

The `plstx` package

Jesse A. Tov
tov@ccs.neu.edu

This document corresponds to `plstx` v0.2, dated 2011/03/26.

Contents

1 Introduction	1
1.1 Requirements	2
2 Command Reference	2
2.1 Configuration Options	4
3 Implementation	7

1 Introduction

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

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

Using the `plstx` environment, I coded that like this:

```
\begin{plstx}
*(type variables): \alpha [\in] \mathit{TVar} \\
*(variables):      x [\in] \mathit{Var} \\
(types): \tau ::= \alpha \mid \tau_1 \to \tau_2 \mid \forall \alpha. \tau \\
(terms): e ::= x \mid e_1 e_2 \mid \lambda x \colon \tau. e
          \mid \Lambda \alpha. e \mid e[\tau] \\
\end{plstx}
```

The `plstx` environment allows redefining much of its behavior. For example, if we prefer \longrightarrow to $::=$ 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.

```
\plstxset{
  is one of=\longrightarrow,
  label style=\textsf
}
```

Then we get:

$\alpha \in TVar$	(type variables)
$x \in Var$	(variables)
$\tau \longrightarrow \alpha \mid \tau_1 \rightarrow \tau_2 \mid \forall \alpha. \tau$	(types)
$e \longrightarrow x \mid e_1 e_2 \mid \lambda x: \tau. e \mid \Lambda \alpha. e \mid e[\tau]$	(terms)

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

$\alpha \in TVar$	<i>(type variables)</i>
$x \in Var$	<i>(variables)</i>
$\tau ::= \alpha \mid \tau_1 \rightarrow \tau_2 \mid \forall \alpha. \tau$	<i>(types)</i>
$e ::= x \mid e_1 e_2 \mid \lambda x: \tau. e$	<i>(terms)</i>
$\mid \Lambda \alpha. e \mid e[\tau]$	

1.1 Requirements

The `plstx` package depends on three other packages. Two are a standard part of the L^AT_EX distribution: `keyval` and `calc`. The third, `listproc`, is non-standard, and may be obtained at <http://www.ccs.neu.edu/~tov/code/latex/>.

2 Command Reference

`\plstxset {⟨plstx-options⟩}`

Takes a comma-separated list of keys and values, in the style of `keyval`:

$$\langle plstx-options \rangle ::= \langle key \rangle_1 = \langle value \rangle_1, \dots, \langle key \rangle_k = \langle value \rangle_k [,]$$

The options available are described in §2.1.

`\begin{plstx} [⟨plstx-options⟩]`
`⟨plstx-cmd⟩ ...`
`\end{plstx}`

The `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 `\plstxset`.

The available commands are:

```

⟨plstx-cmd⟩ ::= ⟨label-text⟩: ⟨nonterm⟩ ::= ⟨rhs⟩ \\  

              | * ⟨label-text⟩: ⟨nonterm⟩ [⟨sep⟩] ⟨rhs⟩ \\  

              | \set {⟨plstx-options⟩}  

              | \intertext {⟨text⟩}  

              | [⟨dimen⟩]

```

where

```

⟨sep⟩      ::= ⟨is-one-of⟩  

              | ⟨is-one-of⟩ , ⟨continue⟩  

⟨rhs⟩      ::= ⟨production⟩  

              | ⟨production⟩ | ⟨rhs⟩

```

If a command starts with `*`, `\set`, `\intertext`, or `[`, 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:

`⟨label-text⟩: ⟨nonterm⟩ ::= ⟨rhs⟩ \\`

A normal nonterminal item consists of a label `⟨label-text⟩` (which is set on the right, in text mode by default); a non-terminal being defined `⟨nonterm⟩` (which is set on the left, in math mode by default); a separator (option **is one of**, default `::=`, and written as `::=` in the command even if it has been configured to appear otherwise), and a right-hand side `⟨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.

`* ⟨label-text⟩: ⟨nonterm⟩ [⟨sep⟩] ⟨rhs⟩ \\`

A special nonterminal item starts with `*`, 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 $\alpha \in TVar$ in the example from §1, I wrote `\alpha [\in] \mathit{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 `[::=, \vert]` is equivalent to writing a normal nonterminal.

`\set {⟨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 {⟨text⟩}`

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

`[⟨dimen⟩]`

Inserts `⟨dimen⟩` vertical space.

Note: The grammar for `⟨plstx-cmd⟩` above was written like this:

```
\begin{plstx}[rhs style=,one per line]
: \meta{plstx-cmd}
  ::= \meta{label-text}\texttt: \meta{nonterm} \defother{::=}
     \meta{rhs} \texttt{\backslash\backslash}
     | \defother{*} \meta{label-text}\texttt: \meta{nonterm}
     \texttt[\meta{sep}\texttt]
     \meta{rhs} \texttt{\backslash\backslash}
     | \defmacro{set} \marg{plstx-options}
     | \defmacro{intertext} \marg{text}
     | \defother{[]}\meta{dimen}\texttt
     | \oarg{dimen}
  \\\
\intertext{where}
: \meta{sep}
  ::= \meta{is-one-of}
     | \meta{is-one-of} \texttt, \meta{continue} \\\
: \meta{rhs}
  ::= \meta{production}
     | \meta{production} {\defother|} \meta{rhs} \\\
\end{plstx}
```

2.1 Configuration Options

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

<code>align continue=⟨<i>cs</i>⟩</code>	<i>default:</i> ‘ <code>\plstx@right</code> ’
<code>continue center</code>	<i>(boolean) default:</i> false
<code>continue left</code>	<i>(boolean) default:</i> false
<code>continue right</code>	<i>(boolean) default:</i> true

To configure the horizontal alignment of the continuation separator (see **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>` \equiv `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   \def\plstx@gutter@left{\kern#1}%
13   \def\plstx@gutter@right{\kern#1}}
14 \define@key{plstx}{gutter left text}{\def\plstx@gutter@left{#1}}
15 \define@key{plstx}{gutter left}{\def\plstx@gutter@left{\kern#1}}
16 \define@key{plstx}{gutter right text}{\def\plstx@gutter@right{#1}}
17 \define@key{plstx}{gutter right}{\def\plstx@gutter@right{\kern#1}}
18 \define@key{plstx}{gutter text}{%
19   \def\plstx@gutter@left{#1}%
20   \def\plstx@gutter@right{#1}}
21 \define@key{plstx}{is one of center}[]{\def\plstx@align@isoneof{\plstx@center}}
22 \define@key{plstx}{is one of left}[]{\def\plstx@align@isoneof{\plstx@left}}
23 \define@key{plstx}{is one of right}[]{\def\plstx@align@isoneof{\plstx@right}}
24 \define@key{plstx}{is one of}{\def\plstx@isoneof{#1}}
25 \define@key{plstx}{label skip text}{\def\plstx@labelskip{#1}}
26 \define@key{plstx}{label skip}{\def\plstx@labelskip{\kern#1}}
```

```

27 \define@key{plstx}{label style}[]{\def\plstx@label@style{#1}}
28 \define@key{plstx}{left margin}[1em]{\def\plstx@margin@left{\kern#1}}
29 \define@key{plstx}{many per line}[]{\let\plstx@one@per@line\@secondoftwo}
30 \define@key{plstx}{margin}[1em]{%
31   \def\plstx@margin@left{\kern#1}%
32   \def\plstx@margin@right{\kern#1}}
33 \define@key{plstx}{nonterm center}[]{\def\plstx@align@nonterm{\plstx@center}}
34 \define@key{plstx}{nonterm left}[]{\def\plstx@align@nonterm{\plstx@left}}
35 \define@key{plstx}{nonterm right}[]{\def\plstx@align@nonterm{\plstx@right}}
36 \define@key{plstx}{nonterm style}[]{\def\plstx@nonterm@style{#1}}
37 \define@key{plstx}{one per line}[]{\let\plstx@one@per@line\@firstoftwo}
38 \define@key{plstx}{or skip text}{\def\plstx@orskip{#1}}
39 \define@key{plstx}{or skip}{\def\plstx@orskip{\kern#1}}
40 \define@key{plstx}{or}{\def\plstx@or{#1}}
41 \define@key{plstx}{rhs style}[]{\def\plstx@rhs@style{#1}}
42 \define@key{plstx}{right margin}[1em]{\def\plstx@margin@right{\kern#1}}

```

`\plstx@set` To set configuration options, we delegate to `\setkeys` from the `keyval` package.

```

\plstxset 43 \newcommand*\plstx@set{\setkeys{plstx}}
          44 \let\plstxset\plstx@set\relax

```

Set the initial options:

```

45 \plstx@set{
46   continue      = \vert,
47   continue right,
48   gutter        = 4pt,
49   is one of     = {::=},
50   is one of center,
51   label skip    = 1pc,
52   label style   = \emph,
53   many per line,
54   margin        = 0pt,
55   nonterm center,
56   nonterm style = \ensuremath,
57   or            = \vert,
58   or skip       = 4pt,
59   rhs style     = \ensuremath,
60 }

```

`\plstx@left` Helper commands for aligning text:

```

\plstx@right 61 \def\plstx@left#1{#1\hfill}
\plstx@center 62 \def\plstx@right#1{\hfill#1}
              63 \def\plstx@center#1{\hfill#1\hfill}

```

`\plstx@parseRHS` 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 `|`.


```

64 \newcommand\plstx@parseRHS[2]{%
65   \let#1=\empty
66   \plstx@parseRHS@loop#2|\plstx@parseRHS@stop\plstx@parseRHS@loop{#1}%
67 }
68 \def\plstx@parseRHS@loop#1|#2\plstx@parseRHS@loop#3{%
69   \SnocTo{#1}{#3}%
70   \ifx#2\plstx@parseRHS@stop
71     \let\plstx@parseRHS@kont=\relax
72   \else
73     \def\plstx@parseRHS@kont{%
74       \plstx@parseRHS@loop#2\plstx@parseRHS@loop{#3}%
75     }%
76   \fi
77   \plstx@parseRHS@kont
78 }

```

`\plstx@additem` 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.

```

79 \newcommand\plstx@additem[1]{%
80   \SnocTo{#1}{\plstx@items}%
81 }

```

`\plstx@dispatch` 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`.

```

82 \def\plstx@dispatch#1{%
83   \ifx#1\end
84     \let\plstx@dispatch@kont\end
85   \else\ifx#1\endplstx
86     \let\plstx@dispatch@kont\endplstx
87   \else\ifx#1\intertext
88     \let\plstx@dispatch@kont\plstx@intertext
89   \else\ifx#1[%
90     \let\plstx@dispatch@kont\plstx@vskip
91   \else\ifx#1\set
92     \let\plstx@dispatch@kont\plstx@set@later
93   \else\ifx#1*%
94     \let\plstx@dispatch@kont\plstx@other
95   \else
96     \def\plstx@dispatch@kont{\plstx@parseprod#1}%
97   \fi\fi\fi\fi\fi\fi
98   \plstx@dispatch@kont
99 }

```

`\plstx@parseprod` This is the command handler for normal productions. Productions are stored in
`::=` the item list as

`\plstx@production{<label-text>}{<nonterm>}{<is-one-of>}{<continue>}{<rhs>}`

It then calls back to `\plstx@dispatch` to have it figure out the next command.

```
100 \def\plstx@parseprod#1:#2: :=#3\{%
101   \plstx@additem{\plstx@production{#1}{#2}{\plstx@isoneof}{\plstx@continue}{#3}}%
102   \plstx@dispatch%
103 }
```

`\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.

```
104 \def\plstx@other#1:#2[#3]#4\{%
105   \let\plstx@other@isoneof\plstx@isoneof
106   \let\plstx@other@continue\plstx@continue
107   \def\plstx@other@todo##1{%
108     \def\plstx@other@isoneof{##1}%
109     \def\plstx@other@continue{##1}%
110     \def\plstx@other@todo###1{%
111       \def\plstx@other@continue{###1}%
112     }%
113   }%
114   \@for\plstx@each:=#3\do{%
115     \expandafter\plstx@other@todo\expandafter{\plstx@each}%
116   }%
117   \def\plstx@other@addthis##1##2{%
118     \plstx@additem{\plstx@production{#1}{#2}{##1}{##2}{#4}}%
119   }%
120   \expandafter\expandafter\expandafter\plstx@other@addthis
121   \expandafter\expandafter
122   \expandafter{\expandafter\plstx@other@isoneof\expandafter}%
123   \expandafter{\plstx@other@continue}%
124   \let\plstx@other@isoneof\@undefined
125   \let\plstx@other@continue\@undefined
126   \let\plstx@other@todo\@undefined
127   \let\plstx@other@addthis\@undefined
128   \plstx@dispatch
129 }
```

`\intertext` Intertext is added to the item list as

```
\plstx@intertext   \plstx@intertext{<text>}

130 \def\plstx@intertext#1{%
131   \plstx@additem{\plstx@intertext{#1}}%
132   \plstx@dispatch%
133 }
```

`\plstx@vskip` To add vertical space, we add

```
[<dimen>]   \plstx@later{\vskip<dimen>}
```

to the list of items.

```
134 \def\plstx@vskip#1{\plstx@additem{\plstx@later{\vskip#1}}\plstx@dispatch}
```

`\set` For `\set`, we add `\plstx@set{<plstx-options>}` directly to the list of grammar items.

```
135 \def\plstx@set@later#1{\plstx@additem{\plstx@set{#1}}\plstx@dispatch}
```

`\plstx@box@a` 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.

```
\plstx@box@b
```

```
136 \newsavebox\plstx@box@a
```

```
137 \newsavebox\plstx@box@b
```

```
138 \newsavebox\plstx@box@c
```

`\plstx@maxnt` 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.
`\plstx@maxisoneof`
`\plstx@availwd` The third dimension register, `\plstx@availwd`, is used to keep track of remaining available width when line breaking the right-hand side.

```
139 \newlength\plstx@maxnt
```

```
140 \newlength\plstx@maxisoneof
```

```
141 \newlength\plstx@availwd
```

`plstx` The main `plstx` environment.

```
142 \newenvironment{plstx}[1] []
```

```
143 {%
```

```
144   \begingroup
```

Make sure that `|` is recognizable as the production separator:

```
145   \catcode'\|=12\relax
```

```
146   \plstx@set{#1}%
```

Initialize the list of items as empty. Then call `\plstx@dispatch` to read in the commands in the grammar.

```
147   \let\plstx@items\empty
```

```
148   \plstx@dispatch
```

```
149 }
```

```
150 {%
```

```
151   \ifx\plstx@items\empty
```

```
152     \PackageWarning{plstx}{grammar must have at least one production}%
```

```
153   \else
```

For both passes through the list of items, we’ll just evaluate the list, so we make `\listitem` a no-op.

```
154   \def\plstx@listitem@noop##1{##1\let\listitem\plstx@listitem@noop}%
```

```
155   \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.

```
156   \setlength{\plstx@maxnt}{0pt}%
```

```

157 \setlength{\plstx@maxisoneof}{0pt}%
158 \def\plstx@production##1##2##3##4##5{%
159 \setlength
160   {\global\plstx@maxnt}
161   {\maxof{\plstx@maxnt}{\widthof{\plstx@nonterm@style{##2}}}}%
162 \setlength
163   {\global\plstx@maxisoneof}
164   {\maxof{\plstx@maxisoneof
165     {\maxof{\widthof{${##3}$}
166       {\widthof{${##4}$}}}}}}%
167   }%
168 \def\plstx@intertext##1{%
169 \def\plstx@later##1{%
170   {\plstx@items}%

```

Now `\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 `\trivlist`, so we'll precede each line with `\item`. We redefine the grammar item callbacks:

```

171 \def\plstx@production##1##2##3##4##5{%

```

The initial available width is the `\linewidth`. We then add the label to `box@a`, and if the resulting box has non-zero width, we prepend `\plstx@labelskip` to it. Then, in either case, we postpend the right margin to it. We update the available width to account for the size of the label and any space around it.

```

172 \setlength{\plstx@availwd}{\linewidth}%
173 \sbox\plstx@box@a{\plstx@label@style{##1}}%
174 \ifdim\wd\plstx@box@a>0pt
175 \sbox\plstx@box@a{\plstx@labelskip\usebox\plstx@box@a}%
176 \fi
177 \sbox\plstx@box@a{\usebox\plstx@box@a\plstx@margin@right}%
178 \addtolength{\plstx@availwd}{-\wd\plstx@box@a}%

```

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

```

179 \sbox\plstx@box@b{%
180 \plstx@margin@left
181 \makebox[\plstx@maxnt]
182   {\plstx@align@nonterm{\plstx@nonterm@style{##2}}}%
183 \plstx@gutter@left
184 \makebox[\plstx@maxisoneof]{\plstx@align@isoneof{${##3}$}}%
185 \plstx@gutter@right
186   }%

```

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

```

187 \plstx@parseRHS\plstx@rhsOut{##5}%
188 \LopTo\plstx@rhsOut\plstx@rhsFirst
189 \sbox\plstx@box@b
190   {\usebox\plstx@box@b

```

```

191         \plstx@rhs@style{\plstx@rhsFirst}}%
192         \addtolength{\plstx@availwd}{-\wd\plstx@box@b}%

```

Now iterate over the remaining productions.

```

193         \@forList\plstx@each:=\plstx@rhsOut\do{%

```

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

```

194         \sbox\plstx@box@c
195         {\plstx@orskip${\plstx@or}$\plstx@orskip
196         \plstx@rhs@style{\plstx@each}}%
197         \plstx@one@per@line
198         {\iftrue}
199         {\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.

```

200         \item\makebox[\linewidth]
201         {\strut\usebox\plstx@box@b\hfill\usebox\plstx@box@a}
202         \setlength{\plstx@availwd}{\linewidth}%
203         \sbox\plstx@box@a{\plstx@margin@right}%
204         \sbox\plstx@box@b{%
205         \plstx@margin@left
206         \makebox[\plstx@maxnt]{}%
207         \plstx@gutter@left
208         \makebox[\plstx@maxisoneof]{\plstx@align@continue{${##4}$}}%
209         \plstx@gutter@right
210         \plstx@rhs@style{\plstx@each}}%
211         }%
212         \addtolength{\plstx@availwd}{-\wd\plstx@box@b}%
213         \else

```

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

```

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

```

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

```

218         \ifdim\wd\plstx@box@b>0pt
219         \item\makebox[\linewidth]
220         {\strut\usebox\plstx@box@b\hfill\usebox\plstx@box@a}
221         \fi
222         }%

```

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

For `\intertext`, we merely drop the text in a fresh `\item`. For items delayed with `\plstx@later`, we evaluate them as is.

```

223     \def\plstx@intertext##1{%
224         \item\strut\ignorespaces##1%
225     }%
226     \def\plstx@later##1{##1}%

```

Finally, we evaluate the list of grammar items in a `\trivlist`:

```

227     \trivlist{\plstx@items}\endtrivlist
228     \fi
229     \endgroup
230 }

```

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	
<code>*</code>	2, 3, (10)
<code>\@forList</code>	193
<code>[</code>	3, 4, (10)
<code>\l</code>	145
A	
<code>align continue</code> (configuration option)	4, (5)
<code>align is one of</code> (configuration option)	4, (5)
<code>align nonterm</code> (configuration option)	5, (6)
C	
<code>\catcode</code>	145
configuration options:	
<code>align continue</code>	4, (5)
<code>align is one of</code>	4, (5)
<code>align nonterm</code>	5, (6)
<code>continue</code>	(4), 5
<code>continue center</code>	4
<code>continue left</code>	4
<code>continue right</code>	4
<code>gutter</code>	5
<code>gutter left</code>	5
<code>gutter left text</code>	5
<code>gutter right</code>	5
<code>gutter right text</code>	5
<code>gutter text</code>	5
<code>is one of</code>	(1), (3, 4), 5
<code>is one of center</code>	4
<code>is one of left</code>	4
<code>is one of right</code>	4
<code>label skip</code>	6
<code>label skip text</code>	6
<code>label style</code>	(1), 6
<code>left margin</code>	6
<code>many per line</code>	(6), 6
<code>margin</code>	(6), 6, (7)
<code>nonterm center</code>	5
<code>nonterm left</code>	5
<code>nonterm right</code>	5
<code>nonterm style</code>	(6), 6
<code>one per line</code>	(6), 6, (12)
<code>or</code>	(6), 6

or skip	6		
or skip text	6		
rhs style	7		
right margin	6		
continue (configuration option)	(4), 5		
continue center (configuration option)	4		
continue left (configuration option)	4		
continue right (configuration option)	4		
D			
<code>\define@key</code>	4, 5, 6, 7, 8, 9, 10, 11, 14, 15, 16, 17, 18, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42		
E			
<code>\emph</code>	(6), 52		
<code>\endplstx</code>	85, 86		
<code>\endtrivlist</code>	227		
<code>\ensuremath</code>	(6, 7), 56, 59		
environments:			
<code>plstx</code>	(1), 2, (4), 142		
G			
gutter (configuration option)	5		
gutter left (configuration option)	5		
gutter left text (configuration option)	5		
gutter right (configuration option)	5		
gutter right text (configuration option)	5		
gutter text (configuration option)	5		
H			
<code>\hfill</code>	61, 62, 63, 201, 220		
I			
<code>\iftrue</code>	198		
<code>\intertext</code>	3, (6), 87, 130		
is one of (configuration option)	(1), (3, 4), 5		
is one of center (configuration option)	4		
is one of left (configuration option)	4		
is one of right (configuration option)	4		
<code>\item</code>	200, 219, 224		
K			
<code>\kern</code>	(5, 6), 12, 13, 15, 17, 26, 28, 31, 32, 39, 42		
L			
label skip (configuration option)	6		
label skip text (configuration option)	6		
label style (configuration option)	(1), 6		
left margin (configuration option)	6		
<code>\linewidth</code>	172, 200, 202, 219		
<code>\listitem</code>	154		
<code>\LopTo</code>	188		
M			
<code>\makebox</code>	181, 184, 200, 206, 208, 219		
many per line (configuration option)	(6), 6		
margin (configuration option)	(6), 6, (7)		
<code>\maxof</code>	161, 164, 165		
N			
<code>\newlength</code>	139, 140, 141		
<code>\newsavebox</code>	136, 137, 138		
nonterm center (configuration option)	5		
nonterm left (configuration option)	5		
nonterm right (configuration option)	5		
nonterm style (configuration option)	6		
O			
one per line (configuration option)	(6), 6, (12)		
or (configuration option)	(6), 6		
or skip (configuration option)	6		
or skip text (configuration option)	6		
P			
<code>\PackageWarning</code>	152		
<code>plstx</code> (environment)	(1), 2, (4), 142		
<code>\plstx@additem</code>	79, 79, 101, 118, 131, 134, 135		
<code>\plstx@align@continue</code>	4, 7, 8, 9, 208		
<code>\plstx@align@isoneof</code>	5, 21, 22, 23, 184		
<code>\plstx@align@nonterm</code>	6, 33, 34, 35, 182		
<code>\plstx@availwd</code>	139, 141, 172, 178, 192, 199, 202, 212, 214		
<code>\plstx@box@a</code>	136, 136, 173, 174, 175, 177, 178, 201, 203, 220		
<code>\plstx@box@b</code>	136, 137, 179, 189, 190, 192, 201, 204, 212, 215, 218, 220		

<code>\plstx@box@c</code>	136, 138, 194, 199, 214, 215	<code>\plstx@parseRHS@kont</code>	71, 73, 77
<code>\plstx@center</code>	(4, 5), 7, 21, 33, 61, 63	<code>\plstx@parseRHS@loop</code>	64, 66, 68, 74
<code>\plstx@continue</code>	10, 101, 106	<code>\plstx@parseRHS@stop</code>	66, 70
<code>\plstx@dispatch</code>	(9), 82, 82, 102, 128, 132, 134, 135, 148	<code>\plstx@production</code>	(9), 101, 118, 158, 171
<code>\plstx@dispatch@kont</code>	84, 86, 88, 90, 92, 94, 96, 98	<code>\plstx@rhs@style</code>	41, 191, 196, 210
<code>\plstx@each</code>	114, 115, 193, 196, 210	<code>\plstx@rhsFirst</code>	188, 191
<code>\plstx@gutter@left</code>	12, 14, 15, 19, 183, 207	<code>\plstx@rhsOut</code>	187, 188, 193
<code>\plstx@gutter@right</code>	13, 16, 17, 20, 185, 209	<code>\plstx@right</code>	(4), 9, 23, 35, 61, 62
<code>\plstx@intertext</code>	(10), 88, 130, 130, 131, 168, 223	<code>\plstx@set</code>	(10), 43, 43, 44, 45, 135, 146
<code>\plstx@isoneof</code>	24, 101, 105	<code>\plstx@set@later</code>	92, 135, 135
<code>\plstx@items</code>	80, 147, 151, 170, 227	<code>\plstx@vskip</code>	90, 134, 134
<code>\plstx@label@style</code>	27, 173	<code>\plstxset</code>	(1, 2), 2, (4), 43, 44
<code>\plstx@labelskip</code>	25, 26, 175	R	
<code>\plstx@later</code>	(10), 134, 169, 226	<code>\RequirePackage</code>	1, 2, 3
<code>\plstx@left</code>	8, 22, 34, 61, 61	rhs style (configuration option)	7
<code>\plstx@listitem@noop</code>	154, 155	right margin (configuration option)	6
<code>\plstx@margin@left</code>	28, 31, 180, 205	S	
<code>\plstx@margin@right</code>	32, 42, 177, 203	<code>\sbox</code>	173, 175, 177, 179, 189, 194, 203, 204, 215
<code>\plstx@maxisoneof</code>	139, 140, 157, 163, 164, 184, 208	<code>\set</code>	2, 3, (4), 91, 135
<code>\plstx@maxnt</code>	139, 139, 156, 160, 161, 181, 206	<code>\setkeys</code>	(8), 43
<code>\plstx@nonterm@style</code>	36, 161, 182	<code>\setlength</code>	156, 157, 159, 162, 172, 202
<code>\plstx@one@per@line</code>	29, 37, 197	<code>\SnocTo</code>	69, 80
<code>\plstx@or</code>	40, 195	<code>\strut</code>	201, 220, 224
<code>\plstx@orskip</code>	38, 39, 195	T	
<code>\plstx@other</code>	94, 104, 104	<code>\trivlist</code>	(12), 227
<code>\plstx@other@addthis</code>	117, 120, 127	U	
<code>\plstx@other@continue</code>	106, 109, 111, 123, 125	<code>\usebox</code>	175, 177, 190, 201, 215, 220
<code>\plstx@other@isoneof</code>	105, 108, 122, 124	V	
<code>\plstx@other@todo</code>	107, 110, 115, 126	<code>\vert</code>	(5, 6), 46, 57
<code>\plstx@parseprod</code>	96, 100, 100	<code>\vskip</code>	(10), 134
<code>\plstx@parseRHS</code>	64, 64, 187	W	
		<code>\widthof</code>	161, 165, 166