



eolang: L^AT_EX Package for Formulas and Graphs of EO Programming Language and φ -calculus*

Yegor Bugayenko
yegor256@gmail.com

2022-10-20, 0.0.1

NB! You must run T_EX processor with `--shell-escape` option and you must have Perl installed.

1 Introduction

This package helps you print formulas of φ -calculus, which is a formal foundation of EO programming language. The calculus was introduced by Bugayenko (2021) and later formalized by Kudasov et al. (2022). Here is how you render a simple expression:

$ \begin{aligned} a &\mapsto \llbracket \\ &\quad b \mapsto \llbracket c \mapsto x(56), \\ &\quad \quad \rho \mapsto \text{hello}(\xi), \\ &\quad \quad \Delta \mapsto 01\text{-FE-C3} \rrbracket \rrbracket, \\ x &\mapsto \llbracket \lambda \mapsto M_1 \rrbracket. \end{aligned} $	<pre> 1 \documentclass{article} 2 \pagestyle{empty} 3 \usepackage{eolang} 4 \begin{document} 5 \begin{phiquestion*} 6 a -> [7 b -> [c -> x (56), 8 \rho -> hello (\xi), 9 \Delta ~> 01-FE-C3]], 10 11 x -> [\lambda ~> M_1]. 12 \end{phiquestion*} 13 \end{document} </pre>
--	---

`phiquestion` The environment `phiquestion` lets you write a φ -calculus expressions using simple plain-text notation, where:

*The sources are in GitHub at [yegor256/eolang.sty](https://github.com/yegor256/eolang.sty)

- “ \rightarrow ” maps to “ \mapsto ” (`\mapsto`),
- “ \dashrightarrow ” maps to “ \mapsto ” (`\mapstochar\dashrightarrow`),
- “[” maps to “ \llbracket ” (`\llbracket`),
- “]” maps to “ \rrbracket ” (`\rrbracket`),
- “|abc|” maps to “abc” (`\texttt{abc}`).

`\phi` The command `\phi` lets you inline a φ -calculus expressions using the same simple plain-text notation:

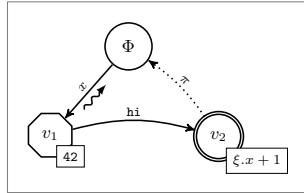
A simple object $x \mapsto$
 $\llbracket \varphi \mapsto y \rrbracket$ is a decorator
of the data object
 $y \mapsto \llbracket \Delta \dashrightarrow 42 \rrbracket$.

```

1 \documentclass{article}
2 \usepackage[paperwidth=2in]{geometry}
3 \pagestyle{empty}
4 \usepackage{eolang}
5 \begin{document}
6 A simple object
7 \phi{x -> [\varphi -> y]}
8 is a decorator of
9 the data object
10 \phi{y -> [\Delta ~> 42]}.
11 \end{document}

```

`figure` The environment `figure` allows you to draw a SODG graph:



```

1 \documentclass{article}
2 \pagestyle{empty}
3 \usepackage{eolang}
4 \begin{document}
5 \begin{figure}
6 v0
7 v1 xy:v0,-2,+1 data:|42|
8 v0->v1 a:$x$ rho
9 v2 xy:v0,+1,+1 atom:$\xi.x+1$
10 v1->v2 a:|hi| bend:-15
11 v2->v0 pi bend:10
12 \end{figure}
13 \end{document}

```

The content of the environment is parsed line by line. Markers in each line are separated by a single space. The first marker is either a unique name of a vertex, like `v1` in the example above, or an edge, like `v0->v1`. All other markers are either unary like `rho` or binary like `atom:$\xi.x+1$`. Binary markers have two parts, separated by colon. The following markers are supported for a vertex:

- “`data: [<box>]`” makes it a data vertex with an optional attached `<box>`,
- “`atom: [<box>]`” makes it an atom with an optional attached `<box>`,
- “`box:<txt>`” attaches a `<box>` to it,
- “`xy:<v>,<r>,<d>`” places this vertex in a position relative to the vertex `<v>`, shifting it right by `<r>` and down by `<d>` centimetres.

The following markers are supported for an edge:

- “rho” places a backward snake arrow to the edge,
- “rrho” places a reverse rho,
- “bend:<angle>” bend it right by the amount of <angle>,
- “a:<txt>” attaches label <txt> to it,
- “pi” makes it dotted, with π label.

2 Implementation

First, we include a few packages:

```
1 \RequirePackage{stmaryrd}
2 \RequirePackage{amsmath}
3 \RequirePackage{amsfonts}
4 \RequirePackage{iexec}
5 \RequirePackage{fancyvrb}
```

`\eolang@env` Then, we define `\eolang@env` supplementary command. It is implemented with the help of `\iexec` from `iexec` package:

```
6 \makeatletter\newcommand\eolang@env[2]{
7   \iexec[trace]{
8     /bin/echo -n '\begin{#1}\begin{split} &';
9     /bin/echo -n '\detokenize{#2}'
10    | perl -pe 's/^\r\\+//g'
11    | perl -pe 's/\\r\\+$/g'
12    | perl -pe 's/->/\\\\mapsto/g'
13    | perl -pe 's/\unexpanded{~}>/\\\\mapstochar\\\\dashrightarrow/g'
14    | perl -pe 's/\\|([^\|]+)\\\\/\\\\texttt{\|1}/g'
15    | perl -pe 's/\\[/\\\\llbracket/g'
16    | perl -pe 's/\\]/\\\\rrbracket/g'
17    | perl -pe 's/\\r\\r\\\\\\\\ \&/g'
18    | perl -pe 's/\\r\\\\\\\\\\\\[-4pt] \&/g'
19    | perl -pe 's/([~& ]) {2}([~ ])\\\\1 \\\2/g'
20    | perl -pe 's/{2}/\\\\quad{}/g'
21    ;
22    /bin/echo -n '\end{split}\end{#1}\endinput'
23  }%
24 }\makeatother
```

`phiquation` Then, we define `phiquation` and `phiquation*` environments:

```
25 \makeatletter
26 \NewDocumentEnvironment{phiquation*}{b}{%
27   \eolang@env{equation*}{#1}
28 }{}
29 \NewDocumentEnvironment{phiquation}{b}{%
30   \eolang@env{equation}{#1}
31 }{}
32 \makeatother
33 \AddToHook{env/phiquation*/begin}{\obeylines\obeyspaces}
34 \AddToHook{env/phiquation/begin}{\obeylines\obeyspaces}
```

`\phiq` Then, we define `\phiq` command:

```
35 \newcommand\phiq[1]{
```

```

36 \iexec[trace]{
37   /bin/echo -n '$';
38   /bin/echo -n '\detokenize{#1}'
39   | perl -pe 's/->/\\\\mapsto/g'
40   | perl -pe 's/\unexpanded{~}>/\\\\mapstochar\\\\dashrightarrow/g'
41   | perl -pe 's/[\\\\llbracket/g'
42   | perl -pe 's/[\\\\rrbracket/g'
43   ;
44   /bin/echo -n '$\\endinput'
45 }%
46 }

```

Perl Then, create a Perl script:

```

47 \begin{VerbatimOut}{\jobname.pl.eolang}
48 $tex = $ARGV[0];
49 $tex =~ s/^\s+|\s+$//g;
50 $tex =~ s/(\\[a-zA-Z]+)\s+/\1/g;
51 $tex =~ s/\\r\s+/\r/g;
52 $tex =~ s/\\|([^\|]+)\\|/\texttt{\1}/g;
53 my @cmds = split (/\\r/g, $tex);
54 print '\begin{phig}', "\n";
55 foreach my $c (@cmds) {
56   my ($head, $tail) = split (/ /, $c, 2);
57   my %opts = {};
58   foreach my $p (split (/ /, $tail)) {
59     my ($q, $t) = split (/:/, $p);
60     $opts{$q} = $t;
61   }
62   if (index($head, '->') == -1) {
63     print '\node[';
64     if (exists $opts{'xy'}) {
65       my ($v, $right, $down) = split(/,/ , $opts{'xy'});
66       print ',below right=';
67       print $down;
68       print 'cm and ';
69       print $right;
70       print 'cm of ';
71       print $v;
72     }
73     if (exists $opts{'data'}) {
74       print ',phi-data';
75       if (not $opts{'data'} eq '') {
76         $opts{'box'} = $opts{'data'};
77       }
78     } elsif (exists $opts{'atom'}) {
79       print ',phi-atom';
80       if (not $opts{'atom'} eq '') {
81         $opts{'box'} = $opts{'atom'};
82       }
83     } else {
84       print ',phi-object';
85     }
86     print ']';
87     print ' (' , $head, ')';

```

```

88     print ' {'$';
89     if ($head eq 'v0') {
90         print '\Phi';
91     } else {
92         print 'v_', substr($head, 1);
93     }
94     print '$}';
95     if (exists $opts{'box'}) {
96         print ' node[phi-box] at (';
97         print $head, '.south east) {';
98         print $opts{'box'}, '}'';
99     }
100 } else {
101     print '\draw[';
102     if (exists $opts{'pi'}) {
103         print ',phi-pi';
104         if (not exists $opts{'a'}) {
105             $opts{'a'} = '$\pi$';
106         }
107     }
108     print ']';
109     my ($from, $to) = split (/->/, $head);
110     print ' (' , $from, ') ' ;
111     if (exists $opts{'bend'}) {
112         print 'edge [bend right=', $opts{'bend'}, ']' ;
113     } else {
114         print '--';
115     }
116     if (exists $opts{'rho'} or exists $opts{'rrho'}) {
117         print ' pic[sloped,phi-rho]{parallel arrow={';
118         print '-' if not exists $opts{'rrho'};
119         print '0.3,-0.15}}';
120     }
121     if (exists $opts{'a'}) {
122         print ' node [phi-attr] {' , $opts{'a'}, '}' ;
123     }
124     print ' (' , $to, ') ' ;
125 }
126 print ";\n";
127 }
128 print '\end{phig}', "\n", '\endinput';
129 \end{VerbatimOut}
130 \message{^^Jeolang: file with Perl script (\jobname.pl.eolang) saved^^J}%
131 \iexec[trace,stdout=/dev/null]{
132     perl -pi -e 's/(\[a-zA-Z\])\s+/\1/g' \jobname.pl.eolang}

```

phigure Then, we include tikz package and its libraries:

```

133 \RequirePackage{tikz}
134 \usetikzlibrary{arrows}
135 \usetikzlibrary{shapes}
136 \usetikzlibrary{decorations}
137 \usetikzlibrary{decorations.pathmorphing}
138 \usetikzlibrary{intersections}
139 \usetikzlibrary{positioning}

```

```

140 \usetikzlibrary{backgrounds}
141 \usetikzlibrary{calc}
142 \usetikzlibrary{shapes.arrows}

```

phig Then, we define internal environment phig:

```

143 \newenvironment{phig}%
144 {\noindent\begin{tikzpicture}[
145   ->,>=stealth',node distance=0,thick,
146   pics/parallel arrow/.style={
147     code={\draw[-latex,phi-rho] (##1) -- (-##1);}}}%
148 {\end{tikzpicture}}
149 \tikzstyle{transforms} = [fill=white!80!black, single arrow,
150   minimum height=0.5cm, minimum width=0.5cm,
151   single arrow head extend=2mm]
152 \tikzstyle{phi-thing} = [thick,inner sep=0pt,minimum height=2.4em,
153   draw,font={\small}]
154 \tikzstyle{phi-object} = [phi-thing,circle]
155 \tikzstyle{phi-data} = [phi-thing,regular polygon,
156   regular polygon sides=8]
157 \tikzstyle{phi-empty} = [phi-object]
158 \tikzstyle{phi-rho} = [draw,decorate,decoration={
159   snake,amplitude=.4mm,segment length=2mm,post length=1mm}]
160 \tikzstyle{phi-pi} = [draw,dotted]
161 \tikzstyle{phi-atom} = [phi-object,double]
162 \tikzstyle{phi-box} = [xshift=-5pt,yshift=3pt,draw,fill=white,
163   rectangle,thin,minimum width=1.2em,anchor=north west,
164   font={\scriptsize}]
165 \tikzstyle{phi-attr} = [midway,sloped,inner sep=0pt,
166   above=2pt,sloped/.append style={transform shape},
167   font={\scriptsize},color=black]

```

phigure Then, create a new environment figure, as suggested [here](#):

```

168 \NewDocumentEnvironment{phigure}{b}{\%
169   \catcode'\ =10 %
170   \catcode'\^M=5 %
171   \iexec[trace,stdout=\jobname.tex.eolang]{
172     perl \jobname.pl.eolang '\detokenize{#1}'}%
173 }{}
174 \AddToHook{env/phigure/before}{\bgroup\obeylines\obeyspaces}
175 \AddToHook{env/phigure/after}{\egroup}

```

References

- Bugayenko, Yegor (2021). *EOLANG and φ -calculus*. arXiv: [2111.13384](#) [cs.PL].
- Kudasov, Nikolai et al. (2022). *φ -calculus: a purely object-oriented calculus of decorated objects*. arXiv: [2204.07454](#) [cs.PL].

Change History

v0.0.1
General: Initial version 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.

Symbols	<code>\endinput</code> 128	P
<code>\&</code> 17, 18	<code>\eolang@env</code> <u>6</u> , 27, 30	<code>\Perl</code> <u>47</u>
<code>\^</code> 170	I	<code>\Phi</code> 90
<code>\ </code> 52	<code>\iexec</code> 7, 36, 131, 171	<code>\phig</code> <u>143</u>
	J	<code>\phigure</code> <u>133</u> , <u>168</u>
<code>_</code> 169	<code>\jobname</code> 47, 130, 132, 171, 172	<code>\phiq</code> <u>35</u>
A		<code>\phiquation</code> <u>25</u>
<code>\AddToHook</code> . 33, 34, 174, 175	M	<code>\pi</code> 105
B	<code>\makeatletter</code> 6, 25	R
<code>\begin</code> 47, 54, 144	<code>\makeatother</code> 24, 32	<code>\RequirePackage</code> 1–5, 133
<code>\bgroup</code> 174	<code>\message</code> 130	S
C	N	<code>\scriptsize</code> 164, 167
<code>\catcode</code> 169, 170	<code>\newcommand</code> 6, 35	<code>\small</code> 153
D	<code>\NewDocumentEnvironment</code> 26, 29, 168	T
<code>\detokenize</code> 9, 38, 172	<code>\newenvironment</code> ... 143	<code>\tikzstyle</code> 149, 152, 154, 155, 157, 158, 160–162, 165
<code>\draw</code> 101, 147	<code>\node</code> 63	
	<code>\noindent</code> 144	
E	O	U
<code>\egroup</code> 175	<code>\obeylines</code> 33, 34, 174	<code>\unexpanded</code> 13, 40
<code>\end</code> 128, 129, 148	<code>\obeyspaces</code> ... 33, 34, 174	<code>\usetikzlibrary</code> 134–142