Forest Quickstart Guide for Linguists

Guido Vanden Wyngaerd
guido.vandenwyngaerd@kuleuven.be

January 14, 2020

Contents

1 Introduction

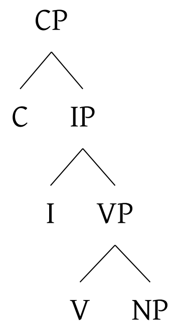
`Forest` is a package for drawing linguistic trees developed by Sašo Živanović. This manual provides a quickstart guide for linguists with just the essential things that you need to get started. More extensive documentation is available from the CTAN-archive. `Forest` is based on the TikZ package; more information about its commands, in particular those controlling the appearance of the nodes, the arrows, and the highlighting can be found in the TikZ documentation.

2 Loading Forest

In your preamble, put

```
\usepackage[linguistics]{forest}
```

The `linguistics` option makes for nice trees, in which the branches meet above the two nodes that they join; it will also align the example number (provided by `linguex`) with the top of the tree:

(1)


3 Basic Usage

`Forest` uses a familiar labelled brackets syntax. The code below will output the tree in (1) above (`\ex.` requires the `linguex` package and provides the example number):

```
\ex. \begin{forest}
[CP[C][IP[I][VP[V][NP]]]]
\end{forest}
```

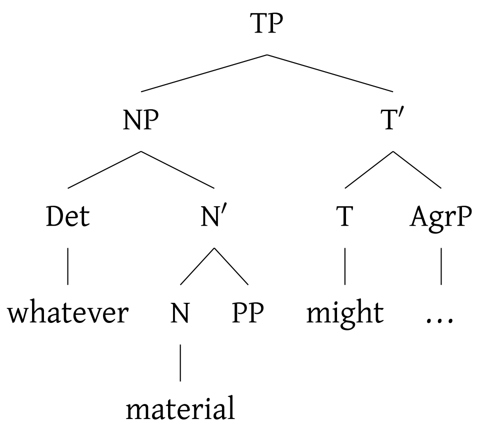
`Forest` will parse the above code without problem, but you will soon get lost in your labelled brackets with more complicated trees if you write the code this way. The better alternative is to arrange the nodes over multiple lines:

```
\ex. \begin{forest}
[CP
        [C]
        [IP
                [I]
                [VP [V] [NP]]
        ]
]
\end{forest}
```

One important caveat: be careful not to insert any empty lines in your code, as `forest` will not be able to parse those.

`Forest` automatically positions nodes in the tree depending on their internal complexity, i.e. the material that they dominate, as shown in the following example. Notice in particular how the horizontal spacing between the nodes varies according to what the nodes dominate.

```
\ex. \begin{forest}
[TP
        [NP
                [Det [whatever]]
                [N$'$
                        [N [material]] [PP]
                ]
        ]
        [T$'$
                [T [might] ]
                [AgrP [\ldots]]
        ]
]
\end{forest}
```

(2)


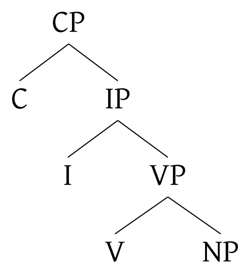
4 Adjusting node spacing

Although `forest` will arrange the nodes in the tree for you, you can still adjust both the horizontal and the vertical spacing, and the empty space around the nodes. The horizontal spacing is controlled by the `s sep` command, the vertical (or level) spacing by the `l` command. The `inner sep` command controls the empty space around the nodes.

You can specify absolute values for these parameters, as in the example below, or increase or decrease their default values as calculated by `forest`. This is done either by multiplication (e.g. `l*=3` multiplies the default level distance by 3), or by addition or subtraction (e.g. `l+=3mm` adds 3mm to the default level distance, `l-=3mm` subtracts 3mm).

These parameters can be applied globally, to the entire tree, as follows:

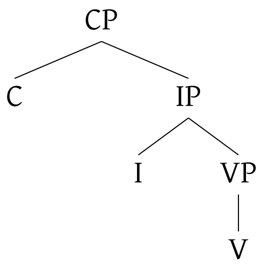
```
\ex. \begin{forest}
for tree={s sep=10mm, inner sep=0, l=0}
[CP
        [C]
        [IP
                [I]
                [VP [V] [NP]]
        ]
]
\end{forest}
```

(3)


This tree has wider horizontal spacing, less level distance, and less empty space around the nodes than the tree in (1).

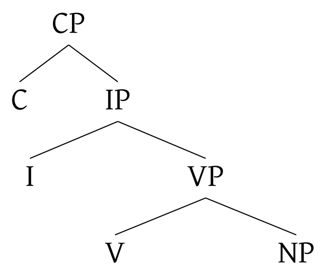
The commands that adjust horizontal and vertical spacing can also be applied locally, either to a single node, or to all the nodes dominated by a node (i.e. a subtree). Applying the parameter to a single node is done by putting a comma after the node label and then issuing the relevant command.

```
\ex. \begin{forest}
for tree={s sep=10mm, inner sep=0, l=0}
[CP, s sep=20mm
        [C]
        [IP
                [I]
                [VP [V]]]]
\end{forest}
```

(4) 

In (4), the daughters of the CP node get an increased horizontal spacing; the setting at the CP node overrides the global setting for the rest of the tree. You can also apply the setting to a subtree by applying the `for tree` command to a particular node:

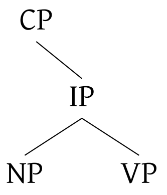
```
\ex. \begin{forest}
for tree={s sep=10mm, inner sep=0, l=0}
[CP
        [C]
        [IP, for tree={s sep=20mm}
                [I]
                [VP [V] [NP]]
        ]
]
\end{forest}
```

(5) 

Here increased horizontal distance has been applied to the subtree dominated by IP.

In order to produce a slanted rather than a vertical line under a nonbranching node, you can insert a phantom node. A phantom node has no label; the `phantom` command tells `forest` not to draw a line to this phantom node, as in the example below. Without the phantom node, a vertical line would connect CP to IP:

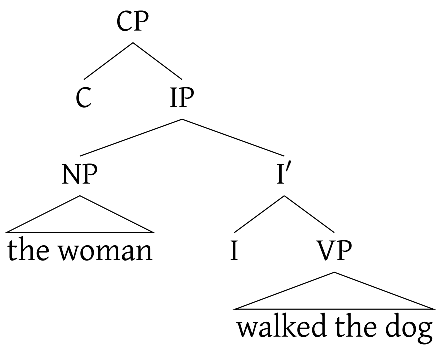
```
\ex. \begin{forest}
for tree={s sep=10mm, inner sep=0, l=0}
[CP     [,phantom]
        [IP
                [NP] [VP]
        ]
]
\end{forest}
```

(6)


5 Triangles

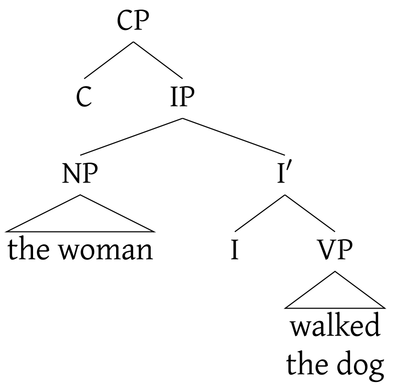
Triangles are easy to produce with the `roof` command, which you put before the closing bracket of the node that you want a triangle above.

```
\ex. \begin{forest}
for tree={s sep=10mm, inner sep=0, l=0}
[CP [C]
        [IP
                [NP [the woman, roof]]
                [I$'$ [I]
                        [VP [walked the dog, roof]]
                ]
        ]
]
\end{forest}
```

(7)


Sometimes you may want to reduce the width of a triangle in order to get a more balanced tree. This is easily done by putting a double backslash before the terminals that you want to move to a line below:

```
\ex. \begin{forest}
for tree={s sep=10mm, inner sep=0, l=0}
[CP
        [C]
        [IP
                [NP [the woman, roof]]
                [I$'$ [I]
                        [VP [walked \\ the dog, roof]]
                ]
        ]
]
\end{forest}
```

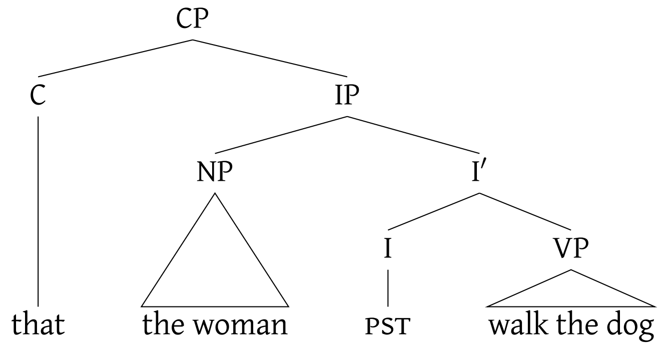
(8)


6 Horizontal alignment of terminals

You can align all the terminals horizontally with the `tier=word` command.

```
\ex. \begin{forest}
for tree={s sep=10mm, inner sep=0, l=0}
[CP
        [C [that, tier=word]]
        [IP
                [NP [the woman, roof, tier=word]]
                [I$'$
                        [I [\textsc{pst}, tier=word]]
                        [VP [walk the dog, roof, tier=word]]
                ]
        ]
]
\end{forest}
```

(9)



7 Arrows

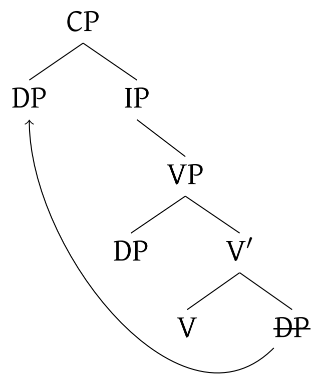
Arrows are drawn using the `\draw` command. This command takes two arguments, which specify the source and the target of the movement, respectively. The source and the target are defined by giving a name to the relevant nodes in the tree with the `name` command. In the example below, the target is named `tgt`, and the source `src`. The `\draw` command tells `forest` to draw a line from `src` to `tgt`. Place the command following the final closing bracket of your tree.

The optional argument of `draw` is used to specify the placement of the arrow (up `[->]`, or down `[<-]`, both `[<->]`, or none `[-]`), and the line style (e.g.

dotted). You can also specify where exactly at the source and target nodes the line should start and end by specifying this as an option with the to command; this is done using the option setting south for the bottom of the node, north for the top, south west for bottom left, etc.

```
\ex.\begin{forest}
for tree={s sep=10mm, inner sep=0, l=0}
[CP
        [DP,name=tgt]
        [IP
                [,phantom]
                [VP
                        [DP]
                        [V$'$ [V] [\sout{DP},name=src]]
                ]
        ]
]
\draw[->] (src) to[out=south west,in=south] (tgt);
\end{forest}
```

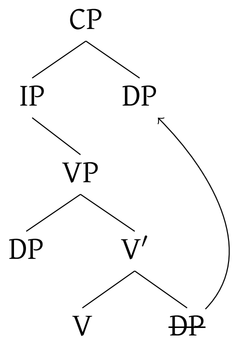
(10)



Do not forget the semicolon at the end of the \draw command. Note that the \sout command (to produce strikeout in the moved DP) requires the following line in your preamble: \usepackage[normalem]{ulem}.

The settings of the in and out options allow you to control the placement of arrows. This is shown in (11), where the arrow appears at the right hand side of the tree.

```
\ex.\begin{forest}
for tree={s sep=10mm, inner sep=0, l=0}
[CP
        [IP [,phantom]
                [VP
                        [DP]
                        [V$'$ [V] [\sout{DP},name=src]]]]
        [DP,name=tgt]]
\draw[->] (src) to[out=north east,in=south east] (tgt);
\end{forest}
```

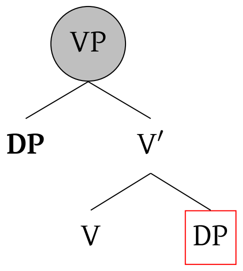
(11)


8 Highlighting

The simplest type of highlighting of individual nodes is done by applying formatting (like bold or italic) to the node labels. Fancier forms of highlighting are achieved with the draw command, as in the following example:

```
\ex. \begin{forest}
for tree={s sep=10mm, inner sep=1mm, l=0}
[VP,circle,draw,fill=lightgray
        [\textbf{DP}]
        [V$'$
                [V]
                [DP,draw,red]
        ]
]
\end{forest}
```

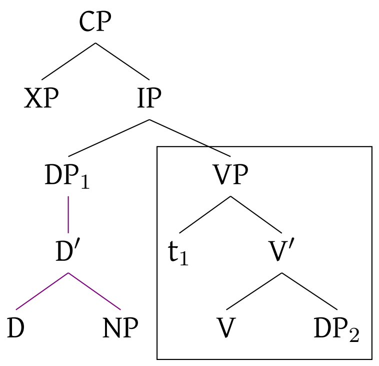
(12)



A subtree can be highlighted with a box around it using the `draw` command, or with colors using the `for tree` command, as in the following example:[1]

```
\ex. \begin{forest}
for tree={s sep=10mm, inner sep=0mm, l=0}
[CP
        [XP] [IP
                [DP$_1$, for tree={color=violet, edge=violet}, edge=black
                        [D$'$
                                [D] [NP]
                        ]
                ]
                [VP,tikz={\node [draw,fit to=tree]{};}
                        [t$_1$]
                        [V$'$
                                [V]     [DP$_2$]
                        ]
                ]
        ]
]
\end{forest}
```

[1]Observe that we first specify the edge of the DP-node to be violet at the tree level (i.e. for the DP subtree); if we did nothing else, the line connecting DP to its parent node would become violet as well. To avoid that, we next specify that the edge of the DP itself be black. Provided the latter command follows the former, black will overwrite violet on the line connecting DP and IP.

(13)


The `color=violet` command specifies the color of the nodes, the `edge=violet` command the color of the branches, i.e. the lines connecting the nodes. Note how TikZ commands are preceded by `tikz` and placed inside `{ }`.

The square around the VP arises by telling TikZ to draw the smallest rectangle around the subtree dominated by VP. You can also highlight sections of the tree that are neither single nodes, nor a complete subtree (as in (13)). In this case, you need to highlight a *nodewalk*. A nodewalk is a series of nodes that stand in some relation to a reference node. The relations are the following:

(14)

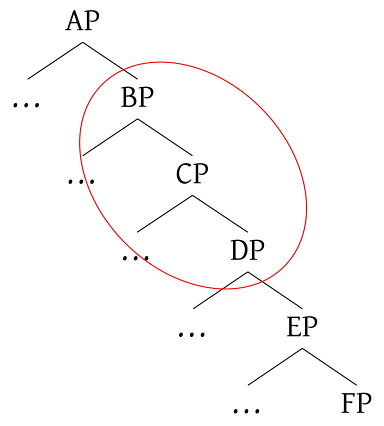
u	parent
p	previous sibling (i.e. sister on the left)
n	next sibling (i.e. sister on the right)
s	sister (only useful in binary trees)
1	first child (i.e. leftmost daughter)
2	second child (i.e. second daughter node counting from left to right)
l	last child (i.e. rightmost daughter)
ll	rightmost daughter of the rightmost daughter
F	first leaf (i.e. the leftmost terminal node) of the current node's descendants
L	last leaf (i.e. the rightmost terminal node) of the current node's descendants
N	next leaf (i.e. terminal node) in the linear order
P	previous leaf (i.e. terminal node) in the linear order

The syntax for specifying a nodewalk can be inferred from the following example, where `fit=` is followed by the nodewalk, i.e. a list of nodes, each preceded by ! and placed between brackets, starting with the reference node `( )` (=BP in the tree; forest infers this from the place where you put the nodewalk in your tree), and followed by a list of related nodes (CP and DP in (15)).

```
\ex.\begin{forest}
```

```
for tree={s sep=10mm, inner sep=0, l=0}
[AP
        [\ldots]
        [BP,tikz={\node
[draw,ellipse,inner sep=-2pt,red,rotate=45,fit=()(!1)(!11)] {};}
                [\ldots]
                [CP
                        [\ldots]
                        [DP
                                [\ldots]
                                [EP [\ldots] [FP]]
                        ]
                ]
        ]
]
\end{forest}
```

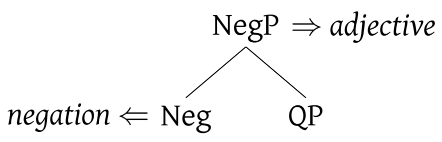
(15)



Other types of highlighting include annotations near nodes in the tree, as in (16) below. In this case, the `\draw` command is put following the closing bracket of the node where you want the annotation to appear, and the entire command needs to be enclosed within `{ }`. The option `node[right]` controls whether the annotation appears to the left or the right of the node. The option `(.west)` controls the point of attachment or anchor of the annotation.

```
\ex. \begin{forest}
for tree={s sep=10mm, inner sep=0, l=0}
[NegP [Neg]
{\draw (.west) node[left]{\textit{negation} $\Leftarrow$}; }
        [QP]
]
{ \draw (.east) node[right]{$\Rightarrow$ \textit{adjective}}; }
\end{forest}
```

(16)



Notice how `forest` automatically moves the entire tree to the right to make room for the annotation on the left side of the tree.