`tikz-qtree`: better trees with TikZ

David Chiang
`chiang@isi.edu`

Version 1.2 (22 Apr 2012)

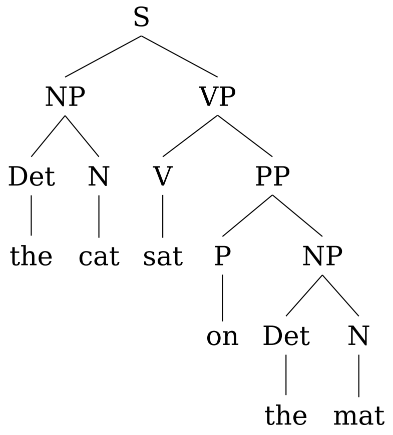
The `tikz-qtree` package provides a macro for drawing trees with TikZ[1] using the easy syntax of Alexis Dimitriadis' Qtree[2]. It improves on TikZ's standard tree-drawing facility by laying out tree nodes without collisions; it improves on Qtree by adding lots of features from TikZ; and it improves on `pst-qtree` in being usable with pdfTeX and XeTeX.[3]

1 Basics

To load the package in LaTeX:

```
\usepackage{tikz}
\usepackage{tikz-qtree}
```

The simplest usage is identical to Qtree:



```
\Tree [.S [.NP [.Det the ] [.N cat ] ]
          [.VP [.V sat ]
               [.PP [.P on ]
                    [.NP [.Det the ] [.N mat ] ] ] ] ]
```

Subtrees are delimited by square brackets. A subtree's root label is joined by a dot (`.`) to its opening bracket.[4] As in Qtree, spaces are required after every (internal or leaf) node label.

`\Tree` works inside or outside a `tikzpicture` environment, but many of the features described below require the explicit `tikzpicture` environment.

---

[1] `http://sourceforge.net/projects/pgf/`

[2] `http://www.ling.upenn.edu/advice/latex/qtree/`

[3] Although XeTeX works with `pst-qtree` using the `xetex-pstricks` package. For typesetting very large trees or a large number of trees, this may be the better option.

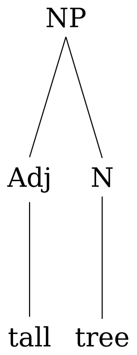
[4] You can also write the label after the closing bracket instead of the opening bracket, or both, or neither. Please see the Qtree documentation for details.

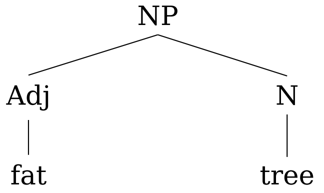
2 Tree options

Some options for standard TikZ trees work for `\Tree` as well:

- `level distance`: vertical distance between the anchors of a parent and its children
- `sibling distance`: horizontal distance between the boundaries of sister subtrees (not the anchors of their roots, as in standard TikZ trees). Note that TikZ nodes already have some horizontal space around them (`inner xsep`, by default `0.3333em`), so even `sibling distance=0pt` leaves some room.

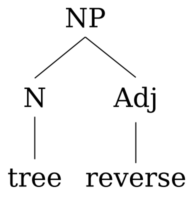
These are set either by writing `\tikzset{`*option*`=`*value*`}` or by writing `[`*option*`=`*value*`]` after a `\begin{tikzpicture}` or `\begin{scope}`.[5] For example:

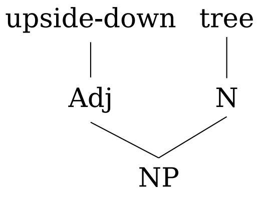




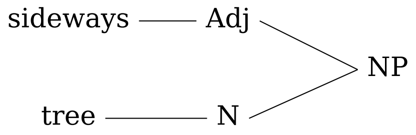
```
\begin{tikzpicture}
\tikzset{level distance=60pt}
\Tree [.NP [.Adj tall ] [.N tree ] ]
\end{tikzpicture}
%
\begin{tikzpicture}[sibling distance=72pt]
\Tree [.NP [.Adj fat ] [.N tree ] ]
\end{tikzpicture}
```

The `grow=`*direction* and `grow'=`*direction* options control the orientation of trees just as for standard TikZ trees. However, *direction* must be one of `up`, `down`, `left`, or `right`. The difference between `grow` and `grow'` is that `grow` places children counterclockwise with respect to their parent and `grow'` places them clockwise:





```
\begin{tikzpicture}[grow'=down]
\Tree [.NP [.Adj reverse ] [.N tree ] ]
\end{tikzpicture}
%
\begin{tikzpicture}[grow'=up]
\Tree [.NP [.Adj upside-down ] [.N tree ] ]
\end{tikzpicture}
```



```
\begin{tikzpicture}[grow=left]
\tikzset{level distance=60pt,sibling distance=18pt}
\tikzset{execute at begin node=\strut}
\Tree [.NP [.Adj sideways ] [.N tree ] ]
\end{tikzpicture}
```

Note that in sideways trees, `level distance` is horizontal and `sibling distance` is vertical. Sideways trees do take a little extra adjusting to look right, since the defaults are geared towards vertically growing trees. The meaning of the option `execute at begin node=\strut` is, before typesetting the label of every node, insert the command `\strut`, which is an invisible box that maximizes the height and depth of the node.

---

[5] Allowing options after `\Tree` would have made sense, but there would be no way to disambiguate the two uses of square brackets.

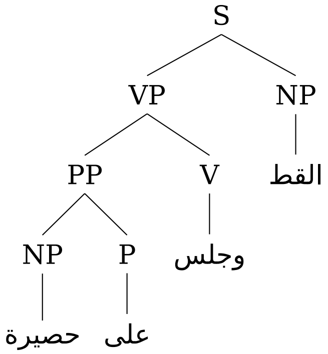
3 Styles

3.1 Node styles

The following TikZ styles are automatically applied to tree nodes, providing a hook for you to change the appearance of nodes or particular kinds of nodes:

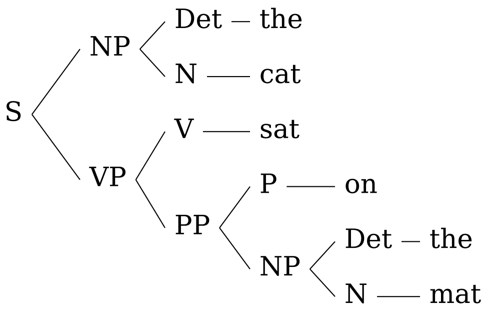
- `every tree node` applies to every node (default: `anchor=base`)
- `every internal node` applies to every internal node
- `every leaf node` applies to every leaf node
- `every level` n `node` applies to every node at level n, where $n = 0$ is the root

The options for nodes are all handled by TikZ and are described in detail in the TikZ documentation. For example, if you have a font named `\ar` and want to set all the leaf labels in this font:



```
\begin{tikzpicture}
\tikzset{grow'=down}
\tikzset{every leaf node/.style={font=\ar}}
\Tree [.S [.NP القط ]
          [.VP [.V وجلس ]
               [.PP [.P على ] [.NP حصيرة ] ] ] ]
\end{tikzpicture}
```

You can make the nodes in a sideways tree line up on their left edge using `anchor=base west`:

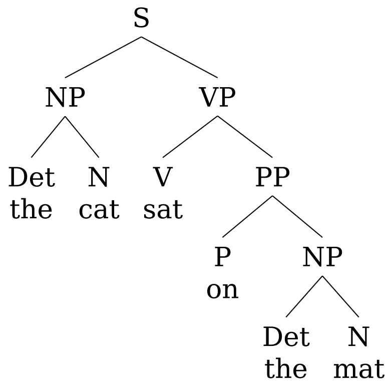


```
\begin{tikzpicture}
\tikzset{grow'=right,level distance=32pt}
\tikzset{execute at begin node=\strut}
\tikzset{every tree node/.style={anchor=base west}}
\Tree [.S [.NP [.Det the ] [.N cat ] ]
          [.VP [.V sat ]
               [.PP [.P on ]
                    [.NP [.Det the ] [.N mat ] ] ] ] ]
\end{tikzpicture}
```

In Qtree, it was allowed to use a line break (\\) inside a node. TikZ nodes by default don't allow this, but the `align` option (in PGF/TikZ version 2.1 or later) enables it as a side effect:[6]

---

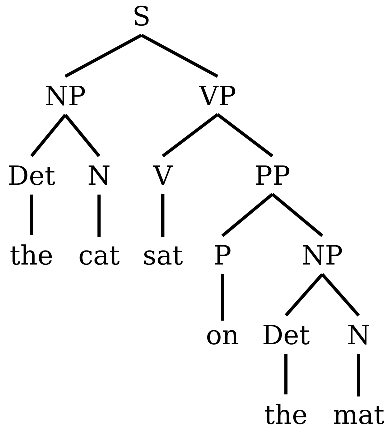
[6] Thanks to Alan Munn for figuring this out. Prior to PGF/TikZ version 2.1, the fix was to use the options `text width=2em,text centered`.



```
\begin{tikzpicture}
\tikzset{every tree node/.style={align=center,anchor=north}}
\Tree [.S [.NP Det\\the N\\cat ]
          [.VP V\\sat
               [.PP P\\on
                    [.NP Det\\the N\\mat ] ] ] ]
\end{tikzpicture}
```

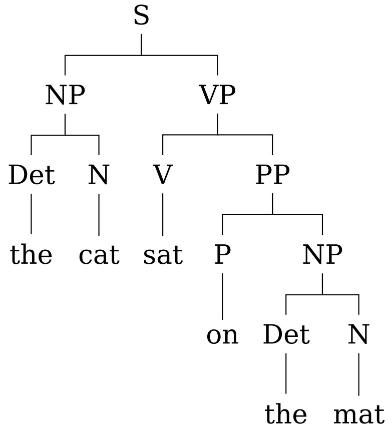
3.2 Edge styles

The `edge from parent` style applies to every edge (default value: `draw`). By defining this style, you can change the appearance of all the edges in a tree. For example, if you want the edges to be a little darker:



```
\begin{tikzpicture}
\tikzset{edge from parent/.append style={very thick}}
\Tree [.S [.NP [.Det the ] [.N cat ] ]
          [.VP [.V sat ]
               [.PP [.P on ]
                    [.NP [.Det the ] [.N mat ] ] ] ] ]
\end{tikzpicture}
```

Note that we must say `.append style` instead of just `.style`, in order to retain the `draw` option without which the edge will be invisible. As a more complex example, edges have an `edge from parent path` option which lets you change the shape of the edge. Its value is a TikZ path expressed in terms of `\tikzparentnode`, the parent node, and `\tikzchildnode`, the child node.

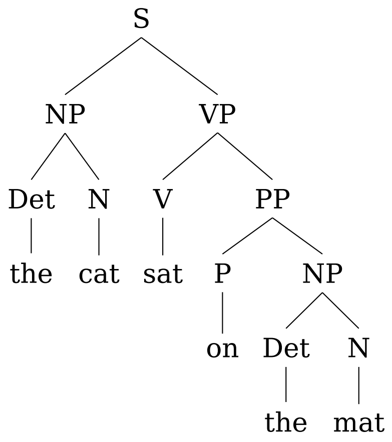


```
\begin{tikzpicture}
\tikzset{edge from parent/.style=
  {draw,
   edge from parent path={(\tikzparentnode.south)
                          -- +(0,-8pt)
                          -| (\tikzchildnode)}}}
\Tree [.S [.NP [.Det the ] [.N cat ] ]
          [.VP [.V sat ]
               [.PP [.P on ]
                    [.NP [.Det the ] [.N mat ] ] ] ] ]
\end{tikzpicture}
```

3.3 Node placement styles

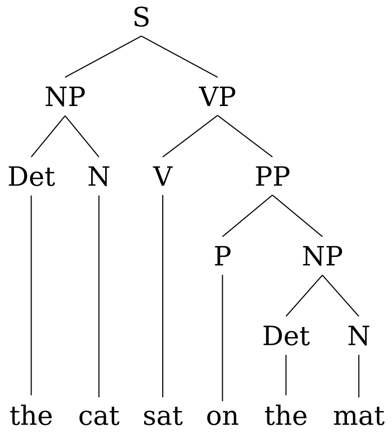
The following styles aren't applied to nodes, but affect the placement of nodes. By defining these styles, you can change the options `level distance` or `sibling distance` for different parts of the tree.[7]

- `level` n applies to the placement of level n (relative to level $n-1$)
- `level` n+ applies to level n and below
- `interior` applies to the placement of internal nodes (except the root)
- `frontier` applies to the placement of leaves



```
\begin{tikzpicture}
\tikzset{level 1/.style={level distance=36pt}}
\tikzset{level 2/.style={level distance=32pt}}
\tikzset{level 3+/.style={level distance=28pt}}
\Tree [.S [.NP [.Det the ] [.N cat ] ]
          [.VP [.V sat ]
               [.PP [.P on ]
                    [.NP [.Det the ] [.N mat ] ] ] ] ]
\end{tikzpicture}
```

In this context, you can also set the option `distance from root`, which positions a level relative to the root instead of the parent level. This is particularly useful for aligning all the leaf nodes:



```
\begin{tikzpicture}
\tikzset{frontier/.style={distance from root=150pt}}
\Tree [.S [.NP [.Det the ] [.N cat ] ]
          [.VP [.V sat ]
               [.PP [.P on ]
                    [.NP [.Det the ] [.N mat ] ] ] ] ]
\end{tikzpicture}
```

Unfortunately, the depth of the deepest leaf node is a global property of the tree and not easy to know during tree rendering, so you do have to specify the absolute depth of the leaf nodes. It will typically be an integer multiple of `level distance`.

---

[7] Thanks to Andrew Stacey for helping with the implementation.

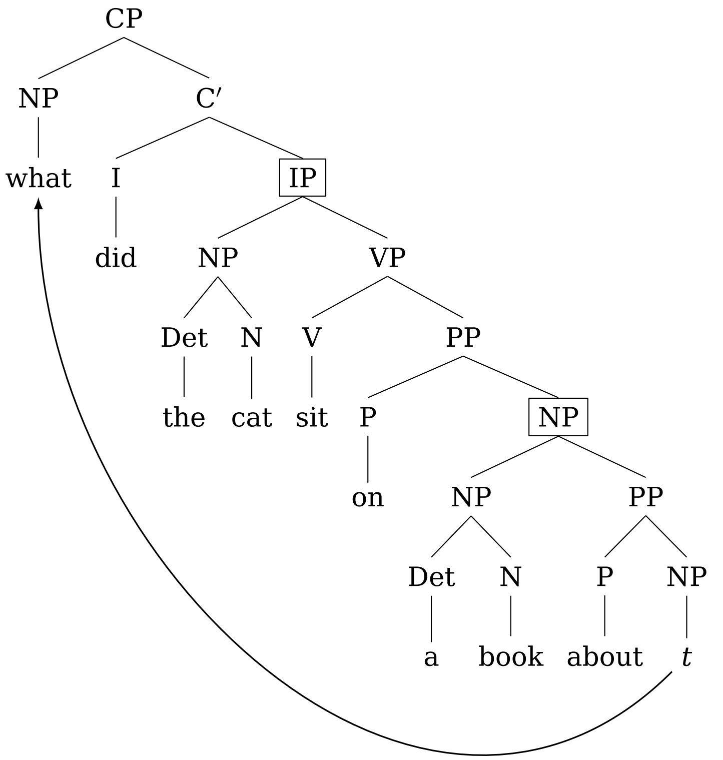
4 Embedding TikZ nodes

Inside a `\Tree`, in place of a node label, you can use a TikZ `\node` command.[8]

`\node [`*options*`] (`*name*`) {`*label*`};`

Don't forget the terminating semicolon. The `[`*options*`]`, which are optional, let you change the appearance of the node; for example, the `draw` option draws a border around the node. The `(`*name*`)`, which is also optional, can be used for drawing lines/arrows to/from the node.

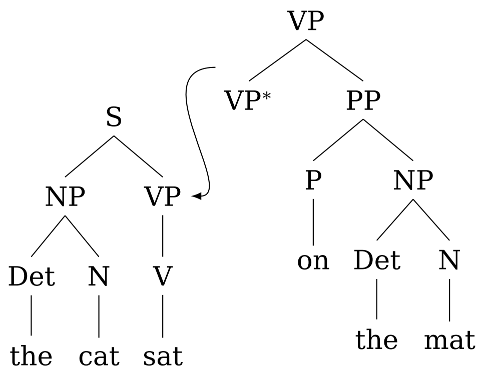
```
\begin{tikzpicture}
\Tree [.CP [.NP \node(wh){what}; ]
           [.C$'$ [.I did ]
                  [.\node[draw]{IP};
                    [.NP [.Det the ] [.N cat ] ]
                    [.VP [.V sit ]
                         [.PP [.P on ]
                              [.\node[draw]{NP};
                                [.NP [.Det a ] [.N book ] ]
                                [.PP [.P about ] [.NP \node(t){$t$}; ] ] ] ] ] ] ] ]
\draw[semithick,->] (t)..controls +(south west:5) and +(south:5)..(wh);
\end{tikzpicture}
```



---

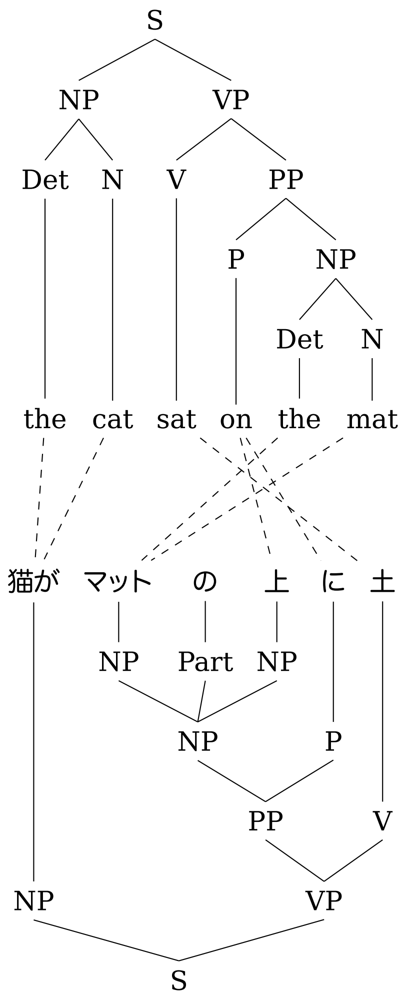
[8] `\Tree` specifically watches out for the token `\node`; do not use `\path node` or other equivalents.

You can also refer to the whole subtree rooted at the node named *name* using `\subtreeof{`*name*`}`:



```
\begin{tikzpicture}
\Tree [.S [.NP [.Det the ] [.N cat ] ]
        [.\node(site){VP}; [.V sat ] ] ]
\begin{scope}[shift={(1in,0.5in)}]
\Tree [.\node(root){VP}; VP$^\ast$
                    [.PP [.P on ]
                        [.NP [.Det the ] [.N mat ] ] ] ]
\end{scope}
\draw[->](\subtreeof{root}.140)..
        controls +(west:1) and +(east:1)..(site);
\end{tikzpicture}
```

Another example for machine translation people:



```
\begin{tikzpicture}
\begin{scope}[frontier/.style={distance from root=150pt}]
\Tree [.S [.NP [.Det \node(e1){the}; ]
        [.N \node(e2){cat}; ] ]
     [.VP [.V \node(e3){sat}; ]
        [.PP [.P \node(e4){on}; ]
           [.NP [.Det \node(e5){the}; ]
              [.N \node(e6){mat}; ] ] ] ] ]
\end{scope}
\begin{scope}[xshift=9pt,yshift=-5in,grow'=up,
        frontier/.style={distance from root=150pt}]
\tikzset{every leaf node/.style={font=\ja}}
\Tree [.S [.NP \node(j1){猫が}; ]
     [.VP [.PP [.NP [.NP \node(j2){マット}; ]
              [.Part \node(j3){の}; ]
              [.NP \node(j4){上}; ] ]
           [.P \node(j5){に}; ] ]
        [.V \node(j6){土}; ] ] ]
\end{scope}
\begin{scope}[dashed]
\draw (e1)--(j1);
\draw (e2)--(j1);
\draw (e3)--(j6);
\draw (e4)--(j4);
\draw (e4)--(j5);
\draw (e5)--(j2);
\draw (e6)--(j2);
\end{scope}
\end{tikzpicture}
```

5 Explicit edges

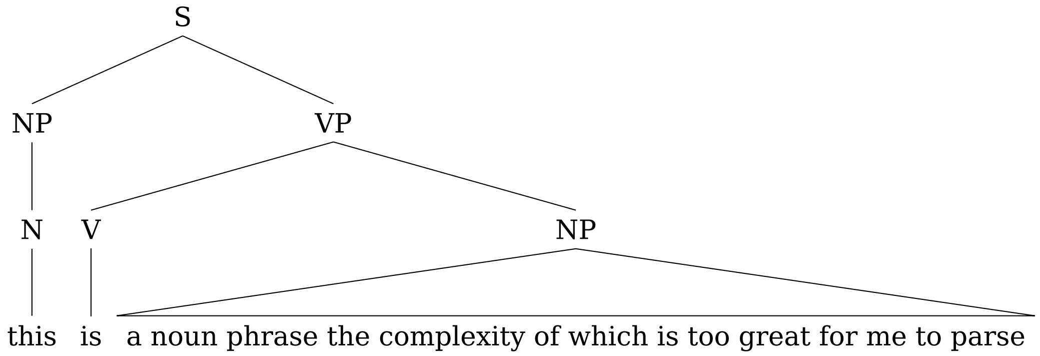
The edge from a parent to a child node is normally automatically drawn for you, but you can do it yourself with an `\edge` command *before* the corresponding child node. It is similar to the TikZ `edge from parent` command.[9]

`\edge [`*options*`];`

Again, don't forget the semicolon. The `[`*options*`]`, which are optional, let you change the appearance of the edge, as described above.

There is a predefined edge style `roof` that draws a triangle-shaped edge over a node, like Qtree's `\qroof`:

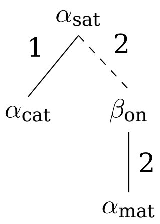
```
\begin{tikzpicture}[level distance=40pt]
\Tree [.S [.NP [.N this ] ]
          [.VP [.V is ]
               [.NP \edge[roof]; {a noun phrase the complexity of which
                                  is too great for me to parse} ] ] ]
\end{tikzpicture}
```


S
NP
VP
N
V
NP
this
is
a noun phrase the complexity of which is too great for me to parse

You can also add a label to the edge, using the following syntax:

`\edge [`*options*`] node [`*options*`] {`*label*`};`

Typically one will use the `auto` option for edge labels, which places the label to the side of the edge.



```
\newcommand{\initial}[1]{\ensuremath{\alpha_{\textrm{\scriptsize #1}}}}
\newcommand{\auxiliary}[1]{\ensuremath{\beta_{\textrm{\scriptsize #1}}}}
\begin{tikzpicture}[level distance=36pt,sibling distance=12pt]
\Tree [.\initial{sat}
        \edge node[auto=right]{1}; \initial{cat}
        \edge[dashed] node[auto=left]{2};
        [.\auxiliary{on}
          \edge node[auto=left]{2}; \initial{mat} ] ]
\end{tikzpicture}
```

The fact that `auto=left` draws a label on the right and `auto=right` draws a label on the left makes sense if you think about the tree growing from the root to the leaves.

---

[9] Except that a TikZ `edge from parent` comes after the child node. I thought it was more logical to put it before.

6 Qtree compatibility

For basic trees, `tikz-qtree` can be used as a drop-in replacement for Qtree, but most of Qtree's advanced features are either not accessed in the same way in `tikz-qtree` or not implemented at all. There is a package `tikz-qtree-compat` which can be loaded to improve compatibility. Supported so far are:

- Superscripts and subscripts outside of math mode, and `\automath`
- The `\0`, `\1`, and `\2` commands, and `\qtreeprimes`
- The `\qroof` command

For unsupported commands, warning messages are printed, but your file should compile.

Acknowledgements

This was all Dan Gildea's idea. Thanks to Alan Munn for his very helpful suggestions, and to Andrew Stacey for modifications.