

# naive-ebnf: L<sup>A</sup>T<sub>E</sub>X Package for EBNF in Plain Text\*

Yegor Bugayenko  
yegor256@gmail.com

2023-06-30, 0.0.7

## 1 Introduction

This package helps render an [Extended Backus-Naur Form](#) using plain text notation:

$\langle \lambda\text{-Expr} \rangle \rightarrow \langle \text{Var} \rangle$ $\quad   \text{"}\lambda\text{" } \langle \text{Var} \rangle \text{" } . \text{" } \langle \text{Expr} \rangle$ $\quad   \text{"}(\text{" } \langle \text{Expr} \rangle \langle \text{Expr} \rangle \text{"})$	<pre>1 \documentclass{minimal} 2 \usepackage{naive-ebnf} 3 \usepackage{mathtools} 4 \begin{document} 5 \begin{ebnf} 6 &lt;\$\lambda\$-Expr&gt; := &lt;Var&gt; \\ 7      "\$\lambda\$" &lt;Var&gt; "." &lt;Expr&gt; \\ 8      "\$\lparen\$" &lt;Expr&gt; &lt;Expr&gt; "\$\rparen\$" 9 \end{ebnf} 10 \end{document}</pre>
---	---

`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 L<sup>A</sup>T<sub>E</sub>X 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 special non-printable terminal symbol, like `'EOL'`;
- `( . . . | . . . )` denotes a series of options to choose from;
- `[ . . . ]` denotes an optional substitution;
- `{ . . . }` denotes a zero or more times repetition;
- `||` denotes an indented vertical bar at the beginning of the string.

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

---

\*The sources are in GitHub at [yegor256/naive-ebnf](#)

- `\lparen` and `\rparen` instead of “(” and “)” (from the [mathtools](#) package);
- `\langle` and `\rangle` instead of “<” and “>”;
- `\lbrace` and `\rbrace` instead of “{” and “}” (also [mathtools](#));
- `\lbrack` and `\rbrack` instead of “[” and “]” (also [mathtools](#));
- `\vert` instead of “|”.

`width` There is an optional argument of `ebnf` environment, which sets the width of the left-hand side of each rule (the default width is 6em):

This EBNF has a larger width of the left hand side than usual: $\langle \text{VeryLongVariable} \rangle \rightarrow \langle X \rangle \mid \langle Y \rangle$ $\langle X \rangle \rightarrow \text{"X"} \text{ EOL}$ $\langle Y \rangle \rightarrow \text{"Y"}$	<pre> 4 This EBNF has a larger width of \ 5 the left hand side than usual: \par 6 \begin{ebnf}[1.5in] 7 &lt;VeryLongVariable&gt; := &lt;X&gt;   &lt;Y&gt; \ 8 &lt;X&gt; := "X" 'EOL' \ 9 &lt;Y&gt; := "Y" \ 10 \end{ebnf} </pre>
--	--

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

The non-terminal $\langle \text{Var} \rangle$ in $\lambda$ -calculus may be equal to $v_1, v_2, \dots$ . Application starts with “(” and ends with “)”.	<pre> 6 The non-terminal \nonterminal{Var} 7 in \$\lambda\$-calculus may be equal 8 to \$v_1, v_2, \dots\$. Application 9 starts with \terminal{() and ends 10 with \terminal{)}. </pre>
---	--

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

If “ $f_1(\lambda\text{-Var})$ ” is always true, then $f_1$ is a tautology.	<pre> 6 If \$\terminal{() f_1 7 \nonterminal{\$\lambda\$-Var} 8 \terminal{)}\$ is always true, then 9 \$f_1\$ is a tautology. </pre>
---	--

Special symbols are interpreted correctly, if they stay inside quotes:

$\langle X \rangle \rightarrow \text{EOL} \text{ " ' "   " }$ $\langle Y \rangle \rightarrow \text{">" "<" "]" "[ " }$ $\langle Z \rangle \rightarrow \text{"\LaTeX" "\$ "}$	<pre> 1 \documentclass{minimal} 2 \usepackage[T1]{fontenc} 3 \usepackage{naive-ebnf} 4 \begin{document}\noindent 5 \begin{ebnf}[1.5in] 6 &lt;X&gt; := 'EOL' " ' "   " \ 7 &lt;Y&gt; := "&gt;" "&lt;" "]" "[ " \ 8 &lt;Z&gt; := "\LaTeX" "\textdollar" \ 9 \end{ebnf} 10 \end{document} </pre>
--	---

## 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 grammar is black-and-white:

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

**trail** The `ebnf` environment is doing pre-processing of the  $\TeX$  commands provided and then let  $\LaTeX$  render them. It may be useful to see the output generated by the pre-processing. The `trail` option (with a file name) asks the package to save the content of the environment after the pre-processing into the file:

```
\usepackage[trail=log.tex]{naive-ebnf}
```

## 3 Implementation

First, we process package options:

```
1 \RequirePackage{pgfopts}
2 \pgfkeys{
3   /ebnf/.cd,
4   bw/.store in=\ebnf@bw,
5   trail/.store in=\ebnf@trail,
6   trail/.default=naive-ebnf.tmp.tex,
7 }
8 \ProcessPgfPackageOptions{/ebnf}
```

Then, we include a few packages, mostly to deal with  $\LaTeX$ 3 expressions:

```
9 \RequirePackage{expl3}
```

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

```
10 \makeatletter\ifdefined\ebnf@bw\else
11   \RequirePackage{xcolor}
12 \fi
13 \newcommand\ebnf@color[2]
14   {\ifdefined\ebnf@bw#2\else\textcolor{#1}{#2}\fi}
15 \makeatother
```

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

```
16 \makeatletter
17 \newcommand\terminal[1]{\{%
18   \relax\ifmmode\else\ttfamily\fi%
19   \ebnf@color{gray}{\relax\ifmmode\textsf{''}\else{\sffamily''}\fi}%
20   #1%
21   \ebnf@color{gray}{\relax\ifmmode\textsf{''}\else{\sffamily''}\fi}}
22 \makeatother
```

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

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

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

```
29 \makeatletter
30 \newcommand\sterminal[1]{\relax\ifmmode\else\ttfamily\fi#1}}%
31 \makeatother
```

Then, we define supplementary commands:

```
32 \makeatletter
33 \newcommand\ebnf@optional[1]
34   {\ebnf@color{gray}{\{#1\ebnf@color{gray}{\}}}}
35 \newcommand\ebnf@repetition[1]
36   {\ebnf@color{gray}{\{#1\ebnf@color{gray}{\}}}}
37 \newcommand\ebnf@grouping[1]
38   {\ebnf@color{gray}{\{#1\ebnf@color{gray}{\}}}}
39 \ExplSyntaxOn
40 \newcommand\ebnf@terminal[1]{
41   \tl_set:Nn \l_ebnf_tl {}
42   \tl_set_rescan:Nnn \l_ebnf_tl {} { #1 }
43   \terminal{\l_ebnf_tl}
44 }
45 \newcommand\ebnf@sterminal[1]{
46   \tl_set:Nn \l_ebnf_tl {}
47   \tl_set_rescan:Nnn \l_ebnf_tl {} { #1 }
48   \sterminal{\l_ebnf_tl}
49 }
50 \newcommand\ebnf@nonterminal[1]{
51   \tl_set:Nn \l_ebnf_tl {}
52   \tl_set_rescan:Nnn \l_ebnf_tl {} { #1 }
53   \nonterminal{\l_ebnf_tl}
54 }
55 \ExplSyntaxOff
56 \newcommand\ebnf@to
57   {\ebnf@color{gray}{\(\to)}}
58 \newcommand\ebnf@alternation
59   {\ebnf@color{gray}{\(\vert)}}
60 \makeatother
```

ebnf Then, we define the ebnf environment:

```
61 \ExplSyntaxOn
62 \cs_generate_variant:Nn \tl_replace_all:Nnn {Nx}
63 \makeatletter
64 \NewDocumentEnvironment{ebnf}{0{4em}+b}
65   {\tl_set:Nn \l__ebnf_tmp_tl{#2}}
66   {%
67   \regex_replace_all:nnN { ([^\ ]< } {\1\\textless{}} \l__ebnf_tmp_tl%
68   \regex_replace_all:nnN { >([^\ ] ) } {\1\\textgreater{}} \l__ebnf_tmp_tl%
69   \regex_replace_all:nnN { ([^\ ])'([^\ ] ) } {\1\\textquotesingle{}} \l__ebnf_tmp_tl%
70   \regex_replace_all:nnN { ([^\ ])\|([^\ ] ) } {\1\\textbar{}} \l__ebnf_tmp_tl%
71   %
72   \regex_replace_all:nnN { \{ \ (.*?) \ } }%
73     {\c{ebnf@repetition}{\1}} \l__ebnf_tmp_tl%
74   \regex_replace_all:nnN { \[ \ (.*?) \ ] }%
75     {\c{ebnf@grouping}{\1}} \l__ebnf_tmp_tl%
76   \regex_replace_all:nnN { \[ \ (.*?) \ ] }%
77     {\c{ebnf@optional}{\1}} \l__ebnf_tmp_tl%
```

```

78 \regex_replace_all:nnN { (<[^\>]+?>\ :=) }%
79   {\c{makebox}[#1][r]{\1}} \l__ebnf_tmp_tl%
80 \regex_replace_all:nnN { <(.*?)> }%
81   {\c{ebnf@nonterminal}{\1}} \l__ebnf_tmp_tl%
82 \regex_replace_all:nnN { "(.+?)" }%
83   {\c{ebnf@terminal}{\1}} \l__ebnf_tmp_tl%
84 \regex_replace_all:nnN { '(.+?)' }%
85   {\c{ebnf@sterminal}{\1}} \l__ebnf_tmp_tl%
86 \regex_replace_all:nnN { \\\(\\) }%
87   {\c{makebox}[#1][r]{ \1 }} \l__ebnf_tmp_tl%
88 \regex_replace_all:nnN { \\ }%
89   {\c{ebnf@alternation}{}} \l__ebnf_tmp_tl%
90 \regex_replace_all:nnN { := }%
91   {\c{ebnf@to}{}} \l__ebnf_tmp_tl%
92 \tl_put_left:Nn \l__ebnf_tmp_tl {\noindent}
93 \tl_put_right:Nn \l__ebnf_tmp_tl {}
94 \ifdefined\ebnf@trail%
95   \newwrite\ebnf@write%
96   \immediate\openout\ebnf@write\ebnf@trail\relax%
97   \immediate\write\ebnf@write{\unexpanded\expandafter{\l__ebnf_tmp_tl}}%
98   \immediate\closeout\ebnf@write%
99   \message{naive-ebnf:\space pre-processed\space TeX
100     \space saved\space to\space "\ebnf@trail"^^J}%
101 \fi%
102 \l__ebnf_tmp_tl}
103 \makeatother
104 \ExplSyntaxOff
105 \endinput

```

## Change History

0.0.1		0.0.4
General: First draft. . . . .	3	<code>ebnf</code> : Any symbols are allowed inside <code>\nonterminal</code> commands and inside the <code>ebnf</code> environment, where non-terminals are mentioned. . . . . 4
0.0.2		0.0.5
General: Proper parsing of grouping. . . . .	3	General: New package option <code>trail</code> added, to enable saving of the generated $\TeX$ content to a file, for debugging purposes. . . . . 3
Substitutions suggested for special symbols. . . . .	3	0.0.6
<code>\nonterminal</code> : New command <code>\nonterminal</code> added, to enable rendering non-terminal symbols outside of the <code>ebnf</code> environment. . . . .	3	<code>\sterminal</code> : New command <code>\sterminal</code> added, to enable rendering of special non-printable terminal symbols outside of the <code>ebnf</code> environment. . . . . 4
<code>\terminal</code> : New command <code>\terminal</code> added, to enable rendering terminal symbols outside of the <code>ebnf</code> environment. . . . .	3	
0.0.3		
<code>\terminal</code> : Quotes fixed in both text and math modes. . . . .	3	

# 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.

<b>Symbols</b>		<b>P</b>
\( ..... 25, 27, 57, 59, 74	\expandafter ..... 97	\pgfkeys ..... 2
\) ..... 25, 27, 57, 59, 74	\ExplSyntaxOff .. 55, 104	\ProcessPgfPackageOptions
\[ ..... 76	\ExplSyntaxOn .... 39, 61	..... 8
\{ ..... 36, 72	<b>I</b>	
\} ..... 36, 72	\ifdefined .... 10, 14, 94	<b>R</b>
\_ 67, 68, 69, 70, 72, 74, 76, 78	\ifmmode ..... 18,	\rangle ..... 27
\] ..... 76	19, 21, 25, 26, 27, 30	\regex ..... 67, 68, 69,
\  ..... 70, 86, 88	\immediate .... 96, 97, 98	70, 72, 74, 76, 78,
		80, 82, 84, 86, 88, 90
<b>Numbers</b>		\relax ..... 18, 19,
\2 ..... 69, 70	\l 41, 42, 43, 46, 47, 48, 51,	21, 25, 26, 27, 30, 96
	52, 53, 65, 67, 68,	\RequirePackage . 1, 9, 11
	69, 70, 73, 75, 77,	
	79, 81, 83, 85, 87,	<b>S</b>
<b>C</b>		\sffamily ..... 19, 21, 26
\c ..... 73, 75, 77, 79,	89, 91, 92, 93, 97, 102	\space ..... 99, 100
81, 83, 85, 87, 89, 91	\langle ..... 25	\sterminal ..... 29, 48
\closeout ..... 98	<b>M</b>	
\cs ..... 62	\makeatletter .....	<b>T</b>
	. 10, 16, 23, 29, 32, 63	\terminal ..... 16, 43
<b>E</b>		\textcolor ..... 14
\ebnf ..... 61	\makeatother .....	\textsf ..... 19, 21, 26
\ebnf@alternation .. 58	15, 22, 28, 31, 60, 103	\tl ..... 41, 42, 46, 47,
\ebnf@bw ..... 4, 10, 14	\message ..... 99	51, 52, 62, 65, 92, 93
\ebnf@color .....	<b>N</b>	
... 10, 19, 21, 25,	\newcommand ..... 13,	\to ..... 57
27, 34, 36, 38, 57, 59	17, 24, 30, 33, 35,	\ttfamily ..... 18, 30
\ebnf@grouping ..... 37	37, 40, 45, 50, 56, 58	
\ebnf@nonterminal .. 50	\NewDocumentEnvironment	<b>U</b>
\ebnf@optional ..... 33	..... 64	\unexpanded ..... 97
\ebnf@repetition ... 35	\newwrite ..... 95	<b>V</b>
\ebnf@sterminal .... 45	\noindent ..... 92	\vert ..... 59
\ebnf@terminal ..... 40	\nonterminal ..... 23, 53	
\ebnf@to ..... 56	<b>O</b>	
\ebnf@trail . 5, 94, 96, 100	\openout ..... 96	<b>W</b>
\ebnf@write . 95, 96, 97, 98		\write ..... 97
\endinput ..... 105		