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

Jay Lee[†] 2023/11/25

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. 1

Grammar	G	::=	P	production
			$P\ ec{\circ}$	production w/ a trailing delimiter
			$P \stackrel{\circ}{\circ} G$	production sequence
Production	P	::=	$L \rightarrow R$	
LHS	L	::=	ν	metavariable
			ν [] c	annotated metavariable
RHS	R	::=		delimiter
			A [] R	alternative sequence
Alternative	A	::=	f	syntactic form
			<i>f</i>	annotated syntactic form
Prod. delimiter	° 9	::=	;;	default symbol
			•••	user-defined
Rule relation	\rightarrow	::=	::=	::=
			->	\rightarrow
			\in	€
				user-defined
Annot. symbol		::=	:	default symbol
			•••	user-defined
Alt. delimiter		::=	I	new-line delimiter
	-		П	single-line delimiter
				user-defined
	v, f, c	€	T _E X tl	valid T _E X token lists

 $^{^*}$ This file describes v1.0.0.

[†]E-mail: jaeho.lee@snu.ac.kr

¹The code is shown in Appendix.

1 Tutorial for the impatient

Typesetting a grammar is as simple as typing the BNF grammar in the bnf environment:

```
\begin{center}
\begin{bnf}
 $\tau$ : \textsf{Type} ::=
  | \texttt{num} : numbers
  | \texttt{str} : strings
 ;;
                                                                                numbers
                                             Type
                                                             num
 $e$ : \textsf{Expr} ::=
                                                                                strings
  | $x$ : variable
                                                                                variable
                                                             \boldsymbol{x}
  | $n$ : numeral
                                                                                numeral
                                                             n
  | \texttt{$e$ + $e$} : addition
                                                                                addition
                                                             e + e
  | \texttt{$e$ * $e$} :
                                                                                multiplication
    multiplication
                                                             e ^ e
                                                                                concatenation
  | \texttt{$e$
                                                                                length
                                                             len(e)
    \textasciicircum{} $e$} :
                                                             let x = e_1 in e_2
                                                                                definition
    concatenation
  | \texttt{len($e$)} : length
  | \text{texttt}\{let $x$ = $e_1$ in
    $e_2$} : definition
\end{bnf}
\end{center}
```

Typically, each column in the BNF grammar has the same font style. In the example above, the comments in the first column, e.g., Type, is typeset in sans-serif, and the second column in math mode, e.g., τ . The column for the right-hand side is (mostly) typeset in typewriter font with a bit of non-terminals like e typeset in math mode.

simplebnf provides a straightforward way to customize the font styles for each column²:

```
\begin{bnf}[
 colspec = {llcll},
 column{1} = {font = \sffamily},
 column{2} = {mode = dmath},
 column{4} = {font = \ttfamily},
 \tau : Type ::=
 | num : numbers
 | str : strings
 ;;
 e : Expr ::=
 | $x$ : variable
 | $n$ : numeral
 | $e$ + $e$ : addition
 | $e$ * $e$ : multiplication
 | $e$ \textasciicircum{} $e$ : concatenation
 | len($e$) : length
 | let $x$ = $e_1$ in $e_2$ : definition
\end{bnf}
```

If you find yourself using the same configuration repeatedly, you can fix the desired configuration using \SetBNFLayout. Once set using \SetBNFLayout, the configuration is applied to all subsequent bnf environments:

²Note that you must provide a manual alignment specification with the key colspec.

```
% Some where above...
\SetBNFLayout{
  colspec = {llcll},
  column{1} = {font = \sffamily},
  column{2} = {mode = dmath},
  column{4} = {font = \ttfamily},
}
% ...
\begin{bnf}
  \tau : Type ::=
  | num : numbers
  | str : strings
  e : Expr ::=
  | $x$ : variable
  | $n$ : numeral
  | $e$ + $e$ : addition
  | $e$ * $e$ : multiplication
  | $e$ \textasciicircum{} $e$ : concatenation
  | len($e$) : length
  | let $x$ = $e_1$ in $e_2$ : definition
\end{bnf}
```

Since these customizations are provided by the backend tabularray³, consult its documentation for more details. The corresponding command is \SetTblrInner.

For more advanced customization, such as changing default delimiters and symbols, continue to section 2 – *Configuration*.

2 Configuration

While the default configuration should be sufficient for most use cases, some grammars may require using different delimiters and symbols. For example, the language itself may contain -> as a syntactic form, which conflicts with one of the default symbols for the rule relation.

To this end, simplebnf provides a number of options to configure the DSL used to typeset the grammar. Some examples of customizable symbols include the delimiters for productions and alternatives, the relation symbol between the left-hand side and the right-hand side, the annotation symbol, and more.

The default configuration is shown in Table 1.

Table 1: Default key-values for the configuration of simplebnf.

Key	Default value
prod-delim	;;
new-line-delim	VI
single-line-delim	//
comment	:
relation	{::= -> :in:}

Continued on next page

³https://github.com/lvjr/tabularray

Table 1: Default key-values for the configuration of simplebnf. (Continued)

The key-value list can be provided as an optional argument to the bnf environment, wrapped in (). The optional key-value list for the layout—wrapped in []—should be provided after the configuration key-value list. See subsection 2.9 – *Layout* for more details.

An exmaple of customizing the configuration is shown below:

```
\begin{center}
\begin{bnf}(
  prod-delim = {--},
  new-line-delim = {\kappa},
  single-line-delim = {\&\&},
  comment = \{//\},
  relation-sym-map =
      {->} = {\ensuremath{\hookrightarrow}},
      {:in:} = {\cdot ensuremath{\cdot in}},
    },
  or-sym = \{\},
)[
  colspec = {llcll},
  column{1} = {font = \sffamily},
  column{2} = {mode = dmath},
  column{4} = {font = \ttfamily},
  \tau // Type ->
  & num // numbers
  & str // strings
  e // Expr ->
  & $x$ // variable
  & $n$ // numeral
  & $e$ + $e$ // addition
  & $e$ * $e$ // multiplication
  & $e$ \textasciicircum{} $e$ // concatenation
  & len($e$) // length
  & let x = e_1 in e_2 // definition
\end{bnf}
\end{center}
```

```
numbers
Type \tau
                num
                                     strings
                 str
                                     variable
Expr e \hookrightarrow
                                     numeral
                                     addition
                e + e
                                     multiplication
                 e * e
                 e ^ e
                                     concatenation
                 len(e)
                                     length
                let x = e_1 in e_2 definition
```

Furthermore, the configuration can be set globally using \SetBNFConfig, similar to \SetBNFLayout:

```
% Some where above...
\SetBNFConfig{
 prod-delim = \{--\},\
 new-line-delim = \{\\&\},
 single-line-delim = {\&\},
 comment = \{//\},
 relation-sym-map =
      {->} = {\ensuremath{\hookrightarrow}},
      {:in:} = {\ensuremath{\in}},
   },
 or-sym = {},
}
% ...
\begin{bnf}[
 colspec = {llcll},
 column{1} = {font = \sffamily},
 column{2} = {mode = dmath},
 column{4} = {font = \ttfamily},
 \tau // Type ->
 & num // numbers
 & str // strings
 e // Expr ->
 & $x$ // variable
 & $n$ // numeral
 & $e$ + $e$ // addition
 & $e$ * $e$ // multiplication
 & $e$ \textasciicircum{} $e$ // concatenation
 & len($e$) // length
 & let x = e_1 in e_2 // definition
\end{bnf}
```

2.1 Production delimiter

The default production delimiter prod-delim is ;;. This will separate different productions in the same grammar.

2.2 New-line alternative delimiter

The default new-line alternative delimiter new-line-delimis $\$, which will actually match verbatim $\$ in the grammar. This will separate different alternatives into different lines in the same production.

2.3 Single-line alternative delimiter

The default single-line alternative delimiter single-line-delim is //. The difference between the single-line alternative delimiter and the new-line alternative delimiter is that the former will not add a new line after the delimiter.

For consecutive alternatives delimited by the single-line alternative delimiter, only the last of them can be annotated.

Note that the single-line alternative delimiter must not contain the new-line alternative delimiter as a substring.

2.4 Annotation delimiter

The default annotation delimiter comment is :. This will separate the syntactic form and the annotation within the alternative.

2.5 Rule relations and the symbol map

simplebnf provides a ::=, ->, and :in: for the rule relation between the left-hand side and the right-hand side. Each of them is mapped to a symbol in the relation-sym-map key-value list; By default, they are mapped to ::= (\ensuremath{\Coloneqq}), \rightarrow (\ensuremath{\to}), and \in (\ensuremath{\in}), respectively.

To provide a custom relation symbol, provide the desired symbol and the mapping to the relation-sym-map key-value list and to the relation.

2.6 Or symbol

The default or symbol or-sym is \$\\$. Probably the most common use case for this option is to remove the or symbol altogether by setting it to {}.

2.7 Production separation

The default production separation prod-sep is 2pt. This will add a vertical space between the productions.

2.8 Row separation

The default row separation row-sep is 0pt. This will add a vertical space between the alternatives in the same production.

2.9 Layout

As mentioned in section 1 – *Tutorial for the impatient*, the layout of the bnf environment can be customized per-environment using the key-value list in the optional argument wrapped in []. When provided, these should be provided after the configuration key-value list, which is wrapped in ().

The optional key-value list for the layout is passed to the backend tabularray package. An example of customizing the layout is shown below:

```
\begin{center}
\begin{bnf}(
  prod-delim = {--},
  or-sym = \{\},
  prod-sep = 5pt,
)[
  colspec = {|[2pt, gray5]11c11},
  column{1} = {font = \sffamily},
  column{2} = {mode = dmath},
  column{4} = {font = \ttfamily},
  column{5} = {font = \interesting},
                                                                                    numbers
                                              Type
                                                              num
                                                         ::=
  cell{9}{4} = {cmd = \backslash fbox},
                                                              str
                                                                                    strings
  row{-} = {bg = azure9},
                                                                                    variable
                                              Expr
                                                         ::=
                                                              \boldsymbol{x}
                                                                                    numeral
                                                              n
  \tau : Type ::=
                                                                                    addition
  | num : numbers
                                                                                    multiplication
  | str : strings
                                                                                    concatenation
                                                               len(e)
                                                                                    length
  e : Expr ::=
                                                                                    definition
                                                               let x = e_1 in e_2
  | $x$ : variable
  | $n$ : numeral
  | $e$ + $e$ : addition
  | $e$ * $e$ : multiplication
  | $e$ \textasciicircum{} $e$ :
    concatenation
  | len($e$) : length
  | let $x$ = $e_1$ in $e_2$ :
    definition
\end{bnf}
\end{center}
```

3 Caveats

The choices of delimiters and symbols should be carefully made, in order to avoid conflicts with the grammar of the target language and our meta-language, T_EX.

Another subtlety to consider while customizing delimiters, is that certain delimiters not contain the others as a substring. For instance, as mentioned in section 2 – *Configuration*, the single-line alternative delimiter must not contain the new-line alternative delimiter as a substring.

When using custom OpenType fonts with the unicode-math package in XTATEX or LualATEX, beware of the fact that many fonts lack the \Coloneqq symbol, :=. The easiest way to fix this is to actually use a font that provides the symbol, such as Garamond-Math, which comes with the TEXLive installation⁴.

```
\setmathfont{Garamond-Math.otf}[
   Scale = MatchUppercase,
   range = {\Coloneq}
]
```

4 The deprecated bnfgrammar environment

Prior to version 1.0.0, the package provided a sole environment bnfgrammar for typesetting grammars. While it is still available, it is now deprecated and will be removed in a future release.

⁴\Coloneq is not a typo of \Coloneqq.

Using it will result in a deprecation warning.

To migrate to the new bnf environment, simply replace the bnfgrammar environment with bnf, and replace \in with :in: and || with //. Moreover, the new environment does not center the grammar by default, so wrapping it in a center environment is necessary to reproduce the same output. While there will be some differences in the spacings and the font styles, it is an easy fix. See section 1 – *Tutorial for the impatient* for more details.

Below is an example of using the deprecated bnfgrammar environment, left here for historical reasons.

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

Appendix

The following is the code used to typeset the metagrammar of the simplebnf DSL in the first page.

```
\begin{bnf}(
  prod-delim = ;;;,
  new-line-delim = !,
  single-line-delim = ?,
  comment = //,
  relation = {:::=|:in:},
  relation-sym-map =
      \{:::=\} = \Coloneqq\$,
      {:in:} = {in},
    },
)[
  colspec = lrcll,
  column{2} = {mode=dmath},
  column{4} = {mode=text, font=\ttfamily},
 G // Grammar :::=
  ! $P$ // production
  ! $P \fatsemi {}$ // production w/ a trailing delimiter
  ! $P \fatsemi G$ // production sequence
;;;
 P // Production :::= $L \rightarrowtriangle R$
;;;
 L // LHS :::=
  ! $v$ // metavariable
 ! $v\!\fatslash\,c$ // annotated metavariable
;;;
 R // RHS :::=
 ! $\talloblong$ // delimiter
 ! $A \talloblong R$ // alternative sequence
;;;
```

```
A // Alternative :::=
  ! $f$ // syntactic form
  ! $f\!\fatslash\,c$ // annotated syntactic form
;;;
  \fatsemi // Prod. delimiter :::=
  ! ;; // default symbol
  ! $\cdots$ // user-defined
;;;
  \rightarrowtriangle // Rule relation :::=
 ! ::= // $\Coloneqq$
 ! -> // $\to$
 ! \texttt{\char`\\in} // $\in$
 ! $\cdots$ // user-defined
;;;
 \fatslash // Annot. symbol :::=
 ! : // default symbol
 ! $\cdots$ // user-defined
 \talloblong // Alt. delimiter :::=
  ! | // new-line delimiter
  ! || // single-line delimiter
 ! $\cdots$ // user-defined
;;;
 v, f, c :in: \text{TeX}{\text{TeX}{}} \ tl} // \ valid \ TeX{} \ token lists
\ensuremath{\mbox{end}\{\ensuremath{\mbox{bnf}}\}}
```