



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

Yegor Bugayenko
yegor256@gmail.com

2022-10-29, 0.2.0

NB! You must run T_EX processor with `--shell-escape` option and you must have [Perl](#) installed. This package doesn't work on Windows.

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 \rho \mapsto \xi.b, \\ &\quad b \mapsto \llbracket c \mapsto \text{fn}(56), \\ &\quad \quad \varphi \mapsto \text{hello}(\xi), \\ &\quad \quad \Delta \mapsto 01\text{-FE-C3} \rrbracket \rrbracket, \\ x &\mapsto \llbracket \alpha_0 \mapsto \emptyset \rrbracket. \end{aligned} $	<pre> 1 \documentclass{article} 2 \pagestyle{empty} 3 \usepackage{eolang} 4 \begin{document} 5 \begin{phiquestion*} 6 a -> [[7 ^ !-> \$.b, 8 b -> [[c -> fn (56), 9 @ -> hello (\$), 10 \Delta ..> 01-FE-C3]]]],\\ 11 x -> [[\alpha_0 -> ?]]. 12 \end{phiquestion*} 13 \end{document} </pre>
---	--

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

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

- “@” maps to “ φ ” (`\varphi`),
- “^” maps to “ ρ ” (`\rho`),
- “\$” maps to “ ξ ” (`\xi`),
- “&” maps to “ σ ” (`\sigma`),
- “?” maps to “ \emptyset ” (`\varnothing`),
- “->” maps to “ \mapsto ” (`\mapsto`),
- “!->” maps to “ \vdash ” (`\vdash`),
- “. .>” maps to “ \vdash ” (`\vdash`),
- “[[” maps to “ \llbracket ” (`\llbracket`),
- “]]” maps to “ \rrbracket ” (`\rrbracket`),
- “|abc|” maps to “abc” (`\texttt{abc}`).

Also, a few symbols are supported for φ PU architecture:

- “-abc>” maps to “ \xrightarrow{ABC} ” (`\xrightarrow{\text{\sffamily\scshape abc}}`),
- “:=” maps to “ \models ” (`\models`).

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

<p>A simple object</p> $x \mapsto \llbracket \varphi \mapsto y \rrbracket$ <p>is a decorator of</p> <p>the data object</p> $y \mapsto \llbracket \Delta \vdash 42 \rrbracket.$	<pre> 1 \documentclass{article} 2 \pagestyle{empty} 3 \usepackage{eolang} 4 \begin{document} 5 A simple object \\\ 6 \phiiq{x -> [[@ -> y]]} \\\ 7 is a decorator of \\\ 8 the data object \\\ 9 \phiiq{y -> [[\Delta .> 42]]}. 10 \end{document} </pre>
--	--

`sodg (env.)` The environment `sodg` allows you to draw a [SODG](#) graph:

	<pre> 1 \documentclass{article} 2 \pagestyle{empty} 3 \usepackage{eolang} 4 \begin{document} 5 \begin{sodg} 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{sodg} 13 \end{document} </pre>
--	---

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

v_1 in the example above, or an edge, like $v_0 \rightarrow v_1$. All other markers are either unary like ρ or binary like $\text{atom}:\$ \backslash \text{xi} . x+1 \$$. Binary markers have two parts, separated by colon. The following markers are supported for a vertex:

- “ $\text{data}:[\langle \text{box} \rangle]$ ” makes it a data vertex with an optional attached $\langle \text{box} \rangle$ (the content of the box may only be numeric data),
- “ $\text{atom}:[\langle \text{box} \rangle]$ ” makes it an atom with an optional attached $\langle \text{box} \rangle$ (the content of the box is a math formula),
- “ $\text{box}:\langle \text{txt} \rangle$ ” attaches a $\langle \text{box} \rangle$ to it,
- “ $\text{xy}:\langle v \rangle, \langle r \rangle, \langle d \rangle$ ” places this vertex in a position relative to the vertex $\langle v \rangle$, shifting it right by $\langle r \rangle$ and down by $\langle d \rangle$ centimetres.

The following markers are supported for an edge:

- “ ρ ” places a backward snake arrow to the edge,
- “ rrho ” places a reverse ρ ,
- “ $\text{bend}:\langle \text{angle} \rangle$ ” bend it right by the amount of $\langle \text{angle} \rangle$,
- “ $\text{a}:\langle \text{txt} \rangle$ ” attaches label $\langle \text{txt} \rangle$ to it,
- “ pi ” makes it dotted, with π label.

$\backslash \text{eolang}$ There is also a no-argument command $\backslash \text{eolang}$ to help you print the name of EO $\backslash \text{phic}$ language. It understands anonymous mode of [acmart](#) and prints itself differently, to $\backslash \text{xmir}$ double-blind your paper. There is also $\backslash \text{phic}$ command to print the name of φ -calculus, also sensitive to anonymous mode. The macro $\backslash \text{xmir}$ prints “XMIR”.

In our research we use XYZ ,
an experimental object-oriented
dataflow language, α -calculus,
as its formal foundation, and XML [’]
— its XML-based presentation.

```
1 \documentclass[anonymous]{acmart}
2 \thispagestyle{empty}
3 \usepackage{eolang}
4 \begin{document}
5 In our research we use \eolang{,}, \
6 an experimental object-oriented \
7 dataflow language, \phic{,}, \
8 as its formal foundation, and \xmir{,} \
9 --- its XML-based presentation.
10 \end{document}
```

$\backslash \text{phiConst}$ A simple commands is defined to help you render an arrow for a constant attribute. It is recommende not to use it directly, but use $! \rightarrow$ instead. However, if you want to use $\backslash \text{phiConst}$, wrap it in $\backslash \text{mathrel}$ for better display:

$\llbracket x \mathrel{\text{H}\!\!\rightarrow} y \rrbracket$

```
1 \documentclass{article}
2 \pagestyle{empty}
3 \usepackage{eolang}
4 \begin{document}
5 \phiq{\llbracket x \mathrel{\phiConst} y \rrbracket}
6 \end{document}
```

2 Package Options

`tmpdir` The default location of temp files is `_eolang`. You can change this using `tmpdir` option:

```
\usepackage[tmpdir=/tmp/foo]{eolang}
```

3 More Examples

The `phiquation` environment treats ends of line as signals to start new lines in the formula. If you don't want this to happen and want to parse the next line as the a continuation of the current line, you can use a single backslash as it's done here:

$\frac{x \mapsto [\varphi \mapsto y] \quad y \mapsto [z \mapsto 42]}{x.z \mapsto 42} R1$	<pre> 1 \documentclass{article} 2 \usepackage{amsmath} 3 \usepackage{eolang} 4 \pagestyle{empty} 5 \begin{document} 6 \begin{phiquation*} 7 \dfrac \{ 8 x->[[@->y]] \quad y->[[z->42]] \} \backslash 9 \{x.z -> 42\} \backslash 10 \text{\sffamily R1} 11 \end{phiquation*} 12 \end{document} </pre>
--	--

The `phiquation` environment may be used together with [acmart](#):

$ \begin{array}{l} x \mapsto [\\ \quad y \mapsto [\\ \qquad z \mapsto \xi, f \mapsto \emptyset]], \\ \beta_1 \models [\psi \xrightarrow{\text{WAIT}} \emptyset]. \end{array} $	<pre> 1 \documentclass{acmart} 2 \usepackage{eolang} 3 \thispagestyle{empty} 4 \begin{document} 5 \begin{phiquation*} 6 x -> [[7 y -> [[8 z !-> \$, f ..> ?]]]],\backslash 9 \beta_1 := [\psi -wait> ?]. 10 \end{phiquation*} 11 \end{document} </pre>
---	--

The `phiquation` environment will automatically align formulas by the first arrow, if there are only left-aligned formulas:

$$\begin{array}{l}
x(\pi) \mapsto \llbracket \lambda \mapsto f_1 \rrbracket, \\
x(a, b, c) \mapsto \llbracket \alpha_0 \mapsto \varnothing, \varphi \mapsto \text{hello}(\xi) \rrbracket, \\
\Delta = 43-09.
\end{array}$$

```

1 \documentclass{acmart}
2 \usepackage{eolang}
3 \thispagestyle{empty}
4 \begin{document}
5 \begin{phiquestion*}
6 x(\pi) -> [[\lambda .> f_1]], \\\
7 x(a,b,c) -> [[ \alpha_0 -> ?, \ \
8   @ -> |hello|($) ]], \\\
9 \Delta = |43-09|.
10 \end{phiquestion*}
11 \end{document}

```

4 Implementation

First, we include a few packages. We need [stmaryrd](#) for `\llbracket` and `\rrbracket` commands:

```
1 \RequirePackage{stmaryrd}
```

We need [amsmath](#) for `equation*` environment:

```
2 \RequirePackage{amsmath}
```

We need [amssymb](#) for `\varnothing` command. We disable `\Bbbk` because it may conflict with some packages from [acmart](#):

```
3 \let\Bbbk\relax\RequirePackage{amssymb}
```

We need [fancyvrb](#) for `\VerbatimEnvironment` command:

```
4 \RequirePackage{fancyvrb}
```

We need [iexec](#) for executing Perl scripts:

```
5 \RequirePackage{iexec}
```

Then, we process package options:

```

6 \RequirePackage{pgfopts}
7 \RequirePackage{ifluatex}
8 \RequirePackage{ifxetex}
9 \pgfkeys{
10   /eolang/.cd,
11   tmpdir/.store in=\eolang@tmpdir,
12   tmpdir/.default=_eolang\ifxetex-xe\else\ifluatex-lua\fi\fi,
13   tmpdir
14 }
15 \ProcessPgfOptions{/eolang}

```

Then, we make a directory where all temporary files will be kept:

```
16 \iexec[null]{mkdir -p "\eolang@tmpdir/\jobname"}%
```

`\eolang@mdfive` Then, we define a command for MD5 hash calculating of a file:

```

17 \RequirePackage{pdftexcmds}
18 \makeatletter\newcommand\eolang@mdfive[1]{\pdf@filemdfivesum{#1}}\makeatother

```

`eolang-phi.pl` Then, we create a Perl script for `phiquestion` processing:

```
19 \makeatletter
```

```

20 \begin{VerbatimOut}{\eolang@tmpdir/eolang-phi.pl}
21 $env = $ARGV[0];
22 open(my $fh, '<', $ARGV[1]);
23 my $tex; { local $/; $tex = <$fh>; }
24 print '% This file is auto-generated', "\n";
25 print '% There are ', length($tex),
26   ' chars in the input: ', $ARGV[1], "\n";
27 print '% ---', "\n";
28 if (index($tex, "\t") > 0) {
29   print "TABS are prohibited!";
30   exit 1;
31 }
32 my @lines = split (/\\n/g, $tex);
33 foreach my $t (@lines) {
34   print '% ', $t, "\n";
35 }
36 print '% ---', "\n";
37 if ($env eq 'phiq') {
38   print '$';
39 } else {
40   print '\begin{', $env, '}\begin{split}';
41 }
42 $tex =~ s/^\s+|\\s+$/g;
43 if ($env ne 'phiq') {
44   $tex =~ s/\\s+\\n\\s*/g;
45   $tex =~ s/\\n\\n\\n\\n/g;
46 }
47 $tex =~ s/([\\s,>\\()([0-9A-F][0-9A-F-]*)\\1|\\2|/g;
48 $tex =~ s/\\?/\\varnothing{/g;
49 $tex =~ s/@/\\varphi{/g;
50 $tex =~ s/&/\\sigma{/g;
51 $tex =~ s/^/\\rho{/g;
52 $tex =~ s/$/\\xi{/g;
53 $tex =~ s/-([a-z]+)>/\\mathrel{\\xrightarrow{\\text{\\sfamily\\scshape \\1}}}/g;
54 $tex =~ s/!->/\\mathrel{\\phiConst}/g;
55 $tex =~ s/->/\\mathrel{\\mapsto}/g;
56 $tex =~ s/:=/\\mathrel{\\vDash}/g;
57 $tex =~ s/..>/\\mathrel{\\phiDotted}/g;
58 $tex =~ s/\\|([~\\|]+)\\|/\\texttt{\\1}/g;
59 $tex =~ s/\\[/\\[/\\llbracket\\mathrel{/g;
60 $tex =~ s/\\]/\\]/\\mathrel{\\rrbracket}/g;
61 if ($env ne 'phiq') {
62   $tex =~ s/\\n\\n/\\&/g;
63   $tex =~ s/\\n/\\&[-4pt]/g;
64   $tex =~ s/([~&\\s])\\s{2}([~\\s])/\\1 \\2/g;
65   $tex =~ s/\\s{2}/ \\quad/g;
66   my @leads = $tex =~ /&[~\\s]+\\s/g;
67   my @eols = $tex =~ /&/g;
68   $tex = '&' . $tex;
69   if (0+@leads == 0+@eols && 0+@eols > 0) {
70     $tex =~ s/&([~\\s]+)\\s/\\1&/g;
71   }
72 }
73 print $tex;

```

```

74 if ($env eq 'phiq') {
75   print '$';
76 } else {
77   print '\end{split}\end{' , $env, '}' ;
78 }
79 print '\endinput', "\n";
80 \end{VerbatimOut}
81 \message{eolang: File with Perl script
82   '\eolang@tmpdir/eolang-phi.pl' saved^~J}%
83 \iexec[trace,null]{perl -pi -e 's/(\\[a-zA-Z])\\s+/\1/g'
84   "\eolang@tmpdir/eolang-phi.pl"}
85 \makeatother

```

phiqutation Then, we define phiqutation and phiqutation* environments through a supplementary \eolang@process command:

```

86 \makeatletter\newcommand\eolang@process[1]{
87   \def\hash{\eolang@mdfive
88     {\eolang@tmpdir/\jobname/phiqutation.tex}}%
89   \iexec[null]{cp "\eolang@tmpdir/\jobname/phiqutation.tex"
90     "\eolang@tmpdir/\jobname/\hash.tex"}%
91   \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-post.tex]{
92     perl "\eolang@tmpdir/eolang-phi.pl"
93     '#1'
94     "\eolang@tmpdir/\jobname/\hash.tex"}%
95 }
96 \newenvironment{phiqutation*}%
97 {\VerbatimEnvironment\begin{VerbatimOut}
98   {\eolang@tmpdir/\jobname/phiqutation.tex}}
99 {\end{VerbatimOut}\eolang@process{equation*}}
100 \newenvironment{phiqutation}%
101 {\VerbatimEnvironment\begin{VerbatimOut}
102   {\eolang@tmpdir/\jobname/phiqutation.tex}}
103 {\end{VerbatimOut}\eolang@process{equation}}
104 \makeatother

```

\phiiq Then, we define \phiiq command:

```

105 \makeatletter\newcommand\phiiq[1]{
106   \iexec[trace,quiet,stdout=\eolang@tmpdir/\jobname/phiq.tex]{
107     /bin/echo '\detokenize{#1}'}
108   \def\hash{\eolang@mdfive
109     {\eolang@tmpdir/\jobname/phiq.tex}}%
110   \iexec[null]{cp "\eolang@tmpdir/\jobname/phiq.tex"
111     "\eolang@tmpdir/\jobname/\hash.tex"}%
112   \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-post.tex]{
113     perl \eolang@tmpdir/eolang-phi.pl 'phiq'
114     "\eolang@tmpdir/\jobname/\hash.tex"}%
115 }\makeatother

```

eolang-sodg.pl Then, we create a Perl script for sodg graphs processing:

```

116 \makeatletter
117 \begin{VerbatimOut}{\eolang@tmpdir/eolang-sodg.pl}
118 open(my $fh, '<', $ARGV[0]);
119 my $tex; { local $/; $tex = <$fh>; }
120 print '% This file is auto-generated', "\n";

```

```

121 print '% There are ', length($tex),
122 ' chars in the input: ', $ARGV[0], "\n";
123 print '% ---', "\n";
124 if (index($tex, "\t") > 0) {
125     print "TABS are prohibited!";
126     exit 1;
127 }
128 $tex =~ s/^\s+|\s+$//g;
129 $tex =~ s/([a-zA-Z]+\s+)/\1/g;
130 $tex =~ s/\n\s+/\n/g;
131 $tex =~ s/|([^\|]+)\|/\\texttt{\1}/g;
132 my @cmds = split (/\\n/g, $tex);
133 foreach my $t (@cmds) {
134     print '% ', $t, "\n";
135 }
136 print '% ---', "\n";
137 print '\begin{picture}', "\n";
138 foreach my $c (@cmds) {
139     my ($head, $tail) = split (/ /, $c, 2);
140     my %opts = {};
141     foreach my $p (split (/ /, $tail)) {
142         my ($q, $t) = split (/:/, $p);
143         $opts{$q} = $t;
144     }
145     if (index($head, '->') == -1) {
146         print '\node[';
147         if (exists $opts{'xy'}) {
148             my ($v, $right, $down) = split(/,/ , $opts{'xy'});
149             print ',below right=';
150             print $down;
151             print 'cm and ';
152             print $right;
153             print 'cm of ';
154             print $v;
155         }
156         if (exists $opts{'data'}) {
157             print ',phi-data';
158             if (not $opts{'data'} eq '') {
159                 my $d = $opts{'data'};
160                 if (index($d, '|') == -1) {
161                     $d = '\texttt{' . $d . '}';
162                 }
163                 $opts{'box'} = $d;
164             }
165         } elsif (exists $opts{'atom'}) {
166             print ',phi-atom';
167             if (not $opts{'atom'} eq '') {
168                 my $a = $opts{'atom'};
169                 if (index($a, '$') == -1) {
170                     $a = '$' . $a . '$';
171                 }
172                 $opts{'box'} = $a;
173             }
174         } else {

```



```

175     print ',phi-object';
176 }
177 print ']';
178 print ' (', $head, ')';
179 print '$';
180 if ($head eq 'v0') {
181     print '\Phi';
182 } else {
183     print 'v_', substr($head, 1);
184 }
185 print '$';
186 if (exists $opts{'box'}) {
187     print ' node[phi-box] at (';
188     print $head, '.south east) {';
189     print $opts{'box'}, ')';
190 }
191 } else {
192     print '\draw[';
193     if (exists $opts{'pi'}) {
194         print ',phi-pi';
195         if (not exists $opts{'a'}) {
196             $opts{'a'} = '$\pi$';
197         }
198     }
199     print ']';
200     my ($from, $to) = split (/>/, $head);
201     print ' (', $from, ')';
202     if (exists $opts{'bend'}) {
203         print 'edge [bend right=', $opts{'bend'}, ')';
204     } else {
205         print '--';
206     }
207     if (exists $opts{'rho'} or exists $opts{'rrho'}) {
208         print ' pic[sloped,phi-rho]{parallel arrow={';
209         print '-' if not exists $opts{'rrho'};
210         print '0.3,-0.15}}';
211     }
212     if (exists $opts{'a'}) {
213         print ' node [phi-attr] {' , $opts{'a'}, ')';
214     }
215     print ' (', $to, ')';
216 }
217 print ";\n";
218 }
219 print '\end{phicture}', "\n", '\endinput';
220 \end{VerbatimOut}
221 \message{eolang: File with Perl script
222 '\eolang@tmpdir/eolang-sodg.pl' saved^^J}%
223 \iexec[trace,null]{perl -pi -e 's/(\\[a-zA-Z])\\s+/\1/g'
224 "\eolang@tmpdir/eolang-sodg.pl"}
225 \makeatother

```

tikz Then, we include tikz package and its libraries:

```

226 \RequirePackage{tikz}

```

```

227 \usetikzlibrary{arrows}
228 \usetikzlibrary{shapes}
229 \usetikzlibrary{decorations}
230 \usetikzlibrary{decorations.pathmorphing}
231 \usetikzlibrary{intersections}
232 \usetikzlibrary{positioning}
233 \usetikzlibrary{calc}
234 \usetikzlibrary{shapes.arrows}

```

picture Then, we define internal environment phicture:

```

235 \newenvironment{phicture}%
236 {\noindent\begin{tikzpicture}[
237   ->,>=stealth',node distance=0,thick,
238   pics/parallel arrow/.style={
239     code={\draw[-latex,phi-rho] (##1) -- (-##1);}}}%
240 {\end{tikzpicture}}
241 \tikzstyle{transforms} = [fill=white!80!black, single arrow,
242   minimum height=0.5cm, minimum width=0.5cm,
243   single arrow head extend=2mm]
244 \tikzstyle{phi-thing} = [thick,inner sep=0pt,minimum height=2.4em,
245   draw,font={\small}]
246 \tikzstyle{phi-object} = [phi-thing,circle]
247 \tikzstyle{phi-data} = [phi-thing,regular polygon,
248   regular polygon sides=8]
249 \tikzstyle{phi-empty} = [phi-object]
250 \tikzstyle{phi-rho} = [draw,decorate,decoration={
251   snake,amplitude=.4mm,segment length=2mm,post length=1mm}]
252 \tikzstyle{phi-pi} = [draw,dotted]
253 \tikzstyle{phi-atom} = [phi-object,double]
254 \tikzstyle{phi-box} = [xshift=-5pt,yshift=3pt,draw,fill=white,
255   rectangle,thin,minimum width=1.2em,anchor=north west,
256   font={\scriptsize}]
257 \tikzstyle{phi-attr} = [midway,sloped,inner sep=0pt,
258   above=2pt,sloped/.append style={transform shape},
259   font={\scriptsize},color=black]

```

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

```

260 \makeatletter\newenvironment{sodg}%
261 {\VerbatimEnvironment\begin{VerbatimOut}
262   {\eolang@tmpdir/\jobname/sodg.tex}}
263 {\end{VerbatimOut}}%
264 \def\hash{\eolang@mdfive
265   {\eolang@tmpdir/\jobname/sodg.tex}}%
266 \iexec[null]{cp "\eolang@tmpdir/\jobname/sodg.tex"
267   "\eolang@tmpdir/\jobname/\hash.tex"}}%
268 \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-post.tex]{
269   perl "\eolang@tmpdir/eolang-sodg.pl"
270   "\eolang@tmpdir/\jobname/\hash.tex"}}%
271 }\makeatother

```

\eolang Then, we define a simple supplementary command to help you print EO, the name of our language.

```

272 \newcommand\eolang{%
273   \ifdefined\anon%

```

```

274 \anon[XYZ]{\sffamily E0}}%
275 \else%
276 {\sffamily E0}%
277 \fi%
278 }

```

`\phic` Then, we define a simple supplementary command to help you print φ -calculus, the name of our formal apparatus.

```

279 \newcommand\phic{%
280 \ifdefined\anon%
281 \anon[ $\alpha$ -calculus]{ $\varphi$ -calculus}%
282 \else%
283  $\varphi$ -calculus%
284 \fi%
285 }

```

`\xmirl` Then, we define a simple supplementary command to help you print XMIR, the name of our XML-based format of program representation.

```

286 \newcommand\xmirl{%
287 \ifdefined\anon%
288 \anon[XML']{XMIR}%
289 \else%
290 XMIR%
291 \fi%
292 }

```

`\phiConst` Then, we define a command to render an arrow for a constant attribute, as suggested [here](#):

```

293 \newcommand\phiConst{%
294 \mathrel{\hspace{.15em}}\mapstochar\mathrel{\hspace{-.15em}}\mapsto}

```

`\phiDotted` Then, we define a command to render an arrow for a special attribute, as suggested [here](#):

```

295 \RequirePackage{trimclip}
296 \RequirePackage{amsfonts}
297 \makeatletter
298 \newcommand\phiDotted{\mapstochar\mathrel{\mathpalette\phiDotted@\relax}}
299 \newcommand\phiDotted@[2]{%
300 \begingroup
301 \settowidth{\dimen\z@}{ $m_{\#1}$ \rightarrow}%
302 \settoheight{\dimen\tw@}{ $m_{\#1}$ \rightarrow}%
303 \sbox\z@{%
304 \makebox[\dimen\z@][s]{%
305 \clipbox{0 0 {0.4\width} 0}%
306 {\resizebox{\dimen\z@}{\height}%
307 { $m_{\#1}$ \dashrightarrow}}}%
308 \hss%
309 \clipbox{{0.69\width} {-0.1\height} 0 {-\height}}{ $m_{\#1}$ \rightarrow}%
310 }%
311 }%
312 \ht\z@=\dimen\tw@ \dp\z@=\z@%
313 \box\z@%
314 \endgroup\makeatother

```

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

0.0.1		
General: First draft.	5	
0.0.2		
sodg: The environment “phigure” renamed to “sodg” for the sake of better semantic. The graph in the picture is solely a SODG graph, that’s why the name “sodg” is better.	10	
eolang-phi.pl: New symbol added for basket slots	5	
Parsing of symbols “@,” “^,” and “&” enabled (varphi, rho, and sigma) . . .	5	
The symbols “[” and “]” replaced with “[[” and “]]” for abstract object brackets, because they conflicted with normal square brackets	5	
eolang-sodg.pl: The Perl file now has a fixed name, which doesn’t depend on the name of the TeX job. This file may be shared among jobs, no need to make it uniquely named. . .	7	
\phiq: Parsing of additional symbols enabled	7	
0.1.0		
General: Parsing of package options introduced.	5	
\eolang: New command “eolang” added to print the name of the language in both normal and anonymous mode of “acmart” . . .	10	
		\eolang@mdfive: New supplementary command added to calculate MD5 sum of a file.
		5
		eolang-phi.pl: A new Perl script “eolang-phi.pl” added for parsing of phi expressions.
		5
		eolang-sodg.pl: There are two Perl scripts now: one for phiquation, another one for sodg.
		7
		\phic: New command “phic” prints the name of φ -calculus in both normal and anonymous mode of “acmart”
		11
		\phiConst: New command “phiConst” added to denote a link to a constant attribute.
		11
		\phiDotted: New command “phiDotted” added to denote a link to a special attribute.
		11
	0.2.0	
	eolang-phi.pl: Numbers automatically render as “texttt”. No need to use vertical bars around them anymore.	5
	eolang-sodg.pl: The content of “atom” and “data” boxes is parsed automatically as formulas and numbers, respectively.	7
	\xmirl: New command “xmirl” prints XMIR in both normal and anonymous mode of “acmart” . . .	11

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		106, 109, 110, 111,	\noindent 236
\\$	52	112, 113, 114, 117,	
\(.....	47	222, 224, 262, 265,	P
\?	48	266, 267, 268, 269, 270	\pdf@filemdfivesum .. 18
\[.....	59		\pgfkeys 9
\]	60	H	\Phi 181
\^	51	\hash 87, 90, 91,	\phic 279
\ 	58, 131	94, 108, 111, 112,	\phiConst 293
		114, 264, 267, 268, 270	\picture 235
Numbers		\height 306, 309	\phiDotted 295
\2	47, 64	\hspace 294	\phiDotted@ 298, 299
A		\hss 308	\phiq 105
\alpha	281	\ht 312	\phiquation 86
\anon	273,	I	\pi 196
	274, 280, 281, 287, 288	\iexec 16, 83, 89, 91, 106,	\ProcessPgfOptions .. 15
		110, 112, 223, 266, 268	
B		\ifdefined .. 273, 280, 287	R
\Bbbk	3	\ifluatex 12	\relax 3, 298
\begin	20, 40, 97,	\ifxetex 12	\RequirePackage ...
	101, 117, 137, 236, 261	 1, 2, 3, 4, 5,
\box	313	J	6, 7, 8, 17, 226, 295, 296
C		\jobname 16, 88,	\resizebox 306
\clipbox	305, 309	89, 90, 91, 94, 98,	\rightarrow . 301, 302, 309
		102, 106, 109, 110,	
D		111, 112, 114, 262,	S
\dashrightarrow ...	307	265, 266, 267, 268, 270	\sbox 303
\def	87, 108, 264	M	\scriptsize 256, 259
\detokenize	107	\m@th ... 301, 302, 307, 309	\settoheight 302
\dimen 301, 302, 304, 306, 312		\makeatletter . 18, 19,	\settowidth 301
\dp	312	86, 105, 116, 260, 297	\sffamily 274, 276
\draw	192, 239	\makeatother .. 18, 85,	\small 245
		104, 115, 225, 271, 314	\sodg 260
E		\makebox 304	
\end	77, 80, 99,	\mapsto 294	T
	103, 219, 220, 240, 263	\mapstochar 294, 298	\t 28, 124
\endinput	79, 219	\mathpalette 298	\texttt 161
\eolang	272	\mathrel 294, 298	\tikz 226
\eolang-phi.pl	19	\message 81, 221	\tikzstyle 241,
\eolang-sodg.pl ...	116		244, 246, 247, 249,
\eolang@mdfive		N	250, 252, 253, 254, 257
	17, 87, 108, 264	\newcommand	\tw@ 302, 312
\eolang@process 86, 99, 103		. 18, 86, 105, 272,	U
\eolang@tmpdir		279, 286, 293, 298, 299	\usetikzlibrary ...
	11, 16, 20,	\newenvironment 227, 228, 229,
	82, 84, 88, 89, 90, 96, 100, 235, 260	230, 231, 232, 233, 234
	91, 92, 94, 98, 102,	\node 146	V
			\varphi 281, 283

<code>\VerbatimEnvironment</code>		X
.....	97, 101, 261	<code>\xmir</code> <u>286</u>
	W	Z
<code>\width</code>	305, 309	<code>\z@</code> 301, 303, 304, 306, 312, 313