RRG trees package
Version 1.1Nov 14 2004

D.J.Gardner

Document date: Dec 14 2004

1 Introduction

This package provides a simplified interface to pstricks module pst-tree for drawing the complex syntactic trees found in RRG.

Role and Reference Grammar is a grammatical (syntactic) theory of human language that, unlike certain other theories, does not make every language look like English. It also acknowledges that certain syntactic phenomena cannot be accounted for except by reference to semantic and pragmatic considerations. As such is well suited to the analysis of languages with features like: free word order; interesting split ergative/accusative systems; pragmatic and/or semantic case assignment; switch-reference systems and minority languages in general. For further details, see e.g. "Syntax", Van Valin and LaPolla, Cambridge University Press:Cambridge 1993.

In LaTeX terms, the most challenging aspect of it is that it allows (nay, rejoices in having!) syntactic trees which have lines that cross, and also has an "operator projection" that is a distinct representation from the syntactic tree of clause, NPs, etc. While it would be possible to draw these with a graphics package, the nice features of LaTeX that might allow the tree to self-scale to the document font, and adjust its parameters depending on the font used and which words are to be displayed would be lost. Such diagrams are not easy to construct using the pst-tree (part of pstricks) macros, but are possible by carefully combining pst-tree with some pst-node commands. The flexibility of pst-tree in allowing any number of decendent nodes, unlike other tree packages I've found, is clearly important for ease of use with language data. This package aims to make RRG trees far easier to construct than would be possible using the raw commands.

The package also provides macros for displaying and typesetting other representations used in RRG, such as the **pred′**() style of elements used in logical structures.

2 Drawing Basic Trees

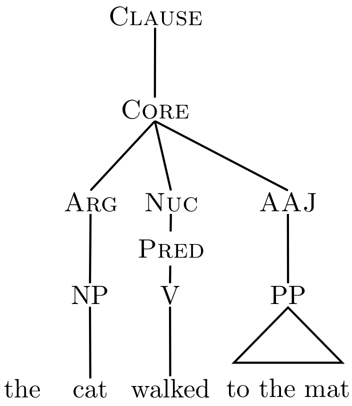
An attempt has been made to make the commands for drawing trees and tree-elements relatively simple, while allowing flexibility. If the full flexibility of the pst-node and pst-tree commands is required, there should be no problem of mixing those in, indeed some complex constructions will need to use direct pst-node and pst-tree commands.

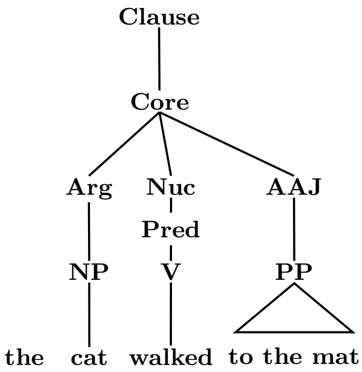
It may be helpful to think of the commands as being divided between two categories: tree-building and line drawing; or alternatively between the operator projection commands, and the syntactic structure commands. Actually these two are pretty much orthogonal, and thus we have four groups of commands, although there is considerable fuzziness at the edges (e.g. line drawing commands which also include tree building elements, and tree elements that require some line drawing).

2.1 Basic Commands

The following example produces the layered structure for 'the cat walked to the mat', but marks the final PP as unanalysed with a triangle. Normal font changing commands work and the separation between levels and nodes of the tree may be changed using the normal pst-tree settings. In the example, the left-hand version was produced by preceeding the commands with `\psset{treesep=2ex}`, the right-hand by preceeding them with `\small\bfseries \psset{treesep=2ex, levelsep=3em}`.

```
\CLAUSE{
  \CORE{
    \OPR{2}{the}
    \ARG{\WORD(NP){cat}}
    \NUC{walked}
    \AAJ{
      \FanEnd{PP}{to the mat}
    }
  }
}
```





The commands `\WORD` and `\NUC` may optionally be followed by an operator projection (see section 2.4), `\FanEnd` may not. `\NUC` also has an optional argument which allows the user to specify non-verb predicating elements, e.g. `\NUC[ADJ]{green}`. `\OPR` is used to make operators, which have no drawn link to the layered structure. It takes two required arguments — the number of levels/layers that should be skipped and the word itself — and an optional argument which is a node name. See section 2.4 for further details.

`\FanEnd` and `\CORE` type commands accept an optional argument that specifies the node name for the Core label. In the case of `\FanEnd`, this label applies to the text at the top of the triangle. Other commands that behave in a similar manner to `\CORE` are: `\SENTENCE`, `\CLAUSE`, `\PP`, `\NP`, `\NCORE`, and `\NNUC`.

By some devious trickery, `\FanEnd` may (in addition to the label) have either two or three arguments. If two arguments are given, as in the example, the the triangle or fan goes one level deep. If three arguments are given, then the first argument specifies how many levels deep the fan should be. I.e either {top}{text} or {levels}{top}{text} may be specified. If for some reason fine adjustments are needed to make the text marking the fan line up with the rest of the sentence, the macro `\rrgfantweak` may be altered from the default of `-1pt`.

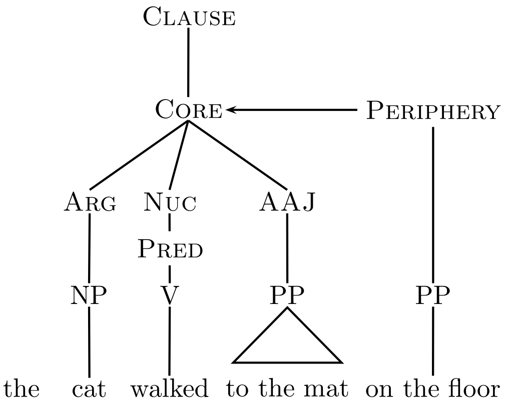
2.2 Periphery and Complementisers

A periphery element may be added using any of four commands, depending on which side (left or right) it is of the relevant core it is to modify, and whether the label 'Periphery' should be positioned at a set distance from the core or whether it should be positioned normally above its decendant nodes.

The 'fixed offset' commands are `\LPERIPH` and `\RPERIPH`, the 'right angle' commands `\lPERIPH`, `\rPERIPH`, for left and right respectively.

The right angle commands take three required arguments: the number of levels/layers to skip, the label for the upper node, and a tree. The fixed distance commands have an additional argument specifying the distance from the modified core. All periphery aelements may have an optional argument to specify the core to which they should apply. If this is used, then the relevant core should be labeled also, but with a prepended 'CORE', as in the example below. Often it is more pleasing to have the periphery decended from an upper invisible node (created by `\TOP`) than to have it distort the tree to which it logically belongs.

```
\psset{treesep=2ex}
\TOP{
  \OPR{4}{the}
  \CLAUSE{
    \CORE[COREa]{
      \ARG{\WORD(NP){cat}}
      \NUC{walked}
      \AAJ{\FanEnd{PP}{to the mat}}
    }
    \rPERIPH[a]{2}{PP}{\WORD{on the floor}}
  }
}
\dolinks
```



Complementisers (or clause linkage markers) are basicly of the same form as the periphery command, except they do not have a tree argument. On the assumption that these usually come before the linked element, there is also a short form: `\CMPL`. which is the same as `\lCMPL`. If a [label] argument is given, this should be the exact label of the core or clause node which the `\CMPL` modifies. The default is to label the item cmpl, but changing `\CMPLtext` will have immediate effect.

Note that the final command in the example is `\dolinks`. This command draws the parts of any periphery and complementisers that were not drawn by the commands themselves. It should come after all linked nodes have been defined but before the next page break.

2.3 Tree Labeling Commands

The following table gives the commands that are used in labeling the tree, with their default definitions and result. The commands may of course be used in the main text and redefined as necessary.:

`\Sentence`	`{\scshape Sentence}`	Sentence
`\Clause`	`{\scshape Clause}`	Clause
`\Core`	`{\scshape Core}`	Core
`\Conj`	`{\scshape Conj}`	Conj
`\CMPLtext`	`{\scshape cmpl}`	cmpl
`\Nuc`	`{\scshape Nuc}`	Nuc
`\Arg`	`{\scshape Arg}`	Arg
`\aaj`	`{\scshape AAJ}`	AAJ
`\pp`	`{\scshape PP}`	PP
`\np`	`{\scshape NP}`	NP
`\Periphery`	`{\scshape Periphery}`	Periphery

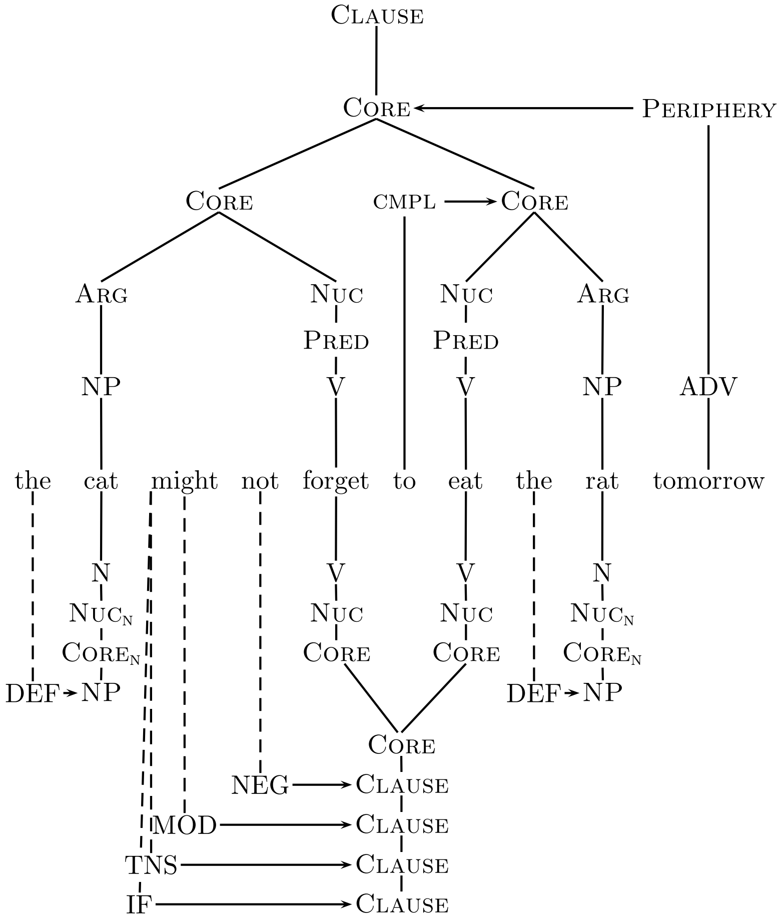
2.4 The Operator Projection

The RRG projection decends below the text line. In this package there are three main commands concerned with the operator projection: `\OP`, `\OPJoin` and `\LINK` (which also has variants `\LLINK`, `\RLINK`, `\lLINK` and `\rLINK`, corresponding to the `\CMPL` commands). `\OP` is used to draw (and label) the operator layers below verbs and nouns; `\OPJoin` is used in complex sentences to join two such stacks together and start a new stack of operator layers (positioned half-way between the two); and `\LINK` draws perpendicular links from operators to these layers, labeling them as specified. As expected, `\LLINK` and `\RLINK` place their text a specified distance from the node in the tree to which they join.

Due to the way that the pst-tree commands work, `\OPJoin` cannot be used within any tree structure, and also it has not been possible thus far to automatically add to the height of the picture in an accurate manner. Thus some calculation is needed to account for extra depth of the structure that `\OPJoin` adds. The extra height it is necessary to allow for at the bottom of the figure is ((2+operator levels in this stack) − Maximum number of operators in other stacks)×`\rrgoprsep`.

In the case that an operator should decend from only part of a word (e.g. an affix) or from a non-word position (e.g. the beginning or end of the text), the pst-node commands `\pnode` and `\Rnode`. should be used. The following example and the table of arguments in the section 3 should make things clearer.

```
\begin{center}
\psset{treesep=2ex}
\TOP{
\OPR[DEFa]{5}{the}
\CLAUSE{
  \CORE{
    \CORE[COREa]{
      \ARG{
        \NP{
          \WORD{cat}{
           \OP{N}{\OP{\Nuc\subN}{\OP{\Core\subN}{\OP[cat]{\np}}}}
          }
        }
      }
      \LINK{DEFa}{cat}{DEF}
      \OPR[might]{2}{\pnode{IF}might}
      \OPR[not]{2}{not}
      \NUC{forget}{\OP{V}{\OP{\Nuc}{\OP[forgetcore]{\Core}}}}
    }
    \CMPL[COREb]{3}{to}
    \CORE[COREb]{
      \NUC{eat}{\OP{V}{\OP{\Nuc}{\OP[eatcore]{\Core}}}}
      \OPR[DEFb]{2}{the}
      \ARG{
        \NP{
          \WORD{rat}{
           \OP{N}{\OP{\Nuc\subN}{\OP{\Core\subN}{\OP[rat]{\np}}}}
          }
          \LINK{DEFb}{rat}{DEF}
        }
      }
    }
  }
}
\rPERIPH{4}{ADV}{\WORD{tomorrow}}
}
\OPJoin{forgetcore}{eatcore}{\OP[a]{\Core}{
        \OP[b]{\Clause}{\OP[c]{\Clause}{\OP[d]{\Clause}}}}}
\LINK{might}{b}{MOD}
\LINK{not}{a}{NEG}
\LLINK{8em}{IF}{d}{IF}
\LINK{IF}{c}{TNS}
\dolinks
\vspace{6\rrgoprsep}
\end{center}
```



3 Tree-Building Commands and Arguments

In the table below, items in italic and their delimiters are optional. Where an argument is labeled “tree”, that argument may be filled with one or more of the commands below, though some obviously make more sence than others. An argument labeled “word” or “text” is anything that should appear in the tree, the only difference between the arguments, is that “text” arguments are intended to be the text of the analysed sentence, and “word” arguments label different parts of the tree structure. Arguments labeled “name” are node names, which may be used for drawing lines to and from by various commands.

Command	**Arguments**	**Summary**	**Item Named**	**Default Name**
`\TOP`	{tree}	Makes (zero sized) logical tree, but top node and lines are not drawn.		
`\SENTENCE`	[*name*]{tree}	New layer of structure	Top	SEN
`\CLAUSE`	[*name*]{tree}	New layer of structure	Top	CLS
`\CORE`	[*name*]{tree}	New layer of structure	Top	CORE
`\ARG`	[*name*]{tree}	New layer of structure	Top	arg
`\AAJ`	[*name*]{tree}	New layer of structure	Top	aaj
`\PP`	[*name*]{tree}	New layer of structure	Top	PP
`\NP`	[*name*]{tree}	New layer of structure	Top	NP
`\NCORE`	[*name*]{tree}	New layer of structure	Top	NC
`\NNUC`	[*name*]{tree}	New layer of structure	Top	NN
`\OPR`	[*name*]{num}{text}	Entry **num** levels below current node	Text	opr
`\FanEnd`	[*name*]{*num*}{word}{text}	Triangle/Fan (i.e. unanalysed) entry below current node, with the apex marked with the word argument.	Top	fan
`\End`	[*name*]{text}{*tree*}	Obsolete simpler version of `\WORD`	Text	x
`\WORD`	[*name*](*word*){text}{*tree*}	Creates one or two level entry below current node.	Text	x
`\NUC`	[*word*]{text}{*tree*}	Predicating Nucleus		
`\OP`	[*name*]{word}{*tree*}	Operator projection layer	Text	op

4 Line Drawing Commands

Command	**Arguments**	**Summary**
`\CMPL`	[*name*]{num}{text}	Also `\rCMPL`. Links to the object named (default: CLAUSE) with right-angle connections.
`\LCMPL`	[*name*]{length}{num}{text}	Also `\RCMPL`. Links to the object named (default: CLAUSE) with CMPL centered length from target.
`\lPERIPH`	[*Extn*]{num}{word}{*tree*}	Also `\rPERIPH`. Links to the object named "COREExtn" with right-angle connections.
`\LPERIPH`	[*Extn*]{length}{num}{word}{*tree*}	Also `\RPERIPH`. Links to the object named "COREExtn" with Periphery centered length from target.
`\lLINK`	[*extra*]{name}{name}{word}	Also `\rLINK`. Links two named objects with right-angle connections.
`\LLINK`	[*extra*]{length}{name}{name}{word}	Also `\RLINK`. Links two named objects with text centered length from target.

In the above table the parameter named *extra* is appended to the optional argument of a `\ncdiag` or `\ncangle` command. It may be used for forcing certain unusual drawing styles, but is best ignored. If used it must start with a comma.

5 Misc. Commands

5.1 Logical Structures

Command	**Result**	**Description**
`\pred{greet}`	**greet′**	Predicate in logical structure
`\prddo`	**do′**	Shorthand for `\pred{do}`
`\prdDO`	**DO′**	Shorthand for `\pred{DO}`
`\SEML`	SEML	Shorthand for `{\textsf{SEML}\,}`
`\INGR`	INGR	Shorthand for `{\textsf{INGR}\,}`
`\BECOME`	BECOME	Shorthand for `{\textsf{BECOME}\,}`
`\CAUSE`	CAUSE	Shorthand for `{\textsf{CAUSE}\,}`
`\NOT`	NOT	Shorthand for `{\textsf{NOT}\,}`

5.2 Other Commands and Lengths

`\textsubscript{}`	Macro	Subscript equivalent for `textsuperscript` command that is unaccountably missing from latex.ltx
`\sup{}`	Macro	Shorthand for `\textsuperscript`
`\sub{}`	Macro	Shorthand for `\textsubscript`
`\subN`	Macro	Shorthand for `\textsubscript{\scshape{n}}`, for labeling e.g. Core$_{N}$ (typed: `\Core\subN`)
`\rrgoprsep`	Length	Separation between levels on operator projection (default is 1.5em)
`\rrgfantweak`	Macro	Fine adjustment of fan (triangle) height. (default -1pt)
`\dolinks`	Macro	Necessary to actually draw and label lines for periphery and complementiser commands.

6 The code itself

```
\ProvidesPackage{rrgtrees}[2004/11/14 RRG tree drawing]
\RequirePackage{pst-node,pst-tree}
```

6.1 Logical structures

Before we start with the tree commands proper, we'll use some shortcuts for writing logical structures…

```
\newcommand{\pred}[2][]{\textbf{#2}\ensuremath{'_{#1}}}
\newcommand\SEML{\textsf{SEML}\,}
\newcommand\INGR{\textsf{INGR}\,}
\newcommand\BECOME{\textsf{BECOME}\,}
\newcommand\CAUSE{\textsf{CAUSE}\,}
\newcommand\NOT{\textsf{NOT}\,}
\newcommand\prddo{\pred{do}}
\newcommand\prdDO{\pred{DO}}
```

… and some tree entries that we want to be consistent

```
\def\Sentence{{\scshape Sentence}}
\def\Clause{{\scshape Clause}}
\def\Core{{\scshape Core}}
\def\Conj{{\scshape Conj}}
\def\CMPLtext{{\scshape cmpl}}
\def\Nuc{{\scshape Nuc}}
\def\Arg{{\scshape Arg}}
\def\aaj{{\scshape AAJ}}
\def\pp{{\scshape PP}}
\def\np{{\scshape NP}}
\def\Periphery{{\scshape Periphery}}
```

6.2 Initial setup

We'll now define a few lengths and sane defaults that will be needed later.

```
\newlength\rrgoprsep
\newlength\rrg@tmpdim
\newlength\rrg@tmpdima
\newlength\rrg@tmpdimb
\setlength{\rrgoprsep}{1.5em}
```

6.3 Generic versions of commands and internally used bits.

```
\def\rrg@empty@toks{\global\let\rrg@dolinks\@empty}
\rrg@empty@toks
```

The `\dolinks` is used to draw things that can only be done once the rest of the tree is defined. Tokens to be executed are deferred into a collecting point by many commands and the macro executes the tokens in the internal collection, and then resets the token list for the next tree.

```
\def\dolinks{\rrg@dolinks\rrg@empty@toks}
\def\@once@tmp{}
\def\rrg@addlinks{\g@addto@macro\rrg@dolinks}
\newcommand{\rrg@newnode}[1]{\@namedef{rrg@node@#1}{\null}}
```

If this had worked, it would have automatically and drawn the line if possible, or added a delay otherwise. It would have been nicer, but it didn't work. Dunno why.

```
%%Broken code:
%%\newcommand{\rrg@do@or@add@link}[3]{\@ifundefined{rrg@node@#1}{\typeout{#1
%% undefined}\rrg@addlinks{#3}}{\@ifundefined{rrg@node@#2}
%% {\typeout{#2 undefined}\rrg@addlinks{#3}}{#3}}}
```

This combines an arbitary level skip, with a right angled join, as used in periphery and clause link marker elements.

Parameters: #1 = Pre-skip; #2 = Name of top node; #3 = Name of link-to node; #4 = Text of top node; #5 = Lower tree; #6 = Mark at Right angle; #7 = Join angle at link-to node.

```
\newcommand{\rightangle@link}[7]{
\settowidth{\rrg@tmpdim}{#4}%
\skiplevels{#1}%
\rrg@newnode{#2}%
\ifdim\rrg@tmpdim>0pt
\pstree{\TR[edge=none, name=#2]{#4}}{%
#5}%
\else%
\TR[edge=none,name=#2]{#5}%
\fi%
\endskiplevels%
\rrg@addlinks{\ncangle[angleA=90,angleB=#7,armB=0]{->}{#2}{#3}%
\ncput*[npos=1]{#6}}}
```

This version produces a diagonal link for close spacing or line-crossing periphery (etc) elements.

```
\newcommand{\diag@link}[8]{
\settowidth{\rrg@tmpdim}{#4}%
\expandafter\typeout{\the\rrg@tmpdim}
\skiplevels{#1}%
\rrg@newnode{#2}%
\ifdim\rrg@tmpdim>0pt
\pstree{\TR[edge=none, name=#2]{#4}}{%
#5}%
\else%
\typeout{#4:#5}%
\TR[edge=none,name=#2]{#5}%
\fi%
\endskiplevels%
\rrg@addlinks{\ncdiagg[angleA=#7,armA=#8]{<-}{#3}{#2}%
\ncput*[npos=1]{#6}}}
```

Different versions of predicating nucleus, depending on presence of an operator projection

```
\newcommand{\@NUCa}[3]{\pstree{\TR{\@do@once\Nuc}}{
\pstree{\TR{#1}}{\pstree{\TR[name=verb]{\ncput*[npos=1.5]{\scshape{Pred}}#2}}{#3}}}}
\newcommand{\@NUCn}[2]{\pstree{\TR{\@do@once\Nuc}}{\pstree{\TR{#1}}%
{\TR[name=verb]{\ncput*[npos=1.5]{\scshape{Pred}}#2}}}}
```

6.4 User commands for periphery, etc.

```
\newcommand{\CMPL}[3][CLAUSE]{\rightangle@link{#2}{CMPL#1}{#1}{}{#3}{\CMPLtext}{180}}
\newcommand{\lCMPL}{\CMPL}
\newcommand{\rCMPL}[3][CLAUSE]{\rightangle@link{#2}{CMPL#1}{#1}{}{#3}{\CMPLtext}{0}}
\newcommand{\LCMPL}[4][CLAUSE]{\diag@link{#2}{CMPL#1}{#1}{}{#3}{\CMPLtext}{180}{#4}}
\newcommand{\RCMPL}[4][CLAUSE]{\diag@link{#2}{CMPL#1}{#1}{}{#3}{\CMPLtext}{0}{#4}}
\newcommand{\lPERIPH}[4][]{\rightangle@link{#2}{PERIPH#1}{CORE#1}%
{#3}{#4}{\Periphery}{180}}
\newcommand{\rPERIPH}[4][]{\rightangle@link{#2}{PERIPH#1}{CORE#1}%
{#3}{#4}{\Periphery}{0}}
\newcommand{\LPERIPH}[5][]{\diag@link{#2}{PERIPH#1}{CORE#1}{#3}%
{#4}{\Periphery}{180}{#5}}
\newcommand{\RPERIPH}[5][]{\diag@link{#2}{PERIPH#1}{CORE#1}{#3}%
{#4}{\Periphery}{0}[#5]}
```

6.5 Predicating Nucleus

```
\newcommand{\NUC}[2][V]{\@ifnextchar\bgroup{\@NUCa{#1}{#2}}{\@NUCn{#1}{#2}}}
```

6.6 More normal tree parts

6.6.1 Basic components

```
\newcommand{\TOP}[1]{\pstree[thislevelsep=0pt,linestyle=none]{\Tn}%
{\psset{linestyle=solid}#1}}
\newcommand{\SENTENCE}[2][SEN]{\pstree{\TR[name=#1]{\Sentence}\@do@once}{#2}}
\newcommand{\CLAUSE}[2][CLS]{\pstree{\TR[name=#1]{\Clause}\@do@once}{#2}}
\newcommand{\CORE}[2][CORE]{\pstree{\TR[name=#1]{\Core}\@do@once}{#2}}
\newcommand{\ARG}[2][arg]{\pstree{\TR[name=#1]{\Arg}\@do@once}{#2}}
\newcommand{\AAJ}[2][aaj]{\pstree{\TR[name=#1]{\aaj}\@do@once}{#2}}
\newcommand{\PP}[2][PP]{\pstree{\TR[name=#1]{\pp}\@do@once}{#2}}
\newcommand{\NP}[2][NP]{\pstree{\TR[name=#1]{\np}\@do@once}{#2}}
\newcommand{\NCORE}[2][NC]{\pstree{\TR[name=#1]{\Core\subN}\@do@once}{#2}}
\newcommand{\NNUC}[2][NN]{\pstree{\TR[name=#1]{\Nuc\subN}\@do@once}{#2}}
```

6.6.2 Word node

`\WORD` tries to be flexible in the arguments it gets. The user may optionally include the (part of speach) a `[name]` for the `{word}` node, and possibly being followed by an `{operator projection}`. E.g. `\WORD{hello}`, `\WORD[hi]{hello}`, `\WORD(N){cat}`, `\WORD(N)[pussy]{cat}` are all permitted, as is any of the above with an operator projection

```
\newcommand{\WORD}{\let\@word@tmp=\empty\@ifnextchar(%) bracket match
{\@word@ii}{\@word@i}}
\newcommand{\@word@i}[2][x]{\@end{#1}{#2}} % Nice easy single level word
\def\@word@ii(#1){\@ifnextchar[%] bracket match % we have a fuller tree.
{\@word@iii{#1}}{\@word@iii{#1}[x]}}
% Here goes!
\def\@word@iii#1[#2]#3{\@ifnextchar\bgroup% operators follow
{\@word@iv{#1}{#2}{#3}}{\pstree{\TR{#1}\@do@once}{\TR[name=#2]{#3}}}}

\newcommand{\@word@iv}[4]{\pstree{\TR{#1}\@do@once}{\pstree{\TR[name=#2]{#3}}{#4}}}
```

`\@do@once` does the delayed linestyle reset required by someone using `\UNJOINED`

```
\newcommand{\@do@once}{\@once@tmp\def\@once@tmp{}}
\newcommand{\UNJOINED}{\def\@once@tmp{\global\let\psedge\@save@edge}%
\let\@save@edge\psedge\def\psedge##1##2{}}
```

6.6.3 Operator node

`\OPR` is used for marking the supplied word as an operator. It has no representation in the upper tree but is described as though it were part of it. The optional firse parameter is its tag label (for drawing linking lines, the second defines the number of logical levels to skip, which is a function of the node placement.

```
\newcommand{\OPR}[3][opr]{\skiplevels{#2}{\TR[edge=none,name=#1]{#3}}\endskiplevels}
```

6.7 Operator tree and linking

`\OP` defines items in the operator projection.

```
\newcommand\OP[2][op]{\psset{levelsep=\rrgoprsep}\@end{#1}{\@do@once#2}}
\newcommand{\oprlink}[4]{%
\ncput*[npos=1 #1]{\rnode{label}{#4}}%
\ncline[linestyle=dashed]{#2}{label}\ncline{->}{label}{#3}%
}
\newcommand{\LLINK}[5][]{%
\ncdiagg[linestyle=none,angleA=180,armA=#2]{#4}{#3}%
\oprlink{#1}{#3}{#4}{#5}
}
\newcommand{\RLINK}[5][]{%
\ncdiagg[linestyle=none,angleA=0,armA=#2]{<-}{#4}{#3}%
\oprlink{#1}{#3}{#4}{#5}
}
\newcommand{\rLINK}[4][]{\ncangle[linestyle=none,angleA=-90,angleB=0 #1]{#2}{#3}%
\oprlink{}{#2}{#3}{#4}
}
\newcommand{\LINK}[4][]{\ncangle[linestyle=none,angleA=-90,angleB=180 #1]{#2}{#3}%
\oprlink{}{#2}{#3}{#4}
}
```

```
\newcommand{\lLINK}{\LINK}
```

The `\End` command is just a simple terminal node, which may have an operator tree following it..

```
\newcommand{\End}[2][x]{\@end{#1}{#2}}
```

First attempt at clever operators. It might even work one day.

```
%\newcommand\Ende{\@ifnextchar[%]
%{\@Ende}{\@Ende[x]}}
%
%\def\@Ende[#1]#2{\def\@End{\TR[name=#1]{#2}}%
%\typeout{@Ende}
%\@check@Ende}
%
%\def\@check@Ende{\@ifnextchar(% ] bracket matching
%{\typeout{Brkt}}{\typeout{No Brkt}}
%\@End}%
%
%\def\@check@OP{%
%\typeout{\@End}
%\@ifnextchar[% ] bracket matching
%{\@OP}{\@ifnextchar<% > bracket matching
%{\@OP[x]}{}}%
%}
%\def\@OP[#1]<#2>{\typeout{OP}\def\@Tmp{\expandafter\pstree{\TR[name=#1]{#2}}%
%{\@End}}\let\@End=\@Tmp}
```

The `\@end` command is run by all things that could be followed by optional decendents. Optional parts are enclosed in {}, just like a normal arg. The reason for this is so that the user dosn't need to define the tree in some horrible inverted order, or remember a whole different set of commands.

```
\newcommand{\@end}[2]{\@ifnextchar\bgroup% matching
{\@are@ops{#1}{#2}}{\@end@{#1}{#2}}}
\newcommand{\@end@}[2]{\TR[name=#1]{#2}}
\def\@are@ops#1#2#3{\pstree{\@end@{#1}{#2}}{#3}}
```

6.8 Macros for fans/triangles

Adjustment in position of fan label due to height of line or something similar.

```
\def\rrgfantweak{-1pt}
```

Draw a fan, but first find out how many arguments we have. I.e. we allow `\FanEnd{depth}{head}{text}` or `\FanEnd{head}{text}` plus an optional [label] for the head.

```
\newcommand{\FanEnd}[3][fan]{%
\@ifnextchar\bgroup{\@fanend{#1}{#2}{#3}}{\@@fanend{#1}{1}{#2}{#3}}}
\def\@fanend#1#2#3#4{\@@fanend{#1}{#2}{#3}{#4}}
\newcommand{\@@fanend}[4]{
\setbox\@tempboxa\hbox{#4}%
\setlength{\rrg@tmpdim}{\wd\@tempboxa}%
\addtolength{\rrg@tmpdim}{-1ex}%
\setlength{\rrg@tmpdimb}{\psk@levelsep}%
\setlength{\rrg@tmpdima}{\psk@levelsep}%
\setlength{\rrg@tmpdimb}{#2\rrg@tmpdima}%
\addtolength{\rrg@tmpdimb}{+0.7ex}%
\addtolength{\rrg@tmpdimb}{-\ht\@tempboxa}%
\addtolength{\rrg@tmpdimb}{-\pslabelsep}%
\addtolength{\rrg@tmpdimb}{\rrgfantweak}%
\pstree[thislevelsep=\rrg@tmpdimb]{\TR{#3}}{\Tfan[fansize=\rrg@tmpdim]%
\nbput*[npos=1.5,name=#1]{#4}}
}
```

6.9 Final bits and pieces

This one should have been defined in latex.ltx

```
\DeclareRobustCommand*\textsubscript[1]{%
  \@textsubscript{\selectfont#1}}
\def\@textsubscript#1{%
  {\m@th\ensuremath{_{\mbox{\fontsize\sf@size\z@#1}}}}}
```

Now I make some shortcuts, and use the `\textsubscript` command

```
\newcommand{\spr}[1]{\textsuperscript{#1}}
\newcommand{\sub}[1]{\textsubscript{#1}}
\newcommand{\subN}{\textsubscript{\scshape{n}}}
```

OPJoin is used to define a joined operator tree - see example in text. Args for `\OPJoin` are: [Offset from centre] {node} {node} {Text/structure}

```
\newcommand{\OPJoin}[4][2\rrgoprsep]{\ncline[linestyle=none]{#2}{#3}%
\nbput[labelsep=#1,npos=0.5]{\Rnode{join-#2-#3}{#4}%
}\ncline[linestyle=solid]{#2}{join-#2-#3}%
\ncdiag[armA=0,armB=0]{#3}{join-#2-#3}}
```

It could get very confusing if the user re-defines these.. curved trees, anyone?

```
\def\psedge{\ncdiag}
\psset{levelsep=3.5em,nodesep=1pt,angleA=-90,angleB=90,armA=0,armB=0}
```