

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 `::=`. It can be redefined using `RenewDocumentCommand`.

`\SimpleBNFDefOr`

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

`\SimpleBNFStretch`

This command is used to control the vertical spacing between consecutive rules. It defaults to 0. It can be redefined using `Renewdocumentcommand`.

`\bnfexpr`

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

`\bnfannot`

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

`\begin{bnfgrammar} text\end{bnfgrammar}`

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

```
term1 ::= rhs1
;;
term2 ::= rhs2
```

*This file describes v0.3.2.

[†]E-mail: jaeho.lee@snu.ac.kr

```
;;
...
termk ::= rhsk
```

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:

<pre>\begin{bnfgrammar} a \in \textit{Vars} : variables ;; expr ::= expr + term : sum term : term ;; term ::= term * a : product a : variable \end{bnfgrammar}</pre>	<pre> a ∈ Vars variables expr ::= expr + term sum term term term ::= term * a product a variable</pre>
--	---

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

<pre>\begin{bnfgrammar} a : Variables \in \textit{Vars} ;; expr : Expressions ::= expr + term term ;; term ::= term * a a \end{bnfgrammar}</pre>	<pre> Variables a ∈ Vars Expressions expr ::= expr + term term term ::= term * a a</pre>
--	--

You can also provide an optional specification to the grammar environment, to redefine alignment or spacing.

```

Variables      a ∈ Vars
expr ::= expr + term  sum
      | term         term
term ::= term * a     product
      | a            variable
```

```

\begin{bnfgrammar}[lr@{\hspace{4pt}}c@{\hspace{2pt}}ll]
  a : Variables \in \textit{Vars}
  ;;
  expr ::=
    expr + term : sum
  | term      : term
  ;;
  term ::=
    term * a : product
  | a       : variable
\end{bnfgrammar}

```

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

<pre> \begin{bnfgrammar} a \in \textit{Vars} ;; expr ::= expr + term term ;; term ::= term * a a \end{bnfgrammar} </pre>	<pre> a ∈ Vars expr ::= expr + term term term ::= term * a a </pre>
--	--

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

```

a      ∈  Vars
expr  ::= expr + term | term
term  ::= term * a
        | a

```

```

\begin{bnfgrammar}[llc1l][\|\|][\|]
  a \in \textit{Vars}
  ;;
  expr ::= expr + term | term
  ;;
  term ::= term * a
  || a
\end{bnfgrammar}

```