

naive-ebnf: L^AT_EX Package for EBNF in Plain Text*

Yegor Bugayenko
yegor256@gmail.com

2026-10-15, 0.1.0

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

1 Introduction

This package renders the [Extended Backus-Naur Form](#) by means of a plain-text notation:

$\begin{array}{l} \langle \lambda\text{-Expr} \rangle \rightarrow \langle \text{Var} \rangle \\ \quad \text{ "}\lambda\text{" } \langle \text{Var} \rangle \text{ "}. " \langle \text{Expr} \rangle \\ \quad \text{ "(" } \langle \text{Expr} \rangle \langle \text{Expr} \rangle \text{ ")" } \end{array}$	<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 *does not* apply any formatting to the paragraph; rather, it merely substitutes plain-text symbols, such as “:=” and “<Var>”, with the appropriate L^AT_EX commands. The following syntax is recognised within the **ebnf** environment:

- := separates the left-hand side from the right-hand side of a production rule;
- <...> denotes a non-terminal (variable);
- "... " denotes a terminal symbol;
- '...' denotes a special, non-printable terminal symbol, such as 'EOL';
- (... | ...) denotes a series of alternatives among which one is selected;
- /.../ denotes a regular expression, such as /[a-z]+/;
- [...] denotes an optional substitution;

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

- $\{...\}$ denotes a repetition of zero or more occurrences;
- $\{...\}^+$ denotes a repetition of one or more occurrences;
- $|||$ denotes an indent at the beginning of a line;
- $||$ denotes an indented vertical bar at the beginning of a line.

Brackets of all three kinds are recognised only when whitespace separates them from their content: $(\langle x \rangle \mid \langle y \rangle)$ is a group, whereas $(\langle x \rangle \mid \langle y \rangle)$ is left as plain text. The same rule is what keeps $"($ a terminal.

Attention: Braces never print within a terminal: a balanced pair vanishes into a $\text{T}_{\text{E}}\text{X}$ group, an unbalanced one stops the compilation. The symbols $(,), <, >, [,]$ and $|$ print literally as long as no space is adjacent to them, as in $"($ or $"a < b"$; once surrounded by spaces, as in $"a < b"$, they are taken for EBNF syntax. The following substitutions are recommended in all other cases:

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

A still more agreeable rendering may be obtained through the following notation:

- $\backslash char \backslash ($ and $\backslash char \backslash)$ instead of $"($ and $)"$;
- $\backslash char \backslash <$ and $\backslash char \backslash >$ instead of $"<"$ and $">"$;
- $\backslash char \backslash \{$ and $\backslash char \backslash \}$ instead of $"{"$ and $"}"$;
- $\backslash char \backslash [$ and $\backslash char \backslash]$ instead of $"["$ and $"]"$.

width The `ebnf` environment accepts an optional argument that determines the width allocated to the left-hand side of each rule (the default value being `4em`):

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

\EbnfTerminal Within the text, terminals, non-terminals, and special terminals may be typeset
\EbnfNonTerminal by means of three supplementary commands:
\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 $)"$.	6 7 8 9 10	The non-terminal $\backslash EbnfNonTerminal\{\text{Var}\}$ in $\$ \backslash lambda \$$ -calculus may be equal to $\$ v_1, v_2, \backslash dots \$$. Application starts with $\backslash EbnfTerminal\{ \}$ and ends with $\backslash EbnfTerminal\{ \}$.
--	------------------------	--

These commands are likewise admissible within math mode, 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\lambda\text{Var}} 8 \EbnfTerminal\{\}\\$ is always true, then 9 \$f_1\$ is a tautology.</pre>
---	--

\EbnfRegex A regular expression is also admissible:

$\langle \text{data} \rangle \rightarrow \langle \text{bool} \rangle \mid \langle \text{integer} \rangle \mid \langle \text{byte} \rangle$ $\langle \text{bool} \rangle \rightarrow \text{"TRUE"} \mid \text{"FALSE"}$ $\langle \text{integer} \rangle \rightarrow /(+ -)?[0-9]+/$ $\langle \text{byte} \rangle \rightarrow /[0-9a-f]{2}/$ $\langle \text{number} \rangle \rightarrow /[1-9]+/ \mid /[0-9]+/$	<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]+/ \mid /[0-9]+/ 12 \end{ebnf}</pre>
---	--

Special symbols print literally when they sit right next to the quotes:

$\langle X \rangle \rightarrow \text{EOL} \text{"'"} \text{" "} \text{"}$ $\langle Y \rangle \rightarrow \text{">"} \text{"<"} \text{"["} \text{"]"} \text{" /"} \text{" /"} \text{"}$ $\langle Z \rangle \rightarrow \text{"\LaTeX"} \text{"\\$"} \text{"}$	<pre> 5 \begin{ebnf} 6 <X> := 'EOL' "' " " \\ 7 <Y> := ">" "<" "[" "]" "/" "/" \\ 8 <Z> := "\LaTeX" "\textdollar" \\ 9 \end{ebnf}</pre>
---	---

Nested brackets are likewise handled correctly:

$\langle x \rangle \rightarrow (\text{"x"} (\text{"y"} \mid (\text{"z"} \mid \langle z \rangle)))$ $\langle y \rangle \rightarrow [\text{"x1"}] \{ / [a-z] + / \}$ $\langle z \rangle \rightarrow \{ \{ \{ \langle x \rangle \}^+ \langle y \rangle \} \langle z \rangle \}^+$ $\langle t \rangle \rightarrow [\langle x \rangle] [\langle y \rangle]$	<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>
--	--

The ||| sequence permits the indentation of text on a new line, thus enabling the breaking of long expressions:

$\langle x \rangle \rightarrow \text{"beginning"}$ $(\langle y \rangle \mid \langle z \rangle)$ "ending"	<pre> 5 \begin{ebnf} 6 <x> := "beginning" \\ 7 (<y> <z>) \\ 8 "ending" \\ 9 \end{ebnf}</pre>
---	--

2 Package Options

The behaviour of the package may be configured by means of several package options:

bw By default, certain colours are employed in the rendered grammar. The **bw** package option, however, disables all colours and ensures that the grammar is

rendered in black and white:

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

trail The `ebnf` environment performs a pre-processing step on the supplied $\text{\TeX}$ commands before delegating their rendering to $\text{\LaTeX}$. Inspection of the output produced by this pre-processing can prove useful. The `trail` option, accompanied by a file name, instructs the package to save the post-processed content of every `ebnf` environment, in document order, to the designated 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}
```

\ebnf@write If the `trail` option is set, we allocate a single output stream for the trail file, since $\text{\TeX}$ has only sixteen of them, open the file once, and close it at the end of the document, so that every `ebnf` environment appends to it:

```
9 \makeatletter
10 \ifdefined\ebnf@trail
11   \newwrite\ebnf@write
12   \immediate\openout\ebnf@write\ebnf@trail\relax
13   \AtEndDocument{\immediate\closeout\ebnf@write}
14 \fi
15 \makeatother
```

Then, we include a few packages, mostly to deal with $\text{\LaTeX3}$ expressions:

```
16 \RequirePackage{expl3}
```

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

```
17 \makeatletter\ifdefined\ebnf@bw\else
18   \RequirePackage{xcolor}
19 \fi
20 \newcommand\ebnf@color[2]
21   {\ifdefined\ebnf@bw#2\else\textcolor{#1}{#2}\fi}
22 \makeatother
```

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

```
23 \makeatletter
24 \newcommand\EbnfTerminal[1]{\{%
25   \relax\ifmmode\else\ttfamily\fi%
26   \ebnf@color{gray}{\relax\ifmmode\textsf{''}\else{\sffamily''}\fi}%
27   #1%
28   \ebnf@color{gray}{\relax\ifmmode\textsf{''}\else{\sffamily''}\fi}}
29 \makeatother
```

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

```

30 \makeatletter
31 \newcommand\EbnfNonTerminal[1]{\%
32   \ebnf@color{gray}{\relax\ifmmode\langle\else\(\langle\)\fi}%
33   \relax\ifmmode\textsf{#1}\else\sffamily#1\fi%
34   \ebnf@color{gray}{\relax\ifmmode\rangle\else\(\rangle\)\fi}}
35 \makeatother

```

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

```

36 \makeatletter
37 \newcommand\EbnfSpecial[1]{\relax\ifmmode\else\ttfamily\fi#1}}%
38 \makeatother

```

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

```

39 \makeatletter
40 \newcommand\EbnfRegex[1]{\relax\ifmmode\else\ttfamily\fi/#1/}%
41 \makeatother

```

Then, we define supplementary commands:

```

42 \makeatletter
43 \newcommand\ebnf@optional[1]
44   {\ebnf@color{gray}{\[]#1\ebnf@color{gray}{\]}}
45 \newcommand\ebnf@repetition[2][]
46   {\ebnf@color{gray}{\{}#2\ebnf@color{gray}{\}}\scriptscriptstyle #1\)}}
47 \newcommand\ebnf@grouping[1]
48   {\ebnf@color{gray}{\{ }#1\ebnf@color{gray}{\}}}
49 \ExplSyntaxOn
50 \newcommand\ebnf@terminal[1]{
51   \tl_set:Nn \l_ebnf_tl {}
52   \tl_set_rescan:Nnn \l_ebnf_tl {} { #1 }
53   \EbnfTerminal{\l_ebnf_tl}
54 }
55 \newcommand\ebnf@special[1]{
56   \tl_set:Nn \l_ebnf_tl {}
57   \tl_set_rescan:Nnn \l_ebnf_tl {} { #1 }
58   \EbnfSpecial{\l_ebnf_tl}
59 }
60 \newcommand\ebnf@nonterminal[1]{
61   \tl_set:Nn \l_ebnf_tl {}
62   \tl_set_rescan:Nnn \l_ebnf_tl {} { #1 }
63   \EbnfNonTerminal{\l_ebnf_tl}
64 }
65 \newcommand\ebnf@regex[1]{
66   \tl_set:Nn \l_ebnf_tl {}
67   \tl_set_rescan:Nnn \l_ebnf_tl {} { #1 }
68   \EbnfRegex{\l_ebnf_tl}
69 }
70 \ExplSyntaxOff
71 \newcommand\ebnf@to
72   {\ebnf@color{gray}{\(\to\)}}
73 \newcommand\ebnf@alternation
74   {\ebnf@color{gray}{\(\vert\)}}
75 \makeatother

```

ebnf Then, we define the ebnf environment:

```

76 \ExplSyntaxOn
77 \makeatletter
78 \NewDocumentEnvironment{ebnf}{0{4em}+b}
79   {\tl_set:Nn\ebnf_tmp{#2}}
80   {%
81     \regex_replace_all:nnN
82       { ([^\s])/( [^\s] ) } {\1\\slash{}\2} \ebnf_tmp%
83     \regex_replace_all:nnN
84       { ([^\s])< } {\1\\textless{}} \ebnf_tmp%
85     \regex_replace_all:nnN
86       { >([^\s]) } {\1\\textgreater{}} \ebnf_tmp%
87     \regex_replace_all:nnN
88       { ([^\s])'([^\s]) } {\1\\textquotesingle{}\2} \ebnf_tmp%
89     \regex_replace_all:nnN { \\|\\|\\| }%
90     {\c{makebox}[#1][r]{ }} \ebnf_tmp%
91     \regex_replace_all:nnN
92       { ([^\s])\|([^\s]) } {\1\\textbar{}\2} \ebnf_tmp%
93     %
94     \regex_replace_all:nnN
95       { /(.[+?)/ }%
96       {\c{ebnf@regexp}{\1}} \ebnf_tmp%
97     \cs_undefine:N\ebnf_curled%
98     \cs_new:Npn\ebnf_curled{%
99       \regex_replace_all:nnNT
100         { \\s(( [^\s]*(\s[^\s]\{ | \s(\) | \{ [^\s])? )*\s\{ ([\+*])? }%
101         {\c{ebnf@repetition}[5]{\1}} \ebnf_tmp \ebnf_curled}%
102       \ebnf_curled%
103     \cs_undefine:N\ebnf_brackets%
104     \cs_new:Npn\ebnf_brackets{%
105       \regex_replace_all:nnNT
106         { \\s(( [^\s]*(\s[^\s]\{ | \s(\) | \{ [^\s])? )*\s\{ ) }%
107         {\c{ebnf@grouping}{\1}} \ebnf_tmp \ebnf_brackets}%
108     \ebnf_brackets%
109     \cs_undefine:N\ebnf_squares%
110     \cs_new:Npn\ebnf_squares{%
111       \regex_replace_all:nnNT
112         { \\s(( [^\s]*(\s[^\s]\{ | \s(\) | \{ [^\s])? )*\s\{ ) }%
113         {\c{ebnf@optional}{\1}} \ebnf_tmp \ebnf_squares}%
114     \ebnf_squares%
115     \regex_replace_all:nnN { (<[>]+>\s:=) }%
116     {\c{makebox}[#1][r]{\1}} \ebnf_tmp%
117     \regex_replace_all:nnN { <(.+?)> }%
118     {\c{ebnf@nonterminal}{\1}} \ebnf_tmp%
119     \regex_replace_all:nnN { "(.+?)" }%
120     {\c{ebnf@terminal}{\1}} \ebnf_tmp%
121     \regex_replace_all:nnN { '(.+?)' }%
122     {\c{ebnf@special}{\1}} \ebnf_tmp%
123     \regex_replace_all:nnN { \\|\\| }%
124     {\c{makebox}[#1][r]{ \1 }} \ebnf_tmp%
125     \regex_replace_all:nnN { \\| }%
126     {\c{ebnf@alternation}{}} \ebnf_tmp%
127     \regex_replace_all:nnN { := }%
128     {\c{ebnf@to}{}} \ebnf_tmp%

```

```

129 \tl_put_left:Nn \ebnf_tmp {\noindent}
130 \ifdefined\ebnf@trail%
131   \immediate\write\ebnf@write{\unexpanded\expandafter{\ebnf_tmp}}%
132   \message{naive-ebnf:\space pre-processed\space TeX
133     \space saved\space to\space "\ebnf@trail"^^J}%
134 \fi%
135 \ebnf_tmp}
136 \makeatother
137 \ExplSyntaxOff

138 \endinput

```

Change History

0.0.1		0.0.3	
General: First draft.	4	<code>\EbnfTerminal</code> : Quotes fixed in both text and math modes. . . .	4
0.0.11		0.0.4	
<code>ebnf</code> : Many bugs fixed in the area of regular expression matching. . .	6	<code>ebnf</code> : Any symbols are allowed inside <code>\EbnfNonTerminal</code> commands and inside the <code>ebnf</code> environment, where non-terminals are mentioned. . .	6
0.0.14		0.0.5	
<code>ebnf</code> : One-or-more repetition introduced with <code>{...}+</code> syntax. . .	6	General: New package option <code>trail</code> added, to enable saving of the generated <code>TEX</code> content to a file, for debugging purposes.	4
0.0.15		0.0.6	
<code>ebnf</code> : The <code>iteration</code> removed, only <code>repetition</code> is left, with the second optional parameter. . .	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.	5
0.0.2		0.0.8	
General: Proper parsing of grouping.	4	<code>\EbnfRegex</code> : New command <code>\EbnfRegex</code> added, to enable rendering of regular expressions outside of the <code>ebnf</code> environment.	5
Substitutions suggested for special symbols.	4		
<code>\EbnfNonTerminal</code> : New command <code>\EbnfNonTerminal</code> added, to enable rendering non-terminal symbols outside of the <code>ebnf</code> environment.	5		
<code>\EbnfTerminal</code> : New command <code>\EbnfTerminal</code> added, to enable rendering terminal symbols outside of the <code>ebnf</code> environment.	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		
\(	32, 34, 46, 72, 74, 106	\ebnf@nonterminal .. 60
\)	32, 34, 46, 72, 74, 106	\ebnf@optional 43
*	 100	\ebnf@regexp 65
\+	 100	\ebnf@repetition .. 45
\[	 112	\ebnf@special 55
\{	 46, 100	\ebnf@terminal 50
\}	 46, 100	\ebnf@to 71
\]	 112	\ebnf@trail
\	 89, 92, 123, 125	5, 10, 12, 130, 133
		\ebnf@write 9, 131
		\EbnfNonTerminal ..
Numbers		 2, 30, 63
\2	 82, 88, 92	\EbnfRegex 3, 39, 68
\5	 101	\EbnfSpecial .. 2, 36, 58
		\EbnfTerminal . 2, 23, 53
A		\endinginput 138
\AtEndDocument	 13	\expandafter 131
B		\ExplSyntaxOff . 70, 137
\bw	 3	\ExplSyntaxOn ... 49, 76
C		
\c	90, 96, 101, 107, 113, 116, 118, 120, 122, 124, 126, 128	
\closeout	 13	
\cs	 97, 98, 103, 104, 109, 110	
E		
\ebnf	.. 1, 76, 79, 82, 84, 86, 88, 90, 92, 96, 97, 98, 101, 102, 103, 104, 107, 108, 109, 110, 113, 114, 116, 118, 120, 122, 124, 126, 128, 129, 131, 135	
\ebnf@alternation	.. 73	
\ebnf@bw	 4, 17, 21	
\ebnf@color	 17, 26, 28, 32, 34, 44, 46, 48, 72, 74	
\ebnf@grouping	 47	
		\ifdefined 10, 17, 21, 130
		\ifmode 25, 26, 28, 32, 33, 34, 37, 40
		\immediate .. 12, 13, 131
L		
\l	51, 52, 53, 56, 57, 58, 61, 62, 63, 66, 67, 68	
\langle	 32	
M		
\makeatletter	. 9, 17, 23, 30, 36, 39, 42, 77	
\makeatother	 15, 22, 29, 35, 38, 41, 75, 136	
\message	 132	
N		
\newcommand	20, 24, 31, 37, 40, 43, 45, 47, 50, 55, 60, 65, 71, 73	
\NewDocumentEnvironment	 78	
		\newwrite 11
		\noindent 129
O		
\openout	 12	
P		
\pgfkeys	 2	
\ProcessPgfPackageOptions	 8	
R		
\rangle	 34	
\regex	 81, 83, 85, 87, 89, 91, 94, 99, 105, 111, 115, 117, 119, 121, 123, 125, 127	
\relax	... 12, 25, 26, 28, 32, 33, 34, 37, 40	
\RequirePackage	1, 16, 18	
S		
\scriptscriptstyle	. 46	
\sffamily	... 26, 28, 33	
\space	 132, 133	
T		
\textcolor	 21	
\textsf	 26, 28, 33	
\tl	51, 52, 56, 57, 61, 62, 66, 67, 79, 129	
\to	 72	
\trail	 4	
\ttfamily	... 25, 37, 40	
U		
\unexpanded	 131	
V		
\vert	 74	
W		
\width	 2	
\write	 131	