Arrows in LaTeX using `tree-dvips`

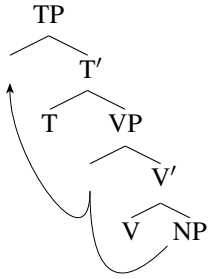
LGCS 105: Syntax

February 6, 2016

1 Introduction

This handout lays out how to draw arrows in trees.

(1)



Drawing arrows requires a new package (`tree-dvips`) which draws arrows on trees built with `qtree`. This introduction assumes that you are using the online LaTeX Previewer that we have been using in this class. To use `tree-dvips`, be sure to add it from the available packages list, the same way you add the `qtree` package.[1]

There are two main components of drawing an arrow in a tree:

1. **Identify Nodes**: you will choose the nodes in your tree where your arrow will begin and end. This code is entered in the code for the tree itself.
2. **Draw the Arrow**: Below (and separate from) the code for the tree, you enter a line of code that draws the arrow.

As you are first learning to draw trees with arrows, I recommend that you first draw a tree using qtree that you are comfortable with, then adding in the node definitions, then add the line of code to draw the tree.

Also worth noting: anything you might want to draw with arrows, you can also draw using traces and/or strike-out (two additional notations for showing movement). So if you find the arrow-drawing overwhelming, you don't have to do it! You can represent movement in some other way.

2 Nodes

Before you draw arrows, you need to define the nodes they connect to. The form of a node is `\node{nodename}{text}`.The `nodename` is what you name your node so that you can refer to it later when telling the arrows where to go. The `text` is simply what you want to show up as the label of that node in the actual tree (i.e. what

[1] If you are using a LaTeX installation on your own machine to write your papers (instead of the online previewer), `tree-dvips` requires that you send the file to a postscript printer, using dvips. You can check out the `tree-dvips` documentation on CTAN for more details.

would be in that part of the tree whether you were drawing arrows or not). You define nodes within the tree itself. So in the examples, below, a N terminal node has been identified as a node, and is named "subj" (so the instruction to draw the line would tell it to go to "subj").

(2) a. ✓ `\node{subj}{N}`
b. ✓ `\node{subj}{N\\John}`

A note about writing the code: `tree-dvips` nodes can't wrap around square brackets (as you can see in (b) below) – so when the node in question is a non-terminal node that is the label of a constituent, you have to identify the node inside the brackets that identify the constituent, as shown in (a) below.

(3) a. ✓ `[.\node{verbphrase}{VP} V NP ]`
b. ✗ `\node{verbphrase}{[.VP V NP ]}`

For any given arrow, you'll need two nodes: a source node where the arrow *comes from* and a target node where the arrow *goes to*. The nodes are defined within the code that you use to draw the tree, which in `qtree` begins with `\Tree`.

3 Arrows

Below where you code your tree and the necessary nodes, you write the code for your arrow. This line of code takes the form:

(4)
```
\anodecurve
[position-on]{source}
[position-on]{target}
{depth}
```

`source` and `target` are `nodenames` of the source and target nodes (i.e. which node the arrow comes **from**, and which node the arrow goes **to**).

`position-on` describes the six different places on a node you can attach arrows to. Use one of the six positions as shown in (5).

(5)
```
tl   t   tr
l  node  r
bl   b   br
```

`depth` will adjust how "ballooney" your curve looks. A bigger `depth` describes a more "ballooney" curve. Try using `0.3-1in` and adjust until you like it.

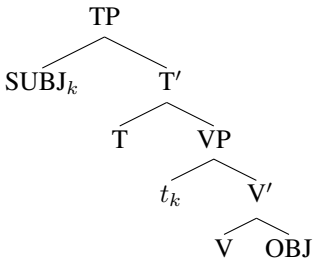
4 Examples of Arrows and More

This section shows a variety of different examples of the code and resulting tree structures. These examples use a couple of additional notations that you may find interesting/useful.

A very basic X'-syntax tree may look like this:

(6) a.
```
\Tree [.TP SUBJ_k [.T\1 T [.VP $t$_k [.V\1 V OBJ ]]]]
```

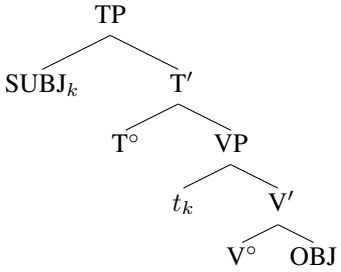
b.



If you want to add the degree symbol to heads, you can do so with the (rather unfortunately designed) command `$^{\circ}$`.

(7) a.
```
\Tree [.TP SUBJ_k [.T\1 T$^{\circ}$ [.VP $t$_k [.V\1 V$^{\circ}$ OBJ ]]]]
```

b.



To make the code easier to read, however, for now I'll leave the degree symbols out (you are welcome to do so on your own trees as well). To draw a tree with an arrow, then, you must do two things:

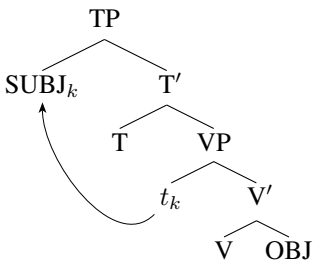
1. Identify the nodes in the code for the tree using the `\node` command
2. Use the `\anodecurve` command to draw the arrow

In the example below, I have identified SUBJ as a node (named SpecTP), and the trace in Spec,VP as a node (named SpecVP). The arrow draws a line between the bottom left of SpecVP to the bottom of SpecTP, with a depth of 0.3 inches.

(8) a.
```
\Tree [.TP \node{SpecTP}{SUBJ_k} [.T\1 T
[.VP \node{SpecVP}{$t$_k} [.V\1 V OBJ ]]]]

\anodecurve[bl]{SpecVP}[b]{SpecTP}{0.3in}
```

b.

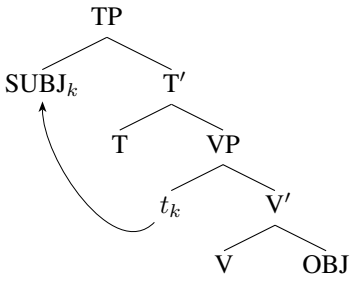


Note in the tree above that the length of the branches gets smaller and smaller as you go down the tree. If you want the length of branches to be similar throughout the entire tree, you have to use the command `!{\qbalance}`.

(9) a.
```
\Tree [.TP \node{SpecTP}{SUBJ_k} [.T\1 T
[.VP \node{SpecVP}{$t$_k} [.V\1 V OBJ !{\qbalance} ]]]]

\anodecurve[bl]{SpecVP}[b]{SpecTP}{0.3in}
```

b.

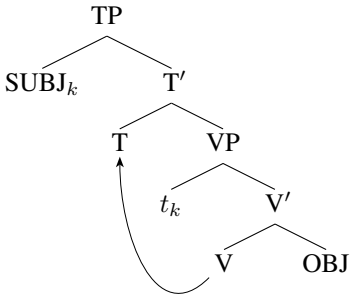


Alternatively, you might want to show some movements only using traces, and others using arrows.

(10) a.
```
\Tree [.TP SUBJ_k [.T\1 \node{T-head}{T}
[.VP $t$_k [.V\1 \node{V-head}{V} OBJ !{\qbalance} ]]]]

\anodecurve[bl]{V-head}[b]{T-head}{0.4in}
```

b.



The example below shows an object NP moving to an empty position in Spec,VP before moving to Spec,TP. The empty position on a tree is marked by a curly bracket structure that is empty `{}`.

(11) a.
```
\Tree
[.TP \node{specTP}{} [.T\1 T [.VP \node{specVP}{}
[.V\1 V \node{object}{NP} ] ] ] ]

\nodecurve[bl]{object}[b]{specVP}{0.4in}
\anodecurve[b]{specVP}[b]{specTP}{0.4in}
```

b.