LaTeX

Ki James

September 28, 2022

1 What is LaTeX

LaTeX, pronounced /latɛx/ (or /latɛk/ if you don't like velar fricatives) is a word processor, much like Microsoft Word or Google Docs. Long ago, in the dark ages of digital writing, a great battle raged between the warriors of What You See Is What You Get (abbreviated to WYSIWYG) and the more traditional markup type setups. While the markup crowd tried their best, they ultimately lost the war. Nowadays, very few people dare to tread far outside the most user friendly of options.

But not you, brave soldier! You're here to learn the dark, forgotten arts. And for good reason - LaTeX is particularly well suited for Linguistics. It's capable of things such as:

- Glosses
- Feature Matrices
- Tableaux
- Syntax Trees

All of these things and more will be shown here in this document; however, we should first discuss a few points of philosophy and history.

1.1 The Hacker Ethic

"Hacker" wasn't always a pejorative. In the early days of computing in the 50's and 60's, a group of students at MiT built a community and philosophy around the limited university hardware they were working with. Primarily, this world view related to things like freedom and transparency, and it had a uniquely libertarian streak to it. This makes sense, of course. Academia is (ostensibly) built on transparency, sharing research, and collaborating towards increasingly optimized results.

This movement didn't grow so much as it festered. It morphed into the open source community, headed by the likes of Richard Stallman. This movement continues into today. If you've ever run into Linux communities, you can know that they're often difficult and particular, but produce some very powerful software.

Importantly for us, hackers, to whatever extent that term makes sense to use today, grandfathered a number of concepts that still hold sway over not only the open source community, but programmers in general. The most critical of these ideas is the "solved problem." To explain this, here's a excerpt from Neil Stephenson's 1999 article "command.txt."

> Nothing is more disagreeable to the hacker than duplication of effort. The first and most important mental habit that people develop when they learn how to write computer programs is to generalize, generalize, generalize. To make their code as modular and flexible as possible, breaking large problems down into small subroutines that can be used over and over again in different contexts.

With almost no exceptions, somebody out there has tried to solve the same problem as you. Your job is to find where they posted the solution. If you're having a hard time drawing an arrow with JTree, somebody has some code that you can shamelessly steal and reapply to your own work. Additionally, the more problems you solve, the more you can reference your own documents.

All that to say, I can't and won't be providing examples of how to do everything you could ever want to do, and that's ok. Somebody else already has.

1.2 Downloading LaTeX

Let's return for a second to Linux. You probably know that Linux is an operating system like Windows. If I want my computer to run on Windows, I go to a Windows store and buy something I can download it off. Linux is not so simple. If you google ”Linux download,” you’ll find a helpful link on linux.org to “25 popular Linux distributions.”

“Distribution?” You might ask, “Is that like a version?” Unfortunately, no. Linux is just the kernel (a technical word that I won’t get into because it’s not relevant), and can’t run anything by itself. Different people take this kernel and add their own bits on top of it. This kernel with the fluff gets a name, names you might have heard of: Ubuntu, Mint, even Android.

It’s the same with LaTeX. By itself, it doesn’t constitute “software” that you can just download and run. You need a TeX distribution that supports LaTeX. This varies from OS to OS, and they probably have different pros and cons. I use MiKTeX, and it works just fine. If you’re curious about which one you need, this is luckily a solved problem.

2 Glosses in LaTeX

Glosses are pretty simple. Here’s some examples of them.

(1)	be͡ɪt	al.ustaːð
	house	the.professor

‘(the) house of the professor’
‘the professor’s house’

(2)	zahrat	faːtˤɪma
	flower	fatima

‘(the) flower of fatima’
‘fatima’s flower’

(3)	χafiːf	adːam
	light	the.blood

‘light of blood’
(idomatic) ‘light hearted’

(4)	ɪbn	ʕam	ʔabiː
	son	uncle	father.my

‘the son of the uncle of my dad’
‘my dad’s cousin’

The plugin I use is called gb4e. It comes with some pretty serious drawbacks that I haven’t figured out how to solve, namely, it automatically numbers each item. In a syntax context, this makes numbering trees somewhat problematic.

3 Feature Matrices

LaTeX is predominantly used by people in the mathematical community, and there's a lot of support for things mathematicians use often. One such example is that of a matrix. For Phonology, our matrices are always one dimensional, so this ends up being pretty simple to do.

$$\begin{bmatrix} \text{+cons} \\ \text{-voice} \\ \text{-spread} \\ \text{-constr} \\ \text{-cont} \\ \text{-nas} \\ \text{-lat} \\ \text{+cor} \\ \text{+ant} \\ \text{-dist} \\ \text{-round} \\ \text{-high} \\ \text{-low} \\ \text{-tense} \end{bmatrix}$$

4 Tableaux

Natively, there's a very powerful way of creating beautiful tables in LaTeX. The only change I've made is an additional outline to make them look a little snappier, but you have options

/χori/	[+dor, -son][+syl] ⇒ [-cont]	*[+dor, -cont]	ID[cont]
☞ [qori]		*	*
[χori]	*		

This also works for derivation tables!

UR	elɨn	tʃølɨn	kulɨn	diʃlarɨn	izlar	kadɨnlarɨn
DICON				diʃlerɨn	izler	
MONOCON	elin	tʃølyn	kulun	diʃlerin		
SR	elin	tʃølyn	kulun	diʃlerin	izler	kadɨnlarɨn

5 Syntax Trees

Syntax Trees are far and away the most complicated of the covered topics. I'm focusing on only the most recurring types of trees I find myself making.

5.1 Qtree

When I first started my forays into making syntax trees, googling ”latex syntax trees” led me to a package called ”Qtree.” I'm here to tell you not to make that mistake. Qtree is a terrible package that is both less usable and less powerful that what I'll be showing off in this document. There are people who swear by it; however, I would caution against touching it.

If you want, you can try both for yourself. I will not be devoting any time to exploring it here, as I've wasted too much time in that environment to ever want to go back.

5.2 jTree

jTree is a package written by a mysterious man named John Frampton. It is a tremendously powerful tool that will more than service all of your wildest dreams. That being said, it's poorly documented, and there isn't a robust community of jTree-ers running around sharing their code.

All that I had written about the hacker ethic flies out the window here. I've resolved to remedy this situation as much as possible, and make available the hard lessons I've learned, but assuredly you'll find yourself traversing some dark path, trying desperately to do something that seems simple that you've seen done before (like convergent nodes) and will come up empty handed.

This isn't to scare you, but to steel your soul for what lies ahead.

5.2.1 Frampton's User Guide

Helpfully, our friend John published a 67 page instructional booklet telling us all the lovely things his application can do. This can easily be found online by searching ”latex jtree documentation,” and downloaded off of a seemingly random university's webpage.[1]

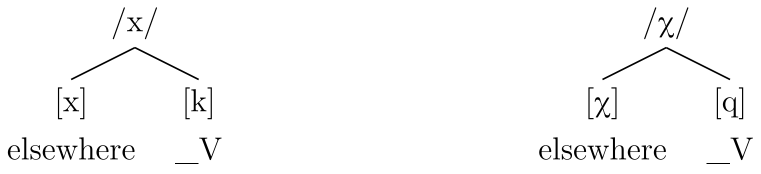
[1] It's best not to ask questions

Unhelpfully, the code he presents does not work. This is for a number of reasons, most frustratingly due to the fact that he seems to have updated the package and not told anybody. It's possible there's another user guide, more updated than the 2006 one I've been working from. I am not aware that such a document exists.

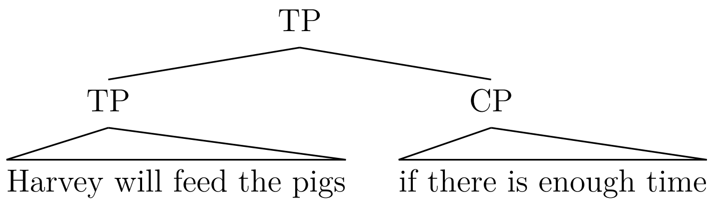
More important than any individual piece of code the manual shows is the type of logic it demonstrates. As you continue to make trees, just consult the cursed text as needed. As you work your way through the problems, you'll begin to develop your own style and conventions.

5.2.2 Basic Trees

Let's start by keeping things very simple - a tree with just two branches. These are great for demonstrating allophonic relationships in Phonology.

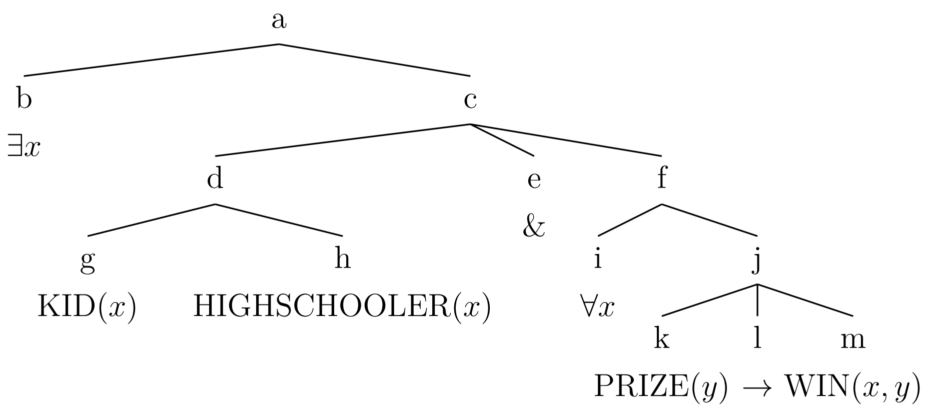


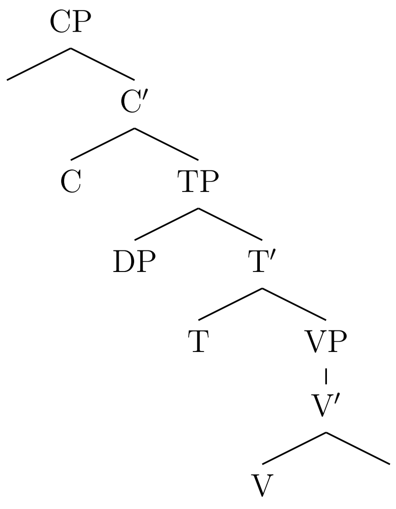
We can also use triangles to simplify what would otherwise be a very complicated tree.



5.2.3 Slightly More Complicated Trees

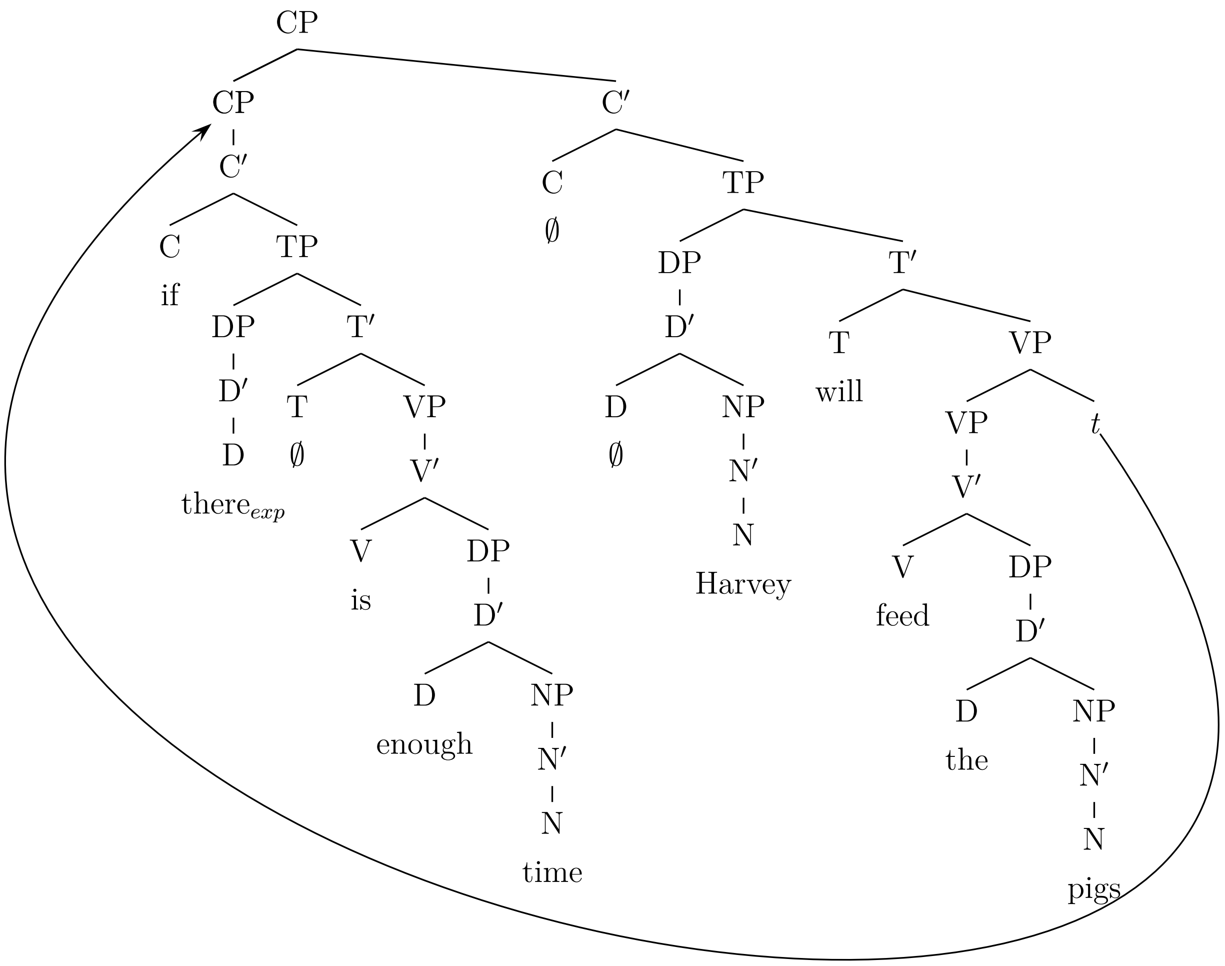
When we get into Syntax or Semantics, we're unlikely to encounter something as simple as we see above. Instead, we're more likely to see something like this, which is really just the same thing with more nodes.



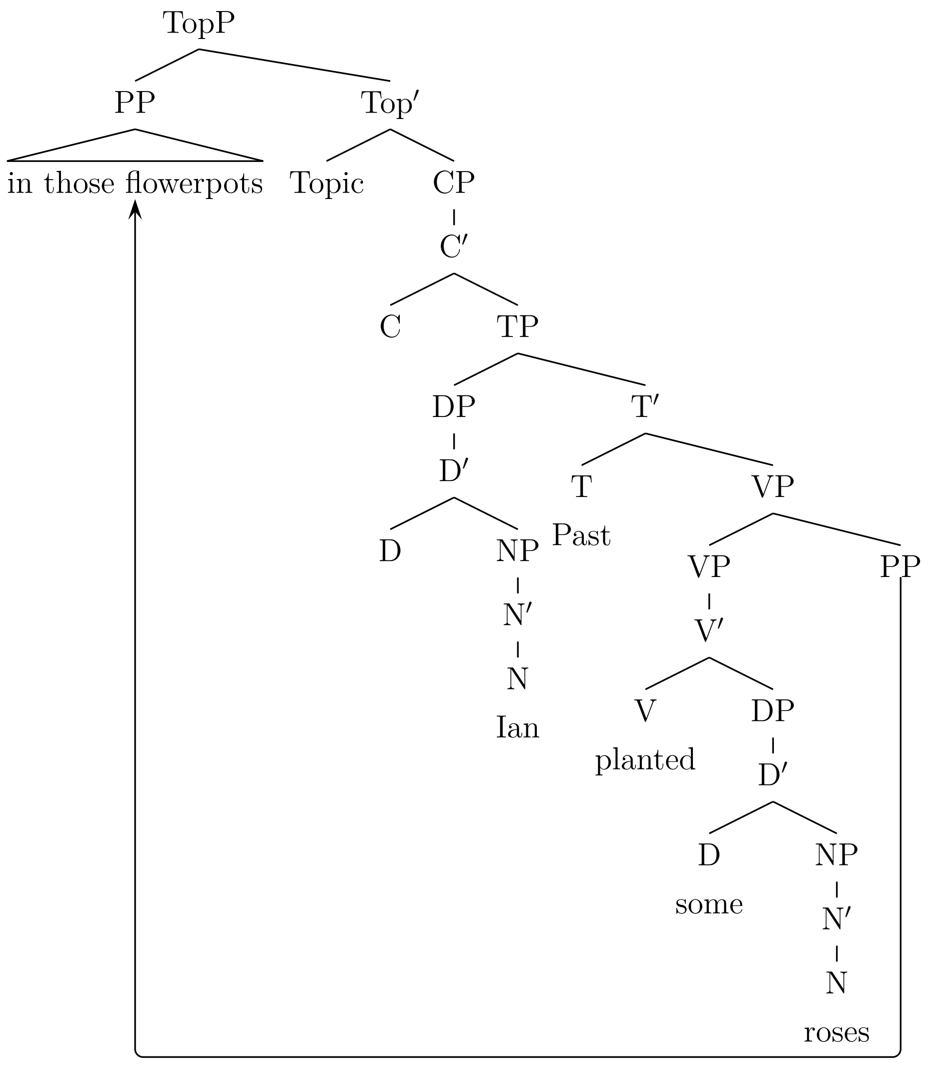


5.2.4 Arrows

Arrows are when things start to get more complicated. One great way of drawing arrows is to use "nccurve." It's a very clunky, cantankerous function, so treat it with caution.



Alternatively, we could use "ncbar" and draw a straight line. This are less pretty, but can be useful if you don't want to bother with getting the angles right on a curve.



One additional point about arrows is that the function works on a sentence just as well as it does a tree.

(5) Who$_\alpha$ did Sally think that Mary said that John stabbed _____$_\beta$