Documentation for qtree, a LaTeX tree macro package[1]

by *Jeffrey Mark Siskind,*
with a front end by *Alexis Dimitriadis*

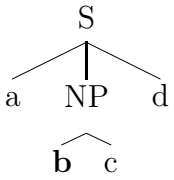
(Version 2)

The *qtree* package consists of QobiTree, a package of tree-drawing macros written by Jeff Siskind, and a front end that allows trees to be specified in bracket notation, using whitespace to separate tokens. Tree nodes, which can have labels of any size of complexity, are automatically arranged on the page, usually with quite good results. Provisions exist for fine-tuning the default layout. The front end also centers trees (by default) and provides some other nice features.

A simple tree may look like this:

```
\Tree [.S a [.NP {\bf b} c ] d ]
```

And produces:



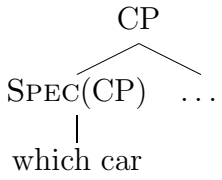
The tree macro file `qtree.sty` is designed to be installed in the directory of style files and included as an optional argument of the `\documentstyle` command. The lines used to draw trees look better if this package is used with the styles `epic.sty` and `eepic.sty` (distributed with *qtree*), which provide enhancements to the LaTeX picture environment. This version of *qtree* will automatically include `eepic.sty` if it can find it, so if `eepic.sty` is in the style files directory, it should not be necessary to give "`eepic`" as a style argument. This document begins with the header:

```
\documentstyle[12pt,qtree]{article}
```

The node labels may be quite complicated; they may contain font changes and math-mode text, line breaks introduced with `\\` (which produce centered lines), etc. The trees produced have a maximum depth of 20, with a maximum of five branches at any one node. Unlike many other tree macros, QobiTree adjusts automatically for the width and height of tree labels, and is pretty good at arranging nodes on the page.

[1]Thanks to Jeff Siskind for permission to distribute the QobiTree code. Please direct any comments to Alexis Dimitriadis, *alexis@babel.ling.upenn.edu*.

To make a multi-word node label, enclose it in braces; note also that TEX discards the spaces immediately after *control sequences* (commands whose name consists of a backslash followed by letters), hence if a node label ends with a control sequence, like `\ldots` in the following example, you need to enclose it in braces too.




```
\Tree [.CP [.{\sc Spec}(CP) {which car} ] {\ldots} ]
```

That's all you really need to get started! Read on for more features, examples and information about complicated uses.

Details and examples

A label for a non-terminal node can be written either after the left bracket or after the right bracket corresponding to that node. Thus the following are equivalent:

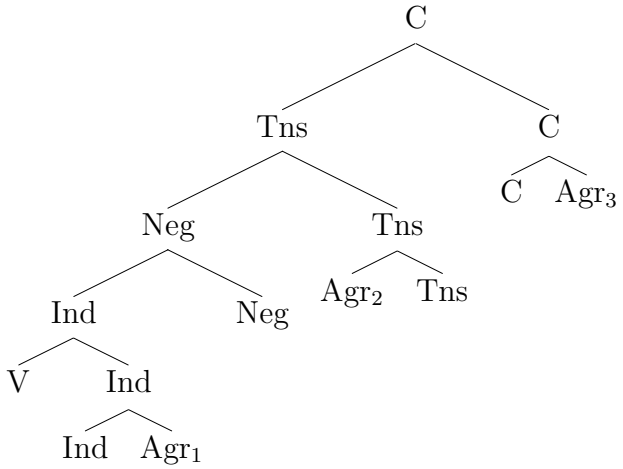
```
\Tree [.S a [.NP b c ]    d ]
\Tree [.S a [    b c ].NP d ]
```

To help keep braces matched when editing large trees, the front end allows the option of writing a label after both the left and the right bracket of the same node, as shown for the node NP below. In this case the two labels provided must be identical, token for token.

```
\Tree [.S a [.NP b c ].NP d ]
```

Numbered examples etc. The front end can be placed in a numbered example environment, in `\parbox`es, inside math formulas, tables, pictures, etc. The tree nodes can also contain arbitrarily complex material. Here is an example using the `enumerate` environment, and the code to generate it.

1. (a)




```
\begin{enumerate} \item (a) \hskip-0.75in
    \Tree[.C [.Tns [.Neg [.Ind V [.Ind Ind Agr_1 ]] Neg ].Neg
      [.Tns Agr_2 Tns ] ].Tns [.C C Agr_3 ] ]
\end{enumerate}
```

For obscure reasons, trees often appear farther to the right than is visually appealing; but not to worry, you can move them sideways by hand. (Note the `\hskip` in the last example, which moves the tree 0.75 inches to the left).

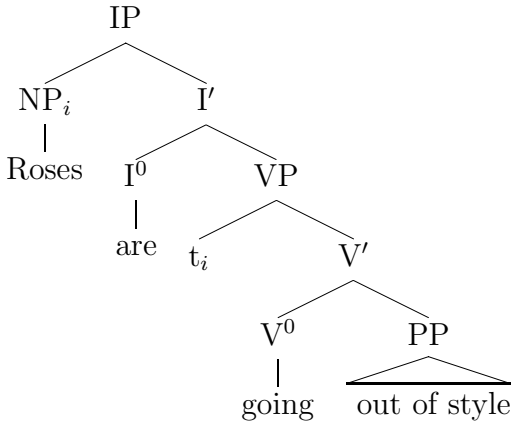
Roofs There is also a provision for drawing a triangular "roof" above a phrase that is treated as a unit. This is done with the command `\qroof`, which can appear anywhere a *leaf* can appear. The slope of the roof is equal to the ratio `\qroofy / \qroofx` (these counters may be reset to any pair of integers between zero and six; the default is 1/3).

To create a roof labeled *NP* over the phrase *the book,* type

```
\qroof{the book}.NP
```

If the phrase contains line breaks introduced with `\\`, the resulting lines are flush left, not centered. Again, it is possible for the "phrase" to be a construction of arbitrary complexity.

Subscripts and superscripts Trees are constructed in a special environment in which things like `NP_i`, `N^0`, automatically format their subscripts or superscripts in math mode, giving NP_i and N^0, respectively. The command that arranges this is called `\automath`, and can be invoked outside the tree environment, if desired. (It is turned off with `\noautomath`). As a further convenience, constructions like `X$'$`, producing X' , can be abbreviated `X\1`. Here is an example using some of these features:

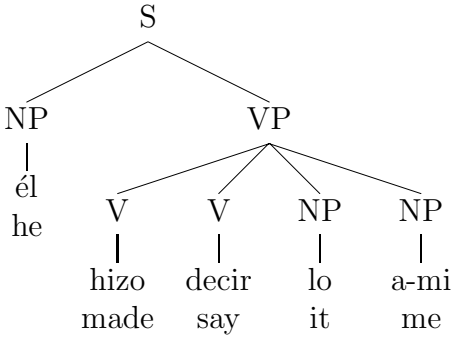
1. 

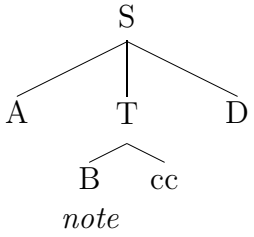

```
\begin{enumerate}
\item \Tree [.IP [ Roses ].NP_i [.I\1 [ are ].I^0
[.VP t_i [ [ going ].V^0 \qroof{out of style}.PP ].V\1 ].VP
          ].I\1 ]
\end{enumerate}
```

Granted, by the time the examples get this big, the bracketed format isn't all that readable, but it's certainly no worse than any other tree format, and you can add white space to make it a little better.

Left adjusted trees If you do not like centered trees, you can turn off the centering option with the command `\qtreecenterfalse`. Normally this would be invoked in the preamble, but it is possible to turn tree centering off and on (with the corresponding `\qtreecentertrue`) at any point. There is no provision for automatic right-adjusted trees.

Side by side trees Multiple trees, or text and trees, can be arranged side by side. This can often be done by just arranging commands one after another (it usually helps to turn off tree centering). If necessary the spacing can be adjusted with `\hskip`.

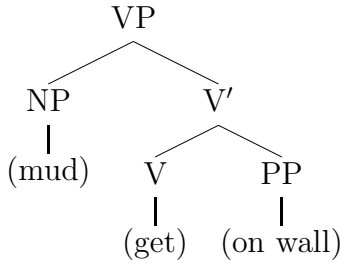
1. (a) 

(b) 

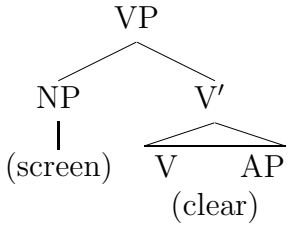
```
\begin{enumerate}
\qtreecenterfalse
\item (a) \hskip -0.3in\Tree [.S [.NP \'el\\he ]
        [.VP [.V hizo\\made ] [.V decir\\say ]
             [.NP lo\\it ] [.NP a-mi\\me ]      ].VP ]
  (b) \Tree[ A [.T {B\\ \em note} cc ].T D ].S
\end{enumerate}
```

Trivia note: If you insist on placing trees in a `\parbox`, here is how to do it. Note the `~`, called a *tie,* before `\Tree` inside the parbox; it keeps LaTeX from breaking the line and placing the tree one line too low.

1. (a)



(b)



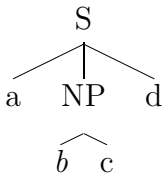

```
\begin{enumerate}
\item \parbox[t]{2.4in}{ (a)~\Tree
   [.VP [.NP (mud) ] [.V\1 [.V (get) ] [.PP {(on wall)} ]]] }
\parbox[t]{2in}{ (b)~\Tree
 [.VP [.NP (screen) ] \qroof{V\qquad AP\\ \hfil(clear)~}.V\1 ] }
\end{enumerate}
```

Escaping the parser There is also a provision for sneaking directives past the front end. If a word begins with an exclamation mark, the next word (i.e., up to the next space) will be passed through unchanged, except for stripping off the "!". (Braces should be used to pass through larger groups). One thing that this may be useful for is the `\faketreewidth` directive. (Note that `\qroof` should *not* be preceded by an exclamation mark).

Fine tuning The command `\faketreewidth{<text>}` tells QobiTree to override its default calculation of the width of the last-defined node and replace it with the width of `<text>` (which again can have '\\' commands etc.) `<text>` is not actually typeset but is used just to compute the fake width of the node on the top of the stack. When you use `\faketreewidth` you are on your own. This can either shrink or enlarge the space taken by the node and may result in trees with overlapping labels.

The low-level interface The guts of *qtree* are the tree macros written by Jeff Siskind, named QobiTree. Using the original interface (which is still usable with this package) the example tree shown on page one would be written like this:

```
\begin{center}
\leaf{a}
\leaf{\bf b} \leaf{c} \branch{2}{NP}
\leaf{d}
\branch{3}{S}
\qobitree
\end{center}
```



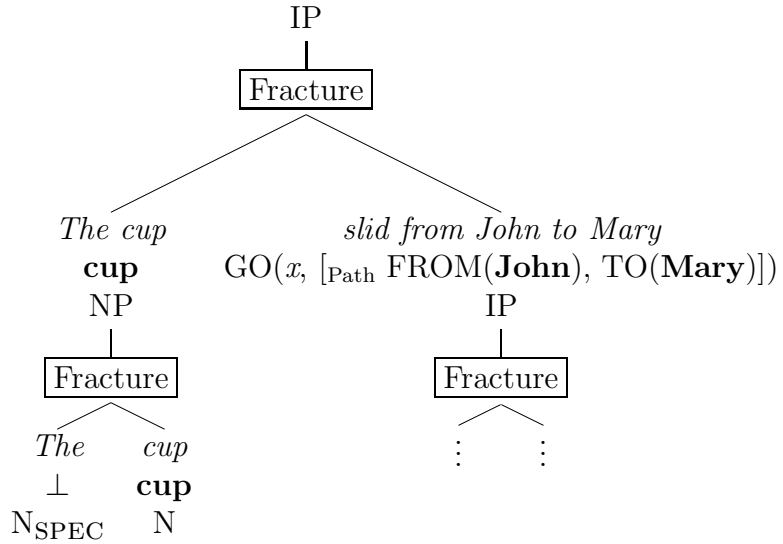
These macros operate like a stack machine. You push TEX boxes onto the stack of tree nodes, then you pop them off to make branching nodes which get pushed back on the stack.

Disclaimer: These macros still have a few rough edges, but are distributed in the belief that they are useful in their present form. Two known problems are the idiosyncratic centering (see above), and a problem with very small leaf nodes: the tree fragment `[ a b ].X` will produce invisible branches from `X` to `a` and `b`. The latter problem appears to be caused by the limited inventory of line slopes in the LATEX picture environment, and disappears if the picture enhancement styles are used, as was done to prepare this document.

I hope to completely fix the centering in the future, and I welcome any comments or reports of other problems or desirable features. But as usual, no guarantees, promises, etc. can be made about the present or future state of this package.

Here is a last demo, illustrating some possibilities. (This example, and parts of the above exposition, was adapted from the original documentation for QobiTree).

1. *The cup slid from John to Mary.*
GO(**cup**, [$_{\text{Path}}$ FROM(**John**), TO(**Mary**)])




```
\def\CUP{{\bf cup}}
\def\Nspec{N$_{\mbox{\sc spec}}$}

\Tree [.{{\em The cup slid from John to Mary.}\\
       {GO(\CUP,  $[_{\rm Path}$ FROM({\bf John}), TO({\bf Mary})])}\\IP}
 [.\fbox{Fracture}
%
   [.{{\em The cup}\\\CUP\\NP}
     [ {{\em The}\\$\bot$\\\Nspec} {{\em cup}\\\CUP\\N} ].\fbox{Fracture}
   ]
   [.{{\em slid from John to Mary}\\
    {GO({\it x},  $[_{\rm Path}$ FROM({\bf John}), TO({\bf Mary})])}\\IP}
     [ $\vdots$ $\vdots$ !\faketreewidth{WWW} ].\fbox{Fracture}
   ]
 ].\fbox{Fracture} % repeated label
]
```