LING 311 – Tree drawing tutorial[*]

Adam Liter
liter@umd.edu

Last updated: January 27, 2020

1 Introduction

This is a short tutorial on how to draw syntax trees on the computer using commonly available resources. We will be reviewing four easy ways of producing trees:

(i) by using `Overleaf`, a cloud-based LaTeX system, and the package `forest` (§2.2.1);

(ii) by using a freely available JavaScript tool called `jsSyntaxTree` (§2.2.2);

(iii) by using the simple drawing functions of word processors like `Microsoft Word` (`LibreOffice` provides a free, open-source, cross-platform alternative to `Microsoft Word`) (§2.2.3); and

(iv) by using a free, open-source, cross-platform tool called `TreeForm` that uses a What You See Is What You Get (WYSIWYG) interface (§2.2.4).

Using the first option, `Overleaf`, is *strongly encouraged*. However, you are of course free to choose whatever method you prefer, or even come up with your own, if you feel so inclined.

2 Tree drawing

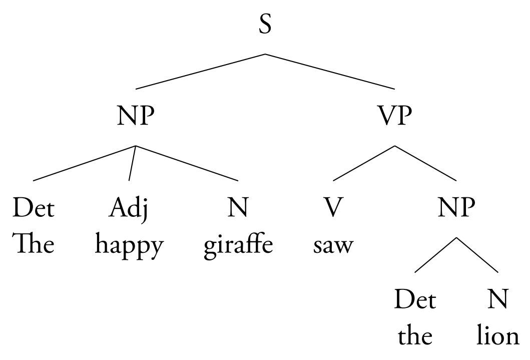
Both of the first two options (cf. §2.2.1 and §2.2.2) require using bracket notation for trees. There are several resources for familiarizing yourself with bracket notation. First, there is §3.1 of Alan Munn's `Overleaf` tree drawing tutorial. Alan's tutorial shows trees that look more like the trees we will be drawing by the end of the course as we develop our syntactic theory. I've adapted Alan's example below in §2.1 for a tree that you might be more familiar with. Second, you can refer to §2.3 of chapter 3 of the textbook for this course, Carnie (2002, 2007, 2013); this starts on p. 84 of the second edition and p. 95 of the third edition. Third, you can refer to the video tutorial put together by Adam Liter for using `Overleaf` and `forest`.

2.1 Bracket notation

Consider the tree in (1).

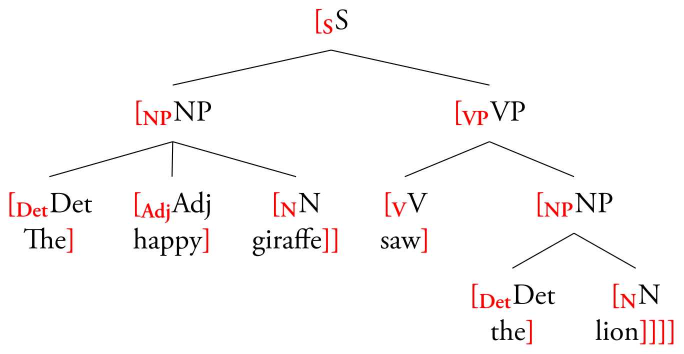
[*]A version of this tutorial was originally written by Diogo Almeida in fall 2006. The section on Microsoft Word greatly benefited from the input of the instructor of the course, Dr. Tonia Bleam. This document was substantially revised by Adam Liter, the LING 311 TA for spring 2020. Any questions or comments about this tutorial can be directed to either the current TA (via email or during office hours) or to Adam Liter (via email). Also, the `.tex` that produced this `.pdf` is available at `https://github.com/adamliter/tree-drawing-tutorial`. Pull requests, issues, and suggested improvements are welcome.

(1)



To convert this tree in to bracket notation, we want include a pair of opening and closing brackets (‘`[`’ and ‘`]`’) for each node in the tree. Moreover, each pair of brackets gets a label, and the label goes with the opening bracket (‘`[`’). It works best to draw your tree by hand on a piece of paper; then, when you go to convert it into bracket notation, you should start drawing your tree from the top, giving each node matching opening and closing brackets and a label. In (2), I’ve shown the brackets in red. Note that there are four closing brackets at the end; this is because all nodes require opening and closing brackets, so these four brackets correspond to closing off the N (*lion*) node, the NP node, the VP node, and the S node. Similarly, there are two closing brackets at the end of the subject NP, which correspond to closing off the N (*giraffe*) node and the subject NP node.

(2)



I encourage you to refer to the video tutorial put together by Adam Liter for using `Overleaf` and `forest`; this tutorial also covers the usage of bracket notation, and it may be easier to follow a video demonstration rather than reading the explanation given here.

Finally, note that this bracket notation can be used with both `Overleaf` and `jsSyntaxTree`; however, there are some slight differences about the bracket notation that only work in one case and not the other. This only applies to more advanced tree drawing, though. Almost everything in this tutorial should work just fine with both `Overleaf` and `jsSyntaxTree` (the one exception deals with how line breaks are handled; see §2.2.2).

2.2 Tree drawing programs

2.2.1 `Overleaf` and `forest`

`Overleaf` is a cloud-based LaTeX system; LaTeX is a document preparation system. Using LaTeX may look a bit like programming, but you should not be intimidated. We are only going to be writing a very basic LaTeX document, and you do not need to install LaTeX on your own computer since we are using `Overleaf`. You only need to create an account on `Overleaf`.[1]

[1] If you are interested in learning more about LaTeX, one good resource is this LaTeX tutorial.

You are encouraged to just watch the the video tutorial put together by Adam Liter for using `Overleaf` and `forest`; however, in what follows, I summarize the video tutorial.

For each tree that you want to draw, you'll need to create a new project on `Overleaf`. You should start each new project with the setup given in Listing 1. You can just copy and paste this in to your new project; alternatively, there is an Overleaf template called "Linguistics: Syntax tree drawing template using forest", which you can search for and use as the basis of a new project.

```
\documentclass[varwidth=true, border=5pt]{standalone}

\usepackage[linguistics]{forest}

\begin{document}
\begin{forest}
[S
]
\end{forest}
\end{document}
```

Listing 1: A template for drawing trees with `Overleaf` and `forest` using the `standalone` document class

Everything between the `\documentclass{...}` declaration and `\begin{document}` is called the preamble of the document; this is where you can load packages and define commands. To draw a tree, you only need to load the `forest` package and specify the optional `linguistics` argument.

The rules for using the `forest` package are straightforward.

(i) Every tree goes in between a line with `\begin{forest}` and a line with `\end{forest}`.

(ii) Every opening bracket needs a matching closing bracket.

(iii) In general, whitespace (spaces and line breaks) does not matter inside the `forest` environment, so you can break your bracketed tree across multiple lines; however, *there are two important exceptions*: (a) there can be no blank lines in the `forest` environment; and (b) the label for a node must appear immediately next to the opening bracket (*i.e.*, '`[S`' is acceptable but '`[ S`' is not).

A final document that produces the tree in (1) is given in Listing 2.

```latex
\documentclass[varwidth=true, border=5pt]{standalone}

\usepackage[linguistics]{forest}

\begin{document}
\begin{forest}
[S
  [NP
    [Det\\The]
    [Adj\\happy]
    [N\\giraffe]
  ]
  [VP
    [V\\saw]
    [NP
      [Det\\the]
      [N\\lion]
    ]
  ]
]
\end{forest}
\end{document}
```

Listing 2: A complete document that produces the tree in (1) using `Overleaf` and `forest`

Once you've finished drawing the tree, you can click the "Download PDF" button in `Overleaf` to download the resulting tree and insert it into a document, such as a `Microsoft Word` document, if that is how you are writing up your homework.

2.2.2 `jsSyntaxTree`

Another tree drawing option that uses bracket notation is called `jsSyntaxTree`.

If you go to the website, all you need to do is input the tree in bracket notation. You can then right click on the resulting image, save it to your computer, and insert it into a document, such as a `Microsoft Word` document, if that is how you are writing up your homework.

In general, you can use the same bracket notation for `jsSyntaxTree` as you would with `Overleaf` and `forest`. One exception is that you cannot use two backslashes (`\\`) for a line break with `jsSyntaxTree`; instead, you just enter the word that is at the leaf of the tree after the label and one space. You can draw the tree in (1) using `jsSyntaxTree` by pasting the bracket notation in Listing 3 into the website.

```
[S
  [NP
    [Det The]
    [Adj happy]
    [N giraffe]
  ]
  [VP
    [V saw]
    [NP
      [Det the]
      [N lion]
    ]
  ]
]
```

Listing 3: Bracket notation for producing the tree in (1) using `jsSyntaxTree`

2.2.3 Word processors

Drawing trees in a word processor involves two simple stages: (a) typing the structure of the tree; and (b) connecting the nodes you typed using the line drawing function. We are going to go through these below.

The first thing you need to do is type the structure of the tree (cf. Figure 1). You can use tabs to space the nodes out, and you will probably need to skip at least one line every time you finish with a structure level. (Notice in Figure 1 that there are actually three blank lines between the subject NP node and the Det, Adj, and N nodes inside that NP; this is because of needing to leave space for the VP structure.)

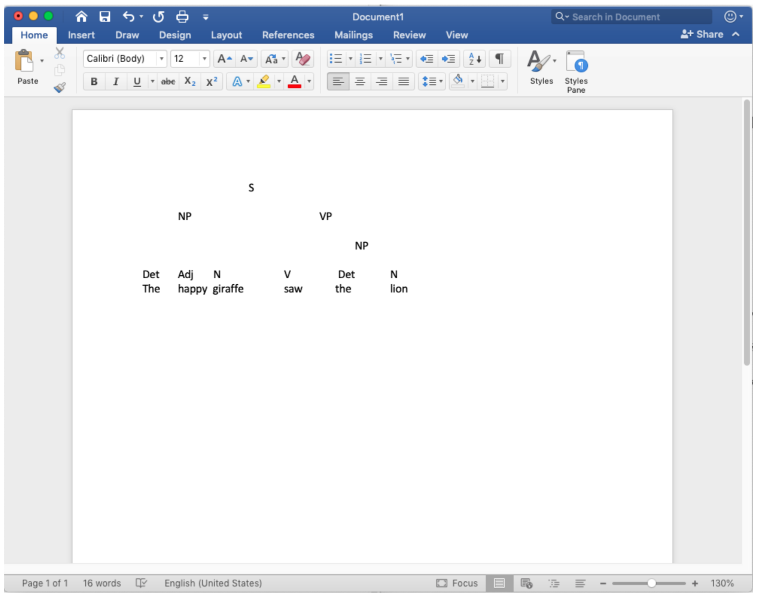


Figure 1: Typing the structure of the tree

Next, you will need to start inserting lines. You can do this by going to the `Insert` tab (or menu item). From there, you can select the `Shapes` dropdown menu and choose the simple `Line` shape (cf. Figure 2).

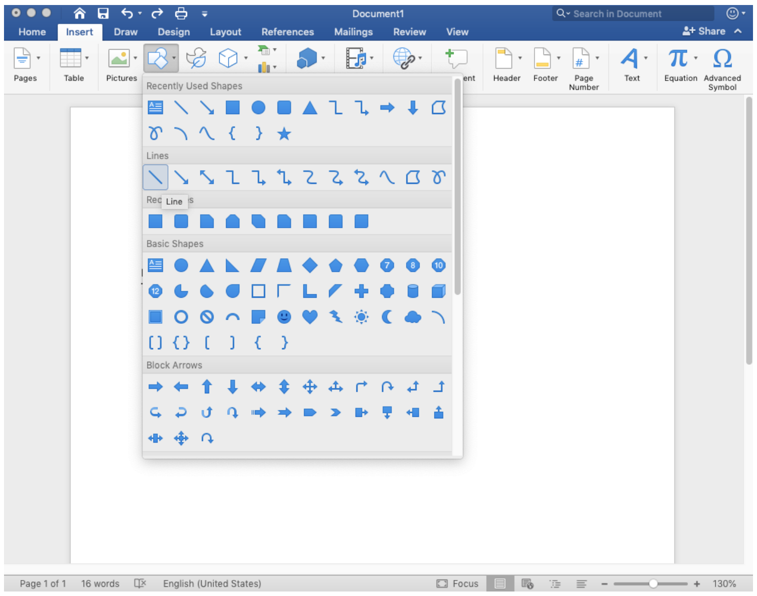


Figure 2: Inserting a line

Now you are ready to connect the nodes. Just draw lines between them (cf. Figure 3). Sometimes you will need to make minor adjustments either on the position of the nodes or on the position of the lines so the tree will look ok.

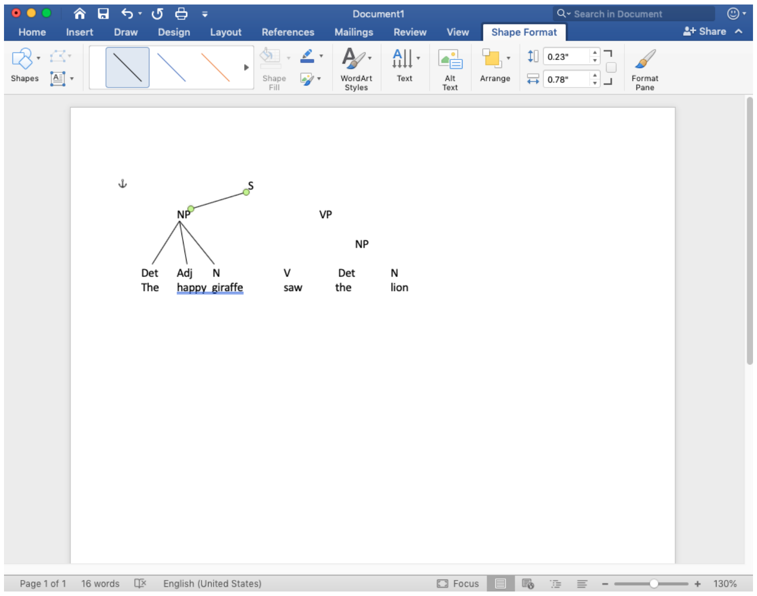


Figure 3: Drawing the lines

The final result should look something like in Figure 4.

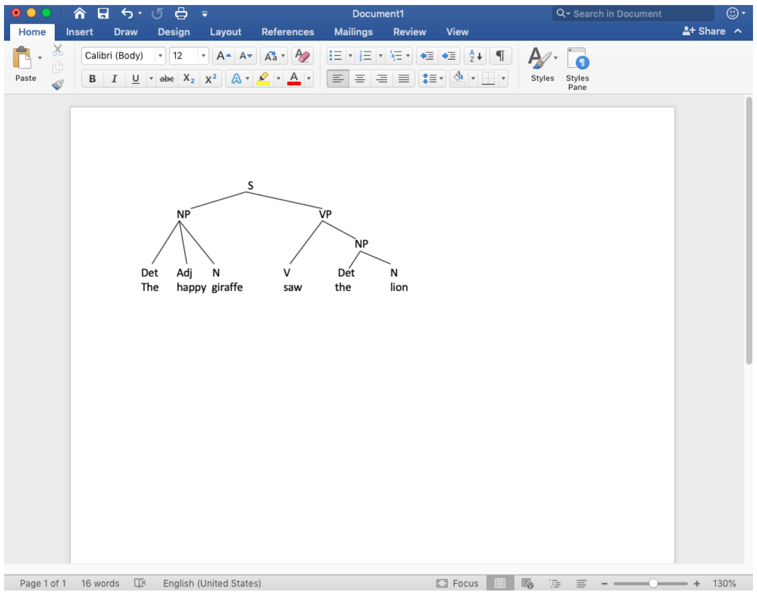
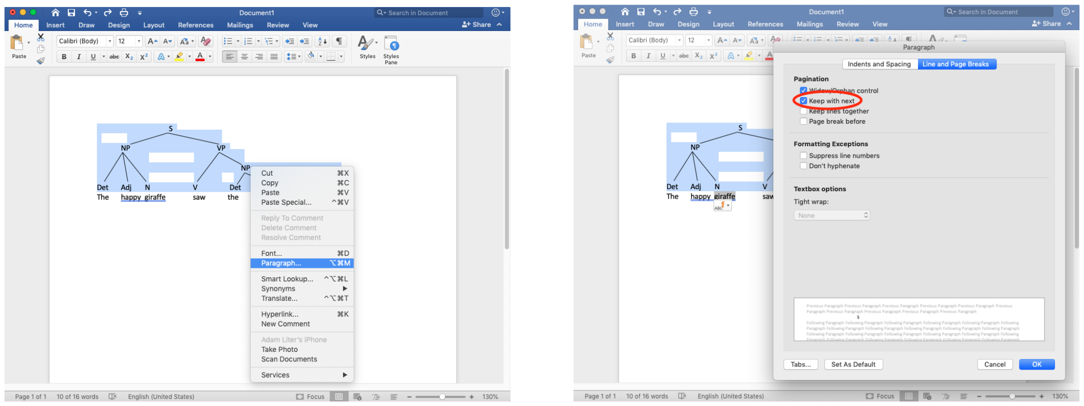


Figure 4: Final tree from using a word processor

The biggest downside to using a word processor to draw your trees is that you'll need to make sure that the tree stays together as you add writing above and below the tree; you'll also need to make sure that the tree does not get separated across multiple pages of the document. The best way to do this is to select all but the last line of your tree, right click, select `Paragraph`, then click on the `Line and Page Break` tab, and select the `Keep with next` option (cf. Figure 5). This should keep your tree together.



(a) Highlight and select `Paragraph` (b) Select `Keep with next`

Figure 5: Keeping a tree together when drawn with a word processor

2.2.4 `TreeForm`

Finally, the last option is to use `TreeForm`. `TreeForm` uses a WYSIWYG approach to drawing trees. As such it is fairly intuitive, and we will not cover it in detail here. There is a video tutorial at the linked website that you can watch.

In brief, you will need to download the program, install it, draw your tree using the interface, and then export it as an image which you can insert into a document, such as a `Microsoft Word` document, if that is how you are writing up your homework.

Note that `TreeForm` is somewhat of an outdated Java application. As of July 2019, the author of the software has begun to try to update and modernize it. However, because of it being somewhat outdated, you are discouraged from using this option. It is unlikely that the TA will be able to help you all that much with this option if you get stuck.

References

Carnie, Andrew. 2002. *Syntax: A generative introduction*. First Edition (Introducing Linguistics 4). Malden, MA: Blackwell Publishing, Ltd.

Carnie, Andrew. 2007. *Syntax: A generative introduction*. Second Edition (Introducing Linguistics 4). Malden, MA: Blackwell Publishing, Ltd.

Carnie, Andrew. 2013. *Syntax: A generative introduction*. Third Edition (Introducing Linguistics 4). Malden, MA: Blackwell Publishing, Ltd.