xytree.sty — A LaTeX package for drawing syntactic trees (version 1.5)

Koaunghi Un

2006/8/15

Abstract

xytree.sty utilizes the XY-pic package to draw syntactic trees. There are currently some similar LaTeX packages for this purpose, but they have strength and weakness, so one allows to draw only binary trees and the other permits to draw only limited sorts of lines. Some packages provide quasi perfect solution in this area, but have the dependency on the other graphic language, which `dvi` driver can not directly process.

The aim of xytree.sty is to draw linguistic syntactic trees with ease and to support hopefully sufficient functionality, that the linguist may need.

In the version of 1.5, a functionality is added for semanticists to draw DRS (Discourse Representation Structure).

1 Introduction

xytree.sty is based on the powerful vector drawing package, XY-pic. Although Vogel (2000) takes the same approach, it's functionality is somehow limited to fulfill linguists' needs.

With parsetree.sty linguists can draw syntactic trees relatively easily, though without curved lines and arrows. Moreover a node can have only three or less daughter. See Arnold (1997).

tree-dvips.sty by Emma Pease would be the best in this contest. Indeed it provides all the functionality, which xytree.sty does. The main strength and weakness in the same time in this package is the use of `\special` command. The PostScript language inside this command draws the lines to construct the desired tree. But, the `\special` command in the `dvi` file can only be processed by `dvi` drivers, which understand PostScript language. Consequently, the output of tree-dvips.sty requires an unpleasant handling. That is, some previewers like `xdvi` can not show the lines drawn by tree-dvips.sty. Secondly, PDF file cannot be produced by `pdflatex`, because the PostScript prologue of tree-dvips.sty isn't compatible with PDF specification. In order to make a PDF file, an intermediate PS file between `dvi` and PDF is indispensable.

Such a cold comfort was due to the weakness of LaTeX in the field of graphics, which can now be supplemented by XY-pic package. See Rose (1999; Rose and Moore (1999).

2 Loading xytree.sty

In order to use xytree.sty, put the following line between `\documentclass` and `\begin{document}` as shown in (1).

(1) `\usepackage{xytree}`

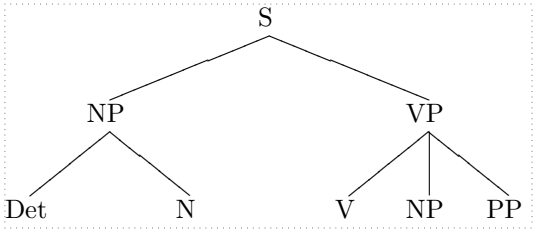
3 Usage

As an experience, it would be advisable to analyze trees as lattice structure before doing the actual task.

			S			
	NP				VP	
Det		N		V	NP	PP

The vertical dotted lines between two adjacent cells in the above diagram can be replaced by `&`'s and the rightmost lines can be replaced by `\\`. The `&`'s delimiting empty cells on the right side of a line are optional and the last `\\` can be also omitted.

The next step is to specify nonempty cells as node. The tree can be finished up by drawing lines, which connect nodes as shown below.[1]



In the following, macros for node declaration, node connection, etc. will be explained.

3.1 \xytree

A tree is drawn with a `\xytree` macro and it's general syntax looks like (2).

(2) `\xytree[col spacing]{matrix}`

`\xytree` takes an optional and a mandatory argument. The optional argument `[col spacing]` controls the horizontal distances between two adjacent nodes. The value should be positive decimal numeral in unit of `pc` (`1pc`=`12pt`). The default is `0.5`. The mandatory argument `matrix` is the actual content of a tree, consisting of nodes and delimiters.

Below, an output of `\xytree` with its source code is shown.

```
\xytree{
  \xynode{A} & \xynode{B} \\
  \xynode{C} & \xynode{D}
}
```

A	B
C	D

The row spacing can be given with `\rowheight` before `\xytree` begins. The default is `2pc`. An example with smaller `\rowheight` is shown below.

```
\rowheight=0.5pc
\xytree{
  \xynode{A} & \xynode{B} \\
  \xynode{C} & \xynode{D}
}
```

A	B
C	D

In case of wider entry in nodes like AVM, `[col spacing]` tends to grow. If a tree should have some labels on the connecting lines between nodes, `\rowheight` will be useful. For AVM, confer Pease (2000) or Manning (1993).

3.2 \xynode

A tree reaches completion, when all nodes are specified and necessary lines are drawn to connect nodes. `\xynode` declares and connects nodes inside one macro. A simple tree can be completed only with `\xynode` command. This command has the form like (3).

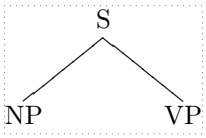
(3) `\xynode[lower nodes]{entry}`

[1] The outer dotted line denotes the range of drawn trees.

`[lower nodes]` is integer numbers delimited by comma. Each number denotes the relative positions of target nodes to be connected from the current node with straight line. The position is counted relatively, e.g. `0` is the node directly under the current node, `1` is the first node rightwards under the current node. Likewise, `-2` is the second node leftwards under the current node.

In the following tree, the node with starting line is S. `-1` connects to the node NP and `1` connects to the node VP.

```
\xytree{
  & \xynode[-1,1]{S} \\
  \xynode{NP} & & \xynode{VP}
}
```



`{entry}` is the content of the current node and in many cases it is the phrasal sign of the syntactic structure.

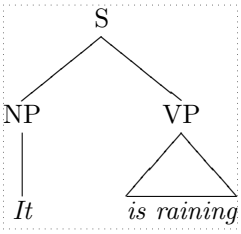
3.3 \xytrinode

A triangle instead of a line is used if the daughter node is not a sort of word. In order to draw a triangle, use `\xytrinode` like (4).

(4) `\xytrinode[width]{entry}`

An example follows:

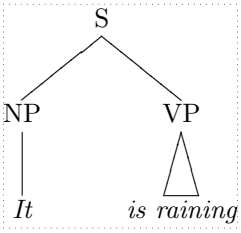
```
\xytree{
  & \xynode[-1,1]{S} \\
  \xynode[0]{NP} & & \xytrinode{VP} \\
  \xyterminal{It} & &
  \xyterminal{is raining}
}
```



As a default the base of the triangle is as wide as the width of the entry. Though, a different value can be specified with `[width]`. If it is a positive real number the triangle will have longer base, and if it is a negative real number the width will be shorter. The unit of `[width]` is mm.

The following example shows an output of a tree with shorter base length using `[5]`.

```
\xytree{
  & \xynode[-1,1]{S} \\
  \xynode[0]{NP} &
  & \xytrinode[-5]{VP}\\
  \xynode{It} & & \xynode{is raining}
}
```



The mandatory argument `{entry}` is the content of the tree node as explained in the `\xynode`.

3.4 \xyterminal

As shown in the above examples, the leaf nodes as phonological representation of the phrasal signs are written in italic shape. The command `\xyterminal` is intended to be used for such leaf nodes. This command replaces `\xynode` command and it has no optional argument for the connecting lines with the daughters.

The general syntax is as (5).

(5) `\xyterminal{entry}`

3.5 \xyconnect

For simple syntactic trees, I think, no more commands need to be added. For complicated syntactic trees, though, this line connecting macro is useful. It's general syntax is as (6).[2]

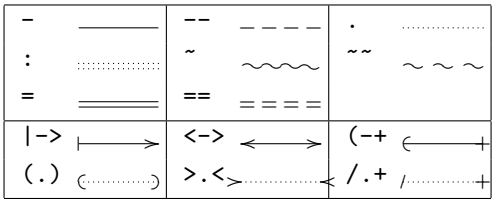
(6) `\xyconnect`
`[arrow form]`
`[bending of line]`
`(start pos, end pos)`
`{target node number}`
`"label of line"`

The fourth argument is mandatory, and all the others are optional. In other word, given the pair of {`target node number`}, a line is drawn with default values.

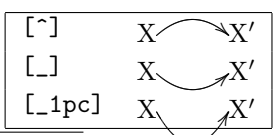
`[arrow form]` is the optional argument for specifying the form of arrow. The default value is `-`, straight line. An arrow consists of tail, shaft, and nib. The nib and tail can be chosen with `<`, `>`, `(`, `)`, `|`, `+`, `/` and the shaft with `-`, `--`, `.`, `~`, `~~`, `=`, `:`.

Without nib and tail, it is a line. But without shaft, no arrow is formed.

Examples forming the arrows with proper combination of tail, shaft, and nib are shown below. The order of the combination is `[tail shaft nib]`, and `tail` and `nib` can be omitted.

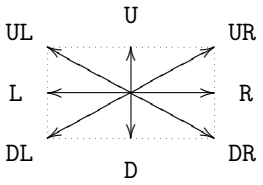


The second optional argument `[bending of line]` is used to draw curved lines.

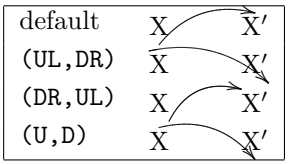


As can be noted from the above example, `^` bends lines to upward direction and `_` to downward direction. This directionality is determined by stretching a line from left to right. If the direction of a line is from right to left, the bending direction reverses. Given by some value of dimension after the direction sign, the amount of the inflection is specified by that value. `1pc` in the above example (the last one) bends the line downwards by `1pc`.

The third optional argument `(start pos, end pos)` consists of a comma-separated pair of position markers. The first marker is for the start position of the line and the second for the end position. The markers are denoted by one or two uppercase letters as shown below.



And some examples follow.

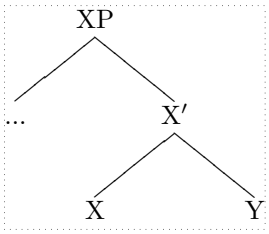


The default is (D,U).

`{target node number}` is the only mandatory argument in this command. It is a pair of integers delimited by a comma. The first integer number is responsible for the relative row number of the target node against the current node, and the second for the relative column number. If the target node is placed above the current node, the relative row number must be negative, and vice versa. Likewise, if the target node is in the left side of the current node, the column number must be negative, and vice versa. This is the same directionality as of `\xynode` command. In case of `\xynode`, the relative row number of the target node is always `1`, so only the relative column number(s) of the target node is(are) to be specified as optional argument.

[2] Because of limited paper width a long line is broken into several ones.

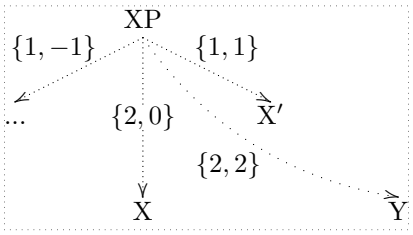
The matrix size of the tree below is 4×3, which means 4 columns and 3 rows.



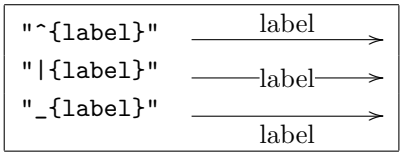
The absolute position of the node XP is represented as `"1,2"`. From this node as current node, all the directionality of `\xyconnect` can be summarized as follows.

node	abs. pos	directionality
...	`"2,1"`	`{1,-1}`
X′	`"2,3"`	`{1,1}`
X	`"3,2"`	`{2,0}`
Y	`"3,4"`	`{2,2}`

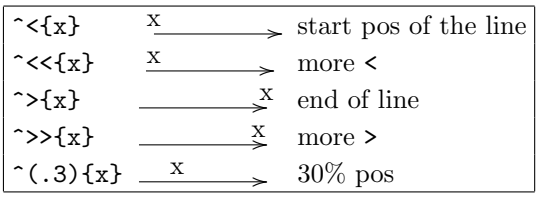
To be clearer, the diagram below will show the relative positions as labels on the connecting lines.



The last argument of `\xyconnect`, `"label of line"` is optional, which puts labels on the connecting line. Labels will be by default centered on the line. Basically, there are three kinds of label formats. As can be noticed in the above diagram, the label on the line between XP and X′ is typeset as `"^{1,1}"`, the label on the line between XP and Y as `"_{2,2}"`, and the label on the line between XP and X as `"|{2,0}"`. The directionality of the label position is the same as the one of the line bending, e.g. if a line starts from left to right, `_` puts labels below the line, `^` above the line. Because `|` breaks the line, the directionality means nothing.



To put labels in the different position on the line than in the middle, type the displacement character directly after the directionality character as follows.



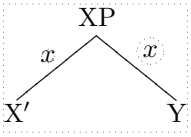
Label itself is given in the curly brace (`{}`). Modification of the label happens if the following modifier is used before `{label}` and after modifier mark `*` is specified.

`+`	increase label size
`+<dimen>`	increase `<dimen>`
`+=`	increase to frame size
`-`	decrease label size
`-<>`	decrease `<>`
`-=`	decrease to frame size
`!`	don't center
`[o]`	oval
`[l] [r] [u] [d]`	align left, right, up, down
`[F] [F=]`	single/double line frame
`[F.] [F--]`	dotted/dashed line frame
`[F-,] [F-:<3pt>]`	shaded/rounded frame (diameter=3pt) straight line

For example, to put rounded (`[o]`) and dotted frame (`[F.]`) around the label, specify the label as follows.

```
_*[o][F.]{x}          ^*+<5pt>[o][F.]{x}
```

The result is shown below. The label with size increment is on the left line and the one with 5pt increment on the right.



3.6 \xytext

For agreement representation, a diagram like (7) is used.

(7) Morpho-syntactic agreement (AGR) can be represented as Det head-noun verb ...

To retain the baseline and spacing on the both side of the text in such a diagram, `\xytext` can be used instead of `\xytree`. `\xytext` requires one mandatory argument as in (8).

(8) `\xytext{row}`

The argument `{row}` is a row of content delimited by `&`.

Above example (7) was typeset as follows.

```
Morpho-syntactic agreement (AGR) can
be represented as \xytext{
  \xybarnode{Det}\xybarconnect{1} &
  \xybarnode{head-noun} &
  \xybarnode{verb} & \xybarnode{...}}
```

The entry in a node should be specified with `\xybarnode` command and the arrows with `\xybarconnect`.

3.7 \xybarnode

As said in the previous subsection, `\xybarnode` is used for nodes in `\xytext`. The major function of this command is to retain the baseline of texts.

Comparing (9a) and (9b) makes it clearer.

(9) a. head-noun agreement (AGR)

b. head-noun agreement (AGR)

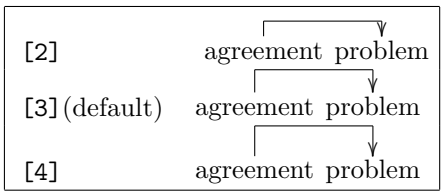
For the word "agreement" in (9a) `\xybarnode` is not used, but in (9b) it is used. In (9a) all node are aligned with all character on it's baseline. In (9b) the character `g` in the word `agreement` displaced the whole word upwards. `\xybarnode` command is necessary to get rid of such unwanted displacement.

3.8 \xybarconnect

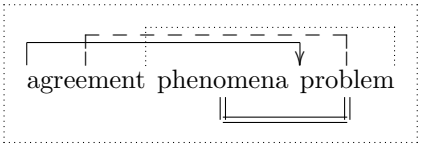
Inside `\xytext` the arrow showing agreement structure is drawn with `\xybarconnect` macro, which have the following general syntax as in (10).

(10)
```
\xybarconnect
        [spacing]
        [line art]
        (start pos, end pos)
        {target node}
        "line label"
```

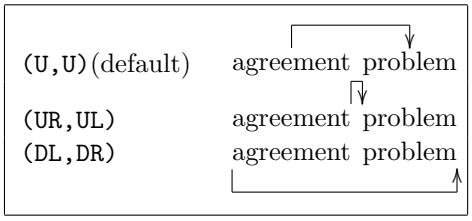
`[spacing]` is the first optional argument. It is used to set spacing between text and horizontal line. A real number specifies the spacing in unit of `pt`. The default is `3`.



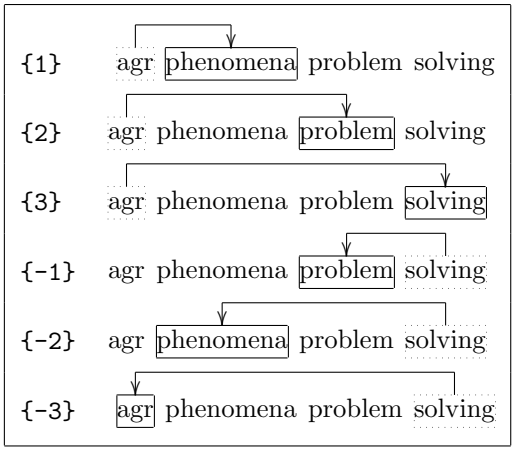
The shape of the line can be specified with the second optional argument `[line art]`. All the line shapes explained in the subsection 3.5 can be used here too. The default one is a straight line with arrow head (`[->]`).



As explained in subsection 3.5, the corner position of the source and target node can be specified with the optional argument `(start pos, end pos)`.



The only mandatory argument `{target node}` in this command is used to specify the relative position number of target node from the current node.



`"line label"`, if given, is placed in the middle of connection line. `line label` should be typeset with leading `_` or `^` and in curly brace. The directionality is the same as explained in subsection 3.5.

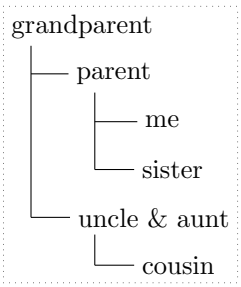
3.9 \yytree

Although not requested by linguists, there is an another tree drawing macro defined in this package. `\yytree` uses indentation and connection lines to draw a hierarchical tree in horizontal direction.

The general syntax is as (11).

(11) `\yytree[indentation]{matrix}`

The optional argument `[indentation]` is the amount of indentation for each depth of the hierarchy. The default is 2 in unit of `pc`. The mandatory argument `{matrix}` is the content of horizontal hierarchical tree. Unlikely `\xytree` only one node per a row should be defined in `\yytree`.



3.10 \yynode

`\yynode` specifies a node in `\yytree` command. The general syntax is as (12).

(12) `\yynode[target node #]{node}`

The optional argument `[target node #]` is comma-separated row numbers, to which a connection line is drawn. As usual, the number is counted relatively to the current node. E.g. `1` denotes the row below current node and `2` the second row below current node.

The tree in subsection 3.9 is typeset as follows.

```
\yytree{
  \yynode[1,4]{grandparent} \\
  & \yynode[1,2]{parent} \\
  & & \yynode{me} \\
  & & \yynode{sister} \\
  & \yynode[1]{uncle \& aunt} \\
  & & \yynode{cousin}}
```

The `[1,4]` of the node grandparent for example connects to the row parent directly under the current node and to the fifth row uncle & aunt (fourth row from the current node) with a line.

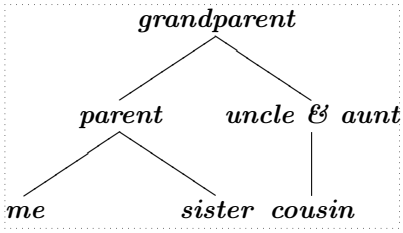
3.11 \xynodefont

This command sets the global font used in the nodes of `\xytree` and `\yytree`.

(13) `\xynodefont{font spec}`

For `font spec`, font changing commands are used. For example, the following command typesets all nodes with italic and bold face font as shown in the diagram below.

```
\xynodefont{\itshape\bfseries}
```



This command affects all the node that appear after this declaration, except the node, which are set with the command `\xyterminal`.

4 Useful macros from XY-pic

The commands from XY-pic package can be directly used with xytree.sty to draw complicated trees. In this section, some useful commands from Rose (1999) are illustrated.

4.1 Frame around node

To enclose a node or tree in a frame, `\drop \frm` after the `\xynode` as shown in (14).

(14) `\drop\frm{line art}`

`{.}`, `{-}`, `{=}`, `{--}`, `{==}`, `{o-}` can be used for `{line art}`. `{o-}` drops rounded frame, whereas others do not. Though, by specifying diameter for the corners as in (15), the others can have also rounded corners.

(15) `\drop\frm<diameter>{line art}`

If `,` follows `line art`, a shadow effect happens. For example, `{,}` draws a shadowed frame without surrounding line and `{-,}` shadowed frame with straight line. The thickness of the shadow can be set as in (16).

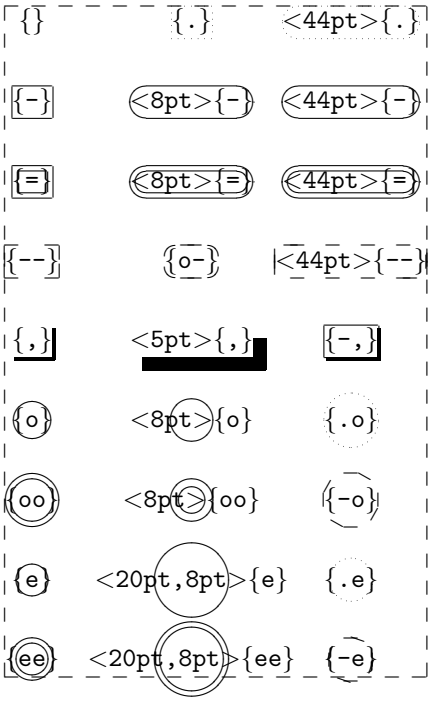
(16) `\drop\frm<thickness>{,}`

For a circled or oval frame, set `{line art}` as `{o}` or `{e}` respectively. `{oo}` or `{ee}` doubles the frame line. The general syntax is shown in (17).

(17) `\drop\frm<diameter>{shape}`

In case of circled frame, `<diameter>` sets the diameter of the circle. In case of oval frame, `<diameter>` sets two comma-separated diameters of the major and minor axis (in the order of x and y axis).

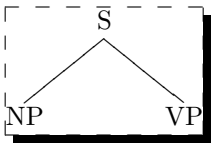
The examples for frame commands follow.



If the framing of node applies to the tree as a whole, enclose the whole tree in between `\xy` and `\endxy`.

(18)
```
\xy
  \xytree{node matrix}
  \drop\frm{shape}
\endxy
```

The following example shows the effect of dashed frame (`\drop\frm{--}`) with shadow (`\drop\frm<3pt>{,}`

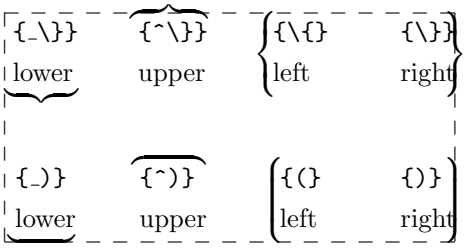


4.2 Extensible braces for nodes

The usual extensible braces in LaTeX are more optimized in XY-pic. The general syntax is as (19).

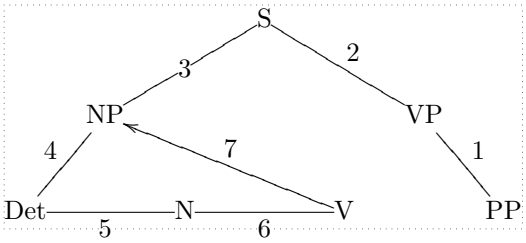
(19) `\drop\frm{brace}`

The following `{brace}` can be used.



4.3 Lines passing through nodes

After the nodes in a tree are defined, it is possible to draw a line, which passes through the nodes. The line breaks at the node positions automatically.



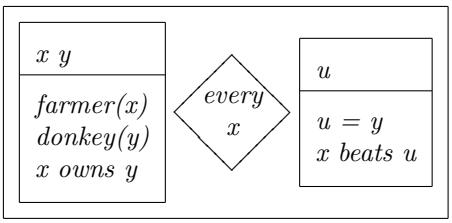
The general syntax for such construction is shown in (20).

(20) `\ar @{line art} '[mid nodes]{label}`
`... [end node]`

`{line art}`, `{label}`, and `[mid/end node(s)]` are the same as shown in the command `\xynodeconnect`. Note that `'` precedes the first mid node. All node positions are relative to the current node. In the above diagram the line, for example, starts from the lower-right node PP. The NP node, which is a mid node and end node, is typeset as `'[-1,-5]` for the mid node, whereas as `[-1,-5]` for the end node.

4.4 Discourse Representation Structure

The semanticists need to draw Discourse Representation Structures (DRS) as illustrated below (Kamp and Reyle, 1993):



The difficulty resides in drawing diamond-shaped box, which is solved with XY-pic module in this package.

The above diagram is coded as follows:

```
\drsrectbox{
  \drsbox{x y}{farmer(x)\\donkey(y)\\
              x owns y}
  \drsdiabox{every\\x}
  \drsbox{u}{u = y\\x beats u}
}
```

For DRS, following three macros are provided.

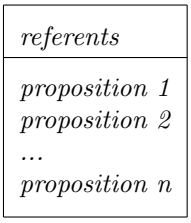
4.4.1 \drsbox

This macro draws a standard rectangular DRS box. The syntax follows:

(21) `\drsbox[line art]`
`{referent}`
`{proposition}`

The sort of lines can be given by the first optional argument `[line art]` (cf. subsection 3.5). The referents used in the propositions will be placed in the separated boxes as shown bellow:

(22)




```
\drsbox{referents}
       {proposition 1\\
        proposition 2\\
        ...\\
        proposition n}
```

Each proposition is separated by \\.

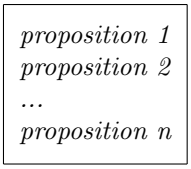
4.4.2 \drsrectbox

This macro draws a simple rectangular box consisting of only propositions without referents.

(23) `\drsrectbox[line art]{proposition}`

The following illustrates the usage:

(24)




```
\drsrectbox{proposition 1\\
            proposition 2\\
            ...\\
            proposition n}
```

4.4.3 \drsdiabox

This macro draws a diamond-shaped box used for representing conditions in DRS. The syntax is shown below:

(25) `\drsdiabox[line art]{condition}`

An example follows:

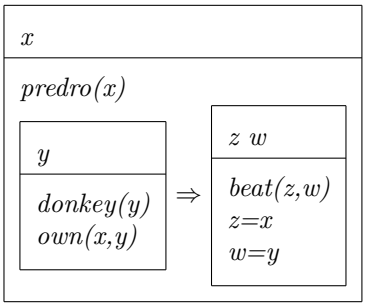
(26)


```
\drsdiabox{condition}
```

4.4.4 Nesting DRS boxes

The boxes made by above DRS macros can be nested in each other as shown in the following example:

(27)




```
\drsbox{x}{predro(x)\\
  \drsbox{y}{donkey(y)\\own(x,y)}
  $\Rightarrow$
  \drsbox{z w}{beat(z,w)\\z=x\\w=y}
```

4.5 Autosegmental Representation

Autosegmental Representation (ASR) can be drawn directly with XY-pic macros with ease[3]. In principle, no further macros need to be provided to draw ASR diagrams, but only for convenience, a macros to align the segmental nodes horizontally is defined in xytree.sty.

(28) `\asrnode{node}`

This macros is identical with `\txt{\strut }` in XY-pic module.

Here is an example using this macro in drawing an ASR diagram with XY-pic macros:

[3]Some skills in using XY-pic module directly are required to draw ASR diagrams though.

(29)


```
\xy
<0pt,0pt>*\asrnode{k}="k",
<1em,0pt>*\asrnode{a}="a",
<2em,0pt>*\asrnode{t}="t",
<0pt,1cm>*\asrnode{$\times$}="x1",
<1em,1cm>*\asrnode{$\times$}="x2",
<2em,1cm>*\asrnode{$\times$}="x3",
<.5em,2cm>*\asrnode{$\mu$}="m1",
<2em,2cm>*\asrnode{$\mu$}="m2",
"k"+U;"x1"+D**\dir{-};
"a"+U;"x2"+D**\dir{-};
"t"+U;"x3"+D**\dir{-};
"x1"+U;"m1"+D**\dir{-};
"x2"+U;"m1"+D**\dir{-};
"x3"+U;"m2"+D**\dir{-};
\endxy
```

5 Notes

5.1 Error message

If a line tries to connect to an undefined node, an error occurs. For example, the matrix below has the size of 3×4.

	(1,2)		
(2,1)		(2,3)	
	(3,2)		(3,4)

If a line tries to connect from node "(1,2)" to the node "(2,4)" with following codes

```
\xytree{
  & \xynode[2]{(1,2)} \\
  \xynode{(2,1)} & & \xynode{(2,3)} \\
  & \xynode{(3,2)} & & \xynode{(3,4)}
}
```

an error occurs as follows because of the undefined node right after the node "(2,3)" (the trailing empty nodes need not necessarily defined).

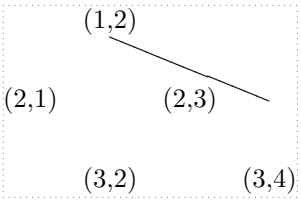
```
! Xy-pic error: in entry "1,2":
No [1,2] (is "2,4") from here.
```

This message says that an error occurred at the entry in the absolute position `"1,2"` and pointing to an entry at the relative position `[1,2]` caused an undefined entry with an absolute position `"2,4"`.

Such an error can be avoided by defining an empty entry at the absolute position `"2,4"` if the line should be appear. Remember that all empty nodes at the right-side of a non-empty nodes can be omitted. Though, if a node somewhere in the Xy-pic matrix point to such an empty node, it must not be omitted.

In many cases, analyzing the error messages is helpful to find out which of undefined target node is unintentionally accessed by a connection line. An example is shown below, where the error is solved by defining an empty node.

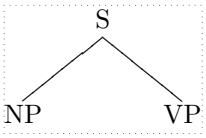
```
\xytree{
  & \xynode[2]{(1,2)} \\
  \xynode{(2,1)} & & \xynode{(2,3)} & \\
  & \xynode{(3,2)} & & \xynode{(3,4)}
}
```



5.2 Problem in left/right alignment

If used barely (without `\xy` and `\endxy`), `\xytree` and `\yytree` don't show inconsistency in alignment with the text, but if used inside `\xy` and `\endxy` or `\begin{xy}` and `\end{xy}` in order to use such command as `\drop\frm`, the tree is aligned to the left side of the text as shown below (raggedleft environment doesn't work).

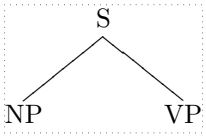
```
\begin{raggedleft}
  \xy
    \xytree{ & \xynode[-1,1]{S} \\
      \xynode{NP} & & \xynode{VP}}
    \drop\frm{.}
  \endxy
\end{raggedleft}
```



This kind of error seems to happen by an internal mechanism of Xy-pic macros, and can be avoid by setting `\hfil` or `\hfill` in front of `\xy` instead of using `center` or `raggedleft` environment.

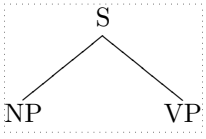
An example of using `\hfil` to align tree to the center of text line is shown below.

```
\hfil
\xy
\xytree{ & \xynode[-1,1]{S} \\
  \xynode{NP} & & \xynode{VP}}
\drop\frm{.}
\endxy
```



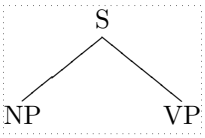
A right-justified example using `\hfill` follows.

```
\hfill
\xy
\xytree{ & \xynode[-1,1]{S} \\
  \xynode{NP} & & \xynode{VP}}
\drop\frm{.}
\endxy
```



5.3 Problem in alignment within \enumsentence

This kind of bottom-aligning problem happens, when a tall object is put inside of `\enumsentence` macro. The `\enumsentence` or `\eenumsentence` macro is provided by lingmacros.sty to make enumerated linguistic examples. The erroneous output looks like as follows. The example numbering is placed under the tree.

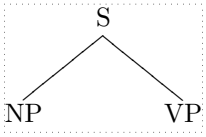


(30)

Because this kind of error occurs so often with lingmacros.sty, it provides an extra macro `\evnup`. See Pease (2000). An example of `\evnup` below aligns the tree correctly with the example number.

```
\enumsentence{\evnup{%
    \xy
    \xytree{
      & \xynode[-1,1]{S} \\
      \xynode{NP} & & \xynode{VP}}
    \drop\frm{.}
    \endxy
  }}
```

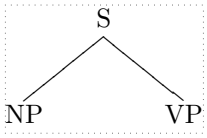
(30′)



Another solution would be using `\leavevmode` in front of `\xy` to escape from vertical mode of TEX. An example using this method follows.

```
\enumsentence{\leavevmode
  \xy
  \xytree{
    & \xynode[-1,1]{S} \\
    \xynode{NP} & & \xynode{VP}}
  \drop\frm{.}
  \endxy
}
```

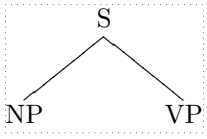
(30″)



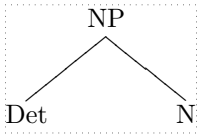
If an example has several sub-examples, `\eenumsentence` macro is used. The sub-examples are introduced by `\item` command. In this case, all sub-examples should be individually handled with `\evnup` or `leavevmode` command for proper alignment. See the following example.

```
\eenumsentence{
\item \evnup{%
    \xy
    \xytree{
      & \xynode[-1,1]{S} \\
      \xynode{NP} & & \xynode{VP}}
    \drop\frm{.}
    \endxy
  }
\item \leavevmode
  \xy
  \xytree{
    & \xynode[-1,1]{NP} \\
    \xynode{Det} & & \xynode{N}}
  \drop\frm{.}
  \endxy
}
```

(31) a.

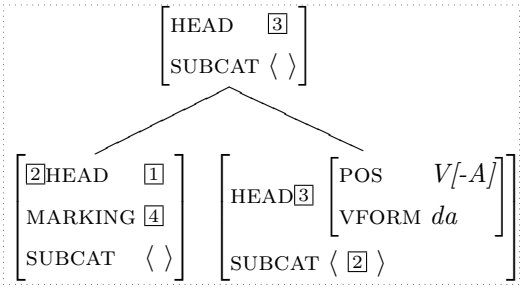


b.



6 Some examples

6.1 Tree with AVM



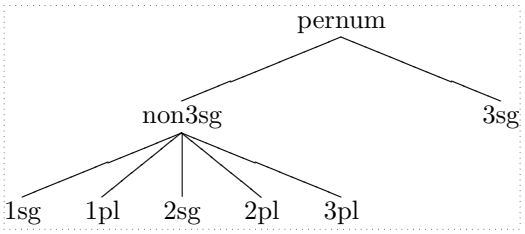

```
\xytree[2.2]{
  & \xynode[-1,1]{\begin{avm}
      \[
      head & \@3 \\
      subcat & \q< \q>
      \]
    \end{avm}} \\
  \xynode{\begin{avm}
      \[
      \@2head & \@1\\
      marking & \@4 \\
      subcat & \q< \q>
      \]
    \end{avm}} & & \xynode{\begin{avm}
      \[
      head\@3 & \[
                pos & V[-A] \\
                vform & da
                \] \\
      subcat & \q< \@2 \q>
      \]
    \end{avm}}
}
```

The overall structuring of the tree can be simplified as follows.

```
\xytree[2.2]{ & \xynode[-1,1]{AVM} \\
  \xynode{AVM} & & \xynode{AVM}
}
```

The connection lines of nodes start from the node in the first row (`[-1,1]`), `-1` connects to the first node in the second row, and `1` connects to the second node in the second row. Each node is typeset using `avm` environment provided by avm.sty.

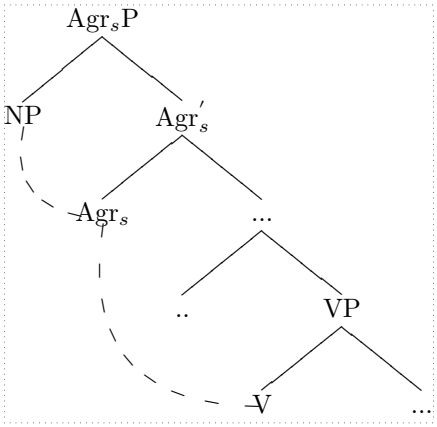
6.2 Multiple daughter node




```
\xytree{
  & & & & \xynode[-2,2]{pernum} \\
  & & \xynode[-2,-1,0,1,2]{non3sg} & & &
  & \xynode{3sg} \\
  \xynode{1sg} & \xynode{1pl}
  & \xynode{2sg} & \xynode{2pl}
  & \xynode{3pl}
}
```

At first, all node positions are fixed. pernum is placed at the fifth cell of the first row, nonsg third cell of the second row, 3sg seventh cell of the second row, and so on. The line from the node pernum connects to the second node leftwards and second node rightwards in the row below, all relatively (`[-2,2]`). The five daughter nodes of non3sg is placed from the first cell in the third row. The connection lines are coded as `[-2,-1,0,1,2]`.

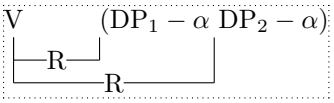
6.3 Special connection line




```
\xytree{
  & \xynode[-1,1]{Agr$_s$P} \\
  \xynode{NP}
  \xyconnect[--][_.7pc](D,L){1,1}
  & & \xynode[-1,1]{Agr$_{s}^{'}$} \\
  & \xynode{Agr$_s$}
  \xyconnect[--][_2pc](D,L){2,2}
  & & \xynode[-1,1]{...} \\
  & & \xynode{..} & & \xynode[-1,1]{VP} \\
  & & & \xynode{V} & & \xynode{...}
}
```

Specifying and connecting nodes with lines are the basic task in drawing trees. In order to draw dash lines, `\xycoonect` macro is used.
`\xyconnect[--][_.7pc](D,L){2,2}`
This is the dash line connecting node NP and Agr_s. The second optional argument is given as `[_.7pc]`, which specifies the amount of inflection of the dash line. The direction of the dash line is from left to right, so the dash line inflects downwards (`_`). Because node Agr_s and V are placed more far away than node NP and Agr_s, the amount of inflection of the dash line is given by more larger value of `[_2pc]`.

6.4 Dependency diagram




```
\xytext{
  \xybarnode{V}
  \xybarconnect[3][-](D,DL){2}"|{R}"
  \xybarconnect[6][-](D,D){2}"|{R}"
  & \xybarnode{\qquad}
  & \xybarnode{(DP$_1-\alpha$ %
                DP$_2-\alpha$)
}
```

Three nodes are specified. Second and third node are connected with line from the first node. Breaking the connection line, the labels are placed in the middle.

7 Summary

In this manual, many macros for drawing syntactic trees for linguistics are explained. A tree is constructed with a `\xytree` command. It is advisable to analyze the tree structure in a lattice structure. A simple tree could be completed using only `\xynode` commands by connecting nodes with it's optional argument. If a tree

has more or less complicated lines, `\xyconnect` command can be used.

For agreement structure, `\xytext` is used. In this case `\xybarnode` is used to specify nodes and `\xybarconnect` is used for the connection lines.

The commands provided by XY-pic package can also be used to draw very complicated trees. Some frequently encountered errors and a possible solutions are also treated in this manual.

As an extension for semanticists, `\drsbox`, `\drsrectbox`, `\drsdiabox` are added to xytree.sty for drawing DRS diagrams. One could also use XY-pic macros directly to draw ASR diagrams. In this case, `\asrnode` could be used efficiently.

With xytree.sty, it is relatively easy to draw linguistic syntactic trees. For more question concerning this package, please mail me (`mailto:koaunghi@kornet.net`).

References

[Arnold1997] Arnold, Doug, 1997. *The `parsetree` Package for Drawing Trees in LaTeX*, 10. available from CTAN site.

[Kamp and Reyle1993] Kamp, Hans and Uwe Reyle. 1993. *From Discourse to Logic*. Kluwer Academic Publishers, Dordrecht.

[Manning1993] Manning, Christopher, 1993. *`avm.sty`*, 12. available from CTAN site.

[Pease2000] Pease, Emma, 2000. *Linguistic Macros*, 12. available from CTAN site.

[Rose1999] Rose, Kristoffer H., 1999. *XY-pic User's Guide*, 2. available from CTAN site.

[Rose and Moore1999] Rose, Kristoffer H. and Ross Moore, 1999. *XY-pic Reference Manual*, 2. available from CTAN site.

[Vogel2000] Vogel, Ralf, 2000. *xyling – LaTeX macros for linguists to draw syntactic trees using xypic module*, 7. available from http://ifla.uni-stuttgart.de/~vogel/latexling.html.