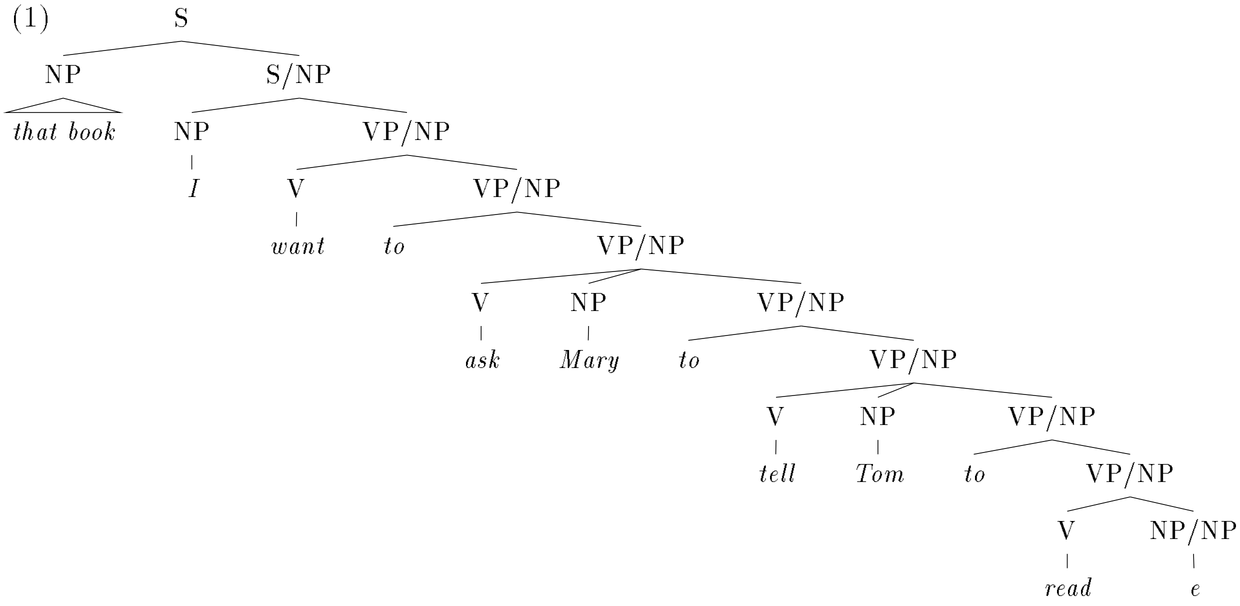
Avery Andrews'

The Trees Preprocessor

Consider the tree shown in (1) drawn from Newmeyer (1986) following GKPS:

(1)



This would require some drama to do by hand, but the `trees`' system makes it reasonably easy.

The system consists of three components: a preprocessor that converts a simple tree format into TEX input; a TEX macro file that lays out the nodes for trees, and Emma Pease's tree macro package that interfaces to Postscript and draws lines, and assorted other fancier sorts of graphics.

1 The trees preprocessor

1.1 Usage

If you give a file-name argument `trees test`, the input will be read from 'test.txp', and written to 'test_tx2.tex', the latter being what you then submit to TEX (the clumsy and complicated name is to make it easy to clean up directories without accidentally deleting source files). Errors go to 'test.err'. The resulting dvi file will be 'test_tx2.dvi': the nuisance of specifying clumy names can be reduced by putting the parts of papers requiring preprocessing into include files.

Alternatively, with no filename argument, the input is read from stdin, output written to stdout, errors written to stderr.

A sample input file is test.tex

A tree begins with `.[`, ends with `.]`, and are preprocessed for joint use with 'tree-dvips.sty' (node boxing & line drawing) and 'trees.sty' (tree layout).

Nesting is indicated by increasing depth of embedding, sisters must be indented equally, i.e.:

```
 .[
S
   NP
     John
```

```
    VP
      V
        bit
      NP
        Fido
.]
```

If a node is to be connected to its mother by a triangle, it is followed by the flag `|tri`. If it is to be given a special tag (for boxing, or whatever), the flag `|tag fred` (or whatever) is supplied.

2 Trees

1. Purpose. Trees is a preprocessor designed to work together with my 'trees.sty' (La)Tex macro package & Emma Pease's 'tree-dvips.sty' package to help with tree-formatting. It probably doesn't do anything that couldn't actually be done entirely in TeX, but for the moment I leave to others to spend their lives figuring out how.

2. Basic Operation. The file needs to invoke trees.sty & tree-dvips.sty, e.g., in LaTex, it might start out:

```
\documentstyle[tree-dvips,trees]{article}
```

(or you could have `\input{tree-dvips.sty}\input{trees.sty}` in the text).

The signal for 'trees' to kick into action is a line beginning with '.['. The tree is then presented as an indented list, with a first daughter appearing (directly) after and more deeply embedded than its mother, and subsequent sisters indented equally to the first. The tree closes with '.]' (at the beginning of the line).

E.g.:

(1)

```
.[
S
  NP
     Det
        a
     Nom
        A
          savage
        Nom
         N
           puppy
  VP
    V
       destroyed
    NP
       Det
        the
       Nom
          shoe
.]
```

What comes out of the preprocessor is input to be handled by 'trees.sty' (which does tree layout) and 'tree-dvips.sty' (which does the line-drawing).

So from (1) we get (2):

(2)

```
\tree{\ntnode{Za}{S},
  {\ntnode{Zb}{NP},
    {\ntnode{Zc}{Det},
      {\tnode{Zd}{a}}
    },
    {\ntnode{Ze}{Nom},
      {\ntnode{Zf}{A},
        {\tnode{Zg}{savage}}
      },
      {\ntnode{Zh}{Nom},
        {\ntnode{Zi}{N},
          {\tnode{Zj}{puppy}}
        }
      }
    }
  },
  {\ntnode{Zk}{VP},
    {\ntnode{Zl}{V},
      {\tnode{Zm}{destroyed}}
    },
    {\ntnode{Zn}{NP},
      {\ntnode{Zo}{Det},
        {\tnode{Zp}{the}}
      },
      {\ntnode{Zq}{Nom},
        {\tnode{Zr}{shoe}}
      }
    }
  }
}
\nodeconnect{Za}{Zb}
\nodeconnect{Za}{Zk}
\nodeconnect{Zb}{Zc}
\nodeconnect{Zb}{Ze}
\nodeconnect{Zc}{Zd}
\nodeconnect{Ze}{Zf}
\nodeconnect{Ze}{Zh}
\nodeconnect{Zf}{Zg}
\nodeconnect{Zh}{Zi}
\nodeconnect{Zi}{Zj}
\nodeconnect{Zk}{Zl}
\nodeconnect{Zk}{Zn}
\nodeconnect{Zl}{Zm}
```

```
\nodeconnect{Zn}{Zo}
\nodeconnect{Zn}{Zq}
\nodeconnect{Zo}{Zp}
\nodeconnect{Zq}{Zr}
```

The preprocessor has converted the visually convenient indented list notation into appropriate TeX grouping for the macro (defined in trees.sty), and has put the node labels into '(nonterminal) and 'tnode' (terminal commands), which feed the ommand (in 'tree-dvips.sty') as discussed briefly below. Each node has also been assigned a tag starting with 'Z', and linedrawing commands ('tree-dvips.sty') issued as appropriate.

3. Options. Options are separated from the node label by a vertical bar ('||'). There are currently two of them. The 'tri' option connecs the node to its mother with a triangle instead of a line. And the 'tag TAG' option, where TAG is any alphabetic string, sets TAG as the tag of the node. Both of these are illustrated in:

(3)

```
.[
S
  NP
     the puppy|tri
  VP|tag vp
     barked
.]
\nodecircle{vp}
```

Putting your own label on the VP node makes it easy to use tree-dvips.sty commands on it freely.

Two additional useful options are handled directly by the trees.sty definitions. First, one sometimes wants to specify the space between the mother and daughters (if the tree is rather wide, for example, the default spacing at the top might be too small). This can be done by putting a command such as `\daughterskip{4ex}` somewhere in the mother node, e.g.:

(4)

```
.[
S\daughtergap{4ex}
  NP
    the annoying child|tri
  VP
    ran away from the crazed puppy|tri
.]
```

It is also possible to specify the distance between a node and its previous sister with a command such as `\sistergap{-2em}`. This can be especially useful in packing 'wide trees' that wouldn't otherwise fit.

The default spacings for sisters & daughters can also be altered, as discussed in trees.sty.

4. `\ntnode & \tnode`. The `\ntnode` and `\tnode` commands do some moderately complicated things that are basically adapted to make it easier to write LFG papers. Both commands operate on node specifications where a bunch of annotations are specified as sitting on top of the main node label. E.g.:

```
\ntnode{a}{(\uparrow SUBJ)=\downarrow/NP}
```

Puts the annotation on top of the node-label, and automatically sets it in math-mode.. The levels are separated by a slash, and there can be as many as you like. The difference between `\ntnode` and `\tnode` is that the latter sets the last level in italics, as is the norm for terminals.

(5)

```
.[
S
  (\uparrow SUBJ)=\downarrow/NP
    \uparrow = \downarrow/N
      Mary
  \uparrow = \downarrow/VP
      yelled
.]
```

For information on altering the operations of `\ntnode & \tnode`, see trees.sty. One could also switch off all of this stuff by redefining them to in effect do nothing:

```
\def\ntnode#1#2{#2}
\def\tnode#1#2{#2}
```

2.1 Assembly

The program has so far run under Sun cc, and Turbo C 2. For Sun cc, the module ansi.c is required, to supply some library functions that appear to be missing. There are some bits of code for the VMS/VAX C version, but the adjustments would need to be made by hand. A executable and source files with a Unix makefile are provided.

2.2 Availability

Emma Pease's tree-dvips may be found in tree.tar.Z in /pub/TeXfiles on csli.stanford.edu. The preprocessor plus trees.sty may be found in pstrees.tar.Z in /pub/TMP on csli.stanford.edu