Documentation for qtree, a LaTeX tree package[1]

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

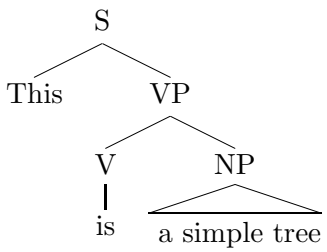
Version 2.20

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 or 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 This [.VP [.V is ] \qroof{a simple tree}.NP ] ]
```

which produces:



The node labels in trees 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, *qtree* automatically adjusts for the width and height of tree labels, and is pretty good at arranging nodes on the page.

Trees are defined using a version of the bracket notation familiar to linguists. Tree elements are delimited by white space; braces can be used to create multi-word labels. *Qtree* does not rely on `\catcode` changes for its operation, allowing trees to be included in footnotes and other moving environments without problems.

1 Invocation

`Qtree.sty` is a LaTeX package designed to be installed in a directory of style files, and included with the LaTeX 2_ε command `\usepackage{qtree}`.

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

Postscript specials *Qtree* relies on LaTeX's `picture` environment to draw the trees. Because this environment is rather limited, the lines used to draw trees look better if *qtree* is used with the package `eepic.sty,` which provides enhancements to the `picture` environment. This version of *qtree* will automatically include `eepic.sty` if it can find it, but automatic inclusion can be suppressed by use of the package option `[noeepic].` This may be necessary if the file will be processed with a driver that does not understand PostScript specials (e.g., *pdflatex*). In that case *qtree* will be included as follows:

```
\usepackage[noeepic]{qtree}
```

Tree centering Trees are centered by default, but you can turn centering off 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. These commands obey normal scoping rules. If used inside, say, an example environment, their effect will only apply until the end of that environment. There is no provision for automatically right-adjusted trees.

2 How to convert a tree to brackets

Reading or writing a complex tree in bracket notation is not terribly easy for humans; it helps to have an editor that can show matching braces as they are typed in. The procedure described here should allow you to easily convert a tree to bracket notation. If you don't have any difficulty with this, just skip this section and do it any way you want!

1. Draw the tree you want to enter on a piece of paper, so you can look at it.

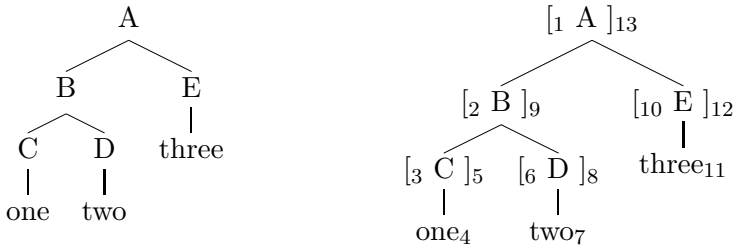
2. Imagine that the tree is a large peninsula, and your pencil is a boat sailing around it. Starting just to the left of the root node, move downwards, following the outline of the tree until you come back to the root node (on the right side, having moved counterclockwise around the tree), without crossing any of the tree's lines.

3. (a) Every time you are at the left side of a non-terminal node, type a left bracket, and the label for that node.

   (b) Every time you are at a leaf node, type in the contents of that node.

   (c) Finally, every time you are on the right of a non-terminal node, type a right bracket (and the node name again, if you want to help keep them straight).

It's difficult to show all this without including a picture, but consider the following tree; in the variant on the right, the numbered subscripts show the order in which the brackets and labels are written.

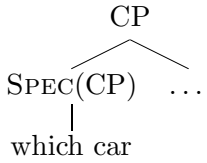


Accordingly, we would generate the tree by typing the following:

```
\Tree [.A [.B [.C one ] [.D two ] ].B [.E three ] ].A
```

3 Usage and features

Syntax The *qtree* front end reads a tree description written in the familiar (to linguists) bracket notation. Tree labels are delimited by whitespace. 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} ]
```

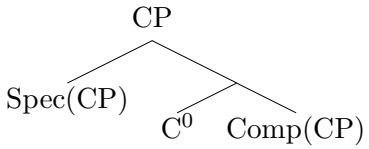
Label matching For convenience, a label for a non-terminal node can be written either after the left bracket or after the right bracket corresponding to that node. Thus the following are equivalent:

```
\Tree [.S when [.NP the cat ]    sleeps ]
\Tree [.S when [    the cat ].NP sleeps ]
```

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 when [.NP the cat ].NP sleeps ]
```

No labels Sometimes we want to draw an abbreviated tree without a label at every intermediate node. *Qtree* now draws such trees properly, as in the following example.




```
\Tree [.CP Spec(CP) [ C^0 Comp(CP) ] ]
```

Roofs It is possible to draw a triangular "roof" above a phrase that is treated as a unit. (See example on page 4). 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,* write

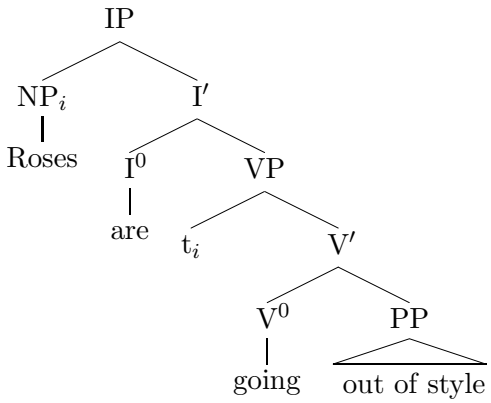
```
\qroof{the book}.NP
```

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; but the roof is implemented as a leaf node (it is not part of the original QobiTree), and so the syntax of `\qroof` does not allow further branches of the tree to appear *under* the roof.

Subscripts, superscripts and primes 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 enabled outside the tree environment, if desired. (It is turned off with `\noautomath`). This feature relies on `\catcode` changes for its operation; in trees that appear in footnotes or floats, all sub- or superscripts must be explicitly placed in math mode, as you would ordinarily do.

As a further convenience, constructions like `X$'$`, producing X', can be abbreviated `X\1`. (If you simply type `X'` you get X', with an apostrophe rather than a prime). There is also `X\2`, producing X'' and `X\0`, producing X^0. These commands also arrange for subtle improvements in the centering of labels that use them.

Here is an example using some of these features:




```
\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 ]
```

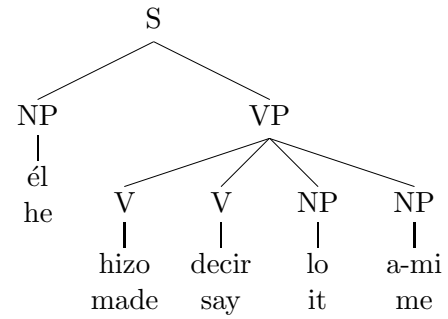
Granted, by the time the examples get this big, the bracketed format isn't all that readable, but it's certainly no worse than any other tree format, and you can add white space to make it a little better.

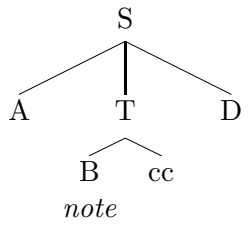
4 Tree placement

Numbered examples etc. A tree generated with *qtree* 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—although, unfortunately, it is not possible to embed a recursive call to *qtree*.

For hard-to-explain 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 next example, which moves the tree 0.5 inches to the left).

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

(2) a. 

b. 

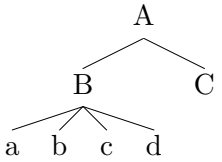

```
\begin{enumerate}
\qtreecenterfalse
\item[(2)] a. \hskip -0.5in\Tree [.S [.NP \'el\\he ]
        [.VP [.V hizo\\made ] [.V decir\\say ]
             [.NP lo\\it ] [.NP a-mi\\me ]      ].VP ]
  b.  \Tree[ A [.T {B\\ \em note} cc ].T D ].S
\end{enumerate}
```

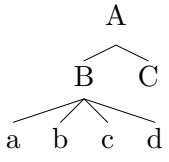
5 Advanced features

Escaping the parser There is 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). This is mainly useful for the manual width-adjustment directives `\faketreewidth` and `\qsetw`, described below. Note that `\qroof` should *not* be preceded by an exclamation mark.

Fine tuning The command `\qsetw{<length>}` (where `<length>` might be `0.5in`, `36pt`, etc.) tells QobiTree to override its default calculation of the width of the *just-finished* node (that's the node ending just to the *left* of where the directive was issued), and instead consider that width to be `<length>`. Similarly, `\faketreewidth{<text>}` sets the width of the last node to be equal to the width of `<text>` (which again can contain '`\\`' 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.

For example, the default placement rules would produce tree (a) below. By setting the width of the subtree headed by B to 1cm, we get tree (b).

a. 

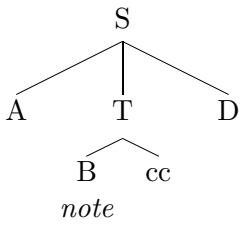
b. 

```
\begin{center}
\qtreecenterfalse
a. \Tree [.A [ a b c d ].B  C ]
\hfil
b. \Tree [.A [ a b c d ].B !\qsetw{1cm} C ]
\end{center}
```

When you use `\qsetw` or `\faketreewidth` you are on your own. They 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 accessible with this package) the example tree shown on page 6 would be written like this:

```
\begin{center}
\leaf{A}
    \leaf{B\\ \em note} \leaf{cc}
    \branch{2}{T}
\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.

6 How do I . . . ?

Make my tree fit in the page? Try one or more of the following: reduce the surrounding font size with `\small` or another size command, *before* you begin the tree; reduce the amount of space between subtrees with `\qsetw` or `\faketreewidth`; consider placing your tree sideways in the page, with one of the packages that provide rotation commands.

Draw movement arrows from one node of a tree to another? Use Emma Pease's *tree-dvips* package. Despite its name it is not a very convenient tool for creating trees, but its many line- and arrow-drawing commands can be used to decorate trees drawn with *qtree.*

Use *qtree* with *pdflatex*? *Pdflatex* does not support the PostScript specials generated by the package `eepic.sty`, which *qtree* loads automatically. At present, you have the following non-ideal options:

1. Do not use *pdflatex;* generate a PDF file by using a PostScript-to-PDF converter. The disadvantage of this solution is that the slideshow and hypertext capabilities of PDF are not available with the resulting files.

2. Suppress the automatic inclusion of *eepic,* by using the package option `[noeepic]`. This unfortunately results in lower-quality graphics, but is probably your best option if you need to use *pdflatex.* (You will also have to do without *tree-dvips* if you adopt this option).

The ideal solution would be to develop a PDF driver for *eepic,* or for some other extention to the picture environment. Please let me know if you know of such a thing.

Line up the text from all the leaf nodes on one horizontal line? As far as I can tell, *qtree*'s design is incompatible with this style of tree. I'd love it if there was an easy way to give *qtree* this capability (or the next one), but if there is, I haven't figured it out.

Draw dashed or dotted branches between certain nodes? Again, I can't see any way to incorporate this functionality into *qtree,* given the syntax of the front end. You can fake it to some extent, by creating lines that are part of a node as far as *qtree* is concerned, but which look like branch lines.

7 Troubleshooting

Disclaimer: This package is distributed in the belief that it is useful in its present form. 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 code.

The following problems are not really the fault of *qtree,* but fortunately they have easy solutions.

Some very short lines are not drawn This problem appears to be caused by the limited inventory of line slopes in the LaTeX picture environment. For example, the tree fragment `[.X a b ]` will produce invisible branch lines from `X` to `a` and `b`, but the lines will reappear if the labels are made wider. Install the picture enhancement styles (`eepic.sty`), and the problem will go away.

Qtree will not work with journal style X Any number of things could be going wrong, of course, but start by checking if the journal's style redefines the `tabular` environment. *Qtree* makes internal calls to `tabular`, so this is a frequent source of problems. Usually the style's writer has saved the original definition of `\tabular` under a different name, so all you need to do is arrange for the original definition to be restored for the calls to `\Tree`.

There is now a hook to make this easier: If you define a command named `\qtreebugfixhook`, it will be implicitly called by `\Tree`, with local scope (so that any redefinitions it causes are automatically cancelled at the end of the call to `\Tree`). For example, the JNLE style (`nle.sty`) saves the commands to begin and end a table as `\oldtabular` and `\endoldtabular`, respectively, and the replacement macros result in $r\,e\,a\,l\,l\,y\quad w\,i\,d\,e$ trees. The following will restore the original definitions for calls to `\Tree` only.

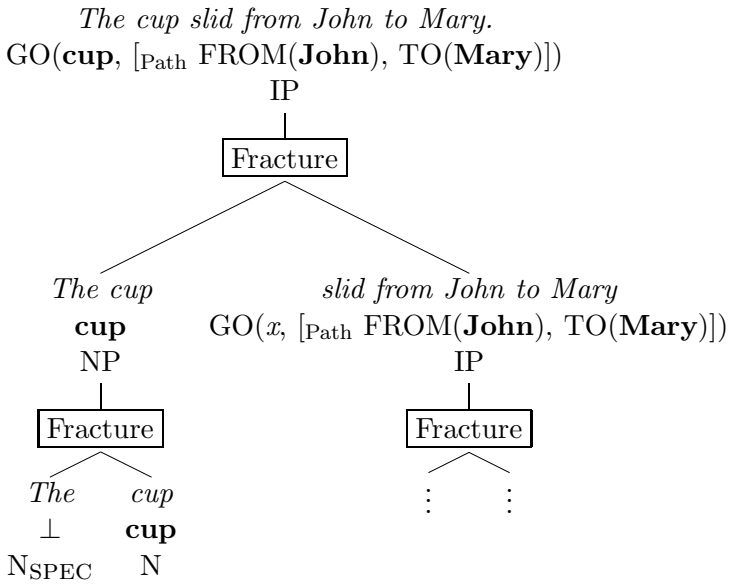
```
\def\qtreebugfixhook{\let\tabular=\oldtabular
         \let\endtabular=\endoldtabular}
```

Kluwer's house style saves the original definitions as `\klu@tabular` and `\klu@endtabular`, so to use *qtree* with it, do the following. (You need the `\makeatletter` call to use commands that contain an @-sign).

```
\makeatletter
\def\qtreebugfixhook{\let\tabular=\klu@tabular
         \let\endtabular=\klu@endtabular}
\makeatother
```

8 Inspiration

Here is a last demo, illustrating some of the things you *can* do with *qtree.* (This example, and parts of the above exposition, were adapted from the original documentation for QobiTree).




```
\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
]
```