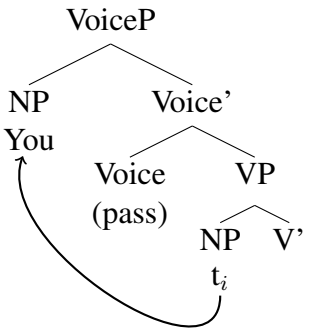
Drawing an arrow between two items on a tree is a little more complicated. Add the following code to your preamble, right after your "tikz" package.

```
\usepackage{tikz}
\tikzstyle{every picture}+=[remember picture]
\def\tnode#1#2{{\tikz\node[shape=rectangle, inner sep=2pt](#1){#2};}}
\def\xnode#1{\tnode{#1}{#1}}
```

This code will allow you to delineate two points in your tree for things to move between. These are named "xnode" and "tnode" respectively here, but you could name them anything by changing their names in the preamble. The arrow is drawn via a command called "tikzpicture." This means that the following code will draw a tree with an arrow.

```
\begin{exe}
\ex \Tree[.VoiceP [.NP\\\xnode{You} ][.Voice'
[.Voice\\(pass) ]
[.VP [.NP\\\tnode{trace}{t_i} ] [.V' ] ] ] ]
\begin{tikzpicture}[overlay]
\path[->,thick](trace) edge [out=-90, in=-110](You);
\end{tikzpicture}
\end{exe}
```

(1)



5 Examples and Glossing

If you would like to put your examples in LaTeX, you can do so with the "exe" environment brought by the "gb4e" package.[2]

[2]For some reason, "gb4e" needs to come last in the list of packages. Do not ask me why.

```
\begin{exe}
\ex Oprah_1 said that we should stop listening to her_1.
\end{exe}
```

(2) Oprah$_1$ said that we should stop listening to her$_1$.

The main benefit to doing this is that LaTeX will auto-number your examples for you, meaning you don't have to re-number all the examples if you want to add something later. If you want to group multiple examples together, you can use the "xlist" command within the "exe" environment. The "xlist" command has to be preceded by an "ex" or it won't work.

```
\begin{exe}
\ex \begin{xlist}
\ex Oprah_1 said that we should stop listening to her_1.
\ex {*} She_1 said that we should stop listening to Oprah_1.
\end{xlist}
\end{exe}
```

(3) a. Oprah$_1$ said that we should stop listening to her$_1$.
b. * She$_1$ said that we should stop listening to Oprah$_1$.

You may be interested in making professional looking interlinear glosses for your final paper. "gb4e" will add the "gll" command possibility to your examples. Doing so allows you to make 3 tiered examples like the following, where everything lines up like you would want. The lines are separated by two backslashes, which is the LaTeX command for "new line."

```
\begin{exe}
\ex \gll Xos-e ta tropaia kato apo to krevati! \\
hole.in-Imp the.Acc.Pl trophies.Acc.Pl under from the.Acc bed.Acc \\
'Get the trophies under the bed!'
\end{exe}
```

(4) Xos-e ta tropaia kato apo to krevati!
hole.in-Imp the.Acc.Pl trophies.Acc.Pl under from the.Acc bed.Acc
'Get the trophies under the bed!'

LaTeX also permits you to label examples. The benefit of doing so if can then reference the label in the text, rather than the actual example number. This means if you change something, your reference will also change. The label will not show up the reader, it is only for your benefit. The labelling command is "label," while the reference command is "ref."[3]

```
\begin{exe}
\ex \label{oprah} Oprah_1 said that we should stop listening to her_1.
\end{exe}

As we can see in \eqref{oprah}, Oprah has low self-esteem.
```

(5) Oprah$_1$ said that we should stop listening to her$_1$.

As we can see in (5), Oprah has low self-esteem.

[3] "eqref," as shown in the example here, automatically puts parentheses around your reference. "ref" will reference it without parentheses