

trees: a tree-drawing package for MetaPost

David Chiang

June 5, 2002

1 Basic trees

To use the `trees` package, include the following lines in your MetaPost file:

```
input boxes;
input trees;
```

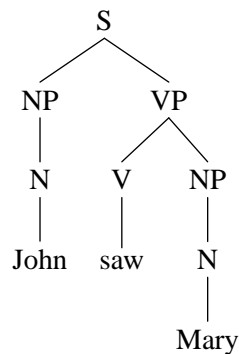
Trees are defined using the `tree` macro, which takes two text arguments: the contents of the root label, and a list of subtrees. You can write `leaf(x)` as shorthand for `tree(x)`. Both of these macros can optionally take a suffix, which becomes the name of that node. The `drawtrees` macro actually draws the trees; its argument is a list of the roots of the trees to be drawn.

```
beginfig(1);

tree.s(btex S etex)
  (tree(btex NP etex)
    (tree(btex N etex)
      (leaf(btex John etex))),
  tree(btex VP etex)
    (tree(btex V etex)
      (leaf(btex saw etex))),
    tree(btex NP etex)
      (tree(btex N etex)
        (leaf(btex Mary etex)))));

drawtrees(s);

endfig;
```



This notation is a little verbose (though not as verbose as the notation described below in Section 5), so there are two devices to simplify things a bit. First, if the `tree` macro is passed a picture instead of a subtree, the picture is automatically made into a leaf. Second, the macros `def_nonterminal` and `def_terminal` let you predefine versions of `tree` and `leaf`, respectively, with the node label built-in.

```
def_nonterminal(N, btex N etex);
def_nonterminal(NP, btex NP etex);
def_nonterminal(V, btex V etex);
```

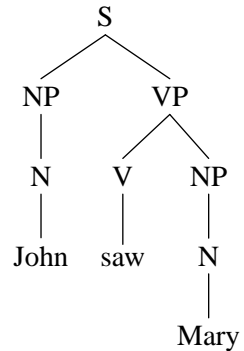
```
def_nonterminal(VP, btex VP etex);
def_nonterminal(S, btex S etex);
```

```
beginfig(2);

S.root(NP(N(btex John etex)),
      VP(V(btex saw etex),
        NP(N(btex Mary etex))));

drawtrees(root);

endfig;
```



2 Placing and connecting nodes

The `tree` macro uses the `boxit` macro to create each node, so individual nodes can be placed or connected just like boxes.

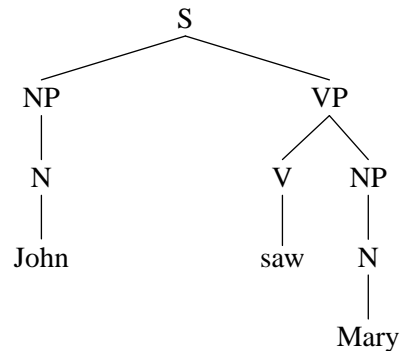
```
beginfig(3);

S.root(NP.subj(N(btex John etex)),
      VP.pred(V(btex saw etex),
        NP(N(btex Mary etex))));

subj.c + (1.5in, 0) = pred.c;

drawtrees(root);

endfig;
```



Any nodes whose positions are unspecified will be positioned automatically, which can sometimes lead to strange results.

```
def_nonterminal(V_, btex V$'$ etex);
def_nonterminal(I, btex I etex);
def_nonterminal(I_, btex I$'$ etex);
def_nonterminal(IP, btex IP etex);
```

```

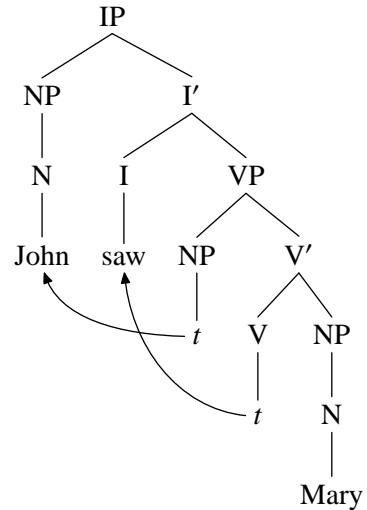
beginfig(4);

IP.root(NP(N(leaf.specip(btex John etex))),
        I_(I(leaf.i(btex saw etex)),
            VP(NP(leaf.specvp(btex $$ etex)),
                V_(V(leaf.v(btex $t$ etex)),
                    NP(N(btex Mary etex))))));

drawtrees(root);
drawarrow v.c{left}..{up}i.c
    cutbefore bpath v cutafter bpath i;
drawarrow specvp.c{left}..{up}specip.c
    cutbefore bpath specvp cutafter bpath specip;

endfig;

```



3 Placing trees

If t is the name of a tree, then the following variables specify the bounding box of the whole tree:

$t.tc$	center
$t.tn$	top center
$t.ts$	bottom center
$t.tw$	left center
$t.te$	right center
$t.tnw$	upper-left corner
$t.tne$	upper-right corner
$t.tsw$	lower-left corner
$t.tse$	lower-right corner

You can use these variables to arrange trees:

```

beginfig(5);

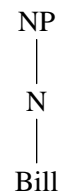
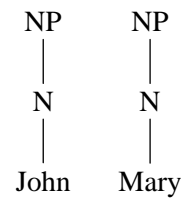
NP.t1(N(btex John etex));
NP.t2(N(btex Mary etex));
NP.t3(N(btex Bill etex));

t2.tnw - t1.tne = (12pt, 0);
0.5[t1.ts,t2.ts] - t3.tn = (0, 12pt);

drawtrees(t1,t2,t3);

endfig;

```



Note that these variables are just like their `boxit` counterparts, but with the letter `t` prefixed. (Perhaps each tree should itself be a box?)

4 Customizing trees

The `levelsep` and `treesep` internal variables control the vertical distance between the bottoms of successive levels of the tree and the horizontal distance between sister subtrees, respectively.

You can specify a different way of drawing edges by setting the `edge` attribute of a node:

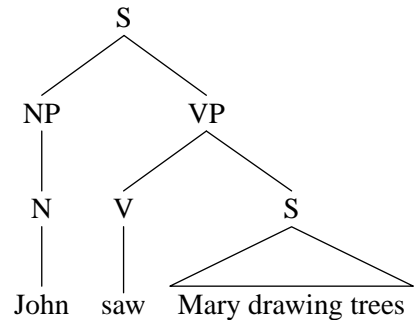
```
beginfig(6);

interim levelsep := 36pt;

S.root(NP(N(btex John etex)),
      VP(V(btex saw etex),
        S(leaf.sub(btex
                Mary drawing trees%
                etex))));

sub.edge := "drawroof";
drawtrees(root);

endfig;
```



where `drawroof` is the name of a macro that takes two suffix arguments, the names of the parent and the child.

5 The low-level interface

```
beginfig(8);  
  
leafit.John(btex John etex);  
treeit.n1(btex N etex)(John);  
treeit.np1(btex NP etex)(n1);  
  
leafit.saw(btex saw etex);  
treeit.v(btex V etex)(saw);  
  
leafit.Mary(btex Mary etex);  
treeit.n2(btex N etex)(Mary);  
treeit.np2(btex NP etex)(n2);  
  
treeit.vp(btex VP etex)(v, np2);  
  
treeit.s(btex S etex)(np1, vp);  
  
drawtrees(s);  
  
endfig;
```

