# CHARTING YOUR GRAMMAR WITH TEX

Michael F. Plass
Xerox Corporation

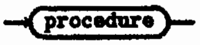
Are you one of those people who would rather look at a syntax chart than a BNF grammar? Do you avoid making a syntax chart for your language because it is too hard to draw or too hard to typeset? If so, this is the article for you. Pay attention, and you will learn how to use the macros below to create your own syntax charts with TEX.

First some basics. Every component of the syntax chart is enclosed in a TEX box, with the entry and exit points on the left and right sides of the box, aligned with the baseline. Usually the entry point is on the left end and the exit is on the right, but not always, as we shall see.

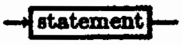
The simplest components are the boxes that represent the terminals and the nonterminals of the language. The terminals are the lowest level pieces of the language that the description deals with, for example, keywords, special characters, letters, digits, and so forth. Nonterminals are the names for the building blocks of the language. Some examples of nonterminal symbols might be number, identifier, expression, statement, program. If you are familiar with BNF, the nonterminals are the things in the angle brackets.

The terminals in a syntax chart are enclosed in boxes with rounded ends, and nonterminals are enclosed in rectangular boxes. This is how you use the syntax chart macros to make both kinds of elementary boxes:

```
\Terminal{procedure}
```
⇒



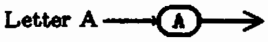
```
\Nonterminal{statement}
```
⇒



(Keep in mind that the case of the first letter in a TEX control sequence is significant.) You can control what fonts are used inside these boxes by defining the control sequences \TerminalFont and \NonterminalFont to be the appropriate font selectors. Notice each of these basic boxes have 'stems' on the left and right sides, one of them being an arrow and the other one just a line. TEX will determine which direction to point the arrow in on the basis of how the box is nested inside of other constructions.

The definition of a nonterminal is specified like this:

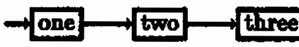
```
\Define{Letter A}\Terminal{A}\EndDef
```
⇒



More complicated charts are built up by means of *sequencing*, *alternation*, and *repetition* of simpler charts. To illustrate the way these work, we w
sume that the control sequence \one has been d
to be \Nonterminal{one}, and so forth.

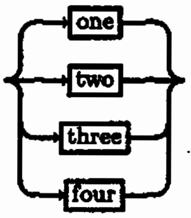
Sequencing is easy to do; just put the sub
together:

```
\one\two\three
```
⇒



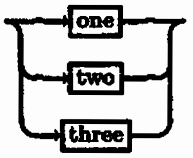
Alternation is a bit more complicated to sp
Here is an example:

```
\Alternatives
 {\Upper{\one}
  \Middle{\two}
  \Lower{\three}
  \Lower{\four}}
```
⇒

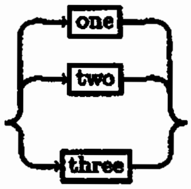


There may be any positive number of choices
as the argument to \Alternatives, and each ch
marked by enclosing it in braces and preced
with \Upper, \Middle, or \Lower. Things marked
\Upper go above the baseline, things marked
\Lower go below the baseline, and something m
with \Middle goes on the baseline. It is permit
omit any one of these three kinds of tags:

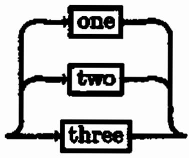
```
\Alternatives
 {\Middle{\one}
  \Lower{\two}
  \Lower{\three}}
```
⇒



```
\Alternatives
 {\Upper{\one}
  \Upper{\two}
  \Lower{\three}}
```
⇒

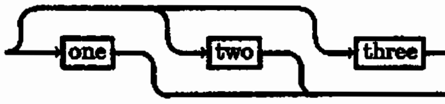


```
\Alternatives
 {\Upper{\one}
  \Upper{\two}
  \Middle{\three}}
```
⇒



Sometimes a stack of alternatives can ge
tall; in this case it might be better to spread
out horizontally:

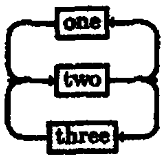
```
\HorzAlternatives{\Alternative{\one}
 \Alternative{\two}\Alternative{\three}}
```
⇒



This one is especially appropriate for a long
short choices.

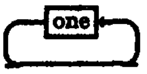
Repetition is specified in almost exactly the same way as alternatives:

```
\Repeat{\Upper{\one}
        \Middle{\two}    ⇒
        \Lower{\three}}
```

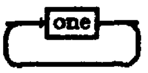


The same rules apply as before, except that you are not allowed to leave out the `\Middle`. This is not the same as specifing an empty middle, which you might often want to do:

```
\Repeat{\Upper{\one}\Middle{}}   ⇒
```



```
\Repeat{\Middle{\one}\Lower{}}   ⇒
```



Sometimes a chart gets too wide for the page, and TeX breaks it up into lines as if it were a paragraph. If this is want you want, fine—but if you are in unrestricted horizontal mode, you can make a nicer transition to a new line by using `\NewLine`. Look at Appendix B for examples of this.

Now you know about all of the constructions. They can be nested in any way you please, up to a depth of four. One way to get around this limit is to save away the innermost parts of the diagram in a box and call it in when you need it. You can say `\AllocBox\cntrlseq` to define `\cntrlseq` to be a digit indicating a box that you can use; this way your box usage won't conflict with that of the syntax chart macros. Remember to use braces to enclose the section in which you use `\AllocBox`, so that the box number can be used again. If you exceed TeX's `varsize` limit, the usual solution is to force a page eject so TeX does not have to keep so much stuff in its memory.

To use the macros, say

```
\input SynChart
```

near the beginning of your file. When you actually want to make a chart, say `\SyntaxChart` to set up the baselineskip, lineskip, and some other stuff. If you are mixing syntax charts with other text, you will want to enclose the `\SyntaxChart` along with the chart definition in braces, so all the funny definitions go away at the end of the group and don't mess up the rest of your document. You will probably also want to say `\parindent 0pt` to eliminate the paragraph indentation.

The call on `\SyntaxChart` also tells TeX to ignore tabs and carriage returns so you can format your more easily. Just be careful where you add spaces—after a control sequence is OK, but not between or after the parameters to a macro. The reason for not ignoring spaces is so terminals and nonterminals can have embedded spaces; if you prefer, you can say `\IgnoreWhiteSpace` to cause spaces to be ignored too, and then put in a control-space where you really want a space.

Now would be a good time to go try making your own simple syntax chart, using the macro definitions in Appendix A. When defining a complex diagram, it is often helpful to first define pieces of it as TeX macros. This makes the source much easier to read, and keeps the nesting of braces down to a reasonable level. This device is used extensively in the example in Appendix B.

Finally, some fine points about spacing. The horizontal lines are actually composed of fixed-width rules and leaders filled with rules. The control sequence `\QLine` defines a fairly short fixed-width line, and `\QQLine` defines one twice as long. `\Fil` defines a line that stretches like glue, with a normal width of zero and the same stretchability as `\hfil`; `\Fill` is similar, but with the stretchability of an `\hfill`. Each component of a repetition or (vertical) alternation construction implicitly has `\Fil` on each side, so if the components are not all the same width, the available space in each of the shorter ones will be distributed evenly between the embedded `\Fil`s and the ends of the component. By using these macros in various combinations, you can get any kind of horizontal spacing you want.

When a terminal box contains only special character or only upper case letters, the result looks best when the contents are centered vertically in the box; this is the default. However, if the same thing is done for lower case letters, the baselines of the words in different boxes may not line up due to the pattern of ascenders and descenders. The way to fix this is by using a *strut*, which is a zero-width box whose height and depth match the extremes of the font. If the keywords in your language are in lower case, say `\Strut` after the keyword in the terminal box—or, better yet, define a macro that does this, as in Appendix B. Struts are already included in nonterminal boxes, because these contain mostly lower-case letters.

Sometimes two rows would look better if they were moved a little bit closer together. You can do this by using `\TrimTop` or `\TrimBot`; these macros take an hlist as a parameter, box it, and then take a little off of the top or the bottom. Look at Appendix B to see how these can be used.

## Appendix A. Listing of SynChart.TEX

```
%%%%%% Beginning of SynChart.TEX

% editor's note - fonts have been adjusted to conform to TUGboat usage;
% the original values are retained, commented out
\font G=dragon10                        % \font >=MANFNT
%                                          \font U=CMTT8
%                                          \font c=CMR8

% The following macros declare how white space is treated.
\def\IgnoreLinebreaks{\chcode'15=9\chcode'11=9}
\def\IgnoreWhiteSpace{\chcode'15=9\chcode'11=9\chcode'40=9}
\def\DontIgnoreWhiteSpace{\chcode'15=5\chcode'11=10\chcode'40=10}

\IgnoreWhiteSpace % so the macro definitions may be freely formatted.

\def\NonterminalFont{\:c}        % font to use for nonterminal symbols
\def\TerminalFont{\:U}                % font to use for terminal symbols
% \def\GraphicFont{\:>}                % font to get quarter circles from;
\def\GraphicFont{\:G}               % font to get quarter circles from;
                                % should have characters a, b, c, and d like
                                % the DRAGON font in the metafont manual.

% The \LeftArrow, \RightArrow, and \ArrowLine macros define the stems
% that get put on each terminal or nonterminal box.  The appropriate arrow
% is put on the entry side of each box, and \ArrowLine is put on the exit side
% to balance out the arrow.  These definitions supply a rather small arrow
% with a long stem, using characters from the standard CMSY10 font.  Users
% that have good arrowheads available in other fonts should redefine these
% macros to take advantage of them.
\def\LeftArrow {\hskip 1.2pt
  \Strikeout{\vbox to 1.9pt{\hbox{\hskip-2pt\:u\char'40\hskip-0.5pt}\vss}}}
\def\RightArrow
 {\Strikeout{\vbox to 1.9pt{\hbox{\hskip-0.5pt\:u\char'41\hskip-2pt}
                           \vss}}\hskip 1.2pt}
\def\ArrowLine{\Strikeout{\hskip 8.7pt}}

% The next two macros define big arrowheads.
\def\BigLeftArrow{\Strikeout{\CenterSink{\hbox{\hskip-2vu\bf<}}}}
\def\BigRightArrow{\Strikeout{\CenterSink{\hbox{\bf>\hskip-2vu}}}}

% The next few macros describe the dimensions of the quarter-circle font.
\def\QThickness{1.1pt} % Thickness of strokes
\def\QSize{5pt} % Height and width of the quarter circles
\def\QQSize{10pt} % Twice \QSize
\varunit\QThickness % 1vu is the stroke thickness.

% The next two groups of macros may need editing if a font other than the
% dragon font is used to get the quarter circles.

% \I, \II, \III, \IV are the quarter circles in the first, second, third and
% fourth quadrants, numbered counterclockwise starting with the upper-right one.
\def\I{\lower \QSize \hbox{\GraphicFont a\BackQ}}
\def\II{\lower \QSize \hbox{\BackQ\GraphicFont d}}
\def\III{\lower \QSize \hbox{\BackQ\GraphicFont c}}
\def\IV{\lower \QSize \hbox{\GraphicFont b\BackQ}}

% \LeftRound and \RightRound are the end caps for Terminal boxes
\def\LeftRound {\BackQ\hbox{\GraphicFont d}\BackQ\BackQ
                \lower\QQSize\hbox{\GraphicFont c}}
\def\RightRound{\hbox{\GraphicFont a}\BackQ\BackQ
                \lower\QQSize\hbox{\GraphicFont b}\BackQ}

% editor's note - page break for TUGboat publication
% \LeftSquare and \RightSquare are the end caps for Nonterminal boxes
\def\LeftSquare
 {\vrule height\QQSize depth\QQSize width 0pt
  \vrule height\QSize depth\QSize width 1vu \hskip -1vu}
\def\RightSquare
 {\hskip -1vu \vrule height\QSize depth\QSize width 1vu}

% These macros are for moving forward and backward in Q units.
\def\BackQ{\hskip-\QSize}
\def\BackQQ{\hskip-\QQSize}
\def\ForwQ{\hskip \QSize}
\def\ForwQQ{\hskip \QQSize}

% Macros in this group are meant to be used directly by the user.
\def\Empty{\vrule height\QSize depth\QSize width 0pt}
\def\Strut{\save0\hbox{Bg}\vrule height 1ht0 depth 1dp0 width 0pt}
\def\QLine{\vrule height 0.5vu depth 0.5vu width \QSize}
\def\QQLine{\vrule height 0.5vu depth 0.5vu width \QQSize}
\def\Fil{\leaders\HorzLine\hfil}
\def\Fill{\leaders\HorzLine\hfill} % use this to swamp a \Fil
\def\FollowedBy{\Fil} % use to separate symbols in a sequence
\def\TrimTop#1{\vbox expand -\Qsize{\vss\hbox{#1}}}
\def\TrimBot#1{\hbox{$\vtop{\hbox{#1}\vskip-\QQSize}$}}

% Some useful formatting primitives, used internally.
\def\HorzLine{\hrule height 0.5vu depth 0.5vu}
\def\Strikeout
  #1{\save0\hbox{#1}\hbox to 1wd0
        {\vrule height 0.5vu depth 0.5vu width 1wd0\hss\unbox0}}
\def\CenterSink#1{\lower\QSize\vbox to \QQSize{\vss\hbox{#1}\vss\null}}
\def\Sandwch#1{\lower\QSize
                \vbox to \QQSize
                 {\vskip-0.5vu\HorzLine\vss
                 \hbox{#1}\vss
                 \HorzLine\vskip-0.5vu\null}}
\def\VertLine#1{\hskip -0.5vu \vrule #1 width 1vu \hskip -0.5vu}
\def\LinetoTop{\VertLine{depth-\QSize}}
\def\LinetoBot{\VertLine{height-\QSize}}
\def\FilVertLine{\hskip -0.5vu \vrule width 1vu \hskip -0.5vu}

% These define some common bits and pieces of the diagrams.
\def\SwitchUp{\IV\BackQ\QQLine}
\def\SwitchDn{\I\BackQ\QQLine}
\def\MergeUp{\QQLine\BackQ\II}
\def\MergeDn{\QQLine\BackQ\III}

% Use \Define to start a section of the syntax diagram.
\def\Define#1{\par${}$\CenterSink
  {\NonterminalFont\Strut#1}\hskip 2vu plus 300pt
  \penalty 0\QQLine}

% And use \EndDef to end it.
\def\EndDef{\QLine\BigRightArrow\par}

% If a section gets too wide, \NewLine will continue the chart on a new line.
\def\NewLine
   {\Fil\I\LinetoBot\ForwQQ\linebreak
    \hbox{\ForwQQ\ForwQQ}\LinetoBot\II
    \Fil\vrule height \QQSize depth \QQSize width 0pt\BigLeftArrow\Fil
    \IV\LinetoTop\ForwQQ\linebreak
    \hbox{\ForwQQ\ForwQQ}\LinetoTop\III\Fil}

% editor's note - page break for TUGboat publication
% The macros \LeftToRight, \RightToLeft, and \SwitchDirection control the
% way arrows are pasted onto the low-level boxes of the chart.
\def\LeftToRight
 {\def\LeftSideArrow{\RightArrow}
  \def\RightSideArrow{\ArrowLine}
  \def\SwitchDirection{\RightToLeft}}
\def\RightToLeft
 {\def\LeftSideArrow{\ArrowLine}
  \def\RightSideArrow{\LeftArrow}
  \def\SwitchDirection{\LeftToRight}}

% The next two macros are used for creating terminal and nonterminal boxes.
\def\Nonterminal
 #1{\LeftSideArrow\LeftSquare\Sandwch
    {\NonterminalFont
     \ #1\Strut\ }\RightSquare\RightSideArrow}
\def\Terminal
 #1{\LeftSideArrow\LeftRound\Sandwch
    {\TerminalFont
     #1}\RightRound\RightSideArrow}

% The next group of macros are used for allocating digits for referring to
% counters and boxes.  These are needed only because counters and boxes in TEX
% do not nest in the nice way that macro definitions do.  The macros are a
% bit on the tricky side, but are short enough that you can figure out how
% they work if you understand the way TEX expands macros.  If you are using
% other packages that use some of these boxes or counters, you may have to
% change some of the numbers in these macros.
\def\Alloc#1#2{\def#2{#1}}
\def\AllocBox
 {\def\AllocBox{\def\AllocBox{\def\AllocBox{\def\AllocBox
 {\def\AllocBox{\def\AllocBox{\def\AllocBox{\def\AllocBox
 {\def\AllocBox{\Overflow
  }\Alloc9}\Alloc8}\Alloc7}\Alloc6}\Alloc5}\Alloc4}\Alloc3}\Alloc2}\Alloc1}
\def\AllocCtr
 {\def\AllocCtr{\def\AllocCtr{\def\AllocCtr{\def\AllocCtr
  {\Overflow}\Alloc8}\Alloc7}\Alloc6}\Alloc5}


% Here're the ones you've been waiting for.  Both \Alternatives and
% \Repeat work in roughly the same fashion, so this description applies
% tc them both.  The single argument is composed of the subcomponents,
% each one enclosed in braces and preceded by a control sequence.  These
% control sequences get defined as local macros, and grab the subcomponents
% as parameters.  Actually, this happens twice: the first time the argument
% is interpreted, the local macros just ignore their parameter, and the
% second time through, the real work is done.  The first pass is used to
% count the subcomponents, since the last one has to be treated specially,
% and also does some validation of the arguments.
%
% There are places in these macros where a \gdef or \xdef is used to get some
% information out of a local scope or to force expansion of a count; since
% the value is used before any other macros are expanded, there is no danger
% of nested macro calls messing things up.
%
% If the structure of the macro isn't tricky enough for you, you can try to
% understand the way the result is built up out of boxes.  Here is the basic
% idea: while the stuff in the middle is being built up in an \halign, the
% connectors on the side are built up in two boxes, one for the left and the
% other on the right.  The baselines of the boxes on the sides are maintained
% to be correct for the final result by doing a \vbox above and at the middle
% and a \vtop after the middle.  The baseline of the stuff in the middle is
% ignored, and when it is all built, it is boxed and forced to line up with
% the boxes on either side.  The \halign is needed since each row has to be
% expanded to the maximum width of everything in the middle.
\def\Alternatives
    #1{{\AllocCtr\AltCtr
```

```
        \setcount\AltCtr 0
        \def\state{T}
        \def\Upper##1{\if L\state{\BadUseOfUpper}\else{}
                      \advcount\AltCtr\def\state{U}}
        \def\Middle##1{\if L\state{\BadUseOfMiddle}\else{}
                       \advcount\AltCtr\def\state{L}}
        \def\Lower##1{\advcount\AltCtr\def\state{L}}
        #1
        \if L\state{}\else{\NoMiddle}
        \def\state{T}
        \def\Upper{\advcount\AltCtr by -1
                   \UpperAlternative}
        \def\Middle{\advcount\AltCtr by -1
                    \MiddleAlternative}
        \def\Lower{\if L\state {\gdef\OMA{}}
                   \else {\gdef\OMA{\OmittedMiddleAlternative}}
                   \OMA % Needed because of scoping in the if statement
                   \advcount\AltCtr by -1
                   \LowerAlternative}
        \AllocBox\L\save\L\null
        \AllocBox\R\save\R\null
        \save0\hbox{$\vtop{\null\halign{##\cr#1\null\cr}}$}
        \hbox{\ForwQ\box\L\raise 1ht\R\box0\box\R\ForwQ}}}

% One extra tricky thing about \Repeat is that the scope of \SwitchDirection
% is terminated by the \cr at the end of the alternatives.  Maybe you didn't
% know about this scoping rule; it was sure news to me.
\def\Repeat
    #1{{\AllocCtr\AltCtr
        \setcount\AltCtr 0
        \def\state{T}
        \def\Upper
            ##1{\if L\state{\BadUseOfUpper}\else{}
                \advcount\AltCtr\def\state{U}}
        \def\Middle
            ##1{\if L\state{\BadUseOfMiddle}\else{}
                \advcount\AltCtr\def\state{L}}
        \def\Lower
            ##1{\if L\state{}\else{\BadUseOfLower}
                \advcount\AltCtr\def\state{L}}
        #1
        \if L\state{}\else{\NoMiddle}
        \def\state{T}
        \def\Upper{\SwitchDirection\advcount\AltCtr by -1
                   \UpperAlternative}
        \def\Middle{\advcount\AltCtr by -1
                    \RepeatBody}
        \def\Lower{\SwitchDirection\advcount\AltCtr by -1
                   \LowerAlternative}
        \AllocBox\L\save\L\null
        \AllocBox\R\save\R\null
        \save0\hbox{$\vtop{\null\halign{##\cr#1\null\cr}}$}
        \hbox{\box\L\raise 1ht\R\box0\box\R}}}

% editor's note - page break for TUGboat publication
```

```
\def\UpperAlternative
 #1{\save0\hbox{#1\Empty}
    \save\L\vbox{\box\L\hbox{\vrule height 1ht0 depth 1dp0 width 0pt
                                 \if U\state{\FilVertLine}\else{\LinetoBot}\II}}
    \save\R\vbox{\box\R\hbox{\vrule height 1ht0 depth 1dp0 width 0pt
                                 \I\if U\state{\FilVertLine}\else{\LinetoBot}}}
    \Fil\unbox0\Fil\cr\def\state{U}}

\def\MiddleAlternative
 #1{\save0\hbox{#1\Empty}
    \xdef\RowsToGo{\count\AltCtr}
    \save\L\vbox{\box\L\hbox{\vrule height 1ht0 depth 1dp0 width 0pt
                             \if U\state{\BackQ\IV\LinetoTop}\else{}
                             \if 0\RowsToGo{}\else{\BackQ\I\LinetoBot}
                             \BackQ\QQLine}}
    \save\R\vbox{\box\R\hbox{\vrule height 1ht0 depth 1dp0 width 0pt
                             \QQLine\BackQ
                             \if U\state{\LinetoTop\III\BackQ}\else{}
                             \if 0\RowsToGo{}\else{\LinetoBot\II\BackQ}}}
    \Fil\unbox0\Fil\cr\def\state{L}}

\def\OmittedMiddleAlternative
   {\save0\hbox{\Empty}
    \xdef\RowsToGo{\count\AltCtr}
    \save\L\vbox{\box\L\hbox{\vrule height 1ht0 depth 1dp0 width 0pt
                             \if U\state{\BackQ\IV\LinetoTop}\else{}
                             \if 0\RowsToGo{}\else{\BackQ\I\LinetoBot}
                             \ForwQ}}
    \save\R\vbox{\box\R\hbox{\vrule height 1ht0 depth 1dp0 width 0pt
                             \ForwQ
                             \if U\state{\LinetoTop\III\BackQ}\else{}
                             \if 0\RowsToGo{}\else{\LinetoBot\II\BackQ}}}
    \hfil\Empty\cr\def\state{L}}

\def\RepeatBody
 #1{\save0\hbox{#1\Empty}
    \xdef\RowsToGo{\count\AltCtr}
    \save\L\vbox{\box\L\hbox{\vrule height 1ht0 depth 1dp0 width 0pt
                             \if U\state{\LinetoTop\III\BackQ}\else{}
                             \if 0\RowsToGo{}\else{\LinetoBot\II\BackQ}
                             \QLine}}
    \save\R\vbox{\box\R\hbox{\vrule height 1ht0 depth 1dp0 width 0pt
                             \QLine
                             \if U\state{\BackQ\IV\LinetoTop}\else{}
                             \if 0\RowsToGo{}\else{\BackQ\I\LinetoBot}}}
    \Fil\unbox0\Fil\cr\def\state{L}}

\def\LowerAlternative
 #1{\save0\hbox{#1}
    \xdef\RowsToGo{\count\AltCtr}
    \save\L\hbox{$\vtop{\box\L\hbox{\vrule height 1ht0 depth 1dp0 width 0pt
            \if 0\RowsToGo {\LinetoTop}\else{\FilVertLine}\III}}$}
    \save\R\hbox{$\vtop{\box\R\hbox{\vrule height 1ht0 depth 1dp0 width 0pt
            \IV\if 0\RowsToGo{\LinetoTop}\else{\FilVertLine}}}$}
    \Fil\unbox0\Fil\Empty\cr\def\state{L}}

% editor's note - page break for TUGboat publication
% Horizontal alternatives work in much the same way as the others,
% but are a little simpler because we don't have to worry about a \Middle
% and we can just use an \hbox instead of \vboxing an \halign and fooling
% around with the position of the baseline.
\def\HorzAlternatives
    #1{{\AllocCtr\AltCtr \setcount\AltCtr 0
        \def\Alternative##1{\advcount\AltCtr}
        #1% count the number of alternatives
        \def\Alternative
          {\advcount\AltCtr by -1\xdef\AltsToGo{\count\AltCtr}
           \if O\AltsToGo{\gdef\HAlt{\LastHorzAlternative}}\else
                            {\gdef\HAlt{\HorzAlternative}}
           \HAlt}
        \def\HorzAlternative
            {\def\HorzAlternative
                {\MiddleHorzAlternative}\FirstHorzAlternative}
        \AllocBox\TopTrack
        \AllocBox\BotTrack
        \save\TopTrack\null
        \save\BotTrack\null
        \save0\hbox{#1}
        \vbox{\box\TopTrack\hbox{$\vtop{\box0\box\BotTrack}$}}}}

\def\FirstHorzAlternative
 #1{\save0\hbox{\QLine#1\I}
    \save\TopTrack\hbox{\box\TopTrack\ForwQ\hbox to 1wd0{\II\Fil}}
    \save\BotTrack\hbox{\box\BotTrack\ForwQ\hskip 1wd0}
    \IV\BackQ\Qline\LinetoTop\box0\LinetoBot}

\def\MiddleHorzAlternative
 #1{\save0\hbox{\III#1\I}
    \save\TopTrack\hbox{\box\TopTrack\I\hbox to 1wd0{\BackQ\Fil}}
    \save\BotTrack\hbox{\box\BotTrack\III\hbox to 1wd0{\Fil\BackQ}}
    \ForwQ\LinetoTop\box0\LinetoBot}

\def\LastHorzAlternative
 #1{\save0\hbox{\III#1\QQLine\BackQ}
    \save\TopTrack\hbox{\box\TopTrack\I}
    \save\BotTrack\hbox{\box\BotTrack\III\hbox to 1wd0{\Fil\IV}}
    \ForwQ\LinetoTop\box0\LinetoBot\II}

\def\SyntaxChart
       {\par
        \varunit\QThickness
        \lineskip0pt
        \baselineskip-1pt
        \parfillskip 0pt plus 3000pt
        \parskip \QQsize plus \Qsize
        \IgnoreLinebreaks
        \LeftToRight}

\DontIgnoreWhiteSpace % so the user can space significantly

%%%%%% End of SynChart.TEX
```

**Appendix B.** Here is a non-trivial example of the use of the syntax chart macros.

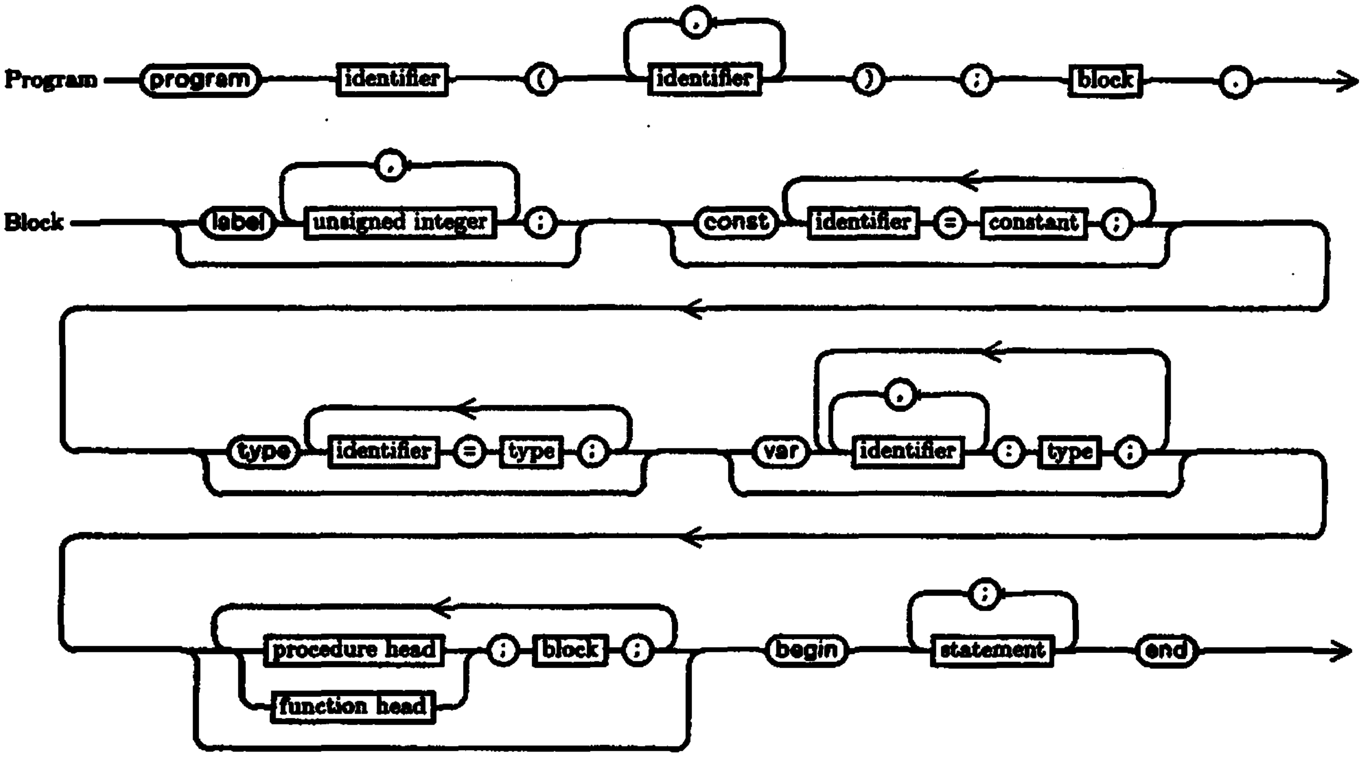
```
   %%%%%% Beginning of PascalSyntax.TEX
\font K=CMSS8 % for the keywords in the chart.

{\SyntaxChart
\AllocBox\subchartbox % for saving pieces of the diagrams

\parindent 0pt

%\def\LeftArrow{\vrule height 0.5vu depth 0.5vu width 2.5vu}
% Turn off the right arrows because they make the charts too wide.
\def\RightArrow{\vrule height 0.5vu depth 0.5vu width 2.5vu}
\def\ArrowLine{\vrule height 0.5vu depth 0.5vu width 2.5vu}
\def\NoArrows{\def\LeftArrow{}\def\RightArrow{}\def\ArrowLine{}}

% \T{<char>} yields a circular terminal node with the character
% aesthetically placed, assuming CMTT8 is used. The 4.5417pt is the
% height of plus and minus, and 4.2pt is the character width.
% Colons and periods should be followed by a \CommaStrut to make
% them line up with semicolons and commas.  For parens, brackets,
% and braces, put an extra \hss on the outer side to center them better.
% This macro may be used for multi-character terminal symbols, but
% \KW should be used for keywords.
\def\T#1{\Terminal
        {\vbox to 4.5417pt{\vss\hbox expand -4.2pt{\hss#1\hss}\vss\vfilneg}}}

\def\CommaStrut{\save0\hbox{,}\lower 1dp0\null}

\def\KW#1{\Terminal{\:K \hskip-2pt#1\Strut\hskip-2pt}} % Use this for keywords.

\def\N{\Nonterminal} % just an abbreviation.

\def\Optional#1{\Alternatives{\Middle{#1}\Lower{\Empty}}}
\def\Rept#1{\Repeat{\Upper{\BigLeftArrow}\Middle{#1}}}
\def\Star#1{\Repeat{\Upper{#1}\Middle{\Empty}}}
\def\LongStar#1{\Optional{\Rept{#1}}}
\def\TwoAlts[#1|#2]{\Alternatives
        {\Upper{\TrimBot{#1}}\Lower{\TrimTop{#2}}}}
\def\RAW#1#2{\Repeat{\Upper{#1}\Middle{#2}}} % Repeat Above With
\def\RBW#1#2{\Repeat{\Middle{#2}\Lower{#1}}} % Repeat Below With
\def\\{\Fil}
\def\-{\QQLine}

\def\goodbreak{\vfil\null\vfilneg}

\def\singlequote{\char'177} % symmetric single quote in tty font.

\def\lpar{\T{\hss(}} \def\rpar{\T{)\hss}}
\def\lbrak{\T{\hss[}} \def\rbrak{\T{]\hss}}
\def\comma{\T{,}}
\def\colon{\T{:\CommaStrut}}
\def\eq{\T{=}}
\def\semicolon{\T{;}}
\def\period{\T{.\CommaStrut}}
\def\gets{\T{\save0\hbox{>}\vbox
               {\hbox{\CenterSink{:}\CenterSink{\vbox to 1ht0{}=}}\null}}}

% editor's note - page break for TUGboat publication
```

```
\def\constant{\N{constant}}
\def\type{\N{type}}
\def\statement{\N{statement}}
\def\id{\N{identifier}}
\def\idlist{\RAW\comma\id}
\def\expression{\N{expression}}
\def\variable{\N{variable}}
\def\character
 {\Alternatives
       {\Middle{\N{character}}
        \Lower{\T{\singlequote\singlequote}}}}
\def\digits{\Rept{\N{digit}}}
\def\blockbody{\KW{begin}\\\RAW\semicolon\statement\\\KW{end}}
\def\sign
{\Alternatives
       {\Upper{\T{+}}
        \Middle{}
        \Lower{\T{-}}}}

\Define{Program}
\KW{program}\\\id\\\lpar\\\RAW\comma\id\\\rpar\\\semicolon\\\N{block}\\\period
\Fil\EndDef

{ % Block
\def\labelpart
 {\Optional{\KW{label}\\\RAW\comma{\N{unsigned integer}}\\\semicolon}}

\def\constpart
 {\Optional{\KW{const}\\\Rept{\id\\\eq\\\constant\\\semicolon}}}

\def\typepart
 {\Optional{\KW{type}\\\Rept{\id\\\eq\\\type\\\semicolon}}}

\def\varpart
 {\Optional{\KW{var}\\\Rept{\idlist\\\colon\\\type\\\semicolon}}}

\def\prochead{\N{procedure head}}
\def\funchead{\N{function head}}
\def\pfhead{\Alternatives{\Middle{\prochead}\Lower{\funchead}}}

\def\pfdeclpart{\LongStar{\pfhead\\\semicolon\\\N{block}\\\semicolon}}


\Define{Block}
\Fil\labelpart\\\constpart\Fil\NewLine
\Fil\typepart\\\varpart\Fil\NewLine
\Fil\pfdeclpart\\\blockbody\Fil\Fil\EndDef

} % end of Block

\vfil\eject
% editor's note - page break for TUGboat publication
```

```
{ % Declaration Extras
\def\simpletype{\N{simple type}}
\def\typelist{\lbrak\\\RAW\comma{\\\simpletype\\}\\\rbrak}
\Define{Type}\Fil
\Alternatives
       {\Middle{\Fil\simpletype}
        \Lower{\Fil\T{\char'136}\\\N{type identifier}}
        \Lower{\Optional{\KW{packed}}\\\KW{array}\\\typelist\\\KW{of}\\\type}
        \Lower{\KW{file}\\\KW{of}\\\type\\}
        \Lower{\KW{set}\\\KW{of}\\\simpletype\\}
        \Lower{\KW{record}\\\N{field list}\\\KW{end}\\}}
\Fil\Fil\EndDef

\Define{Simple type}\Fil
\Alternatives
       {\Middle{\Fil\N{type identifier}}
        \Lower{\Fil\lpar\\\idlist\\\rpar}
        \Lower{\Fil\constant\\\T{..\CommaStrut}\\\constant}}
\Fil\Fil\EndDef

% \vfil\eject

\def\varianthead{\KW{case}\\\id\\\colon\\\N{type identifier}\\\KW{of}}
\def\variant{\RAW\comma\constant\\\colon\\\lpar\\\N{field list}\\\rpar}
\Define{Field list}\Fil
\LongStar{\Optional{\idlist\\\colon\\\type}\\\semicolon}\NewLine
\save\subchartbox\hbox{\Optional{\variant}}\!% this gets too complicated
\Optional{\varianthead\\\LongStar{\box\subchartbox\\\semicolon}}
\Fil\EndDef

\vfil\eject

\def\funorvar{\TwoAlts[\KW{function}|\KW{var}]}
\def\formalparmgroup
{\Alternatives
       {\Middle{\funorvar\\\idlist\\\colon\\\N{type identifier}}
        \Lower{\KW{procedure}\\\RBW\comma\id}}}
\Define{Parameter list}
\save\subchartbox\hbox{\formalparmgroup}\!% again too complicated
\Optional{\lpar\\\RAW\semicolon{\box\subchartbox}\\\rpar}
\Fil\EndDef

} % end of Declaration Extras
% \vfil\eject

{ % Statement
\def\varorfunid{\TwoAlts[\N{variable}|\N{function identifier}]}
\def\assignment{\varorfunid\\\gets\\\expression}

\def\procid{\N{procedure identifier}}
\def\actualarglist
   {\lpar\\\RAW\comma
     {\Alternatives
       {\Middle{\expression}
        \Lower{\procid}}}\\\rpar}
\def\proccall{\procid\\\Optional{\actualarglist}}

\def\ifstatement{\KW{if}\\\expression\\\KW{then}\\\statement
                 \\\Optional{\KW{else}\\\statement}}

\def\casepart{\RBW\comma\constant\\\colon\\\statement}
\def\cases{\Optional{\RAW\semicolon\casepart}}
\def\casestatement{\KW{case}\\\expression\\\KW{of}\\\cases\\\KW{end}}

\def\whilestatement{\KW{while}\\\expression\\\KW{do}\\\statement}

\def\repeatstatement{\KW{repeat}\\\RAW\semicolon\statement
```

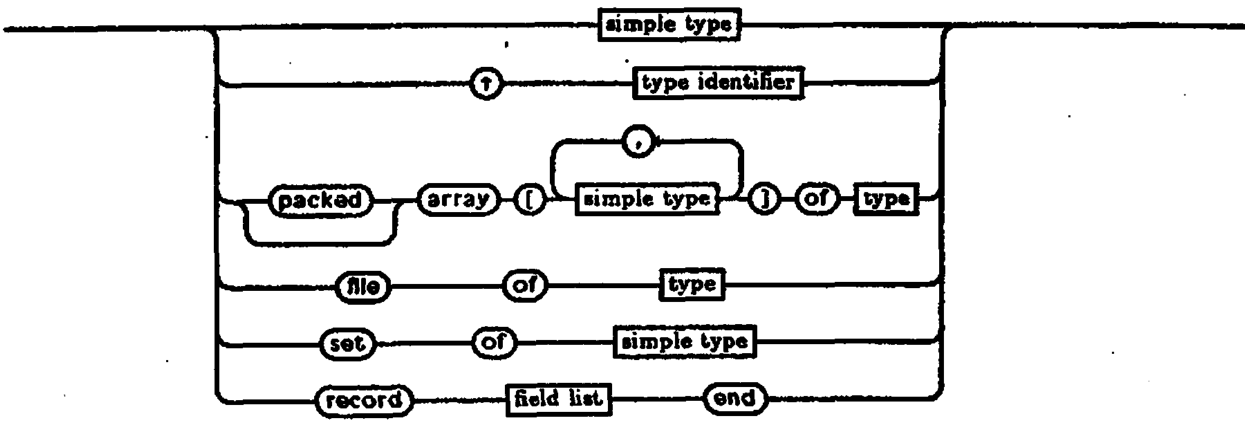
```
                    \\\KW{until}\\\expression}


\def\toordownto{\TwoAlts[\KW{to}|\KW{downto}]}
\def\forstatement{\KW{for}\\\N{variable identifier}\\\gets\\\expression
                 \\\toordownto\\\expression\\\KW{do}\\\statement}

\def\withstatement{\KW{with}\\\RAW\comma\variable\\\KW{do}\\\statement}

\def\gotostatement{\KW{goto}\\\N{unsigned integer}}


\Define{Statement}\Fil
\Optional{\N{unsigned integer}\\\colon}\\\NewLine
\Alternatives
       {\Middle{\assignment\\}
        \Lower{\proccall\\}
        \Lower{\blockbody\\\Fil}
        \Lower{\ifstatement}
        \Lower{\casestatement}
        \Lower{\whilestatement\\}
        \Lower{\repeatstatement\\}
        \Lower{\forstatement\\}
        \Lower{\withstatement\\}
        \Lower{\gotostatement\\\Fil\Fil}
        \Lower{\Empty}}
\Fil\EndDef
} % end of Statement

\vfil\eject

{ % Expression
\def\relop{{\Fil\NoArrows
    \HorzAlternatives
       {\Alternative{\T{=}}
        \Alternative{\T{<}}
        \Alternative{\T{>}}
        \Alternative{\T{<>}}
        \Alternative{\T{<=}}
        \Alternative{\T{>=}}
        \Alternative{\KW{in}}}}}
\Define{Expression}
\N{simple expression}\\
\Optional{\relop\\\N{simple expression}}
\Fil\EndDef

\Define{Simple expression}\Fil
\Alternatives{\Upper{\T{+}}\Middle{\Empty}\Lower{\T{-}}}\\
\Repeat{\Upper{\T{+}}
        \Upper{\T{-}}
        \Middle{\N{term}}
        \Lower{\KW{or}}}
\Fil\Fil\Fil\Fil\EndDef

\def\mulop{{\NoArrows
    \HorzAlternatives
       {\Alternative{\T{*}}
        \Alternative{\T{/}}
        \Alternative{\KW{div}}
        \Alternative{\KW{mod}}
        \Alternative{\KW{and}}}}}
\Define{Term}\Fil
\N{factor}\\\LongStar{\mulop\N{factor}}
\Fil\Fil\Fil\Fil\EndDef

% editor's note - page break for TUGboat publication
```

```
\def\actualarglist
        {\lpar\\\RAW\comma\expression\\\rpar}
\save\subchartbox\hbox
        {\lbrak
         \\\Optional{\RAW\comma
                         {\expression
                          \Optional{\T{..\CommaStrut}\\\!
                                    \expression}}}\\\!
         \rbrak}
\Define{Factor}\Fil
\Alternatives
        {\Middle{\N{unsigned constant}}
         \Lower{\variable}
         \Lower{\N{function identifier}\\\Optional\actualarglist}
         \Lower{\lpar\\\expression\\\rpar}
         \Lower{\KW{not}\\\N{factor}}
         \Lower{\box\subchartbox}}
\Fil\Fil\EndDef

\Define{Variable}\Fil
\TwoAlts[\N{variable identifier}|\N{field identifier}]\\
\Repeat{\Upper{\T{\char'136}}
         \Middle{\Empty}
         \Lower{\N{selector}}}
\Fil\Fil\EndDef

\'efine{Selector}\Fil
\Alternatives
        {\Upper{\period\\\N{field identifier}}
         \Lower{\lbrak\\\RBW\comma\expression\\\rbrak}}
\Fil\Fil\EndDef
} % end of Expression

\vfil\eject

\Define{Constant}\Fil
\TwoAlts[{\sign\\\TwoAlts[\N{constant identifier}|\N{unsigned number}]}
         |\T{\singlequote}\\\Rept{\character}\\\T{\singlequote}]
\Fil\Fil\EndDef

\Define{Unsigned constant}\Fil
\A.ternatives
        {\Middle{\N{constant identifier}}
         \Lower{\N{unsigned number}}
         \Lower{\KW{nil}}
         \Lower{\T{\singlequote}\\\Rept{\character}\\\T{\singlequote}}}
\Fil\Fil\EndDef

\Define{Unsigned number}\Fil
\digits\\\Optional{\period\\\digits}\\\Optional{\T{E}\\\sign\-\digits}
\Fil\Fil\EndDef

\Define{Unsigned integer}\Fil
\digits
\Fil\Fil\EndDef

\Define{Identifer}\Fil
\N{letter}\\
\Repeat{\Upper{\N{letter}}
        \Middle{}
        \Lower{\N{digit}}}
\Fil\Fil\EndDef
}
\vfil\eject
%%%%% End of PascalSyntax.TEX
```

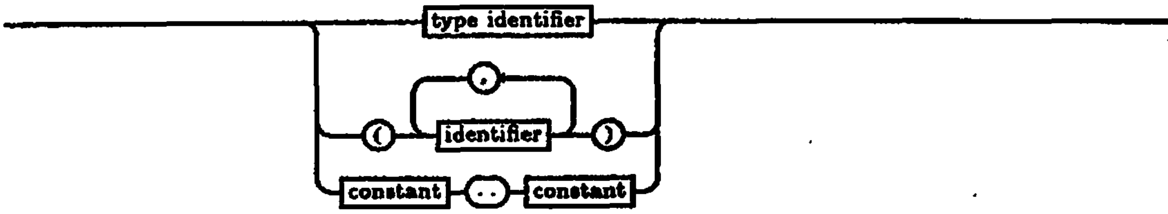


Editor's note: As you may have noticed on the first two pages of this paper, there are some problems with the font used to draw the quarter-circles. The paper copy which Mike Plass sent in with his tape looks quite a bit better, and the remainder of his diagrams are reproduced from that copy, which was prepared on one of Xerox's printers.

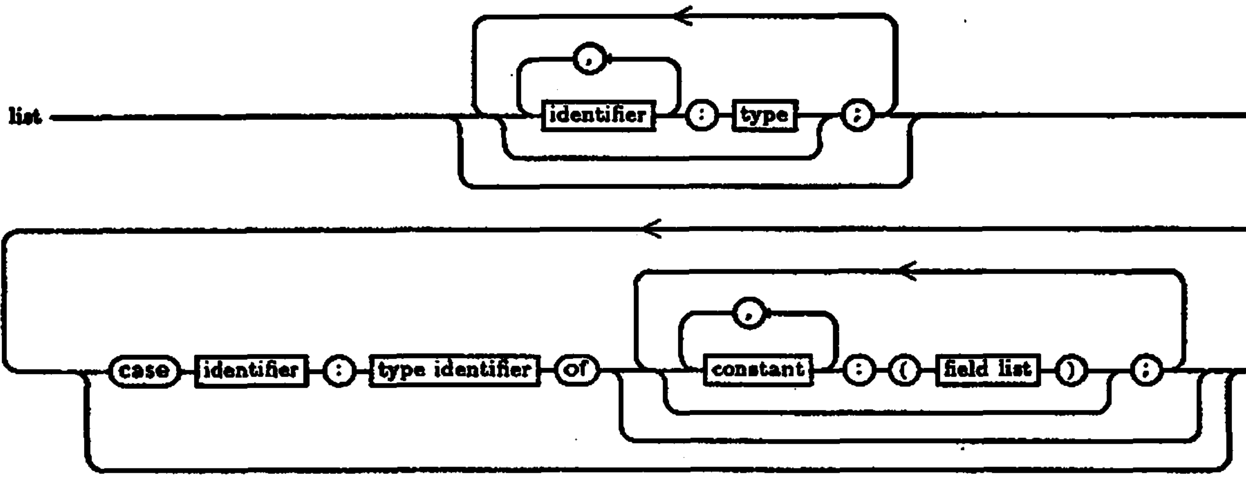
Type



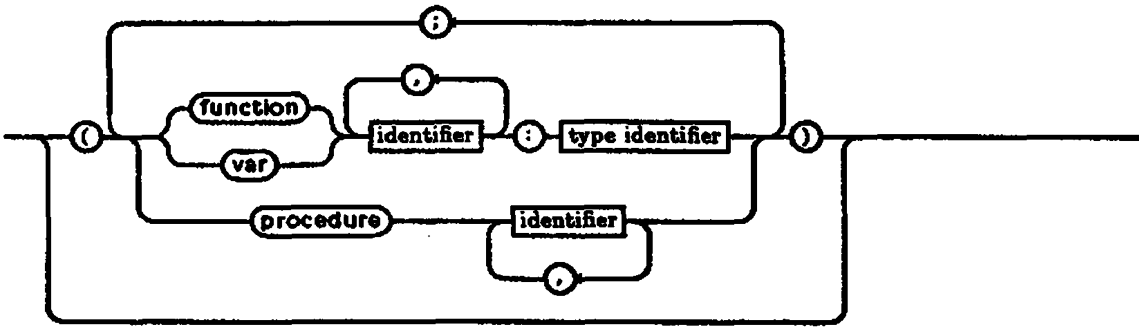
Simple type

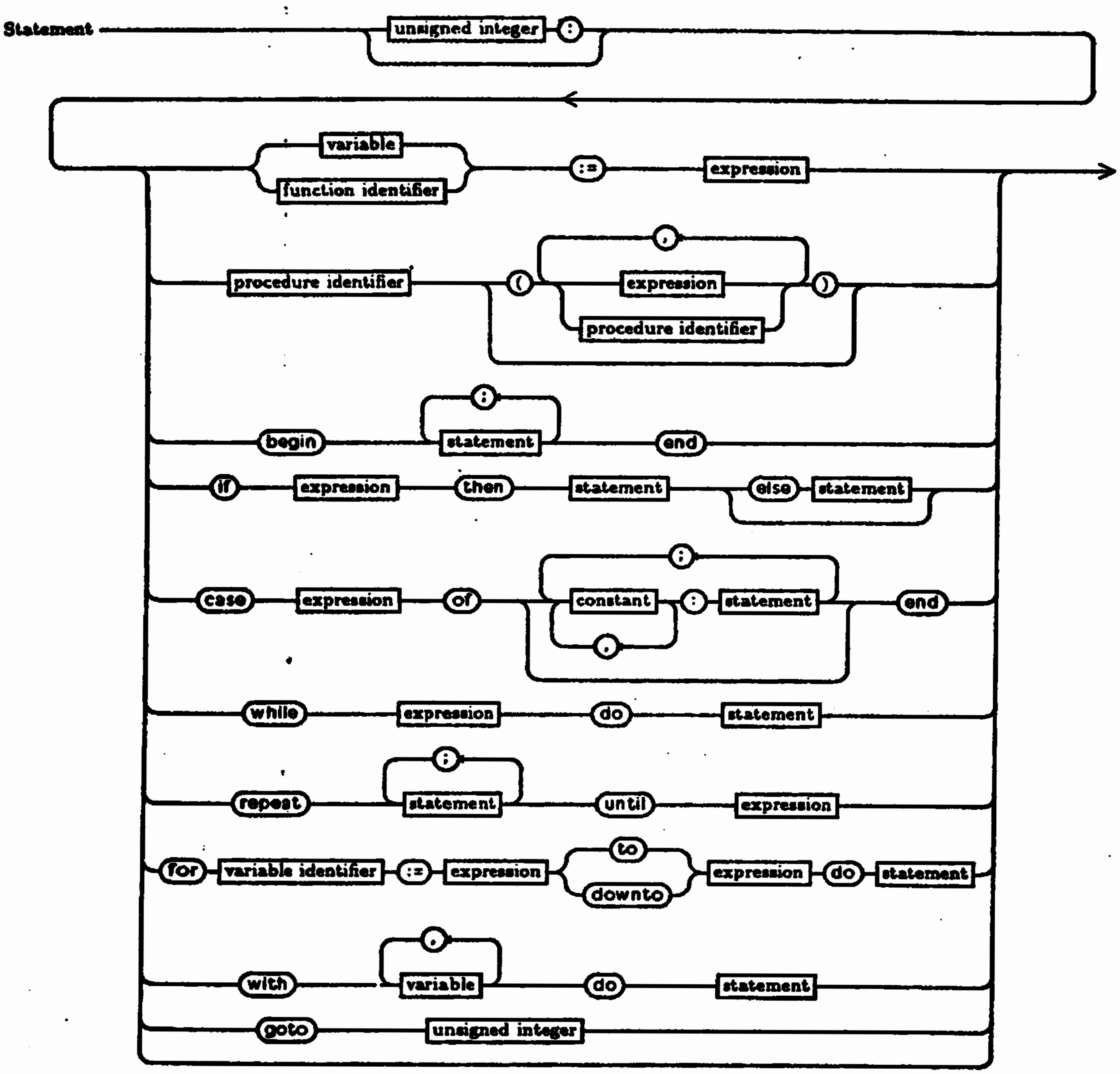


Field list

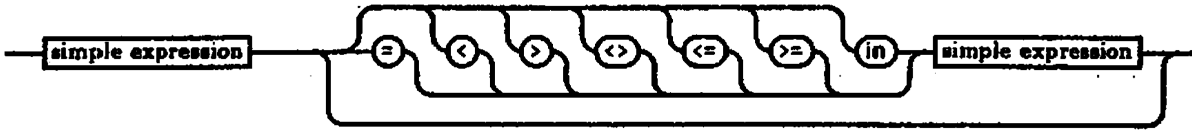


Parameter list

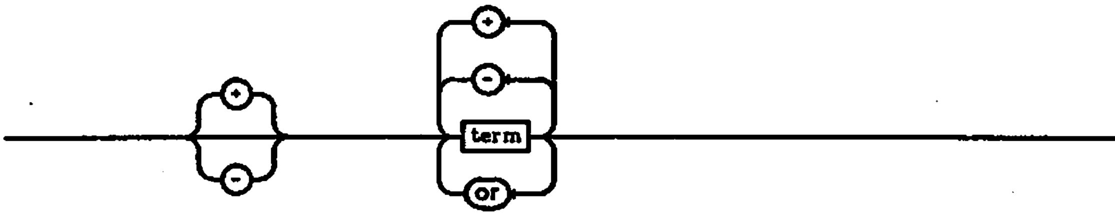



Statement
unsigned integer
:
variable
function identifier
:=
expression
procedure identifier
(
,
expression
procedure identifier
)
begin
;
statement
end
if
expression
then
statement
else
statement
case
expression
of
;
constant
:
statement
,
end
while
expression
do
statement
repeat
;
statement
until
expression
for
variable identifier
:=
expression
to
downto
expression
do
statement
with
,
variable
do
statement
goto
unsigned integer

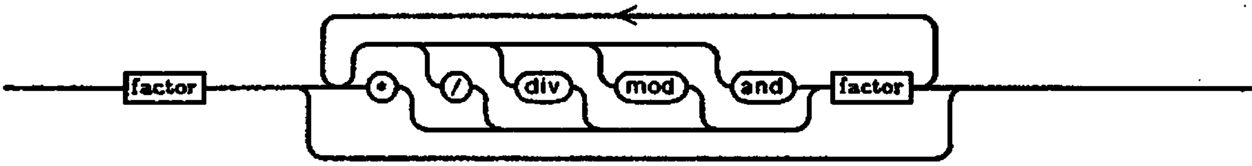
Expression



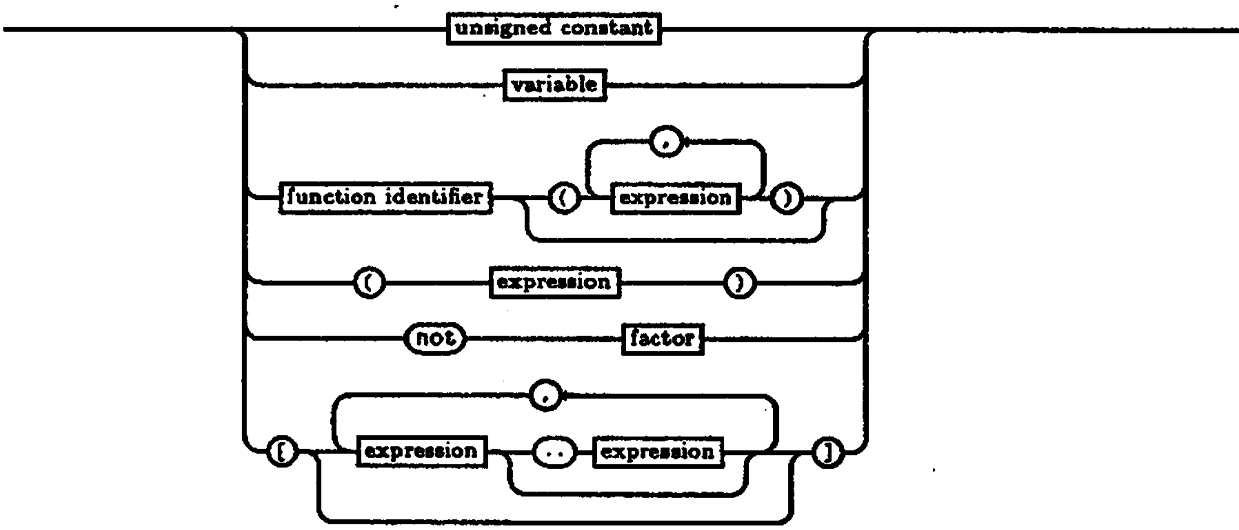
Simple expression



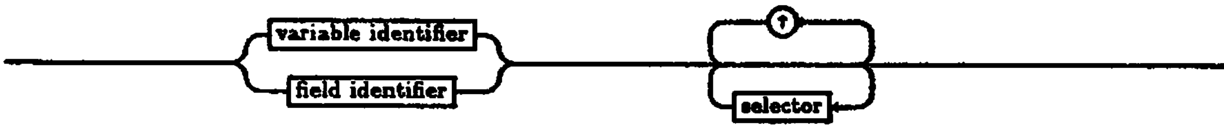
Term



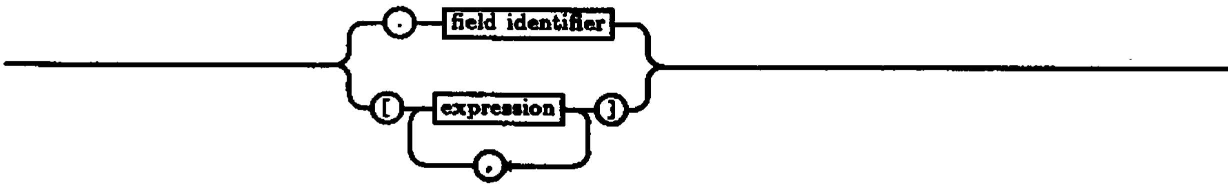
Factor

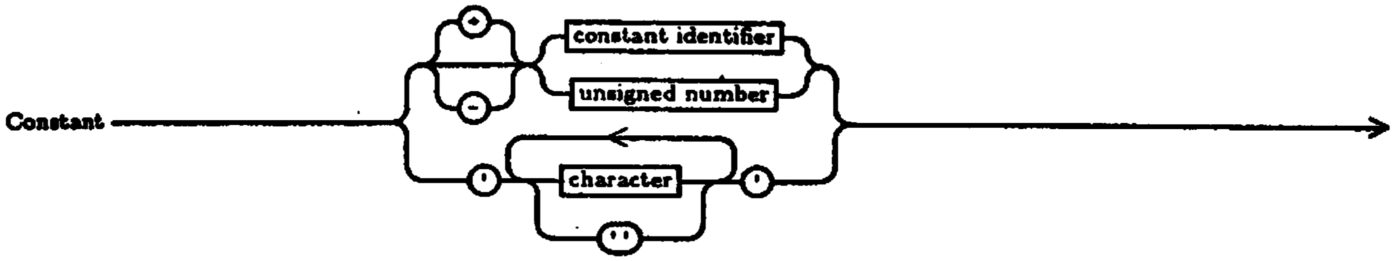


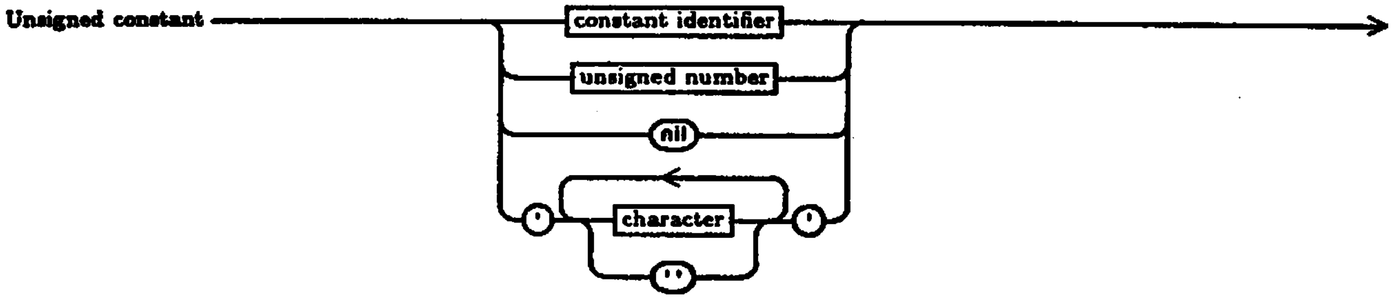
Variable

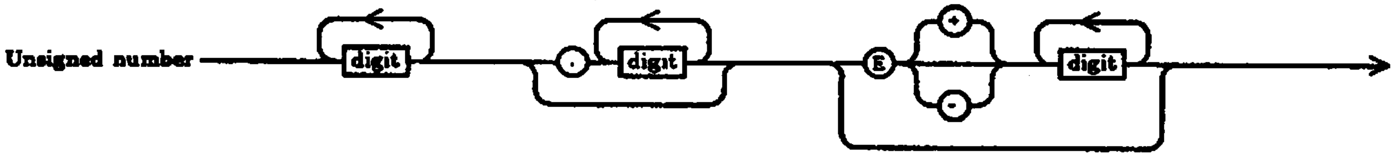


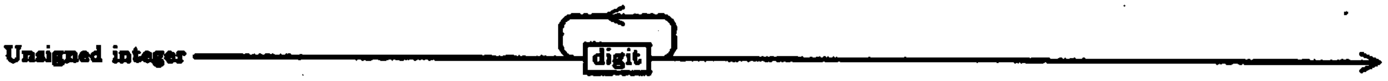
Selector

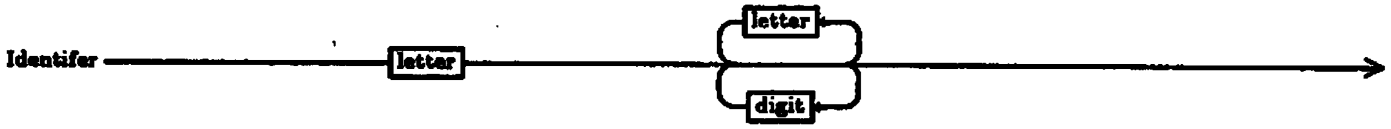



Constant
+
-
constant identifier
unsigned number
'
character
''
'


Unsigned constant
constant identifier
unsigned number
nil
'
character
''
'


Unsigned number
digit
.
digit
E
+
-
digit


Unsigned integer
digit


Identifer
letter
letter
digit