

**simplebnf — A simple package to format Backus-Naur form**

Jay Lee

2023/01/07

This package provides a simple way to typeset grammars written in Backus-Naur form (BNF).

\SimpleBNFDefEq

This command is used to typeset the definition symbol separate a nonterminal from its productions. It defaults to \texttt{::=}. It can be redefined using \texttt{RenewDocumentCommand}.

\SimpleBNFDefOr

This command is used to typeset the separator symbol between productions. It defaults to \texttt{|}. It can be redefined using \texttt{RenewDocumentCommand}.

\SimpleBNFStretch

This command is used to control the vertical spacing between consecutive rules. It defaults to 0. It can be redefined using \texttt{RenewDocumentCommand}.

\bnfexpr

This command is used when typesetting the BNF nonterminal and productions. It defaults to a wrappers around \texttt{\texttt}. It can be redefined to customized output using \texttt{RenewDocumentCommand}.

\bnfannot

This command is used when typesetting the annotations on nonterminals and productions. It defaults to a wrappers around \texttt{\textit}. It can be redefined to customized output using \texttt{RenewDocumentCommand}.

\begin{bnfgrammar} text \end{bnfgrammar}

can be used to typeset BNF grammars. The text inside the environment should be formatted as:

\begin{verbatim}
term1 ::= rhs1
  ;;
term2 ::= rhs2
\end{verbatim}
termk ::= rhs

where each of the rhs represents alternative syntactic forms of the term. An annotation may accompany each alternative in which case the alternative must be separated from its annotation with a colon (\:). If you don’t need annotations, simply omit the colons and annotations altogether. The alternatives themselves are separated using the pipe symbol (\|).

A sample code and the result is shown below:

\begin{bnfgrammar}
a \in \textit{Vars}
\;
expr ::= a \in 

| expr + term : sum
| term : term
\;
term ::= a \in 

| term * a : product
| a :
\;
\end{bnfgrammar}

Annotations can also be provided on left-hand sides, to label the nonterminal instead of a specific production.

\begin{bnfgrammar}
a : Variables \in 

expr : Expressions ::= a \in 

| expr + term :
| term :
\;
term ::= a \in 

| term * a :
| a :
\;
\end{bnfgrammar}

You can also provide an optional specification to the grammar environment, to redefine alignment or spacing.
\begin{bnfgrammar}
\[ \begin{lrdef}
\text{expr} ::= \text{expr} + \text{term} : \text{sum} \\
| \text{term} : \text{term} \\
\end{lrdef} \\
\text{term} ::= \text{term} * \text{a} : \text{product} \\
| \text{a} : \text{variable} \\
\end{bnfgrammar} \\

If you want to typeset multiple productions on a single line, you can use double vertical bars by default.

\begin{bnfgrammar}
\begin{lrdef}
\text{expr} ::= \text{expr} + \text{term} | \text{term} \\
\end{lrdef} \\
\\text{term} ::= \text{term} * \text{a} | \text{a} \\
\end{bnfgrammar} \\

The second and third optional arguments specify regular expressions for the line-breaking and non-breaking RHS separators:

\begin{bnfgrammar}
\begin{lrdef}
\text{expr} ::= \text{expr} + \text{term} | \text{term} \\
\end{lrdef} \\
\\text{term} ::= \text{term} * \text{a} | \text{a} \\
\end{bnfgrammar} \\

3