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

Jay Lee†

2023/11/23

This package provides a simple way for typesetting grammars in Backus-Naur form (BNF). It features a flexible configuration system, allowing for the customization of the domain-specific language (DSL) used in typesetting the grammar. Additionally, the package comes with sensible defaults.

Below is the metagrammar of the DSL as defined in this package, which is typeset using the package itself.

Gramar	G	::=	P	production
		|	P ⁘	production w/ a trailing delimiter
		|	P ⁘ G	production sequence
Production	P	::=	$L \mathrel{-\!\!\triangleright} R$	
LHS	L	::=	v	metavariable
		|	v // c	annotated metavariable
RHS	R	::=	[]	delimiter
		|	A [] R	alternative sequence
Alternative	A	::=	f	syntactic form
		|	f // c	annotated syntactic form
Prod. delimiter	⁘	::=	; ;	default symbol
		|	⋯	user-defined
Rule relation	$\mathrel{-\!\!\triangleright}$	::=	: : =	::=
		|	->	→
		|	\in	∈
		|	⋯	user-defined
Annot. symbol	//	::=	:	default symbol
		|	⋯	user-defined
Alt. delimiter	[]	::=	|	new-line delimiter
		|	| |	single-line delimiter
		|	⋯	user-defined
	v, f, c	∈	TEX tl	valid TEX token lists

*This file describes v1.0.0.

†E-mail: jaeho.lee@snu.ac.kr

1 For the impatient

TBD

`\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
;;
...
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:

```
\begin{bnfgrammar}
  a \in \textit{Vars} :
   variables
  ;;
  expr ::=
    expr + term : sum
  | term        : term
  ;;
  term ::=
    term * a : product
  | a        : variable
\end{bnfgrammar}
```

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

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

```
\begin{bnfgrammar}
  a : Variables \in
   \textit{Vars}
  ;;
  expr : Expressions ::=
    expr + term
  | term
  ;;
  term ::=
    term * a
  | a
\end{bnfgrammar}
```

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

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.

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

$$
\begin{array}{lcl}
\mathsf{a} & \in & \mathsf{Vars} \\
\mathsf{expr} & ::= & \mathsf{expr + term} \mid \mathsf{term} \\
\mathsf{term} & ::= & \mathsf{term * a} \mid \mathsf{a}
\end{array}
$$

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

$$
\begin{array}{lcl}
\mathsf{a} & \in & \mathsf{Vars} \\
\mathsf{expr} & ::= & \mathsf{expr + term} \mid \mathsf{term} \\
\mathsf{term} & ::= & \mathsf{term * a} \\
 & \mid & \mathsf{a}
\end{array}
$$

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