



# eolang: L<sup>A</sup>T<sub>E</sub>X Package for Formulas and Graphs of EO Programming Language and $\varphi$ -calculus\*

Yegor Bugayenko  
yegor256@gmail.com

2022-11-05, 0.4.0

**NB!** You must run T<sub>E</sub>X 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  $\varphi$ -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 -&gt; [[ 7   ^ !-&gt; \$.b, 8   b -&gt; [[ c -&gt;  fn (56), 9     @ -&gt;  hello (\$), 10    \Delta ..&gt; 01-FE-C3 ]]]],\\ 11 x -&gt; [[ \alpha_0 -&gt; ? ]]. 12 \end{phiquestion*} 13 \end{document} </pre>
---	--

`phiquestion (env)` The environment `phiquestion` lets you write a  $\varphi$ -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 “ $\varphi$ ” (`\varphi`),
- “^” maps to “ $\rho$ ” (`\rho`),
- “\$” maps to “ $\xi$ ” (`\xi`),
- “&” maps to “ $\sigma$ ” (`\sigma`),
- “?” maps to “ $\emptyset$ ” (`\varnothing`),
- “->” maps to “ $\mapsto$ ” (`\mapsto`),
- “~>” maps to “ $\rightsquigarrow$ ” (`\phiWave`),
- “!->” maps to “ $\multimap$ ” (`\phiConst`),
- “. .>” maps to “ $\dot{\mapsto}$ ” (`\phiDotted`),
- “[[” maps to “ $\llbracket$ ” (`\llbracket`),
- “]]” maps to “ $\rrbracket$ ” (`\rrbracket`),
- “|abc|” maps to “abc” (`\texttt{abc}`).

Also, a few symbols are supported for  $\varphi$ PU architecture:

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

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

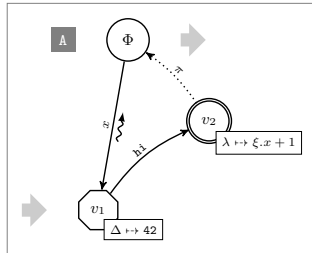
A simple object  $x \mapsto \llbracket \varphi \mapsto y \rrbracket$   
is a decorator of the data object  
 $y \mapsto \llbracket \Delta \mapsto 42 \rrbracket$ .

```

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}

```

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



```

1 \documentclass{article}
2 \pagestyle{empty}
3 \usepackage{eolang}
4 \begin{document}
5 \begin{sodg}
6 v0 \ \ v0=> \ \ v0!A
7 v1 xy:v0,-.8,2.8 data:42 \ \ =>v1
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}

```

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>” (the content of the box may only be numeric data),
- “atom: [<box>]” makes it an atom with an optional attached “<box>” (the content of the box is a math formula),
- “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  $\pi$  label.

It is also possible to put transformation arrows to the graph, with the help of “v0=>v1” syntax. The arrow will be placed exactly between two vertices. You can also put an arrow from a vertex to the right, saying for example “v3=>”, of from the left to the vertex, by saying for example “=>v5.”

You can also put a marker at the left side of a vertex, using “v5!A” syntax, where “v5” is the vertex and “A” is the text in the marker. They are useful when you put a few graphs on a picture explaining how one graph is transformed to another one and so forth.

Be aware, unrecognized markers are simply ignored, without any error reporting.

`\eolang` There is also a no-argument command `\eolang` to help you print the name of EO language. It understands anonymous mode of `\acmart` and prints itself differently, to `\phic` double-blind your paper. There is also `\phic` command to print the name of  $\varphi$ -calculus, also sensitive to anonymous mode. The macro `\xmirl` prints “XMIR”.

In our research we use XYZ,  
an experimental object-oriented  
dataflow language,  $\alpha$ -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 \xmirl{,} \
9 --- its XML-based presentation.
10 \end{document}
```

`\phiConst` A simple commands is defined to help you render an arrow for a constant attribute. It is recommended not to use it directly, but use `!->` instead. However, if you want to use `\phiConst`, wrap it in `\mathrel` for better display:

$$\llbracket x \multimap y \rrbracket$$

```

1 \documentclass{article}
2 \pagestyle{empty}
3 \usepackage{eolang}
4 \begin{document}
5 \phiq{[[ x \mathrel{\phiConst} y ]]}
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 `phiqutation` 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:

$$\frac{x \mapsto \llbracket \varphi \mapsto y \rrbracket \quad y \mapsto \llbracket z \mapsto 42 \rrbracket}{x.z \mapsto 42} R1$$

```

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

```

This is how you can use `\dfrac` from `amsmath` for large inference rules, with the help of `\begin{split}` and `\end{split}`:

$$\frac{x \mapsto [\varphi \mapsto y, z \mapsto 42, \alpha_0 \mapsto \emptyset, \alpha_1 \mapsto 42]}{x \mapsto [\varphi \mapsto y, z \mapsto \emptyset, f \rightsquigarrow \text{pi}(\alpha_0 \mapsto [\psi \mapsto \text{hello}(12)], \alpha_1 \mapsto 42)]} \text{R2.}$$

```

1 \documentclass{article}
2 \usepackage{amsmath}
3 \usepackage{eolang}
4 \pagestyle{empty}
5 \begin{document}
6 \begin{phiquestion*}
7 \dfrac{\begin{split}
8 x->[@->y, z->42,
9 \alpha_0->?, \alpha_1->42]]
10 \end{split}}{\begin{split}
11 x->[@->y, z->?, f ~> |pi|(
12 \alpha_0->[[ \psi !-> |hello|(12) ]],
13 \alpha_1->42)]]
14 \end{split}} \text{R2}.
15 \end{phiquestion*}
16 \end{document}

```

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

$$\begin{array}{l} x \mapsto [ \\ \quad y \mapsto [ \\ \qquad z \mapsto \xi, f \mapsto \emptyset ] ], \\ \beta_1 \models [\psi \xrightarrow{\text{WAIT}} \emptyset]. \end{array}$$

```

1 \documentclass{acmart}
2 \usepackage{eolang}
3 \thispagestyle{empty}
4 \begin{document}
5 \begin{phiquestion*}
6 x -> [[
7   y -> [[
8     z !-> $, f ..> ? ]]]], \\\
9 \beta_1 := [ \psi -wait> ? ].
10 \end{phiquestion*}
11 \end{document}

```

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

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

```

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}

```

Inside `phiquestion` environment you can use labels too (just put `\label` after the formula):

$$x \mapsto \llbracket \Delta \mapsto 42 \rrbracket. \quad (1)$$

Eq. 1 is easy to read.

```

1 \documentclass{article}
2 \usepackage{eolang}
3 \usepackage[paperwidth=2in]{geometry}
4 \pagestyle{empty}
5 \begin{document}
6 \begin{phiqutation}
7 x -> [[\Delta -> 42]]. \label{eq:A}
8 \end{phiqutation}
9 Eq.\ref{eq:A} is easy to read.
10 \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@lineno` Then, we define an internal counter to protect line number from changing:

```
17 \makeatletter\newcounter{eolang@lineno}\makeatother
```

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

```

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

```

eolang-phi.pl Then, we create a Perl script for phiquation processing using VerbatimOut from [fancyvrb](#):

```

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

```

```

74 $tex = '&' . $tex;
75 if (0+@leads == 0+@eols && 0+@eols > 0) {
76     $tex =~ s/&([^\s]+\s)/\1&/g;
77 }
78 }
79 print $tex;
80 if ($env eq 'phiq') {
81     print '$';
82 } else {
83     print '\end{split}\end{' , $env, '}' ;
84 }
85 print '\endinput', "\n";
86 \end{VerbatimOut}
87 \message{eolang: File with Perl script
88   '\eolang@tmpdir/eolang-phi.pl' saved^^J}%
89 \iexec[trace,null]{perl -pi -e 's/(\\[a-zA-Z])\\s+/\1/g'
90   "\eolang@tmpdir/eolang-phi.pl"}
91 \makeatother

```

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

```

92 \makeatletter\newcommand\eolang@process[1]{
93   \def\hash{\eolang@mdfive
94     {\eolang@tmpdir/\jobname/phiqutation.tex}}%
95   \iexec[null]{cp "\eolang@tmpdir/\jobname/phiqutation.tex"
96     "\eolang@tmpdir/\jobname/\hash.tex"}%
97   \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-post.tex]{
98     perl "\eolang@tmpdir/eolang-phi.pl"
99     '#1'
100     "\eolang@tmpdir/\jobname/\hash.tex"}%
101   \setcounter{FancyVerbLine}{\value{eolang@lineno}}%
102 }
103 \newenvironment{phiqutation*}%
104 {\VerbatimEnvironment%
105 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
106 \begin{VerbatimOut}
107   {\eolang@tmpdir/\jobname/phiqutation.tex}}
108 {\end{VerbatimOut}\eolang@process{equation*}}
109 \newenvironment{phiqutation}%
110 {\VerbatimEnvironment%
111 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
112 \begin{VerbatimOut}
113   {\eolang@tmpdir/\jobname/phiqutation.tex}}
114 {\end{VerbatimOut}\eolang@process{equation}}
115 \makeatother

```

\phiiq Then, we define \phiiq command:

```

116 \makeatletter\newcommand\phiiq[1]{%
117   \iexec[trace,quiet,stdout=\eolang@tmpdir/\jobname/phiq.tex]{
118     /bin/echo '\detokenize{#1}'}%
119   \def\hash{\eolang@mdfive
120     {\eolang@tmpdir/\jobname/phiq.tex}}%
121   \iexec[null]{cp "\eolang@tmpdir/\jobname/phiq.tex"
122     "\eolang@tmpdir/\jobname/\hash.tex"}%

```

```

123 \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-post.tex]{
124 perl \eolang@tmpdir/eolang-phi.pl 'phiq'
125 "\eolang@tmpdir/\jobname/\hash.tex"}%
126 }\makeatother

```

eolang-sodg.pl Then, we create a Perl script for sodg graphs processing using VerbatimOut from [fancyvrb](#):

```

127 \makeatletter
128 \begin{VerbatimOut}{\eolang@tmpdir/eolang-sodg.pl}
129 sub num {
130   my ($i) = @_ ;
131   $i =~ s/(\+|-)\./\10./g;
132   return $i;
133 }
134 sub fmt {
135   my ($tex) = @_ ;
136   $tex =~ s/|([^\|]+)|/\textnormal{\texttt{\1}}/g;
137   return $tex;
138 }
139 open(my $fh, '<', $ARGV[0]);
140 my $tex; { local $/; $tex = <$fh>; }
141 print '% This file is auto-generated', "\n";
142 print '% There are ', length($tex),
143   ' chars in the input: ', $ARGV[0], "\n";
144 print '% ---', "\n";
145 if (index($tex, "\t") > 0) {
146   print "TABS are prohibited!";
147   exit 1;
148 }
149 foreach my $t (split (/\\n/g, $tex)) {
150   print '% ', $t, "\n";
151 }
152 print '% ---', "\n";
153 $tex =~ s/^\s+|\s+$//g;
154 $tex =~ s/\\\\/\\n/g;
155 $tex =~ s/([a-zA-Z]+) +/\1/g;
156 $tex =~ s/\\n\s+/\\n/g;
157 my @cmds = split (/\\n/g, $tex);
158 print '\begin{picture}', "\n";
159 foreach my $c (@cmds) {
160   my ($head, $tail) = split (/ /, $c, 2);
161   my %opts = {};
162   foreach my $p (split (/ /, $tail)) {
163     my ($q, $t) = split (/:/, $p);
164     $opts{$q} = $t;
165   }
166   if (index($head, '->') >= 0) {
167     print '\draw[';
168     if (exists $opts{'pi'}) {
169       print ',phi-pi';
170       if (not exists $opts{'a'}) {
171         $opts{'a'} = '\pi';
172       }
173     }

```

```

174 print ']';
175 my ($from, $to) = split (/>/, $head);
176 print '(', $from, ')';
177 if (exists $opts{'bend'}) {
178     print 'edge [bend right=', num($opts{'bend'}), ']';
179 } else {
180     print '--';
181 }
182 if (exists $opts{'rho'} or exists $opts{'rrho'}) {
183     print ' pic[sloped,phi-rho]{parallel arrow={';
184     print '-' if not exists $opts{'rrho'};
185     print '0.3,-0.15}}';
186 }
187 if (exists $opts{'a'}) {
188     my $a = $opts{'a'};
189     if (index($a, '$') == -1) {
190         $a = '$' . fmt($a) . '$';
191     } else {
192         $a = fmt($a);
193     }
194     print ' node [phi-attr] {' , $a, '}' ;
195 }
196 print '(', $to, ')';
197 } elsif (index($head, '>') >= 0) {
198     my ($from, $to) = split (/>/, $head);
199     if ($from eq '') {
200         print '\node [phi-arrow, left=.6cm of ' .
201             $to . ']' ;
202     } elsif ($to eq '') {
203         print '\node [phi-arrow, right=.6cm of ' .
204             $from . ']' ;
205     } else {
206         print '\node [phi-arrow] at ($(' .
207             $from . ')!0.5!(' . $to . ')$)' ;
208     }
209     print '{}';
210 } elsif (index($head, '!') >= 0) {
211     my ($v, $marker) = split (/!/, $head);
212     print '\node [phi-marker, left=.6cm of ' .
213         $v . ']{ ' . fmt($marker) . '}' ;
214 } else {
215     print '\node[';
216     if (exists $opts{'xy'}) {
217         my ($v, $right, $down) = split(/,/ , $opts{'xy'});
218         my $loc = '';
219         if ($down > 0) {
220             $loc = 'below';
221         } elsif ($down < 0) {
222             $loc = 'above';
223         }
224         if ($right > 0) {
225             $loc = $loc . 'right';
226         } elsif ($right < 0) {
227             $loc = $loc . 'left';

```

```

228     }
229     print ', ' . $loc . '=';
230     print abs(num($down)) . 'cm and ' .
231         abs(num($right)) . 'cm of ' . $v;
232 }
233 if (exists $opts{'data'}) {
234     print ',phi-data';
235     if (not $opts{'data'} eq '') {
236         my $d = $opts{'data'};
237         if (index($d, '|') == -1) {
238             $d = '$\Delta\phiDotted\text{' .
239                 '\textnormal{\texttt{' . fmt($d) . '}}}$';
240         } else {
241             $d = fmt($d);
242         }
243         $opts{'box'} = $d;
244     }
245 } elsif (exists $opts{'atom'}) {
246     print ',phi-atom';
247     if (not $opts{'atom'} eq '') {
248         my $a = $opts{'atom'};
249         if (index($a, '$') == -1) {
250             $a = '$\lambda\phiDotted{' . fmt($a) . '$';
251         } else {
252             $a = fmt($a);
253         }
254         $opts{'box'} = $a;
255     }
256 } else {
257     print ',phi-object';
258 }
259 print ']';
260 print ' (', $head, ')';
261 print '{';
262 if (index($head, 'v0') == 0) {
263     print '\Phi';
264 } else {
265     $name = $head;
266     $name =~ s/^v/v_/g;
267     $name =~ s/[^0-9]$/g;
268     print $name;
269 }
270 print '$}';
271 if (exists $opts{'box'}) {
272     print ' node[phi-box] at (';
273     print $head, '.south east) {';
274     print $opts{'box'}, ')';
275 }
276 }
277 print ";\n";
278 }
279 print '\end{phicture}', "\n", '\endinput';
280 \end{VerbatimOut}
281 \message{eolang: File with Perl script

```

```

282 '\eolang@tmpdir/eolang-sodg.pl' saved^^J}%
283 \iexec[trace,null]{perl -pi -e 's/(\[a-zA-Z])\s+/\1/g'}
284 "\eolang@tmpdir/eolang-sodg.pl"}
285 \makeatother

```

FancyVerbLine Then, we reset the counter for [fancyvrb](#), so that it starts counting lines from zero when the document starts rendering:

```

286 \setcounter{FancyVerbLine}{0}

```

tikz Then, we include tikz package and its libraries:

```

287 \RequirePackage{tikz}
288 \usetikzlibrary{arrows}
289 \usetikzlibrary{shapes}
290 \usetikzlibrary{decorations}
291 \usetikzlibrary{decorations.pathmorphing}
292 \usetikzlibrary{intersections}
293 \usetikzlibrary{positioning}
294 \usetikzlibrary{calc}
295 \usetikzlibrary{shapes.arrows}

```

picture Then, we define internal environment picture:

```

296 \newenvironment{picture}%
297 {\noindent\begin{tikzpicture}[
298   ->,>=stealth',node distance=0,thick,
299   pics/parallel arrow/.style={
300     code={\draw[-latex,phi-rho] (##1) -- (-##1);}}}%
301 {\end{tikzpicture}}
302 \tikzstyle{phi-arrow} = [fill=white!80!black, single arrow,
303   minimum height=0.5cm, minimum width=0.5cm,
304   single arrow head extend=2mm]
305 \tikzstyle{phi-marker} = [inner sep=0pt, minimum height=1.4em,
306   minimum width=1.4em, font={\small\color{white}\ttfamily},
307   fill=gray]
308 \tikzstyle{phi-thing} = [thick,inner sep=0pt,minimum height=2.4em,
309   draw,font={\small}]
310 \tikzstyle{phi-object} = [phi-thing,circle]
311 \tikzstyle{phi-data} = [phi-thing,regular polygon,
312   regular polygon sides=8]
313 \tikzstyle{phi-empty} = [phi-object]
314 \tikzstyle{phi-rho} = [draw,decorate,decoration={
315   snake,amplitude=.4mm,segment length=2mm,post length=1mm}]
316 \tikzstyle{phi-pi} = [draw,dotted]
317 \tikzstyle{phi-atom} = [phi-object,double]
318 \tikzstyle{phi-box} = [xshift=-5pt,yshift=3pt,draw,fill=white,
319   rectangle,thin,minimum width=1.2em,anchor=north west,
320   font={\scriptsize}]
321 \tikzstyle{phi-attr} = [midway,sloped,inner sep=0pt,
322   above=2pt,sloped/.append style={transform shape},
323   font={\scriptsize},color=black]

```

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

```

324 \makeatletter\newenvironment{sodg}%
325 {\VerbatimEnvironment%
326 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%

```

```

327 \begin{VerbatimOut}
328 {\eolang@tmpdir/\jobname/sodg.tex}}
329 {\end{VerbatimOut}}%
330 \def\hash{\eolang@mdfive
331   {\eolang@tmpdir/\jobname/sodg.tex}}%
332 \iexec[null]{cp "\eolang@tmpdir/\jobname/sodg.tex"
333   "\eolang@tmpdir/\jobname/\hash.tex"}}%
334 \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-post.tex]{
335   perl "\eolang@tmpdir/eolang-sodg.pl"
336   "\eolang@tmpdir/\jobname/\hash.tex"}}%
337 \setcounter{FancyVerbLine}{\value{eolang@lineno}}%
338 }\makeatother

```

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

```

339 \newcommand\eolang{%
340   \ifdefined\anon%
341     \anon[XYZ]{\sffamily EO}}%
342 \else%
343   {\sffamily EO}%
344 \fi%
345 }

```

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

```

346 \RequirePackage{hyperref}
347 \newcommand\phic{%
348   \ifdefined\anon%
349     \anon[\texorpdfstring{\alpha}{a}-calculus]
350     {\texorpdfstring{\varphi}{phi}-calculus}%
351 \else%
352   \texorpdfstring{\varphi}{phi}-calculus%
353 \fi%
354 }

```

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

```

355 \newcommand\xmirc{%
356   \ifdefined\anon%
357     \anon[XML']{XMIR}%
358 \else%
359   XMIR%
360 \fi%
361 }

```

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

```

362 \newcommand\phiConst{%
363   \mathrel{\hspace{.15em}}%
364   \mapstochar\mathrel{\hspace{-.15em}}\mapsto}

```

`\phiWave` Then, we define a command to render an arrow for a multi-layer attribute, as suggested [here](#):

```

365 \newcommand\phiWave{%
366   \mapstochar\mathrel{\mspace{0.45mu}}\leadsto}

```

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

```

367 \RequirePackage{trimclip}
368 \RequirePackage{amsfonts}
369 \makeatletter
370 \newcommand{\phiDotted}{%
371   \mapstochar\mathrel{\mathpalette\phiDotted@\relax}}
372 \newcommand{\phiDotted@}[2]{%
373   \begin{group}
374     \settowidth{\dimen\z@}{\m@th#1\rightarrow}%
375     \settoheight{\dimen\tw@}{\m@th#1\rightarrow}%
376     \sbox\z@{%
377       \makebox[\dimen\z@][s]{%
378         \clipbox{0 0 {0.4\width} 0}%
379         {\resizebox{\dimen\z@}{\height}%
380           {\m@th#1\dashrightarrow}}}%
381       \hss%
382       \clipbox{{0.69\width} {-0.1\height} 0
383         {-\height}}{\m@th#1\rightarrow}%
384     }%
385   }%
386   \ht\z@=\dimen\tw@ \dp\z@=\z@%
387   \box\z@%
388   \endgroup}\makeatother

```

## References

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

## Change History

0.0.1			
	General: First draft. . . . .	6	
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. 12		
	eolang-phi.pl: New symbol added for basket slots . . . . .	7	
	Parsing of symbols “@,” “^,” and “&” enabled (\varphi, \rho, and \sigma) . . . . .	7	
	The symbols “[” and “]” replaced with “[[” and “]]” for abstract object brackets, because they conflicted with normal square brackets . . . . .	7	
	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. 9		
	\phiq: Parsing of additional symbols enabled. . . . .	8	
0.1.0			
	General: Parsing of package options introduced. . . . .	6	
	\eolang: New command \eolang added to print the name of the language in both normal and anonymous mode of acmart. . . . .	13	
	\eolang@mdfive: New supplementary command added to calculate MD5 sum of a file. . . . .	6	
	eolang-phi.pl: A new Perl script “eolang-phi.pl” added for parsing of phi expressions. . . . .	7	
	eolang-sodg.pl: There are two Perl scripts now: one for phiuation, another one for sodg. . . . .	9	
	\phic: New command \phic prints the name of $\varphi$ -calculus in both normal and anonymous mode of acmart. . . . .	13	
	\phiConst: New command \phiConst added to denote a link to a constant attribute. . . . .	13	
	\phiDotted: New command \phiDotted added to denote a link to a special attribute. . . . .	14	
	0.2.0		
	eolang-phi.pl: Numbers automatically render as \texttt. No need to use vertical bars around them anymore. . . . .	7	
	eolang-sodg.pl: The content of atom and data boxes is parsed automatically as formulas and numbers, respectively. . . . .	9	
	\xmirl: New command \xmirl prints XMIR in both normal and anonymous mode of acmart. . . . .	13	
	0.3.0		
	\eolang@lineno: New counter for protecting lineno. . . . .	6	
	eolang-phi.pl: New arrow added, that looks like \leadsto. . . . .	7	
	\phiWave: New command \phiWave added to denote a link to a multi-layer attribute. . . . .	13	
	0.4.0		
	eolang-sodg.pl: Labels on the edges are automatically printed as math formulas. Also, boxes are prefixed with \Delta and \lambda commands. . . . .	9	
	Relative positioning of vertices fixed. 9		

# 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>			
$\backslash \$$ .....	55	$\backslash \text{eolang@mdfive}$ ....	$\backslash \text{makebox}$ ..... 377
$\backslash ($ .....	50	..... <u>18</u> , 93, 119, 330	$\backslash \text{mapsto}$ ..... 364
$\backslash +$ .....	131	$\backslash \text{eolang@process}$ ...	$\backslash \text{mapstochar}$ . 364, 366, 371
$\backslash .$ .....	131	..... 92, 108, 114	$\backslash \text{mathpalette}$ ..... 371
$\backslash ?$ .....	51	$\backslash \text{eolang@tmpdir}$ ....	$\backslash \text{mathrel}$ 363, 364, 366, 371
$\backslash [$ .....	63	... 11, 16, 23, 88,	$\backslash \text{message}$ ..... 87, 281
$\backslash \{$ .....	66, 67	90, 94, 95, 96, 97,	$\backslash \text{mspace}$ ..... 366
$\backslash \}$ .....	66, 67	98, 100, 107, 113,	
$\backslash ]$ .....	64	117, 120, 121, 122,	<b>N</b>
$\backslash ^$ .....	54	123, 124, 125, 128,	$\backslash \text{newcommand}$ ..... 20,
$\backslash  $ .....	62, 136	282, 284, 328, 331,	92, 116, 339, 347,
		332, 333, 334, 335, 336	355, 362, 365, 370, 372
<b>Numbers</b>		<b>F</b>	$\backslash \text{newcounter}$ ..... 17
$\backslash 2$ .....	50, 70	$\backslash \text{FancyVerbLine}$ .... <u>286</u>	$\backslash \text{newenvironment}$ ...
<b>A</b>			... 103, 109, 296, 324
$\backslash \alpha$ .....	349	<b>H</b>	$\backslash \text{node}$ 200, 203, 206, 212, 215
$\backslash \text{anon}$ .....	340,	$\backslash \text{hash}$ ..... 93, 96, 97,	$\backslash \text{noindent}$ ..... 297
341, 348, 349, 356, 357		100, 119, 122, 123,	
<b>B</b>		125, 330, 333, 334, 336	<b>P</b>
$\backslash \text{Bbbk}$ .....	3	$\backslash \text{height}$ .... 379, 382, 383	$\backslash \text{pdf@filemdfivesum}$ .. 20
$\backslash \text{begin}$ ....	23, 43, 106,	$\backslash \text{hspace}$ ..... 363, 364	$\backslash \text{pgfkeys}$ ..... 9
112, 128, 158, 297, 327		$\backslash \text{hss}$ .....	$\backslash \Phi$ ..... 263
$\backslash \text{box}$ .....	387	381	$\backslash \text{phic}$ .....
<b>C</b>		$\backslash \text{ht}$ .....	346
$\backslash \text{clipbox}$ .....	378, 382	386	$\backslash \text{phiConst}$ .....
$\backslash \text{color}$ .....	306	<b>I</b>	362
<b>D</b>		$\backslash \text{iexec}$ 16, 89, 95, 97, 117,	$\backslash \text{picture}$ ..... 296
$\backslash \text{dashrightarrow}$ ...	380	121, 123, 283, 332, 334	$\backslash \text{phiDotted}$ .. 238, 250, 367
$\backslash \text{def}$ .....	93, 119, 330	$\backslash \text{ifdefined}$ .. 340, 348, 356	$\backslash \text{phiDotted@}$ .... 371, 372
$\backslash \Delta$ .....	238	$\backslash \text{ifluatex}$ .....	$\backslash \text{phiq}$ ..... 116
$\backslash \text{detokenize}$ .....	118	12	$\backslash \text{phiquation}$ .....
$\backslash \text{dimen}$ 374, 375, 377, 379, 386		$\backslash \text{ifxetex}$ .....	92
$\backslash \text{dp}$ .....	386	12	$\backslash \text{phiWave}$ .....
$\backslash \text{draw}$ .....	167, 300		365
<b>E</b>		<b>J</b>	$\backslash \pi$ ..... 171
$\backslash \text{end}$ .....	83, 86, 108,	$\backslash \text{jobname}$ .....	$\backslash \text{ProcessPgfoptions}$ .. 15
114, 279, 280, 301, 329		16, 94,	<b>R</b>
$\backslash \text{endinput}$ .....	85, 279	95, 96, 97, 100, 107,	$\backslash \text{relax}$ ..... 3, 371
$\backslash \text{eolang}$ .....	<u>339</u>	113, 117, 120, 121,	$\backslash \text{RequirePackage}$ .. 1,
$\backslash \text{eolang-phi.pl}$ .....	<u>22</u>	122, 123, 125, 328,	2, 3, 4, 5, 6, 7, 8,
$\backslash \text{eolang-sodg.pl}$ ...	<u>127</u>	331, 332, 333, 334, 336	18, 287, 346, 367, 368
$\backslash \text{eolang@lineno}$ .....	<u>17</u>	<b>L</b>	$\backslash \text{resizebox}$ ..... 379
		$\backslash \lambda$ .....	$\backslash \text{rightarrow}$ . 374, 375, 383
		250	
		$\backslash \text{leadsto}$ .....	<b>S</b>
		366	$\backslash \text{sbox}$ ..... 376
<b>M</b>		$\backslash \text{m@th}$ ... 374, 375, 380, 383	$\backslash \text{scriptsize}$ .... 320, 323
$\backslash \text{makeatletter}$ 17, 19, 22,		92, 116, 127, 324, 369	$\backslash \text{setcounter}$ .... 101,
$\backslash \text{makeatother}$ 17, 21, 91,		115, 126, 285, 338, 388	105, 111, 286, 326, 337
			$\backslash \text{settoheight}$ ..... 375
			$\backslash \text{settowidth}$ ..... 374

