

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

Yegor Bugayenko
yegor256@gmail.com

2023-07-14, 0.0.13

NB! Large ENBF snippets may take too long to render!

1 Introduction

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

$\begin{array}{l} \langle \lambda\text{-Expr} \rangle \rightarrow \langle \text{Var} \rangle \\ \text{"}\lambda\text{" } \langle \text{Var} \rangle \text{"}. \text{" } \langle \text{Expr} \rangle \\ \text{"}(\text{" } \langle \text{Expr} \rangle \langle \text{Expr} \rangle \text{"}) \end{array}$	<pre>1 \documentclass{minimal} 2 \usepackage{naive-ebnf} 3 \usepackage{mathtools} 4 \begin{document} 5 \begin{ebnf} 6 <\$\lambda\$-Expr> := <Var> \\\ 7 "\$\lambda\$" <Var> "." <Expr> \\\ 8 "\char'\" <Expr> <Expr> "\char'\)" 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 \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 special non-printable terminal symbol, like 'EOL';
- (... | ...) denotes a series of options to choose from;
- /.../ denotes a regular expression, like /[a-z]+/;
- [...] denotes an optional substitution;
- {...} denotes a zero or more times repetition;
- || denotes an indented vertical bar at the beginning of the string.

*The sources are in GitHub at [yegor256/naive-ebnf](https://github.com/yegor256/naive-ebnf)

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 “>”;
- `\lbrace` and `\rbrace` instead of “{” and “}” (also [mathtools](#));
- `\lbrack` and `\rbrack` instead of “[” and “]” (also [mathtools](#));
- `\vert` instead of “|”.

They would look even better, if the following notation is used:

- `\char‘\` and `\char‘\` instead of “(” and “)”;
- `\char‘<` and `\char‘>` instead of “<” and “>”;
- `\char‘{` and `\char‘}` instead of “{” and “}”;
- `\char‘[` and `\char‘]` instead of “[” and “]”.

`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 <VeryLongVariable> := <X> <Y> \ 8 <X> := "X" 'EOL' \ 9 <Y> := "Y" \ 10 \end{ebnf}</pre>
--	---

`\EbnfTerminal` Inside the text, terminals, non-terminals, and special terminals may be formatted using three supplementary commands:
`\EbnfNonTerminal`
`\EbnfSpecial`

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

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

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

`\EbnfRegex` A regular expression is possible too:

<pre> ⟨data⟩ → ⟨bool⟩ ⟨integer⟩ ⟨byte⟩ ⟨bool⟩ → 'TRUE' 'FALSE' ⟨integer⟩ → / (+ -)?[0-9]+/ ⟨byte⟩ → /[0-9a-f]{2}/ ⟨number⟩ → /[1-9]+/ / [0-9]+/ </pre>	<pre> 6 \begin{ebnf} 7 <data> := <bool> <integer> <byte> \\ 8 <bool> := "TRUE" "FALSE" \\ 9 <integer> := /(+\char' -)?[0-9]+/ \\ 10 <byte> := /[0-9a-f]\char'{2}\char'\}/ \\ 11 <number> := /[1-9]+/ / [0-9]+/ 12 \end{ebnf} </pre>
--	---

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

<pre> ⟨X⟩ → EOL " ' " " ⟨Y⟩ → ">" "<" "[" "]" " / " " / " ⟨Z⟩ → "⋈" "\$" </pre>	<pre> 5 \begin{ebnf} 6 <X> := 'EOL' " ' " " \\ 7 <Y> := ">" "<" "[" "]" " / " " / " \\ 8 <Z> := "⋈" "\$" \\ 9 \end{ebnf} </pre>
---	---

Nested brackets work fine too:

<pre> ⟨x⟩ → ("x" ("y" ("z" ⟨z⟩))) ⟨y⟩ → ["x1"] {/[a-z]+/} ⟨z⟩ → {{{⟨x⟩} ⟨y⟩} ⟨z⟩} ⟨t⟩ → [⟨x⟩] [⟨y⟩] </pre>	<pre> 5 \begin{ebnf} 6 % There is no meaning in this: 7 <x> := ("x" ("y" ("z" <z>))) \\ 8 <y> := [["x1"] {/[a-z]+/}] \\ 9 <z> := { { { <x> } <y> } <z> } \\ 10 <t> := [<x>] [<y>] \\ 11 \end{ebnf} </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 gammar is black-and-white:

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

trail The **ebnf** environment is doing pre-processing of the \TeX commands provided and then let \TeX 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 L^AT_EX3 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

```

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

```

16 \makeatletter
17 \newcommand\EbnfTerminal[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

```

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

```

23 \makeatletter
24 \newcommand\EbnfNonTerminal[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

```

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

```

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

```

`\EbnfRegex` Then, we define a command to render a regular expression:

```

32 \makeatletter
33 \newcommand\EbnfRegex[1]{\relax\ifmmode\else\ttfamily\fi/#1/}%
34 \makeatother

```

Then, we define supplementary commands:

```

35 \makeatletter
36 \newcommand\ebnf@optional[1]
37   {\ebnf@color{gray}{[ ]#1\ebnf@color{gray}{ ]}}
38 \newcommand\ebnf@repetition[1]
39   {\ebnf@color{gray}{\{ \}#1\ebnf@color{gray}{ \}}}
40 \newcommand\ebnf@grouping[1]
41   {\ebnf@color{gray}{( )#1\ebnf@color{gray}{ )}}
42 \ExplSyntaxOn
43 \newcommand\ebnf@terminal[1]{
44   \tl_set:Nn \l_ebnf_tl {}
45   \tl_set_rescan:Nnn \l_ebnf_tl {} { #1 }

```

```

46 \EbnfTerminal{\l_ebnf_tl}
47 }
48 \newcommand\ebnf@special[1]{
49 \tl_set:Nn \l_ebnf_tl {}
50 \tl_set_rescan:Nnn \l_ebnf_tl {} { #1 }
51 \EbnfSpecial{\l_ebnf_tl}
52 }
53 \newcommand\ebnf@nonterminal[1]{
54 \tl_set:Nn \l_ebnf_tl {}
55 \tl_set_rescan:Nnn \l_ebnf_tl {} { #1 }
56 \EbnfNonTerminal{\l_ebnf_tl}
57 }
58 \newcommand\ebnf@regex[1]{
59 \tl_set:Nn \l_ebnf_tl {}
60 \tl_set_rescan:Nnn \l_ebnf_tl {} { #1 }
61 \EbnfRegex{\l_ebnf_tl}
62 }
63 \ExplSyntaxOff
64 \newcommand\ebnf@to
65 {\ebnf@color{gray}{\(\to\)}}
66 \newcommand\ebnf@alternation
67 {\ebnf@color{gray}{\(\vert\)}}
68 \makeatother

```

ebnf Then, we define the ebnf environment:

```

69 \ExplSyntaxOn
70 \cs_generate_variant:Nn \tl_replace_all:Nnn {Nx}
71 \makeatletter
72 \NewDocumentEnvironment{ebnf}{0{4em}+b}
73 {\tl_set:Nn\ebnf_tmp{#2}}
74 {%
75 \regex_replace_all:nnN
76 { ([^s])/([~s]) } {\1\\slash{}\2} \ebnf_tmp%
77 \regex_replace_all:nnN
78 { ([^s])< } {\1\\textless{}} \ebnf_tmp%
79 \regex_replace_all:nnN
80 { >([~s]) } {\1\\textgreater{}} \ebnf_tmp%
81 \regex_replace_all:nnN
82 { ([~s])'([~s]) } {\1\\textquotesingle{}\2} \ebnf_tmp%
83 \regex_replace_all:nnN
84 { ([~s])\|([~s]) } {\1\\textbar{}\2} \ebnf_tmp%
85 %
86 \regex_replace_all:nnN
87 { /(.)?/ }%
88 {\c{ebnf@regex}{\1}} \ebnf_tmp%
89 \cs_new:Npn\ebnf_curled{%
90 \regex_replace_all:nnNT
91 { \{s([~s]*([s[~\]\{\}\|s(\)\|\{([~s])?)*)s\} }%
92 {\c{ebnf@repetition}{\1}} \ebnf_tmp \ebnf_curled}%
93 \ebnf_curled%
94 \cs_new:Npn\ebnf_brackets{%
95 \regex_replace_all:nnNT
96 { \([s([~s]*([s[~\]\(\)\|s(\)\|\{([~s])?)*)s\)) }%
97 {\c{ebnf@grouping}{\1}} \ebnf_tmp \ebnf_brackets}%

```

```

98 \ebnf_brackets%
99 \cs_new:Npn\ebnf_squares{%
100 \regex_replace_all:nnNT
101 { \[\s(([\^\\s]*(\s[\^\\]\[\]| \s\[\]| \[\][\^\\s]))?)*\s\] }%
102 {\c{ebnf@optional}{\1}} \ebnf_tmp \ebnf_squares}%
103 \ebnf_squares%
104 \regex_replace_all:nnN { (<[\^>]+?>\s:=) }%
105 {\c{makebox}[#1][r]{\1}} \ebnf_tmp%
106 \regex_replace_all:nnN { <(.+?)> }%
107 {\c{ebnf@nonterminal}{\1}} \ebnf_tmp%
108 \regex_replace_all:nnN { "(.+?)" }%
109 {\c{ebnf@terminal}{\1}} \ebnf_tmp%
110 \regex_replace_all:nnN { '(.+?)' }%
111 {\c{ebnf@special}{\1}} \ebnf_tmp%
112 \regex_replace_all:nnN { \|(\|) }%
113 {\c{makebox}[#1][r]{ \| }} \ebnf_tmp%
114 \regex_replace_all:nnN { \| }%
115 {\c{ebnf@alternation}{}} \ebnf_tmp%
116 \regex_replace_all:nnN { := }%
117 {\c{ebnf@to}{}} \ebnf_tmp%
118 \tl_put_left:Nn \ebnf_tmp {\noindent}
119 \tl_put_right:Nn \ebnf_tmp {}
120 \ifdefined\ebnf@trail%
121 \newwrite\ebnf@write%
122 \immediate\openout\ebnf@write\ebnf@trail\relax%
123 \immediate\write\ebnf@write{\unexpanded\expandafter{\ebnf_tmp}}%
124 \immediate\closeout\ebnf@write%
125 \message{naive-ebnf:\space pre-processed\space TeX
126 \space saved\space to\space "\ebnf@trail"^^J}%
127 \fi%
128 \ebnf_tmp}
129 \makeatother
130 \ExplSyntaxOff

131 \endinput

```

Change History

0.0.1	General: First draft.	3	0.0.4	<code>ebnf</code> : Any symbols are allowed inside <code>\EbnfNonTerminal</code> commands and inside the <code>ebnf</code> environment, where non-terminals are mentioned.	5
0.0.11	<code>ebnf</code> : Many bugs fixed in the area of regular expression matching.	5			
0.0.2	General: Proper parsing of grouping. . .	3	0.0.5	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			
	<code>\EbnfTerminal</code> : New command <code>\EbnfNonTerminal</code> added, to enable rendering non-terminal symbols outside of the <code>ebnf</code> environment.	4	0.0.6	<code>\EbnfSpecial</code> : New command <code>\EbnfSpecial</code> added, to enable rendering of special non-printable terminal symbols outside of the <code>ebnf</code> environment.	4
	New command <code>\EbnfTerminal</code> added, to enable rendering terminal symbols outside of the <code>ebnf</code> environment.	4	0.0.8	<code>\EbnfRegex</code> : New command <code>\EbnfRegex</code> added, to enable rendering of regular expresions outside of the <code>ebnf</code> environment. . .	4
0.0.3	<code>\EbnfTerminal</code> : Quotes fixed in both text and math modes.	4			

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.

Symbols		O	
\(..... 25, 27, 65, 67, 96	\ebnf@write 122
\) 25, 27, 65, 67, 96	...	121, 122, 123, 124
\[..... 101	\EbnfNonTerminal	.. 24, 56
\{ 39, 91	\EbnfRegex 32, 61
\} 39, 91	\EbnfSpecial 29, 51
\] 101	\EbnfTerminal	.. 16, 23, 46
\ 84, 112, 114	\endinput 131
Numbers		\expandafter 123
\2 76, 82, 84	\ExplSyntaxOff	.. 63, 130
C		\ExplSyntaxOn 42, 69
\c	88, 92, 97, 102, 105, 107, 109, 111, 113, 115, 117	R	
\closeout 124	\rangle 27
\cs 70, 89, 94, 99	\regex 75, 77, 79, 81, 83, 86, 90, 95, 100, 104, 106, 108, 110, 112, 114, 116
E		\relax 18, 19, 21, 25, 26, 27, 30, 33, 122
\ebnf	. 69, 73, 76, 78, 80, 82, 84, 88, 89, 92, 93, 94, 97, 98, 99, 102, 103, 105, 107, 109, 111, 113, 115, 117, 118, 119, 123, 128	\RequirePackage	.. 1, 9, 11
\ebnf@alternation	.. 66	S	
\ebnf@bw 4, 10, 14	\sffamily 19, 21, 26
\ebnf@color 10, 19, 21, 25, 27, 37, 39, 41, 65, 67	\space 125, 126
\ebnf@grouping 40	T	
\ebnf@nonterminal	.. 53	\textcolor 14
\ebnf@optional 36	\textsf 19, 21, 26
\ebnf@regex 58	\tl	. 44, 45, 49, 50, 54, 55, 59, 60, 70, 73, 118, 119
\ebnf@repetition	.. 38	\to 65
\ebnf@special 48	\ttfamily 18, 30, 33
\ebnf@terminal 43	U	
\ebnf@to 64	\unexpanded 123
L		V	
\l	. 44, 45, 46, 49, 50, 51, 54, 55, 56, 59, 60, 61	\vert 67
\langle 25	W	
M		\write	
\makeatletter	... 10, 16, 23, 29, 32, 35, 71 123	
\makeatother 15, 22, 28, 31, 34, 68, 129		
\message 125		
N			
\newcommand	. 13, 17, 24, 30, 33, 36, 38, 40, 43, 48, 53, 58, 64, 66		
\NewDocumentEnvironment 72		
\newwrite 121		
\noindent 118		