# Forest Quickstart Guide for Linguists

Guido Vanden Wyngaerd
guido.vandenwyngaerd@kuleuven.be

v2, 2020-06-28

## Contents



## 1 Introduction

`Forest` is a package for drawing linguistic (and other) tree diagrams 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.

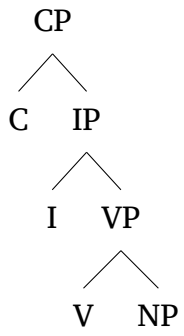
## 2 Loading Forest

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 are likely to 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:

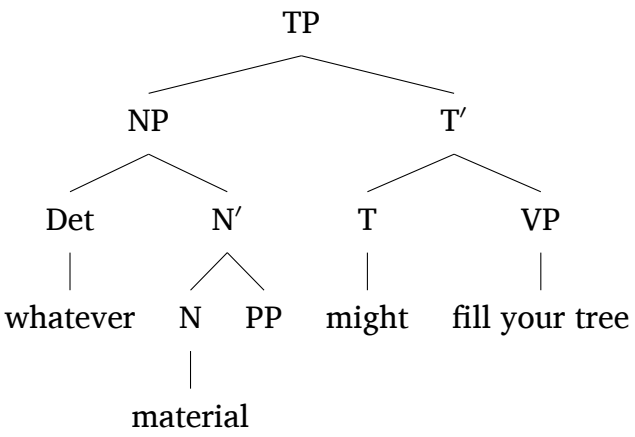
```
\ex. \begin{forest}
[CP
        [C]
        [IP
                [I]
                [VP [V] [NP]]
        ]
]
\end{forest}
```

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] ]
                [VP [fill your tree]]
        ]
]
\end{forest}
```

(2)




# 4 Adjusting node spacing

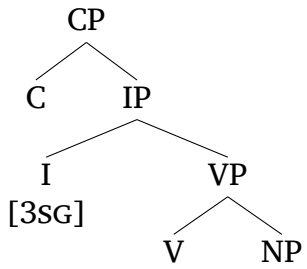
Although `forest` will arrange the nodes in the tree for you, you can still adjust both the horizontal and the vertical spacing, and the empty space around the nodes. The horizontal spacing is controlled by the `s sep` command, the vertical (or level) spacing by the `l` command. The `inner sep` command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

Also notice how we get square brackets in the tree, as in the [3SG] feature set under I. Normally, `forest` would interpret square brackets as instructions to create a new node, but by enclosing them between `{ }` they will appear as such in the output.

If you like this way the tree looks, you might want to use the `level`, `inner sep` and `s sep` settings for all your trees. This can be done by adding the relevant settings to your preamble (somewhere after the line with `\usepackage[linguistics]{forest}`):

```
\forestset{
default preamble={for tree={s sep=10mm, inner sep=0, l=0}
}
```

More options can be added inside the `\forestset` environment: just separate them by a comma.

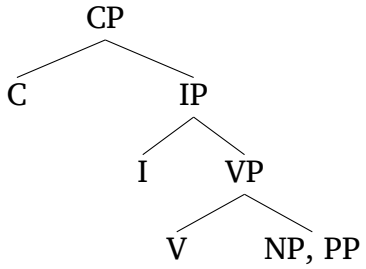
The commands that adjust horizontal and vertical spacing can also be applied locally, either to a single node, or to all the nodes dominated by a node (i.e. a subtree). Applying the parameter to a single node is done by putting a comma after the node label and then issuing the relevant command, as shown in (4).

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

```
\end{forest}
```

The comma following the node label CP is an instruction to `forest` that whatever follows it is not part of the node label, but an instruction for typesetting, in this case to adjust the value for the `s sep` at this node. We will come across such commas on multiple occasions later on. This means your node labels must be placed between `{ }` if they contain commas.

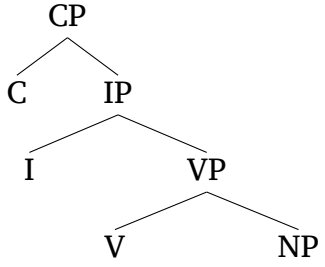
(4)




In (4), the daughters of the CP node get 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)

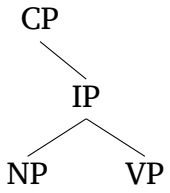


Here increased horizontal distance has been applied to the subtree dominated by IP.

In order to produce a slanted rather than a vertical line under a non-branching 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)



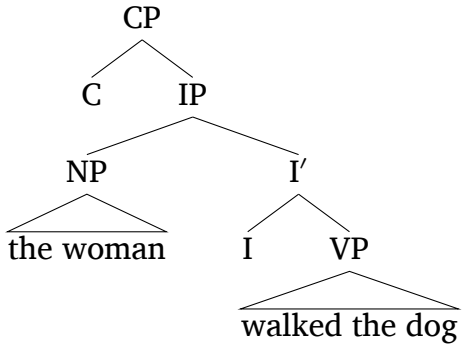

# 5 Triangles

Triangles are easy to produce with the `roof` command, which you put after a comma which follows the node label 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}
```

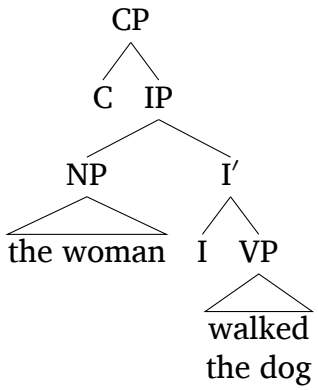
(7)




Sometimes you may want to reduce the width of a triangle in order to get a more balanced tree. This is achieved 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}
```

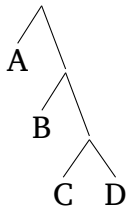
(8)




# 6 Unlabelled nodes

Sometimes you want to leave nodes in the tree unlabelled. Just leaving out these labels is fine as far as the `forest` syntax is concerned, but the result shows a rather uncomfortable gap where the node label would normally be:

(9)




The solution is to define a style, which is called `fairly nice empty nodes`, and which you can put in the document preamble.

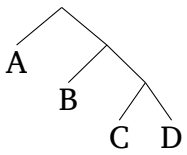
```
\forestset{
fairly nice empty nodes/.style={
delay={where content={}
{shape=coordinate, for siblings={anchor=north}}{}},
for tree={s sep=4mm}
}
}
```

We can now specify this is style in the code of the above tree, as follows:

```
\ex.
\begin{forest}
fairly nice empty nodes,
for tree={inner sep=0, l=0}
[[A]
        [[B]
                [[C][D]]
        ]
]
\end{forest}
```

The result is shown in (10).

(10)

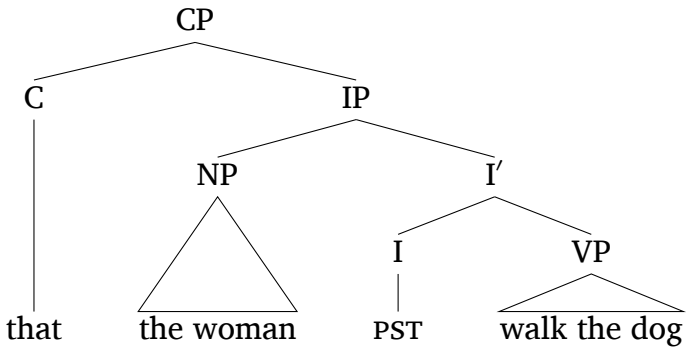



# 7 Horizontal alignment of terminals

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}
```

(11)




# 8 Arrows

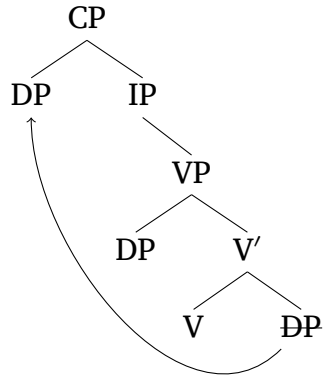
Arrows are made using the `\draw` command, which follows TikZ syntax. This command takes two arguments, which specify the nodes in the tree that are the source and the target of the arrow, respectively. These nodes are defined by giving a name to them with the `name` command. In the example below, the 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}
```

(12)




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 default arrow tips are rather tiny, which may be improved upon by specifying the `stealth'` option in the `\draw` command, as shown in (13) below. The use of this option requires that you add the following lines to your preamble:

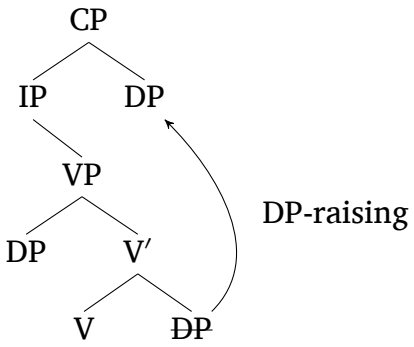
```
\usepackage{tikz}
\usetikzlibrary{arrows}
```

The settings of the `in` and `out` options allow you to control the placement of arrows. This is shown in (13), where the arrow appears at the right hand side of the tree, and with a curve that is the mirror image of the one in (12).

```
\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[->,>=stealth'] (src) to[out=north east,in=south east]
node[pos=0.5,xshift=14mm]{DP-raising} (tgt);
\end{forest}
```

(13)

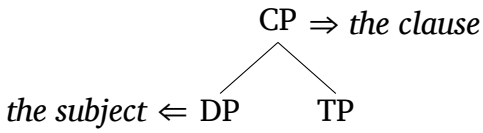



This example also adds an annotation to the arrow, by adding a `node` preceding the target. The option `pos` lets you position the annotation nearer the source (0) or the target (1) of the arrow (`pos=0.5` places it in the middle), the `xshift` option controls horizontal placement. The default is that the annotation will be placed on the line, so you will probably want to shift it a bit to the left or right to put it beside the line. Vertical spacing is controlled with `yshift`. If you specify more than one option, they need to be separated by commas.

You can also add annotations near nodes in the tree, as in (14) 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}
[CP
[DP] { \draw (.west) node[left]{\textit{the subject} $\Leftarrow$}; }
[TP]
] { \draw (.east) node[right]{$\Rightarrow$ \textit{the clause}}; }
\end{forest}
```

(14)



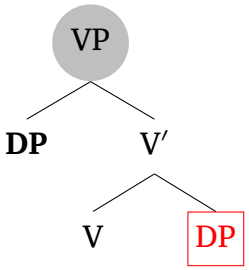

Notice how `forest` automatically moves the entire tree to the right to make room for the annotation on the left hand side of the tree.

# 9 Highlighting

You can highlight either nodes in the tree, or subtrees. The simplest type of highlighting of the nodes is done by applying formatting (like bold or italic) to the node labels. Fancier forms of highlighting are achieved as in the example below. `draw` is an instruction to draw a border around the node. The default shape of a node is a box, but this may be changed into a `circle` or an `ellipse`. The size of the node is controlled by `inner sep`. You can also specify a `fill` color of the node.

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

(15)

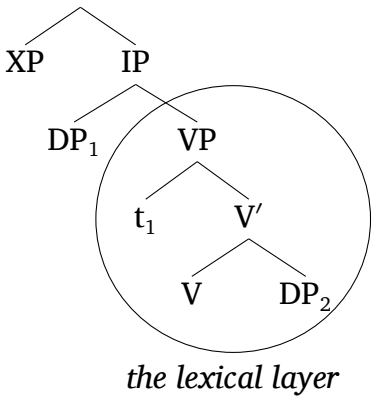



A subtree can be highlighted with a box, circle, or ellipse around it by specifying a TikZ command following the node label, as in the following

example:

```
\ex. \begin{forest}
for tree={s sep=10mm, inner sep=0mm, l=0}
[CP
[XP]
[IP
        [DP$_1$]
        [VP,tikz={\node [draw,ellipse,inner sep=-1pt,fit to=tree,
                label=below:\emph{the lexical layer}] {};}
                [t$_1$]
                [V$'$ [V] [DP$_2$]]
]]]
\end{forest}
```

(16)




The portion of the tree so highlighted may be accompanied by a label, as in the example. The positioning of the label may be controlled by `below`, `above`, `left`, `right`, or, more finely, by any positive or negative number between 0 and 360. This number specifies the angle with respect to a horizontal radius pointing from the center of the circle to the right (0, 360 and -360 place the label to the right of the circle, 90 and -270 place it above, 270 and -90 below, and so on).