naive-ebnf: LaTeX Package for EBNF in Plain Text*

Yegor Bugayenko
yegor256@gmail.com

2023-01-30, 0.0.3

1 Introduction

This package helps render an Extended Backus-Naur Form using plain text notation:

⟨Expr⟩ → ⟨Var⟩
  | "λ" ⟨Var⟩ "." ⟨Expr⟩
  | "(" ⟨Expr⟩ ⟨Expr⟩ ")"

```
\documentclass{minimal}
\usepackage{naive-ebnf}
\usepackage{mathtools}
\begin{document}
\begin{ebnf}
<Expr> := <Var>
  | "$\lambda$" <Var> "." <Expr>
  | "$\lparen$" <Expr> <Expr> "$\rparen$"
\end{ebnf}
\end{document}
```

`ebnf` The `ebnf` environment *doesn't* add any formatting to the paragraph, but only replaces the plain text symbols, such as "`:=`" and "`<Var>`" with proper LaTeX commands. The following syntax is understood inside the `ebnf` environment:

- `:=` separates the left-hand side from the right-hand side of the production rule;
- `<...>` denotes a non-terminal (variable);
- `"..."` denotes a terminal symbol;
- `(...|...)` denotes a series of options to choose from;
- `[...]` denotes an optional substitution;
- `{...}` denotes a zero or more times repetition;

Attention: The usage of some symbols is prohibited inside terminals. Instead, the following substitutions are recommended:

- `$\lparen$` and `$\rparen$` instead of "`(`" and "`)`" (from the mathtools package);
- `$\langle$` and `$\rangle$` instead of "<" and ">";

*The sources are in GitHub at yegor256/naive-ebnf

- `$\lbrace$` and `$\rbrace$` instead of "{" and "}" (also `mathtools`);
- `$\lbrack$` and `$\rbrack$` instead of "[" and "]" (also `mathtools`);
- `$\vert$` instead of "|".

`\terminal` `\nonterminal` Inside the text, terminals and non-terminals may be formatted using two supplementary commands:

The non-terminal ⟨Var⟩ in λ-calculus may be equal to $v_1, v_2, \ldots$. Application starts with "(" and ends with ")".

```
The non-terminal \nonterminal{Var}
in $\lambda$-calculus may be equal
to $v_1, v_2, \dots$. Application
starts with \terminal{(} and ends
with \terminal{)}.
```

It's possible to use them in math-mode too, for example:

If "(" f_1 ⟨Var⟩ ")" is always true, then f_1 is a tautology.

```
If $\terminal{(} f_1 \nonterminal{Var}
\terminal{)}$ is always true, then
$f_1$ is a tautology.
```

2 Package Options

It's possible to configure the behavior of the package with the help of a few package options:

`bw` By default, some colors are used in the rendered grammar. However, the `bw` package option disables any colors and makes sure the gammar is black-and-white:

```
\usepackage[bw]{naive-ebnf}
```

3 Implementation

First, we process package options:

```
\RequirePackage{pgfopts}
\pgfkeys{
  /ebnf/.cd,
  bw/.store in=\ebnf@bw,
}
\ProcessPgfPackageOptions{/ebnf}
```

Then, we include a few packages, mostly to deal with LaTeX3 expressions:

```
\RequirePackage{filecontentsdef}
\RequirePackage{expl3}
```

`\ebnf@color` Then, we include xcolor to colorize the output a bit:

```
\makeatletter\ifdefined\ebnf@bw\else
  \RequirePackage{xcolor}
\fi
\newcommand\ebnf@color[2]
  {\ifdefined\ebnf@bw#2\else\textcolor{#1}{#2}\fi}
\makeatother
```

`\terminal` Then, we a command to render a single terminal:

```
\makeatletter
\newcommand\terminal[1]{{%
  \relax\ifmmode\else\ttfamily\fi%
  \ebnf@color{gray}{\relax\ifmmode\textsf{‘‘}\else{\sffamily‘‘}\fi}%
  #1%
  \ebnf@color{gray}{\relax\ifmmode\textsf{’’}\else{\sffamily’’}\fi}}}
\makeatother
```

`\nonterminal` Then, we a command to render a single non-terminal:

```
\makeatletter
\newcommand\nonterminal[1]{%
  \ebnf@color{gray}{\relax\ifmmode\langle\else\(\langle\)\fi}%
  \relax\ifmmode\textsf{#1}\else{\sffamily#1}\fi%
  \ebnf@color{gray}{\relax\ifmmode\rangle\else\(\rangle\)\fi}}
\makeatother
```

Then, we define supplementary commands:

```
\makeatletter
\newcommand\ebnf@optional[1]
  {\ebnf@color{gray}{[}#1\ebnf@color{gray}{]}}
\newcommand\ebnf@repetition[1]
  {\ebnf@color{gray}{\{}#1\ebnf@color{gray}{\}}}
\newcommand\ebnf@grouping[1]
  {\ebnf@color{gray}{(}#1\ebnf@color{gray}{)}}
\ExplSyntaxOn
\newcommand\ebnf@terminal[1]{
  \tl_set:Nn \l_ebnf_tl { }
  \tl_set_rescan:Nno \l_ebnf_tl { } { #1 }
  \terminal{\l_ebnf_tl}
}
\ExplSyntaxOff
\newcommand\ebnf@to
  {\ebnf@color{gray}{\(\to\)}}
\newcommand\ebnf@alternation
  {\ebnf@color{gray}{\(\vert\)}}
\newcommand\ebnf@eol{\\}
\makeatother
```

`ebnf` Then, we define the `ebnf` environment:

```
\ExplSyntaxOn
\cs_generate_variant:Nn \tl_replace_all:Nnn {Nx}
\NewDocumentEnvironment{ebnf}{}{\filecontentsdefmacro\l__ebnf_tmp_tl}{
  \endfilecontentsdefmacro
  \str_set:NV \l__ebnf_tmp_tl \l__ebnf_tmp_tl
  \str_set:Nx \l__ebnf_tmp_tl {\str_range:Nnn \l__ebnf_tmp_tl {1} {-2}}
  \regex_replace_all:nnN { \{(.+?)\} }
    {\c{ebnf@repetition}{\1}} \l__ebnf_tmp_tl
  \regex_replace_all:nnN { \((.+?)\) }
    {\c{ebnf@grouping}{\1}} \l__ebnf_tmp_tl
  \regex_replace_all:nnN { \[([^\]]+?)\] }
    {\c{ebnf@optional}{\1}} \l__ebnf_tmp_tl
  \regex_replace_all:nnN { <([A-Za-z][a-z-]+)> }
    {\c{nonterminal}{\1}} \l__ebnf_tmp_tl
```

```
  \regex_replace_all:nnN { "([^"]+)" }
    {\c{ebnf@terminal}{\1}} \l__ebnf_tmp_tl
  \regex_replace_all:nnN { \^^M\s*\| }
    {\^^M :=} \l__ebnf_tmp_tl
  \regex_replace_all:nnN { \| }
    {\c{ebnf@alternation}{}} \l__ebnf_tmp_tl
  \regex_replace_all:nnN { \^^M\s*:= }
    {\^^M \c{-}\c{hspace}{3em}\c{ebnf@alternation}{}} \l__ebnf_tmp_tl
  \regex_replace_all:nnN { := }
    {\c{ebnf@to}{}} \l__ebnf_tmp_tl
  \regex_replace_all:nnN { \^^M }
    {\c{ebnf@eol}{}} \l__ebnf_tmp_tl
  \tl_put_left:Nn \l__ebnf_tmp_tl {}
  \tl_put_right:Nn \l__ebnf_tmp_tl {}
  \l__ebnf_tmp_tl
}
\ExplSyntaxOff

\endinput
```

Change History

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.