

Notes on using `pst-jftree`

The commands described in this note are contained in the file `pst-jftree.tex`, available at <http://www.math.neu.edu/ling/tex/>, as is the latest version of these notes to the user. A LaTeX `.sty` wrapper for the `.tex` file is also available for LaTeX users. Familiarity with (or at least willingness to become familiar with) PSTricks is assumed. Most of the commands take parameters which are defined and explained in the PSTricks documentation. See <http://www.tug.org/applications/PSTricks/> for information on PSTricks.

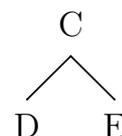
Branch definitions and node definitions

A tree is typeset by defining the branches that it needs to use, and defining its nodes and their connections in terms of the defined branches.

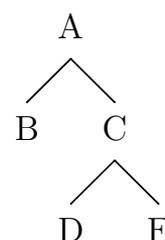
```
\defbranch\medleft(15pt)(1)
\defbranch\medright(15pt)(-1)

\defnode\firstnode{A}\medleft\secondnode\medright\thirdnode\cr
\defnode\secondnode{B}\cr
\defnode\thirdnode{C}\medleft\fourthnode\medright\fifthnode\cr
\defnode\fourthnode{D}\cr
\defnode\fifthnode{E}\cr
```

Given the definitions, evaluating `\thirdnode` then produces the tree at the right. The branches are 15 pts high, the rightbranch has slope -1 , and the left branch has slope 1 , as specified by the `\defbranch` commands.



Evaluating `\firstnode` produces the full tree. This evaluation uses the definition of `\firstnode`, which in turn uses the definitions of `\medleft`, `\secondnode`, `\medright`, and `\thirdnode`.



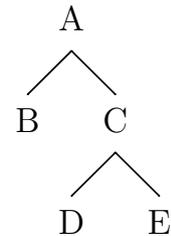
Simply evaluating `\firstnode` is awkward, because its output is dimensionless. `pst-jftree` provides the `\jftree ... \endjftree` construction which handles the dimensions and setting the baseline and encloses all the definitions in a group. It also makes `\!` equivalent to `\setnode`. Branch definitions are usually made outside the `\jftree` group so that they can be used in multiple trees. Node definitions are usually made local to a particular tree. The tree above could be drawn by:

```

\defbranch\medleft(15pt)(1)
\defbranch\medright(15pt)(-1)

\jftree
\!\0{A}\medleft\1\medright\2\cr
\!\2{C}\medleft\3\medright\4\cr
\!\1{B}\cr
\!\3{D}\cr
\!\4{E}\cr
\endjftree

```

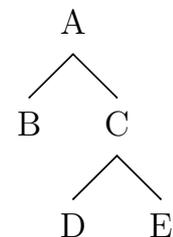


An equivalent description is allowed, which saves some keystrokes, but loses some logical clarity, since it loses the one to one correspondence between nodes and node definitions. Terminal node definitions can be eliminated.

```

\jftree
\!\0{A}\medleft{B}\medright\2\cr
\!\2{C}\medleft{D}\medright{E}\cr
\endjftree

```



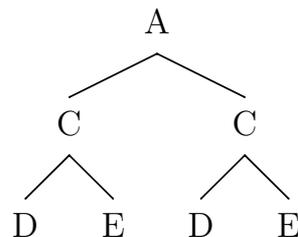
It is worth noting that the same (literally the same) node can appear in two different places in the tree!

```

\defbranch\longleft(15pt)(.5)
\defbranch\longright(15pt)(-.5)

\jftree
\!\0{A}\longleft\2\longright\2\cr
\!\2{C}\medleft{D}\medright{E}\cr
\endjftree

```



Branch definitions accept a height specification in Postscript units. This has the very big advantage that the typeset tree that a given tree specification produces can depend upon the Postscript environment dimensions. Suppose we define

```

\defbranch\medleft(1)(1)
\defbranch\medright(1)(-1)

```

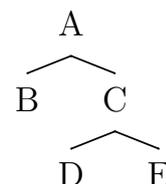
Different settings of the ps units then produce different tree pictures.

```

\psset{xunit=15pt,yunit=6pt}

\jftree
\!\0{A}\medleft{B}\medright\2\cr
\!\2{C}\medleft{D}\medright{E}\cr
\endjftree

```



```

\jftree[xunit=6pt,yunit=15pt]
\!\0{A}\medleft{B}\medright\2\cr
\!\2{C}\medleft{D}\medright{E}\cr
\endjftree

```



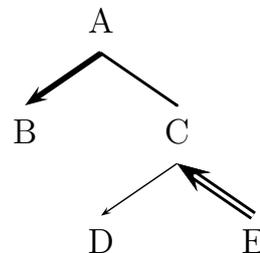
The node identifiers, $\backslash 0$, $\backslash 1$, $\backslash 2$, $\backslash 3$, and $\backslash 4$ which are used above are arbitrary. The first line after $\backslash jftree$ is assumed to be the definition of the root node. The other node definitions can occur in any order and there is no requirement that they must be used. Using $\backslash 0$, $\backslash 1$, \dots , $\backslash 9$, $\backslash A$, $\backslash B$, and so on (in order) makes it much easy to keep track of things. The structure $\backslash jftree \dots \backslash endjftree$ creates a Tex group, so definitions of the node identifiers will not spread outside their use in tree construction. The following is perfectly fine, if your style tends toward the verbose.

Parameters

`\jftree`, `\defbranch`, and uses of branches inside node definitions all take parameters in the usual PSTricks fashion.

```
\defbranch[linewidth=2pt,arrows=->]\thickmedleft(1)(1)
```

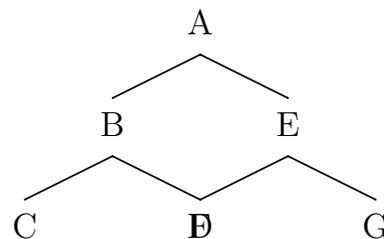
```
{\psset{linewidth=5pt}
\jftree[xunit=26pt,yunit=18pt,linewidth=1pt]
\!0{A}\thickmedleft{B}\medright\2\cr
\!2{C}\thickmedleft[linewidth=.5pt]{D}
\medright[arrows=<-,doubleline=true]{E}\cr
\endjftree}
```



Parameter settings attached to `\jftree` take precedence over parameter settings in the environment (i.e. those set by `\psset` outside the tree). Those attached to a branch definition take precedence over those attached to the `\jftree` in which the branch is used. The highest precedence is taken by parameter settings attached to a particular use of a branch in a node definition. It is worth making sure that you know why the code above produces the tree that it does.

Changes in `\unit`, `\xunit`, and `\yunit` do not apply to branch definitions and the uses of branches inside node definitions. But the parameter `scaleby` is provided. The code below is obviously unsuccessful.

```
\jftree[xunit=30pt,yunit=15pt]
\!0{A}\medleft\1\medright\2\cr
\!1{B}\medleft{C}\medright{D}\cr
\!2{E}\medleft{F}\medright{G}\cr
\endjftree
```

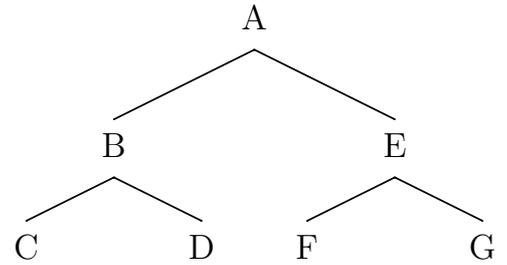


Use of `\scaleby` fixes the problem.

```

\jftree[xunit=30pt,yunit=15pt]
\!\0{A}\medleft[scaleby=1.6]
  \1\medright[scaleby=1.6]\2\cr
\!\1{B}\medleft{C}\medright{D}\cr
\!\2{E}\medleft{F}\medright{G}\cr
\endjftree

```



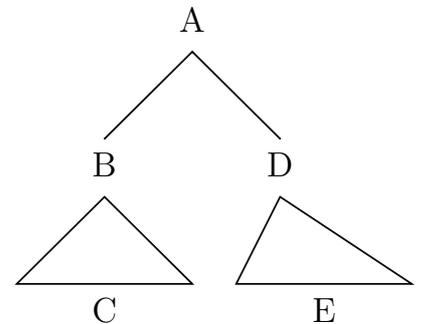
Triangles and vartriangles

Along with *branches*, there are two kinds of *triangles* available (via `\deftriangle` and `\defvartriangle`). The syntax of ordinary triangles is more or less like that of branches, with the downstairs node centered (as the default).

```

\deftriangle\medtri(1)(1)(-1)
\deftriangle\medoffsettri(1)(2)(-2/3)
\jftree[unit=30pt]
\!\0{A}\medleft\1\medright\3\cr
\!\1{B}\medtri{C}\cr
\!\3{D}\medoffsettri{E}\cr
\endjftree

```

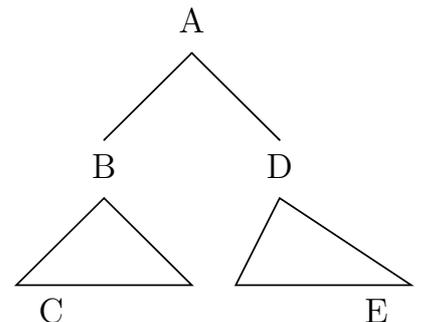


The horizontal position of the node at the base of an ordinary triangle is determined by a PSTricks parameter `triratio` (whose default value is `.5`). So:

```

\jftree[unit=30pt]
\!\0{A}\medleft\1\medright\3\cr
\!\1{B}\medtri[triratio=.2]{C}\cr
\!\3{D}\medoffsettri[triratio=.8]{E}\cr
\endjftree

```

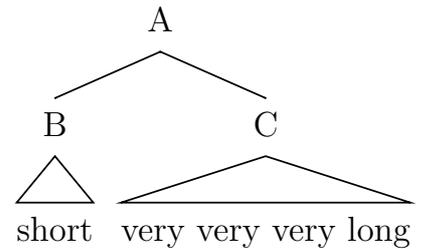


In addition to ordinary triangles, there are *vartriangles* whose base width expands to exactly match the width of the “stuff” that it covers. The syntax is restricted. Only stuff can follow a vartriangle command, not another branch or triangle command.

```

\defvartriangle\medvartri(1)
\jftree[xunit=36pt,yunit=16pt]
\!\0{A}\medleft\1\medright\3\cr
\!\1{B}\medvartri{short}\cr
\!\3{C}\medvartri{very very very long}\cr
\endjftree

```

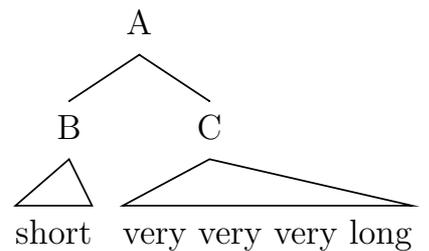


The parameter `triratio` has a different meaning for vartriangles, which always have the “stuff” centered. Its meaning can be deduced from:

```

\jftree[xunit=24pt,yunit=16pt]
\!\0{A}\medleft\1\medright\3\cr
\!\1{B}\medvartri[triratio=.7]{short}\cr
\!\3{C}\medvartri[triratio=.3]
{very very very long}\cr
\endjftree

```

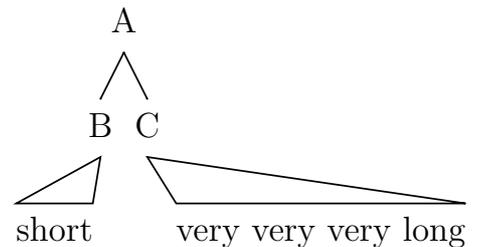


The parameter `triratio` can be negative or larger than 1.

```

\jftree[xunit=8pt,yunit=16pt]
\!\0{A}\medleft\1\medright\3\cr
\!\1{B}\medvartri[triratio=1.1]{short}\cr
\!\3{C}\medvartri[triratio=-.1]
{very very very long}\cr
\endjftree

```



Evaluating an ordinary triangle (not a vartriangle) defines a macro `\jfttriwd` which expands to the width of the triangle. Additionally, the following definition is in `pst-jftree.tex`.

```

\def\triline#1{\hbox to\jfttriwd{#1}}

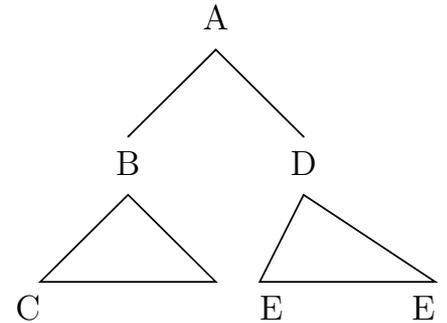
```

This can be handy for positioning material along the base of the triangle.

```

\jftree[unit=30pt]
\!\0{A}\medleft\1\medright\3\cr
\!\1{B}\medtri\2\cr
\!\2{\triline{\llap{C}\hfil}}\cr
\!\3{D}\medoffsettri\4\cr
\!\4{\triline{E\hfil E}}\cr
\endjftree

```



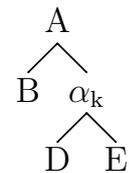
Node spacing

There are two tools for adjusting the nodespacing, the distance between the node connections and the node material. Three PSTricks parameters are defined for this purpose: `nodetopgap`, `nodebotgap`, and `everynode` (which has other functions as well). One technique is to set the gaps to 0 and to use `everynode` to insert a strut into each node. The contents of the token register `\jfteverynode` are inserted at the front of the node “stuff”. Setting the PSTricks parameter `everynode` sets the contents of `\jfteverynode`. So, for example:

```

\jftree[unit=10pt,nodetopgap=0,nodebotgap=0,
everynode=\strut]
\!\0{A}\medleft{B}\medright\2\cr
\!\2{\alpha_k}\medleft{D}\medright{E}\cr
\endjftree

```

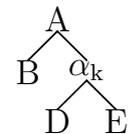


This should be compared with:

```

\jftree[unit=10pt,nodetopgap=0,nodebotgap=0,
everynode=]
\!\0{A}\medleft{B}\medright\2\cr
\!\2{\alpha_k}\medleft{D}\medright{E}\cr
\endjftree

```

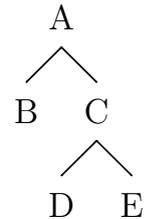


The node gaps can be expressed in ps units. Their values, however, depend on the ps units at the point of parameter setting, not at the point that nodes are typeset.

```

\jftree[unit=30pt,nodetopgap=.2,nodebotgap=.2,
  everynode={},unit=12pt]
\!\0{A}\medleft{B}\medright\2\cr
\!\2{C}\medleft{D}\medright{E}\cr
\endjftree

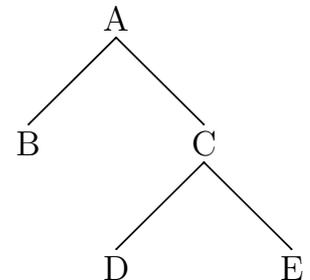
```



```

\jftree[unit=12pt,nodetopgap=.2,nodebotgap=.2,
  everynode={},unit=30pt]
\!\0{A}\medleft{B}\medright\2\cr
\!\2{C}\medleft{D}\medright{E}\cr
\endjftree

```



What seems to work best is some combination of non-zero `nodetopgap` and `nodebotgap`, with `\jfteverynode` containing a strut (which can be put there with the `everynode` parameter). The file `pst-jftree.tex` source contains

```

\psset{arrows=c-c,linewidth=.6pt,nodetopgap=3pt,
  nodebotgap=3pt,triratio=.5,everynode=\strut}

```

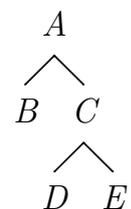
The setting `linewidth=.6pt` determines the width of the connecting lines. The setting `arrows=c-c` makes the ends of lines disk shaped, with the disk centered on the endpoint. This is good for the usual narrow connecting lines, because it makes for a smoother juncture where lines meet, but extends the lines too far if thick lines are used. It also distorts dotted lines, so `arrows=-` should be used in dotted line styles. Consult the PSTricks manual for the range of possible settings. Experimentation is necessary.

The parameter `everynode` can also be used to set the font that the nodes are typeset in. For example:

```

\jftree[unit=10pt,everynode={\it\strut}]
\!\0{A}\medleft{B}\medright\2\cr
\!\2{C}\medleft{D}\medright{E}\cr
\endjftree

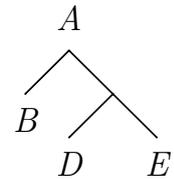
```



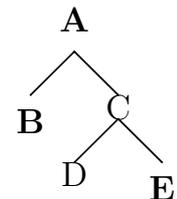
There is one other option for controlling node spacing. If the first item in the node stuff is `\omit`, the `nodegaps` are

turned off and `\everynode` is not inserted. This is crucial for trees that have unlabeled nodes. For example:

```
\jftree[unit=15pt,everynode={\it\strut}]
\!\0{A}\medleft{B}\medright\2\cr
\!\2{\omit}\medleft{D}\medright{E}\cr
\endjftree
```



```
\jftree[unit=15pt,everynode={\bf\strut}]
\!\0{A}\medleft{B}\medright\2\cr
\!\2{\omit C}\medleft{\omit D}\medright{E}\cr
\endjftree
```



Using `pst-node` in conjunction with `pst-jftree`

Labeled nodes can be placed at judiciously chosen positions and the power of `pst-node` brought to bear in constructing tree annotations. Several examples from Sandra Chung's book *The Design of Agreement* follow. First, some definitions (assuming that `pst-node.tex` has been loaded):

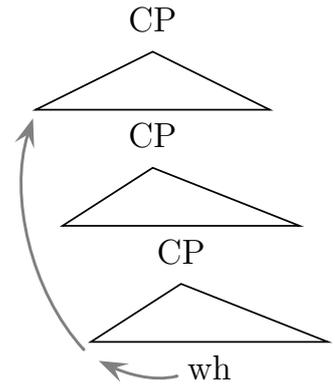
```
\deftriangle\offsettri(1)(1.3)(-.8)
\newsstyle{treearrow}{arrowlength=1.4,arrowsize=3pt 4,
arrows=cc->,linewidth=1pt,linecolor=gray}
\psset{nodesep=3pt}
\def\rpnode(#1)#2{\rput(#1){\pnode{#2}}}
```

The parameter `nodesep` is used in determining how closely node connectors approach nodes. See the `PSTricks` documentation for the details. No default is determined in `pst-jftree.tex` because it is not assumed that `pst-node.tex` has been loaded. The macro `\rpnode` is helpful in positioning `pst-node` labels.

```

\jftree[xunit=40pt,yunit=20pt]
\!\0{CP}\medtri\1\cr
\!\1{\triline
  {\rpnode(0,2.5ex){A1}\hfil CP\hfil}}\offsettri\2\cr
\!\2{CP}\offsettri\3\cr
\!\3{\triline
  {\rpnode(0,1.5ex){A2}\hfil\rnode{A3}{wh}\hfil}}\cr
\endjftree
\ncarc[style=treenarrow,arcangle=20]{A3}{A2}
\ncarc[style=treenarrow,arcangle=30]{A2}{A1}

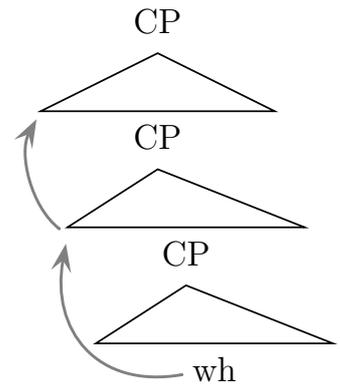
```



```

\jftree[xunit=40pt,yunit=20pt]
\!\0{CP}\medtri\1\cr
\!\1{\triline
  {\rpnode(0,2.5ex){A1}\hfil CP\hfil}}%
  \offsettri\2\cr
\!\2{\triline
  {\rpnode(0,2ex){A2}\hfil CP\hfil}}%
  \offsettri\3\cr
\!\3{\rnode{A3}{wh}}\cr
\endjftree
\ncurve[style=treenarrow,angleA=190,
  angleB=-100,ncurv=1]{A3}{A2}
\ncarc[style=treenarrow,arcangle=40]{A2}{A1}

```

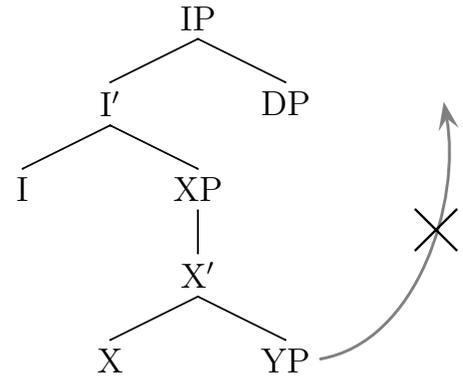


In order to exploit the full range of possibilities, it is necessary to understand pst-node's extensive ability to attach labels to arrows. Below, the "crossout" mark is placed on the arrow as a label using `\lput`. See the PSTricks documentation for the full range of label tricks, which are tricky indeed.

```

\def\crossout#1{\psset{linewidth=.8pt}
  \qline(-#1,-#1)(#1,#1)
  \qline(#1,-#1)(-#1,#1)}}%
\psset{everynode={\everymath={\rm}}}
\defbranch\medvert(1)(1/0)
\jftree[xunit=30pt,yunit=15pt]
\!0{IP}\medleft1\medright2\cr
\!1{I'}\medleft3\medright4\cr
\!2{DP}\rpnode(1.5,.5){A2}\cr
\!3{I}\cr
\!4{XP}\medvert5\cr
\!5{X'}\medleft6\medright7\cr
\!6{X}\cr
\!7{\rnode[r]{A1}{YP}}\cr
\endjftree
\ncurve[style=treetarrow,angleA=10,
  angleB=-80,ncurv=.7]{A1}{A2}
\lput{0}(.6){\crossout{1.4ex}}

```



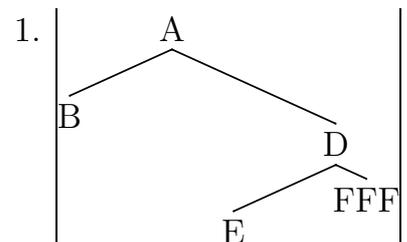
`\jftree... \endjftree`; its size and baseline

Unlike the PSTricks style, `pst-jftree` tries to keep track of the size of a tree that it constructs. Fussing with bounding boxes when you should be thinking about what goes in the boxes is a distraction. It does a fairly good job, provided that there are no arcs drawn by `pst-node` which project outside the rectangle determined by the tree connectors and nodes. The baseline is the baseline of the top node, which makes it easy to line trees diagrams up with example numbers and the like.

```

1. \hbox{\vrule\jftree[xunit=35pt,yunit=16pt]
\!0{A}\medleft{B}\medright[scaleby=1.6]\2\cr
\!1{B}\medtri\3\cr
\!2{D}\medleft{E}\medright[scaleby=.3]{FFF}\cr
\endjftree\vrule}

```

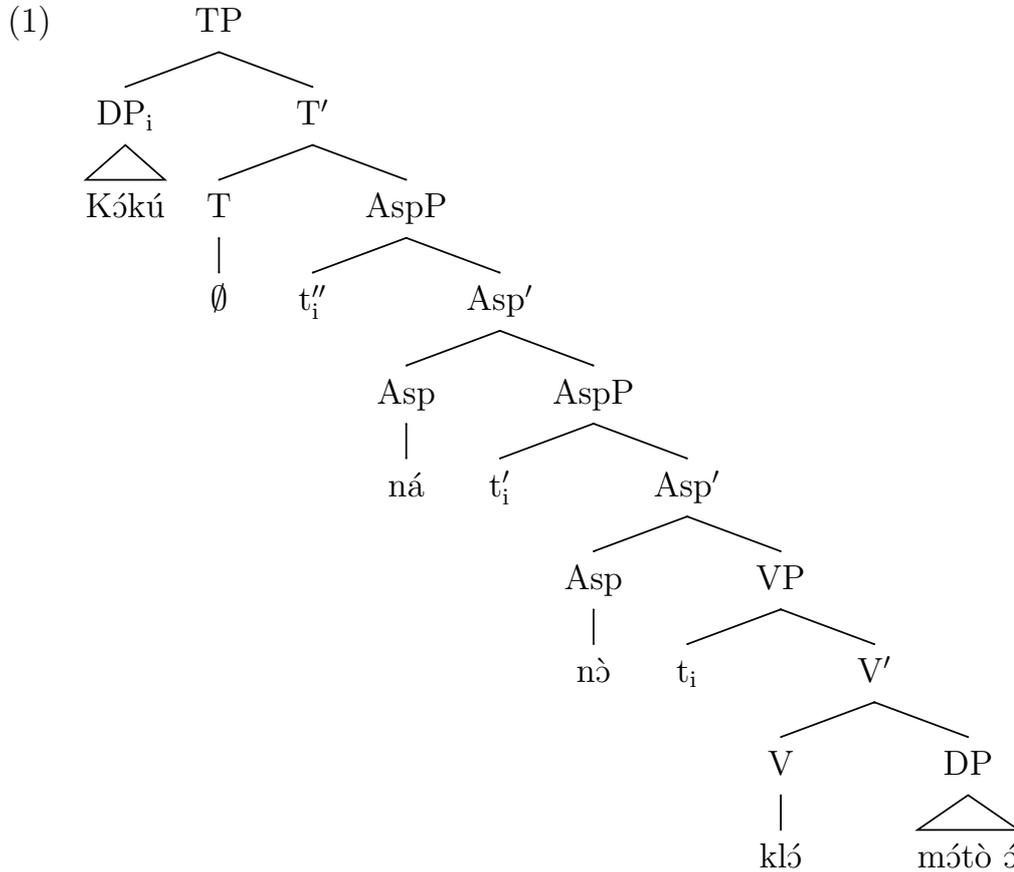


The extra space at the top and bottom stems from the inclusion of `nodetopgap` and the invisible strut in the calculation of the height and depth.

Case Studies

Several complex examples follow. My intention is to add to this over time. If you have complex examples that pose interesting problems, send them to me and they can be incorporated.

Example 1



(Ndayiragije, Juvénal. 2000. Strengthening PF, *Linguistic Inquiry* 31.3:492.)

```

(1)\quad \everymath={\rm}%
\jftree[xunit=32pt,yunit=12pt,everynode={\strut}]
\!0{TP}\medleft\1\medright\2\cr
\!1{DPi}\medvar{\hbox{K{\ipr }'0k'u}}\cr
\!2{T'}\medleft\3\medright\4\cr
\!3{T}\medvert{\emptyset}\cr
\!4{AspP}\medleft{t''i}\medright\5\cr
\!5{Asp'}\medleft\6\medright\7\cr
\!6{Asp}\medvert{n'a}\cr
\!7{AspP}\medleft{t'i}\medright\8\cr
\!8{Asp'}\medleft\9\medright\A\cr
\!9{Asp}\medvert{\ipr n'0}\cr
  
```

```

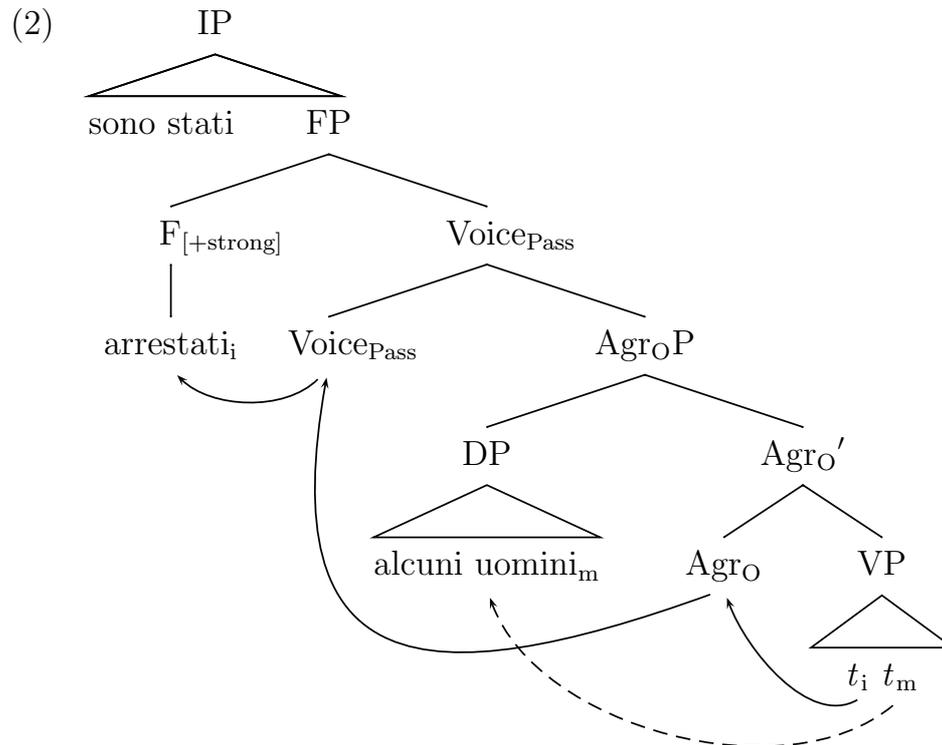
\!\A{VP}\medleft{\$t_i\$}\medright\B\cr
\!\B{\$V'\$}\medleft\C\medright\D\cr
\!\C{V}\medvert{\\ipr kl\ '0}\cr
\!\D{DP}\medvar{\\ipr m\ '0t\ 'o \ '0}\cr
\endjftree

```

In the Tex setup I use, `\ipr` above begins typesetting in IPA fonts. You will have to adapt this to whatever method you use for typesetting with IPA fonts.

Example 2

This and the remaining examples require the `pst-node` add-on to PSTricks. Tex users need `\input pst-node` and LaTeX users need `\usepackage{pst-node}`.



(Caponigro, Ivano and Carson Schütze. 2003. Parametrizing Passive Participle Movement, *Linguistic Inquiry* 34.2:300.)

```

\defbranch\steepmedright(1)(-2)
\defbranch\steepmedleft(1)(2)
\deftriangle\fixedtri(.8)(1)(-1)

```

The following macro works like `\rnode`, but it puts the node 5 pt below the baseline of the node material.

```

\def\overnode#1#2{\vtop{\offinterlineskip
  \halign{\hfil##\hfil\cr
    \vrule width0pt depth5pt #2\cr \pnode{#1}\cr}}}

(2)\quad \everymath={\rm}%
\jftree[xunit=54pt,yunit=22pt,triratio=.5,everynode={\strut}]
\!\0{IP}
  \fixedtri[triratio=.95]\1
  \fixedtri{\triline{sono stati\hfil}}\cr
\!\1{FP}\medleft\2\medright\3\cr
\!\2{\mathbf{F}_{\rlap{\mathscr{style}+strong}}}}{\$}
  \medvert{\overnode{A1}{\$arrestati_i}}\cr
\!\3{Voice}_{\rlap{\mathscr{style} Pass}}{\$}
  \medleft{\hskip1.4em\overnode{A2}{Voice}_{\$Pass}}{\$}
  \medright\6\cr
\!\6{\$Agr_0P}\medleft\7\medright\8\cr
\!\7{DP}\medvar{\overnode{A3}{alcuni uomini\$_m}}\cr
\!\8{\$Agr_0}'{\$}
  \steepmedleft{\overnode{A4}{\$Agr_0}}
  \steepmedright\9\cr
\!\9{VP}\medvar{\quad\overnode{A5}{\it t}_{\,i}}{\$}
  \quad\overnode{A6}{\it t}_{\,m}}{\quad}\cr
\endjftree
{\psset{nodesepA=0,nodesepB=3pt,arrows=->}
\ncarc[arcangle=45,nodesepA=5pt]{A2}{A1}
\ncurve[angleA=-150,angleB=-70]{A5}{A4}
\ncurve[angleA=-140,angleB=-70,nodesepA=3pt,
  linestyle=dashed]{A6}{A3}
\ncurve[angleA=-160,angleB=-100,ncurv=1.4,nodesepA=5pt]{A4}{A2}}

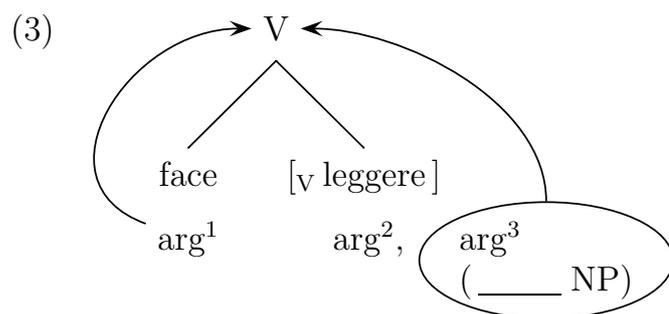
```

Note that `\fixedtri` is used twice at node `\0`, solving the problem of correctly positioning *sono stati*.

The authors go to some trouble to boldface the label of `\2`. This has not been done here, since this typesetting problem is not a tree drawing problem. Depending on the TeX setup you use, it can be hard or easy to typeset boldfaced math formulas.

The size and shape of arrows can be easily adjusted in PSTricks. Various characteristics of dashed lines can also be adjusted. The default settings were used above.

Example 3



(Zubizaretta, Maria Luisa. 1985. Morphology and Morphosyntax: Romance Causatives, *Linguistic Inquiry* 16.2:276.)

The problem in typesetting (3) is to correctly position the material inside the ellipse. The macro `\stuff` builds the appropriate box, with the same baseline as the material on its top line. It is defined inside the `\jftree... \endjftree` construction, so its definition is local. Invisible vertical branches are used to position `arg1` and `arg2`.

```
\defbranch[linestyle=none,arrows=-]\phantomvert(2pt)(1/0)
```

```
(3)\hskip3em \jftree
```

```
\def\stuff{\vtop{\halign{###$\hfil\cr
```

```
arg3\cr
```

```
(\,\underline{\hskip2.4em}\;NP)\cr}}%
```

```
\def\ovalstuff{\ovalnode
```

```
[linestyle=solid,framesep=1pt]{A1}{\stuff}}%
```

```
\!\0{\rnode{A2}{V}}\medleft\1\medright\2\cr
```

```
\!\1{face}\phantomvert\3\cr
```

```
\!\3{\rnode{A3}{arg1}}\cr
```

```
\!\2{$_V$, leggere\,$}\phantomvert\4\cr
```

```
\!\4{arg2\rlap{,}\rput[B](4.4em,0){\ovalstuff}}\cr
```

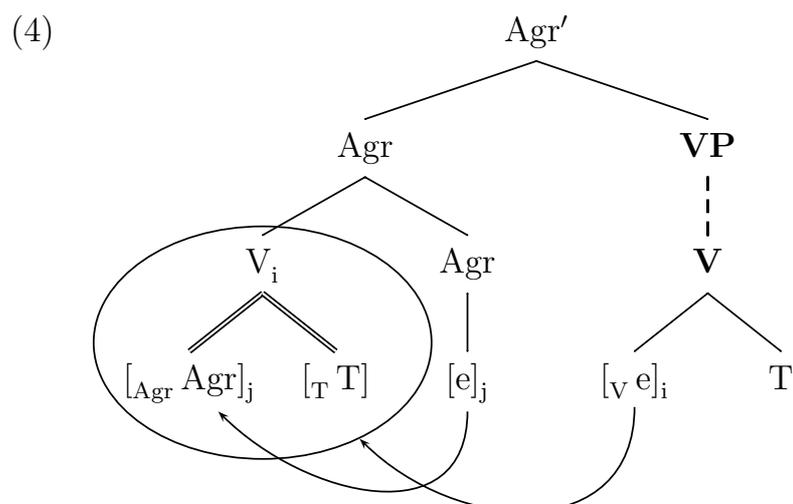
```
\endjftree
```

```
{\psset{nodesep=4pt,arrows=->,arrowscale=2}
```

```
\nccurve[nodesepA=0,angleA=90]{A1}{A2}
```

```
\nccurve[angleA=160,angleB=180,ncurv=1.1]{A3}{A2}}
```

Example 4



(Koopman, Hilda. 1995. On Verbs That Fail to Undergo V-Second, *Linguistic Inquiry* 26.1:150.)

```

\defbranch\medshallowleft(1)(.6)
\defbranch\medshallowright(1)(-.6)
\defbranch\medsteepleft(1)(1.4)
\defbranch\medsteepright(1)(-1.4)

\def\clap#1{\hbox to 0pt{\hss#1\hss}}

(4)\qqquad \jftree[xunit=35pt,yunit=20pt]
† \fontdimen16\twelvesy=3.2pt
\everymath={\rm}%
\def\oval{\ovalnode{A4}
  {\vrule height0pt depth0pt width2.2\psxunit
   \vrule height2.6\psyunit depth0pt width0pt}}%
\!0{\$Agr'\$}\medshallowleft\1\medshallowright\2\cr
\!1{\$Agr}\medleft\3\medright\4\cr
\!3{\clap{\$V_i\$}}\rput(0,-1.2){\oval}
  \medsteepleft[doubleline=true,arrows=-]\5
  \medsteepright[doubleline=true,arrows=-]\6\cr
\!5{\rnode{A1}{\$[_Agr]\,Agr]_j\$}}\cr
\!6{\$[_T\, T\$}}\cr
\!4{\$Agr}\medvert{\rnode{A2}{\$[e]_j\$}}\cr
\!2{\bf VP}\medvert[linestyle=dashed,linewidth=1pt]\7\cr
\!7{\bf V}
  \medsteepleft{\rnode{A3}{\$[_V\, e]_i\$}}
  \medsteepright{T}\cr
\endjftree

```

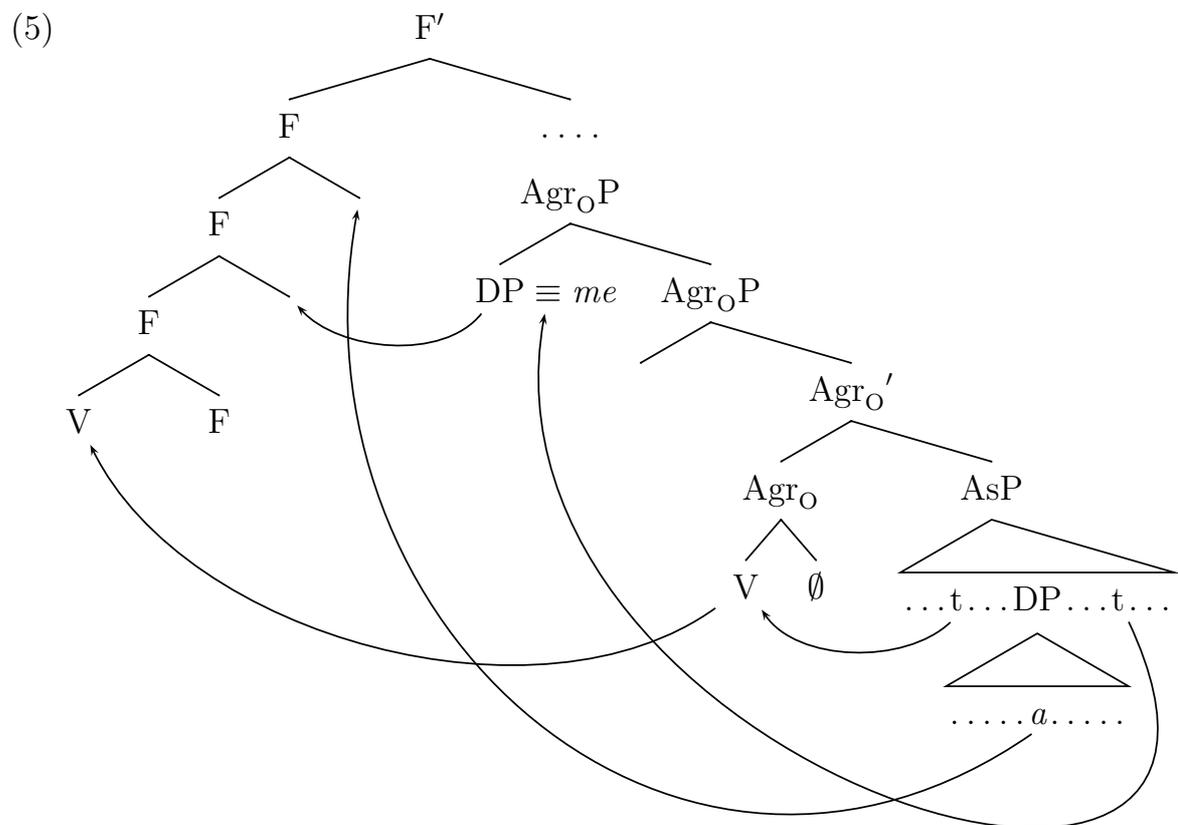
```

\ncurve[angleA=-90,angleB=-45,ncurv=1,arrows=->,nodesep=3pt]
  {A2}{A1}
\ncurve[angleA=-90,angleB=-45,ncurv=1,arrows=->,nodesepA=3pt,
  nodesepB=0]{A3}{A4}

```

Note †: Tex uses `\fontdimen16` to locate the vertical position of subscripts (in the case where there is no superscript). The setting above produces lower subscripts than usual. Category symbols set as subscripts on brackets look much better when they are a bit lower. LaTeX removes this ability to easily affect such font matters, so LaTeX users who might want to experiment with the tree above should remove that line. This is one of many reasons why many users prefer Tex without a LaTeX superstructure.

Example 5



(Uriagereka, Juan. 1995. Syntax of Clitic Placement in Western Romance, *Linguistic Inquiry* 26.1:115.)

```

\defbranch\shallowmedleft(1)(1/2)
\defbranch\shallowmedright(1)(-1/2)
\defbranch\steepmedleft(1)(2)
\defbranch\steepmedright(1)(-2)
\deftriangle[scaleby=1.3]\slantmedtri(1)(1)(-1/2)

```

```

\newpsstyle{phantom}{linestyle=none,arrows=-}

(5) \jftree[xunit=24pt,yunit=14pt]
\everymath={\rm}%
\def\*{\xleaders\hbox{\kern1pt.\kern1pt}\hfil}%
\!\0{\$F'$}\shallowmedleft\1\shallowmedright\4\cr
\!\1{F}\medleft\2\medright{\omit\node{L1}}\cr
\!\2{F}\medleft\3\medright{\omit\node{L2}}\cr
\!\3{F}\medleft{\rnode{L3}{V}}\medright{F}\cr
\!\4{\hbox to 2em{\*}}
\medvert[style=phantom,scaleby=.2]\5\cr
\!\5{\$Agr_OP$}
\medleft
{\rnode{M1}{DP}\rlap{\rnode{M2}{\$;\equiv\;$}\it me}}
\shallowmedright\BB\cr
\!\BB{\$Agr_OP$}\medleft{\}\shallowmedright\6\cr
\!\6{\$Agr_0}'$}\medleft\7\shallowmedright\8\cr
\!\7{\$Agr_0$}
\steepmedleft{\rnode{A2}{V}}
\steepmedright{\$\emptyset$}\cr
\!\8{AsP}\slantmedtri\9\cr
\!\9{\vrule depth5pt width0pt}
\triline{\*t\node{T1}\*DP*\rnode{T2}{t}\*}}
\medtri\AA\cr
\!\AA{\triline{\*{\it a}\node{A1}\*}}\cr
\endjftree
\begingroup \psset{arrows=->,nodesep=4pt}
\ncarc[arcangle=50,nodesepA=6pt]{T1}{A2}
\ncarc[arcangle=50]{M1}{L2}
\ncarc[arcangle=50,nodesepA=6pt]{A2}{L3}
\ncurve[angleA=-65,angleB=-100,ncurv=1.4]{T2}{M2}
\ncurve[angleA=-145,angleB=-100,ncurv=1,nodesepA=6pt]{A1}{L1}
\endgroup

```