The \texttt{plstx} package

Jesse A. Tov
tov@ccs.neu.edu

This document corresponds to \texttt{plstx} v0.2, dated 2011/03/26.

Contents

1 Introduction \hfill 1
\hspace*{1em} 1.1 Requirements \hfill 2

2 Command Reference \hfill 2
\hspace*{1em} 2.1 Configuration Options \hfill 4

3 Implementation \hfill 7

1 Introduction

The purpose of this package is to provide a facility for typesetting grammars for programming language syntax, like this:

\[
\begin{align*}
\alpha & \in \mathit{TVar} \quad \text{(type variables)} \\
x & \in \mathit{Var} \quad \text{(variables)} \\
\tau & ::= \alpha \mid \tau_1 \rightarrow \tau_2 \mid \forall \alpha.\tau \quad \text{(types)} \\
e & ::= x \mid e_1 e_2 \mid \lambda x:\tau.\ e \mid \Lambda\alpha.e \mid e[\tau] \quad \text{(terms)}
\end{align*}
\]

Using the \texttt{plstx} environment, I coded that like this:

\begin{verbatim}
\begin{plstx}
\(*\text{(type variables): } \alpha \in \mathit{TVar} \)
\(*\text{(variables): } x \in \mathit{Var} \)
\(*\text{(types): } \tau ::= \alpha \mid \tau_1 \rightarrow \tau_2 \mid \forall \alpha.\tau \)
\(*\text{(terms): } e ::= x \mid e_1 e_2 \mid \lambda x:\tau.\ e \mid \Lambda\alpha.e \mid e[\tau] \)
\end{plstx}
\end{verbatim}
The \texttt{plstx} environment allows redefining much of its behavior. For example, if we prefer \texttt{\rightarrow} to \texttt{::=} in our grammars, we can change the “is one of” symbol. Perhaps we also want to change the formatting for the descriptions on the right.

\begin{verbatim}
\plstxset{
  is one of=\longleftarrow,
  label style=\textsf
}
\end{verbatim}

Then we get:

\[
\begin{array}{ll}
\alpha \in T\text{Var} & \text{ (type variables)} \\
x \in \text{Var} & \text{ (variables)} \\
\tau \rightarrow \alpha \mid \tau_1 \rightarrow \tau_2 \mid \forall \alpha.\tau & \text{ (types)} \\
e \rightarrow x \mid e_1 e_2 \mid \lambda x:\tau.e \mid \Lambda \alpha.\tau \mid e[\tau] & \text{ (terms)}
\end{array}
\]

The environment also handles breaking lines when all the productions won’t fit on one line, like this:

\[
\begin{array}{ll}
\alpha \in T\text{Var} & \text{ (type variables)} \\
x \in \text{Var} & \text{ (variables)} \\
\tau ::= \alpha \mid \tau_1 \rightarrow \tau_2 \mid \forall \alpha.\tau & \text{ (types)} \\
e ::= x \mid e_1 e_2 \mid \lambda x:\tau.e \mid \Lambda \alpha.\tau \mid e[\tau] & \text{ (terms)}
\end{array}
\]

1.1 Requirements

The \texttt{plstx} package depends on three other packages. Two are a standard part of the \LaTeX{} distribution: \texttt{keyval} and \texttt{calc}. The third, \texttt{listproc}, is non-standard, and may be obtained at \url{http://www.ccs.neu.edu/~tov/code/latex/}.

2 Command Reference

\begin{enumerate}
\item \texttt{\plstxset{(plstx-options)}}
\end{enumerate}

Takes a comma-separated list of keys and values, in the style of \texttt{keyval}:

\[
\langle\text{plstx-options}\rangle ::= \langle\text{key}\rangle_1 = \langle\text{value}\rangle_1, \ldots, \langle\text{key}\rangle_k = \langle\text{value}\rangle_k, [,]
\]

The options available are described in §2.1.

\begin{enumerate}
\item \texttt{\begin{plstx} (plstx-options) \end{plstx}}
\end{enumerate}
The \texttt{plstx} environment takes an optional argument, which is a list of options as keys and values, as described in §2.1. These are the same options that may be provided to \texttt{\plstxset}.

The available commands are:

\begin{verbatim}
⟨plstx-cmd⟩ ::= ⟨label-text⟩: ⟨nonterm⟩ ::= ⟨rhs⟩ \\
  | * ⟨label-text⟩: ⟨nonterm⟩ [⟨sep⟩] ⟨rhs⟩ \\
  | \texttt{\set} {⟨plstx-options⟩}
  | \texttt{\intertext} {⟨text⟩}
  | [(dimen)]
\end{verbatim}

where

\begin{verbatim}
⟨sep⟩ ::= ⟨is-one-of⟩
  | ⟨is-one-of⟩ , ⟨continue⟩
⟨rhs⟩ ::= ⟨production⟩
  | ⟨production⟩ | ⟨rhs⟩
\end{verbatim}

If a command starts with \texttt{*}, \texttt{\set}, \texttt{\intertext}, or \texttt{[}, then it is taken to be one of those four commands—otherwise, it is treated as the first case, which handles normal nonterminal item. We’ll consider the available commands in order:

\begin{verbatim}
⟨label-text⟩: ⟨nonterm⟩ ::= ⟨rhs⟩ \\
\end{verbatim}

A normal nonterminal item consists of a label \texttt{⟨label-text⟩} (which is set on the right, in text mode by default); a non-terminal being defined \texttt{⟨nonterm⟩} (which is set on the left, in math mode by default); a separator (option \texttt{is one of}, default ::=, and written as ::= in the command even if it has been configured to appear otherwise), and a right-hand side \texttt{⟨rhs⟩}, which is a sequence of productions separated by |, each set in math mode by default. The nonterminal and label are set first, and then productions from the right-hand side are added one at a time until there’s no more space remaining, at which point it may add continuation lines.

\begin{verbatim}
* ⟨label-text⟩: ⟨nonterm⟩ [⟨sep⟩] ⟨rhs⟩ \\
\end{verbatim}

A special nonterminal item starts with \texttt{*}, after which the syntax is the same as a normal nonterminal, with one exception. Rather than write ::= for the “is one of” separator, it expects a separator for use in just that case to appear in square brackets. For example, to get \texttt{α ∈ TVar} in the example from §1, I wrote \texttt{\alpha [\in] \textit{TVar}}. Optionally, the square brackets may contain a second item, after a comma, which indicates the separator to use for continuation lines if the right-hand side wraps. Writing a special * nonterminal item with separator \texttt{[::=,\vert]} is equivalent to writing a normal nonterminal.
\set \{
\texttt{\textit{plstx-options}}\}

This allows changing the options in the middle of a grammar, using the options described in §2.1. Changes made by \set last only until the end of the current plstx environment.

\intertext \{
\texttt{\textit{text}}\}

Escapes from the normal grammar typesetting to allow including arbitrary text between grammar items. (This is similar to amsmath’s \intertext command.)

\[\]

Inserts \texttt{(dimen)} vertical space.

Note: The grammar for \texttt{(plstx-cmd)} above was written like this:

\begin{plstx} [rhs style=, one per line]
  : \meta{plstx-cmd} ::= \meta{label-text}\texttt{: \meta{nonterm} \defother{::=}
  \texttt{\slashed{\bslash}{\bslash}}
  | \defother{*} \meta{label-text}\texttt{: \meta{nonterm}
  \texttt{\slashed{\meta{sep}}\texttt{\slashed{\bslash}{\bslash}}}
  | \meta{rhs}\texttt{\texttt{\slashed{\bslash}{\bslash}}}
  | \defmacro{set} \texttt{\meta{dimen}}\texttt{\texttt{\slashed{\bslash}{\bslash}}}
  | \oarg{dimen}\texttt{\texttt{\slashed{\bslash}{\bslash}}}
  | \defother{[}\texttt{\meta{dimen}}\texttt{\texttt{\slashed{\bslash}{\bslash}}}
  | \defother{]}\texttt{\meta{dimen}}\texttt{\texttt{\slashed{\bslash}{\bslash}}}
\end{plstx}

\begin{plstx} [where]
  : \meta{sep} ::= \meta{is-one-of}
  | \meta{is-one-of} \texttt{, \meta{continue}} \texttt{\texttt{\texttt{\slashed{\bslash}{\bslash}}}
  : \meta{rhs} ::= \meta{production}
  | \meta{production} { \defother{} } \meta{rhs} \texttt{\texttt{\texttt{\texttt{\slashed{\bslash}{\bslash}}}}
\end{plstx}

2.1 Configuration Options

In this section, we document the configuration options that may be passed to \texttt{plstxset}, \texttt{set}, or environment \texttt{plstx}.

\begin{verbatim}
align continue=⟨cs⟩ default: ‘plstx@right’
continue center (boolean) default: false
continue left (boolean) default: false
continue right (boolean) default: true
\end{verbatim}

To configure the horizontal alignment of the continuation separator (see \texttt{continue}). The default is to right align it. It’s possible to specify different
alignment using one of the boolean options, or supply a command to format
the continuation separator using align continue.

align is one of = ⟨cs⟩

default: ‘\plstx@center’

is one of center (boolean) default: true

is one of left (boolean) default: false

is one of right (boolean) default: false

To configure the horizontal alignment of the “is one of” separator (see is
one of). The default is to center it.

align nonterm = ⟨cs⟩

default: ‘\plstx@center’

nonterm center (boolean) default: true

nonterm left (boolean) default: false

nonterm right (boolean) default: false

To configure the horizontal alignment of each nonterminal. The default is
to center them.

continue = ⟨text⟩

default: ‘\vert’

The “is one of” separator for continuation lines in normal grammar items.
When the right-hand side spills onto additional lines, this is used in the
separator column for each additional line. To change this for just one item,
use the * command to get a special grammar item. The value of continue
is set in math mode.

continue center (boolean) see align continue

continue left (boolean) see align continue

continue right (boolean) see align continue

gutter = ⟨dimen⟩

default: ‘4pt’

gutter left = ⟨dimen⟩

default: ‘4pt’

gutter right = ⟨dimen⟩

default: ‘4pt’

gutter left text = ⟨text⟩

default: ‘\kern4pt’

gutter right text = ⟨text⟩

default: ‘\kern4pt’

gutter text = ⟨text⟩

default: ‘\kern4pt’

These options are for specifying the gutters, which are the space to the left
and right of the “is one of” separator. The text versions of the options set
exactly what will be placed to the left or right (or both) of the separator,
whereas the non-text versions allow supplying a length to be kerned. For
example, each of these pairs is equivalent:

gutter left = ⟨dimen⟩ ≡ gutter left text = \kern ⟨dimen⟩
gutter right = ⟨dimen⟩ ≡ gutter right text = \kern ⟨dimen⟩
gutter = ⟨dimen⟩ ≡ gutter left = ⟨dimen⟩, gutter right = ⟨dimen⟩

is one of = ⟨text⟩

default: ‘::=’
The separator for normal grammar items. To change this for just one item, use the \* command to get a special grammar item. The value of is one of is set in math mode.

**is one of center** *(boolean)* see align is one of

**is one of left** *(boolean)* see align is one of

**is one of right** *(boolean)* see align is one of

**label skip=(dimen)** default: ‘1pc’

**label skip text=(text)** default: ‘\kern1pc’

This specifies the space to the left of the label, which separates the label from the right-hand side. Option label skip text takes the exact text to put to the left of (non-empty) labels, whereas label skip merely needs a length. The latter is defined in terms of the former: label skip=(dimen) \equiv label skip text=\kern(dimen).

**label style=(cs)** default: ‘\emph’

Command used to style grammar labels. Providing this key with no value sets the option to empty.

**left margin=(dimen)** see margin

**many per line** *(boolean)* default: true

Set as many right-hand side productions as will fit on each line before wrapping. This option does not take a value; the opposite option is one per line.

**margin=(dimen)** default: ‘0pt’

**left margin=(dimen)** default: ‘0pt’

**right margin=(dimen)** default: ‘0pt’

Sets the margin on one or both sides of the grammar. This margin applies only to items (normal and special), not to \intertext. If no value is supplied, the margin is set to 1em.

**nonterm center** *(boolean)* see align nonterm

**nonterm left** *(boolean)* see align nonterm

**nonterm right** *(boolean)* see align nonterm

**nonterm style=(cs)** default: ‘\ensuremath’

Commands used to style nonterminals. By default, nonterminals are set in math mode using \ensuremath. Providing this key with no value sets the option to empty.

**one per line** *(boolean)* default: false

Set only one right-hand side production on each line, regardless of space. This option does not take a value; the opposite option is many per line.
or=⟨text⟩  
*default: ‘\vert’*

Used to separate productions in a right-hand side. Set in math mode.

or skip=⟨dimen⟩  
*default: ‘4pt’*

or skip text=⟨text⟩  
*default: ‘\kern4pt’*

This specifies the space around the production separator (option or). Option or skip text takes the exact text to put on each side of the production separator, whereas or skip merely needs a length. The latter is defined in terms of the former: or skip=⟨dimen⟩ ≡ or skip text=\kern⟨dimen⟩.

rhs style=⟨cs⟩  
*default: ‘\ensuremath’*

Commands used to style each right-hand side production. By default, productions are set in math mode using \ensuremath. Providing this key with no value sets the option to empty.

right margin=⟨dimen⟩  
*see margin*

## 3 Implementation

We begin by requiring packages:

1 \RequirePackage{keyval}
2 \RequirePackage{calc}
3 \RequirePackage{listproc}

Set up the configuration options for keyval:

4 \define@key{plstx}{align continue}{\def\plstx@align@continue{#1}}
5 \define@key{plstx}{align is one of}{\def\plstx@align@isoneof{#1}}
6 \define@key{plstx}{align nonterm}{\def\plstx@align@nonterm{#1}}
7 \define@key{plstx}{continue center}{\def\plstx@align@continue{\plstx@center}}
8 \define@key{plstx}{continue left}{\def\plstx@align@continue{\plstx@left}}
9 \define@key{plstx}{continue right}{\def\plstx@align@continue{\plstx@right}}
10 \define@key{plstx}{continue}{\def\plstx@continue{#1}}
11 \define@key{plstx}{gutter}{\%}
12 \define@key{plstx}{gutter left}{\kern#1}
13 \define@key{plstx}{gutter left text}{\kern#1}
14 \define@key{plstx}{gutter right}{\kern#1}
15 \define@key{plstx}{gutter right text}{\kern#1}
16 \define@key{plstx}{gutter right text}{\kern#1}
17 \define@key{plstx}{is one of center}{\def\plstx@align@isoneof{\plstx@center}}
18 \define@key{plstx}{is one of left}{\def\plstx@align@isoneof{\plstx@left}}
19 \define@key{plstx}{is one of right}{\def\plstx@align@isoneof{\plstx@right}}
20 \define@key{plstx}{label skip text}{\def\plstx@labelskip{#1}}
21 \define@key{plstx}{label skip}{\def\plstx@labelskip{\kern#1}}
To set configuration options, we delegate to \setkeys from the keyval package.

\plstx@set
\plstxset
\newcommand*{\plstx@set}{\setkeys{plstx}}
\let\plstxset\plstx@set\relax

Set the initial options:
\plstx@set{
  continue = \vert,  
  continue right, 
  gutter = 4pt, 
  is one of = {::=}, 
  is one of center, 
  label skip = 1pc, 
  label style = \emph, 
  many per line, 
  margin = 0pt, 
  nonterm center, 
  nonterm style = \ensuremath, 
  or = \vert, 
  or skip = 4pt, 
  rhs style = \ensuremath, 
}

Helper commands for aligning text:
\plstxleft
\plstxright
\plstxcenter

The right-hand side is provided by the user delimited by |. We need to break it into productions, carefully, in order to line break it as necessary. Command \plstx@parseRHS breaks #1 into productions and stores them as a list in #2. It does this by calling \plstx@parseRHS@loop, which uses \TeX\’s argument pattern matching to find each |.
The `plstx` environment accumulates grammar items in a list, so that it can measure all of them before it chooses the widths of various parts. This macro adds an item to the accumulating list of items.

This macro is used inside the `plstx` environment to figure out which ⟨plstx-command⟩ comes next. It takes one argument, and then dispatches to the handler for the correct command. It has to deal with an additional case not mentioned in the user documentation: it detects the control sequences `\end` and `\endplstx` to detect when the environment is ending. If nothing matches, it dispatches to the normal item parser `\plstx@parseprod`.

This is the command handler for normal productions. Productions are stored in the item list as
\plstx@production{\langle label-text \rangle}{\langle nonterm \rangle}{\langle is-one-of \rangle}{\langle continue \rangle}{\langle rhs \rangle}

It then calls back to \plstx@dispatch to have it figure out the next command.
\def\plstx@parseprod#1:#2::=#3\{%
\plstx@additem{\plstx@production{#1}{#2}{\plstx@isoneof}{\plstx@continue}{#3}}%
\plstx@dispatch%
\}

\plstx@other

The command handler for special grammar items. Almost all the complexity is about figuring out whether the separator(s) in the square brackets are a single separator to use for both “is one of” and “continuation” separators, or two with a comma in between.
\def\plstx@other#1:#2[#3]#4\{%
\let\plstx@other@isoneof\plstx@isoneof
\let\plstx@other@continue\plstx@continue
\def\plstx@other@todo##1{%
\def\plstx@other@isoneof{##1}%
\def\plstx@other@continue{##1}%
\def\plstx@other@todo####1{%
\def\plstx@other@continue{####1}%
\}
\@for\plstx@each:=#3\do{%
\expandafter\plstx@other@todo\expandafter{\plstx@each}%
}\def\plstx@other@addthis##1##2{%
\plstx@additem{\plstx@production{#1}{#2}{##1}{##2}{#4}}%
\expandafter\expandafter\expandafter\plstx@other@addthis
\expandafter\expandafter\expandafter{\plstx@other@isoneof}%
\expandafter\plstx@other@continue%
\let\plstx@other@isoneof\@undefined
\let\plstx@other@continue\@undefined
\let\plstx@other@todo\@undefined
\let\plstx@other@addthis\@undefined
\plstx@dispatch%
\}
\plstx@intertext

Intertext is added to the item list as
\def\plstx@intertext#1{%
\plstx@additem{\plstx@intertext{#1}}%
\plstx@dispatch%
\}
\plstx@vskip

To add vertical space, we add

[\langle dimen \rangle] \plstx@later{\vskip(dimen)}
to the list of items.
\def\plstx@vskip#1\}{\plstx@additem{\plstx@later{\vskip#1}}\plstx@dispatch}

For a set, we add \plstx@set\{⟨plstx-options⟩\} directly to the list of grammar items.
\def\plstx@set@later#1\{\plstx@additem{\plstx@set\{#1\}}\plstx@dispatch\}

We require three boxes: box@a is used for labels, box@b for the nonterminal and productions, and box@c as a temporary box as needed.
\newsavebox\plstx@box@a
\newsavebox\plstx@box@b
\newsavebox\plstx@box@c

We use two dimension registers for calculating the maximum width of the non-terminals and the maximum width of the “is one of” and “continue” separators. The third dimension register, \plstx@availwd, is used to keep track of remaining available width when line breaking the right-hand side.
\newlength\plstx@maxnt
\newlength\plstx@maxisoneof
\newlength\plstx@availwd

The main plstx environment.
\newenvironment{plstx}[1][]{\begingroup\catcode'|=12\relax\plstx@set{#1}%%
\let\plstx@items\empty\plstx@dispatch\}\%}{\ifx\plstx@items\empty%%
\PackageWarning{plstx}{grammar must have at least one production}%%
\else%%
For both passes through the list of items, we’ll just evaluate the list, so we make \listitem a no-op.
\def\plstx@listitem@noop##1{##1\let\listitem\plstx@listitem@noop}%%
\plstx@listitem@noop\relax%%

We’re going to compute the width of the widest nonterminal and widest “is one of.” We do this by defining \plstx@production to measure each nonterminal and “is one of.” The other grammar item callbacks are defined to do nothing for now.
\setlength{\plstx@maxnt}{Opt}}
Now \texttt{plstx@maxnt} is the widest nonterminal.

For the second pass, we actually output each item. We’re going to wrap the whole thing in a \texttt{trivlist}, so we’ll precede each line with \texttt{item}. We redefine the grammar item callbacks:

\begin{verbatim}
\def\plstx@production##1##2##3##4##5{\
  \setlength{%\global\plstx@maxnt}%
  {\maxof{\plstx@maxnt}\{\widthof{\plstx@nonterm@style{##2}}\}}%\
  \setlength{%\global\plstx@maxisoneof}%
  {\maxof{\plstx@maxisoneof}}
  {\maxof{\widthof{$##3$}}}
    {\maxof{\widthof{$##4$}}}\
  }%\
\def\plstx@intertext##1{}%\
\def\plstx@later##1{}%\
  {\plstx@items}\
\end{verbatim}

Now we begin with the nonterminal. In \texttt{box@b}, we add the left margin, the non-terminal in a box of size \texttt{plstx@maxnt} (formatted and aligned according to the options), the left gutter, the “is one of” separator, and finally the right gutter.

\begin{verbatim}
\sbox\plstx@box@b{\
  \plstx@margin@left
  \makebox[\plstx@maxnt]{\plstx@align@nonterm{\plstx@nonterm@style{##2}}}%
  \plstx@gutter@left
  \makebox[\plstx@maxisoneof]{\plstx@align@isoneof{$##3$}}%
  \plstx@gutter@right
}\
\end{verbatim}

Parse the right-hand side into a list of productions. We take the first production out of the list, postpend it to \texttt{box@b}, and update the available width.

\begin{verbatim}
\plstx@parseRHS\plstx@rhsOut{##5}\
\LopTo\plstx@rhsOut\plstx@rhsFirst\
\sbox\plstx@box@b{\
  {\usebox\plstx@box@b}
}\
\end{verbatim}
Now iterate over the remaining productions.

\foreach \plstx@each := \plstx@rhs@Out \do{

Place the next production in box@c along with the production separator. If option \texttt{one per line} is set, then we don’t need to check, but otherwise, we check whether box@c will exceed the available space.

\sbox\plstx@box@c
\gdef\plstx@orskip\{}% \plstx@orskip\{}% \plstx@orskip
\plstx@one@per@line
\iftrue
\ifdim\wd\plstx@box@c > \plstx@availwd
In this case, either box@c won’t fit or we’re in one-per-line mode. So we stick box@a and box@b together and output them. Then, to start the next line, we reinitialize box@a with the right margin and box@b with the “continue” separator and the current production.

\item\makebox[\linewidth]{\strut\usebox\plstx@box@b\hfill\usebox\plstx@box@a}
\setlength{\plstx@availwd}{\linewidth}
\sbox\plstx@box@a{\plstx@margin@right}
\sbox\plstx@box@b{% \plstx@margin@left
\makebox[\plstx@maxnt]{}
\plstx@gutter@left
\makebox[\plstx@maxisoneof]{\plstx@align@continue{${##4}$}}
\plstx@gutter@right
\plstx@rhs@style{\plstx@each}%
\addtolength{\plstx@availwd}{-\wd\plstx@box@b}%
\else
\fi}

Otherwise, we add box@c to box@b and update the available width.

\addtolength{\plstx@availwd}{-\wd\plstx@box@c}%
\sbox\plstx@box@b{\usebox\plstx@box@b\usebox\plstx@box@c}
\fi
\}% end \do

When we’ve iterated through all the productions, we flush box@b if it isn’t empty:

\ifdim\wd\plstx@box@b > 0pt
\item\makebox[\linewidth]{\strut\usebox\plstx@box@b\hfill\usebox\plstx@box@a}
\fi
\}%

That’s the end of the main grammar item callback.

For \texttt{\intertext}, we merely drop the text in a fresh \texttt{\item}. For items delayed with \texttt{\plstx@later}, we evaluate them as is.
Finally, we evaluate the list of grammar items in a \trivlist:

\trivlist{\plstx@items}\endtrivlist

\fi
\endgroup

Change History

v0.1
  General: Initial documented release 1

v0.2
  General: Included listproc.sty .... 1

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols
* ........................................ 2, 3, (10)
\@forList .................................. 193
[ ........................................ 3, 4, (10)
\| ........................................ 145

A
align continue (configuration option) ........................................ 4, (5)
align is one of (configuration option) ........................................ 4, (5)
align nonterm (configuration option) ........................................ 5, (6)

C
\catcode .................................. 145
configuration options:
  align continue ............ 4, (5)
  align is one of ............ 4, (5)
  align nonterm ............. 5, (6)
  continue ..................... (4), 5
  continue center ........... 4
  continue left .............. 4
  continue right ............. 4
  gutter ......................... 5
  gutter left .................. 5
  gutter right ............... 5
  gutter text ................. 5
  gutter text ................. 5
  is one of ..................... (1), (3), 4, 5
  is one of center ......... 4
  is one of left .......... 4
  is one of right ......... 4
  label skip ............... 6
  label skip text .......... 6
  label style ............. (1), 6
  left margin ............. 6
  many per line ........... (6), 6
  margin .................. (6), 6, (7)
  nonterm center ........... 5
  nonterm left ............ 5
  nonterm right .......... 5
  nonterm style .......... 6
  one per line ........... (6), 6, (12)
  or ................... (6), 6