Decorating Trees: Arrows for Syntax Diagrams with Forest

Charis Kim and Michael Diercks (Pomona College)
December 16, 2022

Contents

1 Introduction

This template/instructional guide is designed to help syntacticians of various levels learn to make fine-grained adjustments to syntactic trees and diagrams that are constructed using the forest package.

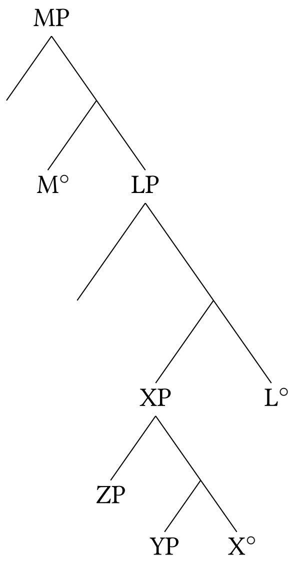
- This is not meant as a beginner’s introduction for LaTeX itself, nor is it an introduction for drawing trees with forest.
- If you need an introduction to LaTeX more generally (specifically for linguists), we recommend the Pomona Linguistics Quick Reference Guide.
- If you already are comfortable working in LaTeX and just need to learn trees in Forest, we suggest the Forest quickstart guide.[1]
- This introduction is designed to be accessed via Overleaf, and is published on Overleaf as a template.
- As on Overleaf template, when you open it in your account, the project now belongs to you: you can edit, adjust, and experiment to your heart’s content.
- If you break the document, don’t worry! You can just delete it and open up a new copy of the template.

2 Basic Forest Trees

- As mentioned above, a proper guide to the basics of forest trees for linguists is available here.
- A basic (abstract) syntax tree drawn with forest is given in (1).

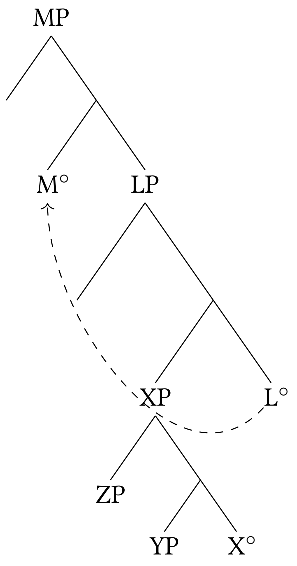
[1]You can always investigate the full documentation for the forest package, but it’s quite overwhelming.

(1)



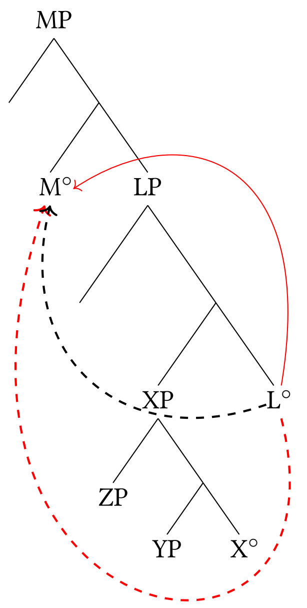
- The goal of this document is to illustrate how arrows can be drawn onto forest trees in visually appealing ways.
- Arrows are used in syntax diagrams to illustrate movement, agreement, and other kinds of relations-at-a-distance.
- As explained in the the forest quickstart guide, arrows can be created using the `\draw{}` command.
  - First, name the nodes in the tree you are attempting to connect via an arrow using ", name=X". This is illustrated in the tree in (2).
  - The command for drawing the arrow is then inserted after the tree but still inside the forest environment (before `\end{forest}`).
  - the `\draw` command identifies the source node, the target node, and can identify the exit and entry angles (using cardinal directions) from the source and target nodes, respectively, as illustrated below:
  - `\draw[->] (src) to[out=south west,in=south] (tgt);`
- As (2) shows, there are many instances where the automatically drawn path of the arrow is quite visually unappealing, even at times disrupting the communicative intent of the tree diagram.
- This primer offers instructions for more fine-grained control of the path of arrows in forest trees.

(2)



- The tree in (3) offers some alternatives.
  - The dashed black line in (3) is just slightly moved around the XP node.
  - The solid red arrow still crosses the tree, but is moved up around the LP node, out of the way of the central parts of the structure.
  - The dashed red arrow loops around the bottom of the tree.

(3)



- If you look at the `\draw` commands in (3), you can see that they look quite complex!
- This is what this document will teach, in relatively accessible ways (we hope): how to invent commands like the `\draw` commands in (3) so you can put arrows where you want them in your trees.

3 Arrow color, style, etc

- Arrows can take on a variety of appearances: in (3) there are already three distinct arrow styles.
- These styles are customizable with the `\draw` command by changing the contents of the square brackets [] immediately after `\draw`.
  - -> indicates an arrow that faces the target node, and [<-] indicates an arrow facing the source node.
    - Using `<->` would give you a double pointed arrow.
    - If you want a line with no tip, you can leave off the < or >: [-].
    - If you want the tip to be a circle, you can use [-*], [*-], or [*-*].
    - If you want the tip to be an open circle, use [-o], [o-], or [o-o].
  - Typing a range of **colors** (try red, blue, green, etc.) will allow you to change the color of the arrow.
  - **Line weight** can be determined using `thin, thick, ultra thick` and more.
  - If you scroll up to (3) and change the options and recompile, you will see the different effects.
- See this wiki page for nearly all the available formatting options.

4 Moving the “anchor” of an arrow to a specific part of a node

Each node on a forest tree can be labeled and used as a start or end point of an arrow. If you want that start or end point to be from a specific part of the node, here are some ways you can adjust that.

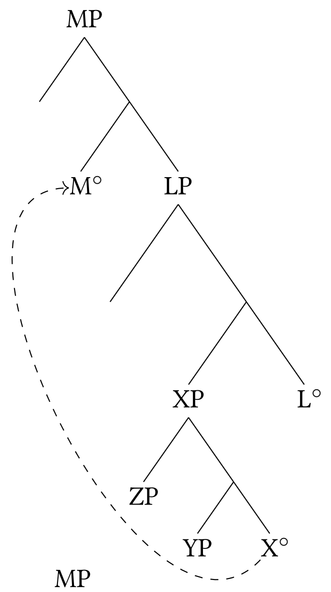
4.1 Cardinal directions

A basic arrow from A to B would simply be created using `\draw[->] (A) to (B);`, where (A) is the label of the source node (where the arrow comes out of), and B is the label of the target node (where the arrow goes into). A and B in the draw command therefore reference the “anchor” of the arrow, the point the arrow lands at: at anchor positions can be modified. If you wanted the arrow to start from the bottom point of A and end at the top right corner of B, you would use cardinal directions (e.g. north, south, east, etc). Using cardinal directions under the ’to’ function would also make the arrow start at the bottom of A and end at the top right of B, as in (4).

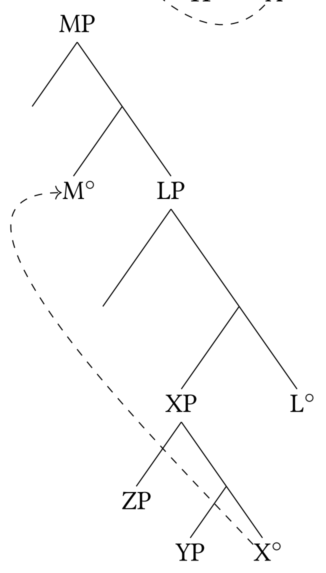
(4) `\draw[->] (A) to[out=south, in=north east] (B);`

The examples in (5) and (6) are identical apart from the “out” anchor for the arrow: it is “south west” in (5), and “north west” in (6). You can try changing the cardinal directions in the “to[]” command to see how the anchor points of the arrows change, and how the arc of the arrow changes as a result.

(5)



(6)



Adjusting the anchor points with cardinal directions does not directly control the arc of the arrow, but it is probably the easiest way to get the arc of an arrow to change, sometimes dramatically (as can be seen in the difference between (5) and (6)).

4.2 Fine-tuning anchor position

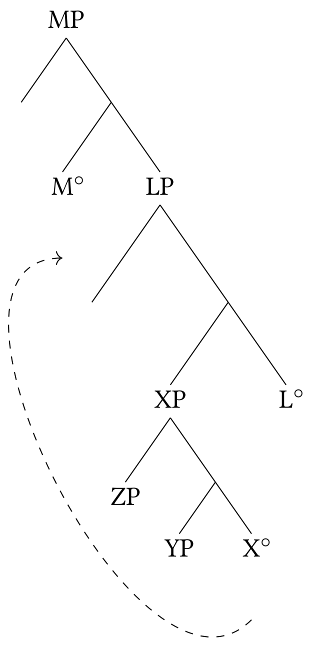
If the eight cardinal directions aren't specific enough, you can start from one cardinal direction and shift the anchor point (where the arrow originates from / goes to) from there in any direction. We do this using `xshift` and `yshift`. This modifies the anchor by moving it a specified amount on the x axis (horizontal) or y axis (vertical), respectively. For example, if we wanted an anchor set on the bottom of node A, then moved upwards 1em, we could use `([yshift=1em] A.south)`. If we wanted to use `xshift` and `yshift` simultaneously, we could do that with `([yshift=1em, xshift=-1.5em] A.south)`

Recall that the basic format of an arrow command as introduced here is as in (7):

(7)
```
\draw[->] (Source Anchor) to[out=cardinal, in=cardinal] (Target Anchor);
```

Therefore commands to move the source anchor will apply to the source anchor portion of the draw command, and likewise for the target anchor. Example (8) moves both the source anchor and the target anchor down (using a negative number) 2em. The result is somewhat ridiculous but it is at least obvious:

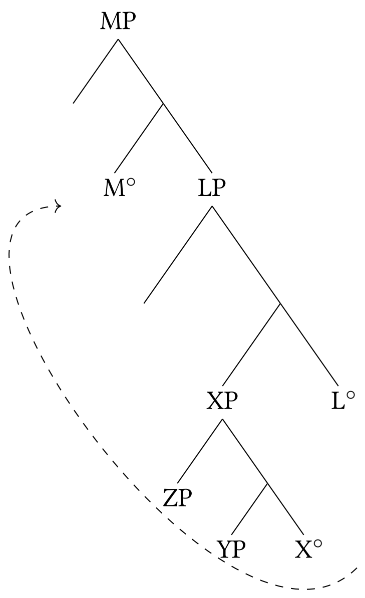
(8)



Adjust the value of the yshift for both source and target in (8) to see what happens! use positive and negative numbers, also decimal fractions for fine-grained control.

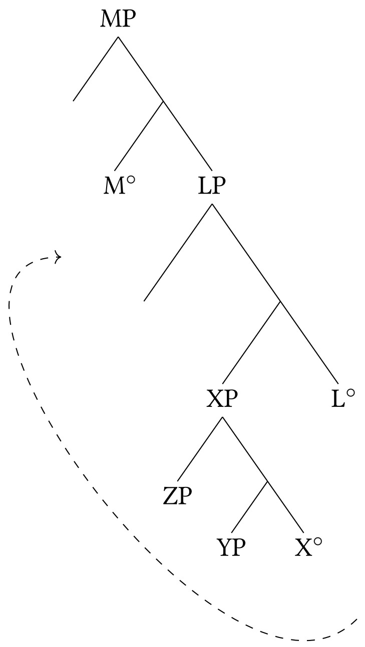
The xshift command works similar, but moves anchor points horizontally. The example in (9) is the same as above but uses xshift instead of yshift (and moves the source anchor 2em, and target anchor -2em). The result is that the source anchor (at X) moves to the right, and the target anchor (at M) moves to the left.

(9)



And it is possible to use xshift and yshift in the same command to move an anchor both vertically and horizontally. (10) illustrates this by combining the shifts from the previous examples.

(10)



In all of the examples above, you can adjust the xshift and yshift values and see what the result is. Meddle to your heart's content (and remember that if the document breaks and you can't solve it, you can always just click on the main template link (LINK) again and open a fresh copy).

We will also point out that in each instance, here, we've moved the anchor point from the *south* point on the relevant node (X.south, Y.south). This is not the only way to do this, as will become relevant in what comes below.

5 Arc of an arrow

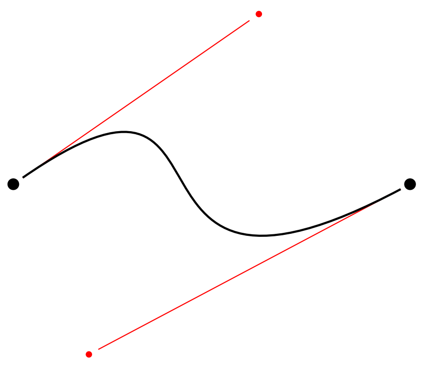
5.1 Adjusting leaving/entering angles

As we mentioned above, the most basic way to control the shape of an arrow is by specifying its entry and exit angles from its start and end points. We can do this using cardinal directions. For example, if we wanted an arrow to go straight down from its starting point and then enter its endpoint from the right, we could use `\draw[->] (A) to[out=south, in=east] (B);`. The `out=south` makes the arrow leave its starting point in the southward (downward) direction, and the `in=east` makes the arrow arrive at its endpoint from the eastward (rightward) direction. The leaving and arriving angles of arrows can be further fine-tuned using specific degree angles, as described in §4 and illustrated in the examples there.

5.2 Bezier curves

If you need an arrow to curve specifically around an obstacle (e.g. if an arrow is overlapping another element on your chart), Bezier curves are the way to go. A Bezier curve is a curve created by up to four points: a start point, an end point, a control point associated with the start point, and a control point associated with the end point. In (11), the black points represent the start and end points, the red points represent the control points, and the black curve is the arrow that they create.

(11)



This explainer about Bezier curves allows you to practice dragging the control points around and seeing what the result on the curve is. If you are going to use a Bezier curve in a LaTeX tree, we highly recommend reading the explainer first, and practicing adjusting control points there first: the LaTeX implementation will make a lot more sense afterwards. (You can also practice adjusting control points here.)

When you create an arrow in Forest, you can use Bezier curve controls to decide how it's shaped. Rather than using `\draw[->] (A) to (B)` like in previous examples, you would format the arrow as such:

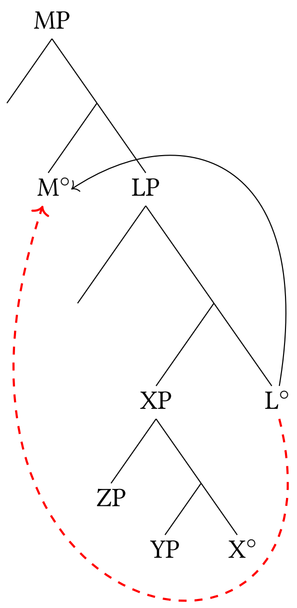
(12)
```
\draw[->] (Source Anchor) .. controls +(x, y) and +(x, y) .. (Target Anchor);.
```

The same controls for anchors above apply here (xshift, yshift, and the cardinal direction that the anchor sits at for the relevant node on the tree). But instead of the "to[out,in]" commands, the arc of the curve is controlled by "controls" and the entry/exit points for the arrows at the nodes is controlled by xshift,yshift, and the cardinal direction noted in the node.

You may notice the new `+(x, x)` format – this is how we determine the control points on our Bezier curve (again, see the explainer to learn what those are). The first `+(x, y)` creates the control point that is set in relation to the start point (i.e. the node on your tree where your arrow starts). The second (`+(x, y)`) is set in relation to the end point (i.e. the node on your tree where your arrow terminates). The + indicated how many units to the in the x and y direction the **control point** is located from its start/end point. For example, the control point `+(3, 1)` would be located 3 units to the right and 1 unit up from either the start or end point of the arrow.

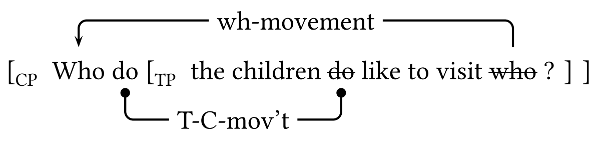
As we allude to above, the effect on the curve is often not immediately obvious based on the numbers entered to adjust the control points. Practicing with a bezier curve in general here or perhaps here will be helpful, but you can also adjust the controls for the examples in this document to practice. A general point is that in order to create the kinds of arcs syntacticians usually want the control points to both be on the same side of the arrow's arc.

The example in (13) is repeated from above, but should make more sense now. Adjust the control points and see what happens to the arrows! And if the result on the arrow curves is inscrutable, check out the linked explainers above re: Bezier curves.

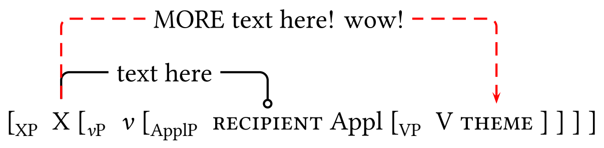
(13)


6 Arrows on linear structures

You may sometimes want to use a linear version of a tree in order to highlight certain connections (movement, agreement, etc). In this case, we use a slightly different method to format arrows. In order to create a start and end point for each arrow, we use the `\rnode{Node Name}{Text Shown}` command. For each word that you would like to set as a start or end point of an arrow, simply create an `\rnode` and name it accordingly. To draw the arrow, use `\ncbar`, and to place text on that arrow, use `\ncput*` in the following line. (14) offers a basic example:

(14)

wh-movement
[$_{CP}$ Who do [$_{TP}$ the children ~~do~~ like to visit ~~who~~ ?]]
T-C-mov't

Another example is given in (15), with colored arrows, dotted lines, different heights, and arrows facing the other direction.

(15)

MORE text here! wow!
text here
[$_{XP}$ X [$_{vP}$ v [$_{ApplP}$ RECIPIENT Appl [$_{VP}$ V THEME]]]]

You can modify the style of arrows with a few different commands. `linestyle`, `linecolor`, and `arm` will be the most useful to you (check the .tex to the left to see how they are used). See if you can change the red arrow from (15) to be dotted and blue, and you can try changing the tips of the arrows as well (all of these use the same commands as introduced earlier).