



eolang: \LaTeX Package for Formulas and Graphs of EO Programming Language and φ -calculus*

Yegor Bugayenko
yegor256@gmail.com

2023-12-28, 0.17.0

NB! You must run \TeX processor with `--shell-escape` option and you must have [Perl](#) installed. If you omit the `--shell-escape` option, the package will try to use cached files, if they exist. If they don't, compilation will crash. Thus, when you must prepare your document for a compilation without the `--shell-escape` option, run it locally with the option and then package all files (including the files in the `_eolang` directory) into a single ZIP archive.

If `--shell-escape` is set, 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:

*The sources are in GitHub at [objectionary/eolang.sty](#)

$$\begin{aligned} \text{app} &\mapsto \llbracket \\ &\quad \rho \mapsto \xi.b.^2, \alpha_0 | t \rightsquigarrow \text{TRUE}, \\ &\quad b \mapsto \llbracket \alpha_* \mapsto \Phi.\text{fn}(56), \\ &\quad \quad \varphi \mapsto \dot{\Phi}.\text{string.trim}(\xi), \\ &\quad \quad \Delta \mapsto 01\text{-FE-C3} \rrbracket, \\ &\quad x \mapsto \llbracket \lambda \mapsto \emptyset \rrbracket. \end{aligned}$$

```

1 \documentclass{minimal}
2 \usepackage{eolang}
3 \begin{document}
4 \begin{phiquestion*}
5 app -> [[ % it's abstract!
6   ^ !-> $.b.^{~2}, 0/t~> TRUE,
7   b -> [[ *-> Q.fn(56),
8     @ -> QQ.string.trim($),
9     D> 01-FE-C3 ]]],\
10 x -> [[ \lambda .> ? ]].
11 \end{phiquestion*}
12 \end{document}

```

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

- “@” maps to “ φ ” (`\varphi`),
- “^” maps to “ ρ ” (`\rho`),
- “\$” maps to “ ξ ” (`\xi`),
- “&” maps to “ σ ” (`\sigma`),
- “?” maps to “ \emptyset ” (`\varnothing`),
- “Q” maps to “ Φ ” (`\Phi`),
- “QQ” maps to “ $\dot{\Phi}$ ” (`\dot{\Phi}`),
- “->” maps to “ \mapsto ” (`\mapsto`),
- “~>” maps to “ \rightsquigarrow ” (`\rightsquigarrow`),
- “!->” maps to “ \mapsto ” (`\mapsto`),
- “.>” maps to “ \mapsto ” (`\mapsto`),
- “D>” maps to “ $\Delta \mapsto$ ” (`\Delta \mapsto`),
- “L>” maps to “ $\lambda \mapsto$ ” (`\lambda \mapsto`),
- “[[” maps to “ \llbracket ” (`\llbracket`),
- “]]” maps to “ \rrbracket ” (`\rrbracket`),
- “==” maps to “ \equiv ” (`\equiv`),
- “|abc|” maps to “ abc ” (`\text{abc}`).

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

- “<<” maps to “ \langle ” (`\langle`),
- “>>” maps to “ \rangle ” (`\rangle`),
- “-abc>” maps to “ $\xrightarrow{\text{abc}}$ ” (`\xrightarrow{\text{abc}}`),
- “:=” maps to “ \vDash ” (`\vDash`).

Before any arrow you can put a number, which will be rendered as α with an index, for example `\phiq{0->x}` will render “ $\alpha_0 \mapsto x$ ”. Instead of a number you can use asterix too.

You can append a slash and a title to the number of an attribute, such as $0/g \mapsto x$. this will render as $\alpha_0|g \mapsto x$. You can use fixed-width words too, for example `\phiq{0/|f|->x}` will render as “ $\alpha_0|f \mapsto x$ ”. It’s also possible to use an asterisk instead of a number, such that `\phiq{* /g->x}` renders as “ $\alpha_*|g \mapsto x$ ”

Numbers are automatically converted to fixed-width font, no need to always decorate them with vertical bars.

TRUE and FALSE are automatically converted to fixed-width font too.

Object names are automatically converted to fixed-width font too, if they have more than one letter.

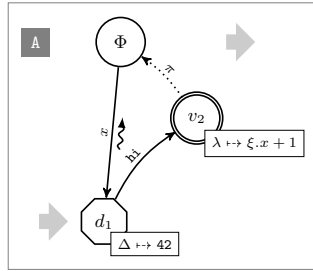
Texts in double quotes are automatically converted to fixed-width font too.

`\phiq` The command `\phiq` lets you inline a φ -calculus expressions using the same simple plain-text notation. You can use dollar sign directly too:

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

```
4 \begin{document}
5 A simple object
6 \phiq{x -> [[@ -> y]]} \
7 is a decorator of
8 the data object \
9 $y -> [[\Delta ..> 42]]$.
10 \end{document}
```

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



```
1 \documentclass{standalone}
2 \usepackage{eolang}
3 \begin{document}
4 \begin{sodg}
5 v0 \ \ v0==> \ \ v0!!A
6 v1 xy:v0,-.8,2.8 data:42 tag:d_1
7 v0->v1 a:x rho \ \ =>v1
8 v2 xy:v0,+1,+1 atom:\xi.x+1
9 v1->v2 a:|hi| bend:-15
10 v2->v0 pi bend:10 % a comment
11 \end{sodg}
12 \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 “ $v0 \rightarrow v1$.” All other markers are either unary like “ ρ ” or binary like “ $\text{atom}:\xi.x+1$.” Binary markers have two parts, separated by colon.

The following markers are supported for a vertex:

- “`tag:<math>`” puts a custom label `<math>` into the circle;
- “`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;
- “+:<v>” makes a copy of an existing vertex and all its kids;
- “edgeless” removes the border from the vertex;
- “style:{...}” adds this TikZ style to the vertex \node.

The following markers are supported for an edge:

- “rho” places a backward snake arrow to the edge,
- “bend:<angle>” bend it right by the amount of “<angle>,”
- “a:<txt>” attaches label “<txt>” to it,
- “pi” makes it dotted, with π label;
- “style:{...}” adds this TikZ style to the edge \path.

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.” If you want the arrow to stay further away from the vertex than usually, use a few “=” symbols, for example “==>v0.”

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. You can make a distance between the vertex and the marker a bit larger by using a few exclamation marks, for example “v5!!!A” will make a distance three times bigger.

You can make a clone of an existing vertex together with all its dependants, by using this syntax: “v0+a.” Here, we make a copy of “v0” and call it “v0a.” See the example below.

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
`\phic` language. It understands the anonymous package option and prints itself differently, to
`\xmirl` double-blind your paper. There is also `\phic` command to print the name of φ -calculus,
also sensitive to anonymous mode. The macro `\xmirl` 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.

```

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

```

Without the anonymous option there will be no orange color:

In our research we use EO,
an experimental object-oriented
dataflow language, φ -calculus, as its
formal foundation, and XMIR —
its XML-based presentation.

```

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

```

`\phiConst` A few simple commands are defined to help you render arrows. It is recommended
`\phiWave` not to use them directly, but use `!->` instead. However, if you want to use `\phiConst`,
`\phiDotted` wrap it in `\mathrel` for better display:

If x is an identifier and y is an object,
then $x \mapsto y$ makes y a constant,
 $x \rightsquigarrow y$ makes it a decoratee
of an arbitrary number of objects,
while $x \vdash y$ makes it a special attribute.

```

6 If $x$ is an identifier and $y$ is
7 an object, then $x \phiConst y$
8 makes $y$ a constant,
9 $x \phiWave y$ makes it a decoratee
10 of an arbitrary number of objects,
11 while $x \phiDotted y$ makes it
12 a special attribute.

```

`\phiOset` If you want to put a text over an arrow or under it, use `\phiOset` and `\phiUset`
`\phiUset` respectively:

When the names of attributes and
their values don't matter, we use
an arrow with a star, for example:

$$[[\mapsto*]]$$

```

6 When the names of attributes and their
7 values don't matter, we use an arrow
8 with a star, for example:
9 \begin{phiquestion*}
10 [[ \phiOset{*}{->} ]]
11 \end{phiquestion*}

```

`\phiMany` Sometimes you may need to simplify the way you describe an object (the typesetting
is a bit off, but this is not because of us, but because of [this](#)):

The expression $[\alpha_1 \mapsto x_1, \alpha_2 \mapsto x_2, \dots, \alpha_n \mapsto x_n]$
and expression $[\alpha_i \mapsto_{i=1}^n x_i]$ are
syntactically different but semanti-
cally equivalent.

```

6 The expression
7 \phiiq{[[ 1-> x_1,
8 2-> x_2, \dots,
9 \alpha_n -> x_n ]]}
10 and expression
11 \phiiq{[[ \alpha_i
12 \phiMany{->}{i=1}{n} x_i ]]}
13 are syntactically different but
14 semantically equivalent.

```

`\phiSaveTo` If you want to use `phiquestion` or `sodg` environments inside `tabular` or any other
`\sodgSaveTo` environment or command, you won't be able to do this, because `phiquestion` and `sodg`
are “verbatim” environments. `\phiSaveTo` and `\sodgSaveTo` commands will help you
in this situation. You use them right before `\begin{phiquestion}` or `\begin{sodg}`
respectively — the content of the equation or the graph won't be rendered, but instead
saved to the file. Later, inside `tabular`, you can use it through the `\input` macro (don't
forget the `\parbox`):

| | |
|--------|---|
| Free: | $\llbracket x \mapsto \emptyset \rrbracket$ |
| Bound: | $\llbracket x \mapsto \llbracket \Delta \mapsto 42 \rrbracket \rrbracket$ |

```

5 \phiSaveTo{a}
6 \begin{phiquestion*}
7 [[ x -> [[D>42]] ]]
8 \end{phiquestion*}
9 \begin{tabular}{p{.5in}l}
10 Free: & $[[x -> ?]]$ \\
11 Bound: & \parbox{1in}{\input{a}} \\
12 \end{tabular}

```

`\eoAnon` You may want to hide some of the content with the help of the anonymous package option. The command `\eoAnon` may help you with this. It has two parameters: one mandatory and one optional. The mandatory one is the content you want to show and the optional one is the substitution we will render if the anonymous package option is set.

2 Package Options

`tmpdir` The default location of temp files is `_eolang`. You can change this with the help of the `tmpdir` package option:

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

`nodollar` You may disable the special treatment of the dollar sign by using the `nodollar` package option:

```
\usepackage[nodollar]{eolang}
```

`anonymous` You may anonymize `\eolang`, `\XMIR`, and `\phic` commands by using anonymous package option (they all use the `\eoAnon` command mentioned earlier):

```
\usepackage[anonymous]{eolang}
```

3 More Examples

The `phiquestion` 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:

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

```

6 \begin{phiquestion*}
7 \dfrac \{
8 {x->[[@->y]] \quad y->[[z->42]]} \{
9 {x.z -> 42} \}
10 \text{\sffamily R1}
11 \end{phiquestion*}

```

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

$$\frac{x \mapsto \llbracket \varphi \mapsto y, z \mapsto 42, \alpha_0 \mid g \mapsto \emptyset, \alpha_1 \mid \text{foo} \mapsto 42 \rrbracket}{x \mapsto \llbracket \varphi \mapsto y, z \mapsto \emptyset, f \rightsquigarrow \text{pi}(\alpha_0 \mapsto \llbracket \psi \mapsto \text{hello}(12) \rrbracket, \alpha_1 \mapsto 42) \rrbracket} \text{R2.}$$

```

6 \begin{phiuation*}
7 \dfrac{\begin{split}
8 x->[[@->y, z->42,
9 0/g->?, 1/foo->42]]
10 \end{split}}{\begin{split}
11 x->[[@->y, z->?, f ~> |pi|(
12 0->[[ \psi !-> |hello|(12) ]],
13 1->42)]]
14 \end{split}}\text{R2}.
15 \end{phiuation*}

```

You can use the `matrix` environment too, in order to group a few lines:

$$\text{foo} \mapsto \left\{ \begin{array}{c} \emptyset \\ \llbracket \lambda \mapsto \rho \times \xi. \alpha_0 \rrbracket \\ \llbracket \Delta \mapsto 42 \rrbracket \end{array} \right\}$$

```

5 \begin{phiuation*}
6 foo -> \left\{\begin{matrix} \backslash
7 ? \backslash\backslash
8 [[ L> ~ \times $. \alpha_0 ]] \backslash
9 [[ D> 42 ]] \backslash
10 \end{matrix}\right\}
11 \end{phiuation*}

```

The `cases` environment works too:

$$\beta \models \begin{cases} [v_2, \varphi \xrightarrow{\text{DTZD}} 42] \\ [v_{33}] \end{cases}$$

```

5 \begin{phiuation*}
6 \beta := \begin{cases} \backslash
7 [ v_2, @ -dtzd> 42 ] \backslash\backslash
8 [ v_{33} ] \backslash
9 \end{cases}
10 \end{phiuation*}
11 \end{document}

```

The `phiuation` environment may be used together with the [acmart](#) package:

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

```

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

```

It's possible to use `\label` inside the `phiuation` environment (pay attention to how you can disable our custom parsing of math formulas by means of curled brackets around the “4” number):

Discriminant can be calculated using the following simple formula:

$$D = b^2 - 4ac. \quad (1)$$

Eq. 1 is also widely used in number theory and polynomial factoring.

```
6 Discriminant can be calculated using
7 the following simple formula:
8 \begin{phiuation}
9 D = b{^2} - {4}ac.
10 \label{d}
11 \end{phiuation}
12 Eq.\ref{d} is also widely used in
13 number theory and polynomial factoring.
```

You can add comments to your equations, using the `&&` command (pay attention, the text inside `\text{}` is not processed and treated like a plain text):

| | |
|--|-------------------|
| $\llbracket \alpha_0 \mapsto x \rrbracket$ | This is formation |
| $\llbracket \alpha_0 \mapsto \emptyset \rrbracket$ | Abstraction |
| $x(\Delta \mapsto 42)$ | Application |

```
6 \begin{phiuation*}
7 [[ 0->x ]] && \text{This is formation}
8 [[ 0->? ]] && \text{Abstraction}
9 x(D>42) && \text{Application}
10 \end{phiuation*}
```

If you don't use `nodollar` package option, you can still use normal parsing of the dollar sign, by means of `\(...\)` syntax:

The object formation $\llbracket \alpha_0 \mapsto x \rrbracket$ may be replaced with a formula $Q \times a^2$.

```
6 The object formation $[[0->x]]$
7 may be replaced with a formula
8 \(( Q \times a^2 \)).
```

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

| |
|---|
| $x(\pi) \mapsto \llbracket \lambda \mapsto f_1 \rrbracket,$ |
| $x(a, b, c) \mapsto \llbracket \alpha_0 \mapsto \emptyset, \varphi \mapsto \text{hello}(\xi), x \mapsto \text{FALSE} \rrbracket,$ |
| $\Delta = 43-09,$ |
| $x(y) \equiv x(\alpha_0 \mapsto y).$ |

```
5 \begin{phiuation*}
6 x(\pi) -> [[\lambda \mapsto f_1]], \\\
7 x(a,b,c) -> [[ \alpha_0 -> ?, \ \
8   @ -> |hello|($), x -> |FALSE| ]], \\\
9 \Delta = |43-09|,
10 x(y) == x(0-> y).
11 \end{phiuation*}
```

If not a single line is indented in `phiuation`, all formulas will be centered:

| |
|---|
| $\llbracket b \mapsto \emptyset \rrbracket,$ |
| $\llbracket \varphi \mapsto \text{TRUE}, \Delta \mapsto 42 \rrbracket,$ |
| $\psi = \langle \pi, 42 \rangle.$ |

```
5 \begin{phiuation*}
6 [[ b -> ? ]],
7 [[ @ -> TRUE, \Delta \mapsto 42 ]], \\\
8 \psi = << \pi, 42 >>.
9 \end{phiuation*}
```

It is possible to use “manual splitting” mode in the `phiuation` environment by starting the body with `\begin{split}`:

$$x(\pi) \mapsto 4$$

$$x(a,b,c) \mapsto \llbracket \alpha_0 \mapsto \emptyset \rrbracket$$

```

5 \begin{phiuation*}
6 \begin{split}
7 x(\pi) &\mapsto 4 \\
8 x(a,b,c) &\mapsto \llbracket \alpha_0 \mapsto \emptyset \rrbracket \\
9 \end{split}
10 \end{phiuation*}

```

When necessary to use a percentage sign, prepend it with a backward slash:

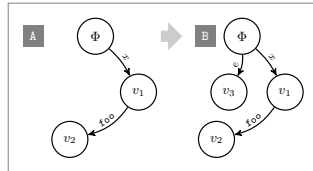
$$x \mapsto \text{sprintf}(\text{"Hello, \%s!"}, \text{name})$$

```

5 \begin{phiuation*}
6 x \mapsto \text{sprintf}(\text{"Hello, \%s!"}, \text{name})
7 \end{phiuation*}
8 \end{document}

```

You can make a copy of a vertex together with its kids:

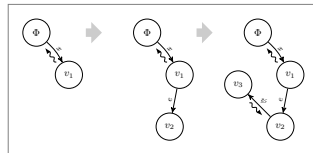


```

5 \begin{sodg}
6 v0 \\\ v0!!A
7 v1 xy:v0,.7,1
8 v0->v1 a:x bend:-10
9 v2 xy:v1,-1.3,.8
10 v1->v2 a:|foo| bend:-20
11 v0+a xy:v0,3,0
12 v3a xy:v0a,-.7,1
13 v0a->v3a a:e bend:-15
14 v0=>v0a \\\ v0a!B
15 \end{sodg}

```

You can make a copy from a copy:

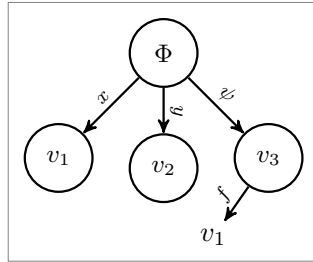


```

5 \begin{sodg}
6 v0
7 v1 xy:v0,.7,1
8 v0->v1 a:x bend:-10 rho
9 v0+a xy:v0,3,0 \\\ v0=>v0a
10 v2a xy:v1a,-.8,1.3
11 v1a->v2a a:e
12 v0a+b xy:v0a,3,0 \\\ v0a=>v0b
13 v3b xy:v2b,-1,-1
14 v2b->v3b a:\psi{} rho
15 \end{sodg}

```

You can have “broken” edges, using “break” attribute of an edge. The attribute must have a value, which is the percentage of the path between vertices that the arrow should take (can’t be more than 80 and less than 20). This may be convenient when you can’t fit all edges into the graph, for example:

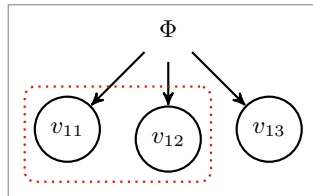


```

5 \begin{sodg}
6 v0
7 v1 xy:v0,-1,1
8 v0->v1 a:x
9 v2 xy:v0,0,1
10 v0->v2 a:y
11 v3 xy:v0,1,1
12 v0->v3 a:\psi{}
13 v3->v1 a:f bend:-75 break:30
14 \end{sodg}

```

You can add [TikZ](#) commands to `sodg` graph, for example:

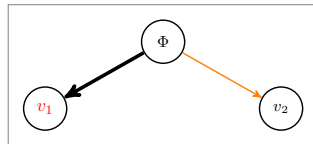


```

6 \begin{sodg}
7 v0 edgeless
8 v11 xy:v0,-1,1 \\\ v0->v11
9 v12 xy:v0,0,1 \\\ v0->v12
10 v13 xy:v0,1,1 \\\ v0->v13
11 \node[draw=red,rounded corners,\
12 dotted,fit=(v11) (v12)] {};
13 \end{sodg}

```

You can modify TikZ style yourself (make sure `style:` stays at the end of the line!), for example:



```

6 \begin{sodg}
7 v0
8 v1 xy:v0,-2,1 style:font=\color{red}
9 v2 xy:v0,2,1
10 v0->v1 style:line width=2pt
11 v0->v2 style:draw=orange
12 \end{sodg}

```

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   nocomments/.store in=\eolang@nocomments,
14   anonymous/.store in=\eolang@anonymous,
15   tmpdir
16 }
17 \ProcessPgfpPackageOptions{/eolang}

```

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

```

18 \ifnum\ShellEscapeStatus=1%
19   \iexec[null]{mkdir -p "\eolang@tmpdir/\jobname"}%
20 \else%
21   \message{eolang: Temporary directory "\eolang@tmpdir/\jobname"
22     is not created, because --shell-escape is not set^^J}%
23 \fi%

```

\eolang@lineno Then, we define an internal counter to protect line number from changing:

```

24 \makeatletter\newcounter{eolang@lineno}\makeatother

```

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

```

25 \RequirePackage{pdftexcmds}
26 \makeatletter
27 \newcommand\eolang@mdfive[1]{\pdf@filemdfivesum{#1}}
28 \makeatother

```

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

```

29 \makeatletter
30 \begin{VerbatimOut}{\eolang@tmpdir/eolang-phi.pl}
31 $macro = $ARGV[0];
32 open(my $fh, '<', $ARGV[1]);
33 my $tex; { local $/; $tex = <$fh> }
34 print "% This file is auto-generated by 0.17.0\n";
35 print '% There are ', length($tex),
36   ' chars in the input: ', $ARGV[1], "\n";
37 print '% ---', "\n";
38 if (index($tex, "\t") > 0) {
39   print "TABS are prohibited!";
40   exit 1;
41 }
42 my @lines = split (/\\n/g, $tex);
43 foreach my $t (@lines) {
44   print '% ', $t, "\n";
45 }
46 print '% ---', "\n";
47 $tex =~ s/(?<\\)\%.*\\n\\n/g;
48 $tex =~ s/^\\s+|\\s+$/g;
49 my $splitting = $tex =~ /^\\begin\\{split\\}/;

```

```

50 if ($splitting) {
51   print '% The manual splitting mode is ON since \begin{split} started the text' . "\n";
52 }
53 my $indents = $tex =~ /\n +/g;
54 my $gathered = (0 == $indents);
55 if ($gathered) {
56   if ($splitting) {
57     print '% The "gathered" is NOT used because of manual splitting' . "\n";
58     $gathered = 0;
59   } else {
60     print '% The "gathered" is used since all lines are left-aligned' . "\n";
61   }
62 } else {
63   print '% The "gathered" is NOT used because ' .
64     $indents . " lines are indented\n";
65 }
66 my $align = 0;
67 print '% The "align" is NOT used by default' . "\n";
68 if (index($tex, '&&') >= 0) {
69   $macro =~ s/equation/align/g;
70   $align = 1;
71   print '% The "align" is used because of && seen in the text' . "\n";
72 }
73 if ($macro ne 'phiq') {
74   if (not $splitting) {
75     $tex =~ s/\\\\\\n/\n\n/g;
76     $tex =~ s/\\n\s*//g;
77   }
78   $tex =~ s/\n*(\\label\{[^\}]+\})\n*/\1/g;
79   $tex =~ s/\n{3,}/\n\n/g;
80 }
81 my @texts = ();
82 sub trep {
83   my ($s) = @_;
84   my $open = 0;
85   my $p = 0;
86   for (; $p < length($s); $p++) {
87     $c = substr($s, $p, 1);
88     if ($c eq '}') {
89       if ($open eq 0) {
90         last;
91       }
92       $open--;
93     }
94     if ($c eq '{') {
95       $open++;
96     }
97   }
98   push(@texts, substr($s, 0, $p));
99   return '{TEXT' . (0+@texts - 1) . '}' . substr($s, $p + 1);
100 }
101 $tex =~ s/\\text\{(.+)/trep("$1")/ge;
102 if (not $splitting) {
103   $tex =~ s/(?<![&])&(?![&])/\\sigma{/g;

```

```

104 }
105 $tex ~ s/([~\{a-z0-9\}|^)\QQ(?![a-z0-9])/1\dot{\Phi}/g;
106 $tex ~ s/([~\{a-z0-9\}|^)\Q(?![a-z0-9])/1\Phi/g;
107 $tex ~ s/([~\{a-z0-9\}|^)\D>/1\Delta{}/g;
108 $tex ~ s/([~\{a-z0-9\}|^)\L>/1\lambda{}/g;
109 $tex ~ s/"([~"]+)"|"\1"/g;
110 $tex ~ s/(\^(?<=[\s](\[\[,.>\/]))([a-zA-Z][a-z0-9]+)(?=[\s](\[\[,.>\/])|$\)|2|/g;
111 $tex ~ s/([~_]|^)([0-9]+|\*)\/(\^[a-z]+|_|[a-z]+|_|)
112 (->|\.\. >|~>|:=|!->)/1\alpha_{2}\vert{3}\space{4}/xg;
113 $tex ~ s/([~_]|^)([0-9]+|\*)
114 (->|\.\. >|~>|:=|!->)/1\alpha_{2}\space{3}/xg;
115 if ($macro ne 'phiq') {
116   if (not $splitting) {
117     $tex ~ s/\begin{split}\n\begin{split}&/g;
118     $tex ~ s/\n\s*\end{split}\end{split}/g;
119     $tex ~ s/\n\n\\&/g;
120     $tex ~ s/\n/\phiEOL{}\n&/g;
121     $tex ~ s/\\$/g;
122     $tex ~ s/\\/\n/g;
123     $tex ~ s/([~\s])\s{2}([~\s])/1 \2/g;
124     $tex ~ s/\s{2}/ \quad/g;
125     $tex = '&' . $tex;
126   }
127   my $lead = '[~\s]+\s(?:->|:=|!<=)\s';
128   my @leads = $tex ~ /&${lead}/g;
129   my @eols = $tex ~ /\n/g;
130   if (0+@leads == 0+@eols && 0+@eols > 1) {
131     $tex ~ s/&($lead)/1~/g;
132     $gathered = 0;
133     print '% The "gathered" is NOT used because all ' .
134       (0+@eols) . ' lines are ' . (0+@leads) . " leads\n";
135   }
136 }
137 if ($macro ne 'phiq') {
138   sub strip_tabs {
139     my ($env, $tex) = @_;
140     $tex ~ s/&/g;
141     return "\begin{$env}" . $tex . "\end{$env}";
142   }
143   foreach my $e (('matrix', 'cases')) {
144     $tex ~ s/\begin{(\Q$e\E*?)\}(.\+)\end{(\Q$e\E*?)\}/strip_tabs($1, $2)/sge;
145   }
146 }
147 $tex ~ s/\$/\xi{/g;
148 $tex ~ s/(?!{ }\^(!{ }\)/\rho{/g;
149 $tex ~ s/[[/\llbracket\mathbin{/g;
150 $tex ~ s/[ ]/\mathbin{\rrbracket}/g;
151 $tex ~ s/([~\s,>()]{0-9A-F}{2}(?:-[0-9A-F]{2})+|
152 [0-9]+(?:\.[0-9]+)?)?(?!{ }\)/1|2|/xg;
153 $tex ~ s/TRUE/|TRUE|/g;
154 $tex ~ s/FALSE/|FALSE|/g;
155 $tex ~ s/\?/\varnothing{/g;
156 $tex ~ s/@/\varphi{/g;
157 $tex ~ s/-([a-z]+)>/\mathrel{\phiSlot{1}}/g;

```

```

158 $tex =~ s/!->/\mathbin{\!\!\phiConst}/g;
159 $tex =~ s/->/\mathbin{\!\!\mapsto}/g;
160 $tex =~ s/~>/\mathbin{\!\!\phiWave}/g;
161 $tex =~ s/:=\mathrel{\!\!\vDash}/g;
162 $tex =~ s/==\mathrel{\!\!\equiv}/g;
163 $tex =~ s/\.\.\>/\mathbin{\!\!\phiDotted}/g;
164 $tex =~ s/<</\langle/g;
165 $tex =~ s/>>/\rangle/g;
166 $tex =~ s/|{2,}/|/g;
167 $tex =~ s/|([^\|]+)\|/\textnormal{\!\!\texttt{\!1}}\!/g;
168 $tex =~ s/{TEXT(d+)}\'\' \text{\' . @texts[$1] . \'\'}/ge;
169 if ($macro eq 'phiq') {
170   print '$' if ($tex ne '');
171 } else {
172   print '\begin{\' , $macro, "}\n";
173   if (not($align)) {
174     if ($gathered) {
175       print '\begin{gathered}' . "\n";
176     } elsif (not $splitting) {
177       print '\begin{split}' . "\n";
178     }
179   }
180 }
181 if ($gathered and not($align)) {
182   $tex =~ s/^&/g;
183   $tex =~ s/\n&/\n/g;
184 }
185 print $tex;
186 if ($macro eq 'phiq') {
187   print '$' if ($tex ne '');
188 } else {
189   if (not($align)) {
190     if ($gathered) {
191       print "\n" . '\end{gathered}';
192     } elsif (not $splitting) {
193       print "\n" . '\end{split}';
194     }
195   }
196   print "\n" . '\end{\' . $macro . \'\'';
197 }
198 print '\endinput';
199 \end{VerbatimOut}
200 \message{eolang: File with Perl script
201   '\eolang@tmpdir/eolang-phi.pl' saved~^J}%
202 \makeatother

```

`\phiSaveTo` Then, we define the `\phiSaveTo` command to instruct the `phi`uation environment that the output should not be sent to the document but saved to the file instead:

```

203 \makeatletter
204 \newcommand\phiSaveTo[1]{\def\eolang@phiSaveTo{#1}}
205 \makeatother

```

`\eolang@ifabsent` Then, we define the `\eolang@ifabsent` command, which if a given file is absent, runs a processing command, otherwise just inputs it:

```

206 \makeatletter
207 \newcommand\eolang@ifabsent[2]{%
208   \IfFileExists
209     {#1}
210     {%
211       \message{eolang: File "#1" already exists ^^J}%
212       \input{#1}}
213   {%
214     \ifnum\ShellEscapeStatus=1\else%
215       \message{eolang: The --shell-escape command line
216         option is not provided, most probably compilation
217         will fail now:^^J}%
218       \fi%
219     #2%
220   }%
221 }
222 \makeatother

```

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

```

223 \makeatletter\newcommand\eolang@process[1]{
224   \def\hash{\eolang@mdfive
225     {\eolang@tmpdir/\jobname/phiquation.tex}-\the\inputlineno}%
226   \eolang@ifabsent
227     {\eolang@tmpdir/\jobname/\hash-post.tex}
228     {%
229       \iexec[null]{cp "\eolang@tmpdir/\jobname/phiquation.tex"
230         "\eolang@tmpdir/\jobname/\hash.tex"}%
231       \message{Start parsing 'phi' at line no. \the\inputlineno^^J}
232       \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-post.tex]{
233         perl "\eolang@tmpdir/eolang-phi.pl"
234         '#1'
235         "\eolang@tmpdir/\jobname/\hash.tex"
236         \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\n|$)//g'\fi
237         \ifdefined\eolang@phiSaveTo > \eolang@phiSaveTo\fi}%
238     }%
239   \setcounter{FancyVerbLine}{\value{eolang@lineno}}%
240   \def\eolang@phiSaveTo{\relax}%
241 }
242 %
243 \newenvironment{phiquation*}%
244 {\catcode'\|=12 \VerbatimEnvironment%
245 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
246 \begin{VerbatimOut}
247   {\eolang@tmpdir/\jobname/phiquation.tex}}
248 {\end{VerbatimOut}\eolang@process{equation*}}
249 %
250 \newenvironment{phiquation}%
251 {\catcode'\|=12 \VerbatimEnvironment%
252 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
253 \begin{VerbatimOut}
254   {\eolang@tmpdir/\jobname/phiquation.tex}}
255 {\end{VerbatimOut}\eolang@process{equation}}
256 \makeatother

```

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

```

257 \RequirePackage{xstring}
258 \makeatletter\newcommand\phi[1]{%
259   \StrSubstitute{\detokenize{#1}}{' '}{''''}[\clean]%
260   \def\hash{\pdf@mdfivesum{\clean}-\the\inputlineno}%
261   \ifdefined\eolang@nodollar\else\catcode'\$=3 \fi%
262   \eolang@ifabsent
263     {\eolang@tmpdir/\jobname/\hash-phi-post.tex}
264     {%
265       \iexec[log,trace,quiet,stdout=\eolang@tmpdir/\jobname/phi.tex]{
266         /bin/echo '\clean'}%
267       \iexec[quiet,null]{cp "\eolang@tmpdir/\jobname/phi.tex"
268         "\eolang@tmpdir/\jobname/\hash-phi.tex"}%
269       \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-phi-post.tex]{
270         perl \eolang@tmpdir/eolang-phi.pl '\phi'
271         "\eolang@tmpdir/\jobname/\hash-phi.tex"
272         \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\n|$)//g' \fi}%
273     }%
274   \ifdefined\eolang@nodollar\else\catcode'\$=\active\fi%
275 }\makeatother

```

`nodollar` Then, we redefine dollar sign:

```

276 \ifdefined\eolang@nodollar\else
277   \begingroup
278   \catcode'\$=\active
279   \protected\gdef$#1${\phi{#1}}
280   \endgroup
281   \AtBeginDocument{\catcode'\$=\active}
282 \fi

```

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

```

283 \makeatletter
284 \begin{VerbatimOut}{\eolang@tmpdir/eolang-sodg.pl}
285 sub num {
286   my ($i) = @_;
287   $i =~ s/(\+|-)\./10./g;
288   return $i;
289 }
290 sub fmt {
291   my ($tex) = @_;
292   $tex =~ s/\\|([^\|]+)\\|\\textnormal{\texttt{\1}}/g;
293   return $tex;
294 }
295 sub vertex {
296   my ($v) = @_;
297   if (index($v, 'v0') == 0) {
298     return '\Phi';
299   } else {
300     $v =~ s/~v/v_/g;
301     $v =~ s/[^0-9]$/g;
302     return $v . ' ';
303   }
304 }

```



```

305 sub tailor {
306   my ($t, $m) = @_;
307   $t =~ s/<([A-Z]?${m}[A-Z]?)([~>+)>/\2/g;
308   $t =~ s/<[A-Z]+:[~>+>>//g;
309   return $t;
310 }
311 open(my $fh, '<', $ARGV[0]);
312 my $tex; { local $/; $tex = <$fh>; }
313 if (index($tex, "\t") > 0) {
314   print "TABS are prohibited!";
315   exit 1;
316 }
317 print '% This file is auto-generated', "\n%\n";
318 print '% --- there are ', length($tex),
319   ' chars in the input (', $ARGV[0], "):\n";
320 foreach my $t (split /\n/g, $tex) {
321   print '% ', $t, "\n";
322 }
323 print "% ---\n";
324 $tex =~ s/\\\\\\/\n/g;
325 $tex =~ s/\\\n//g;
326 $tex =~ s/(\\[a-zA-Z]+)\s+/\1/g;
327 $tex =~ s/\n{2,}/\n/g;
328 my @cmds = split /\n/g, $tex;
329 print '% --- before processing:' . "\n";
330 foreach my $t (split /\n/g, $tex) {
331   print '% ', $t, "\n";
332 }
333 print '% ---';
334 print ' (' . (0+@cmds) . " lines)\n";
335 print '\begin{picture}', "\n";
336 for (my $c = 0; $c < 0+@cmds; $c++) {
337   my $cmd = $cmds[$c];
338   $cmd =~ s/^\\s+//g;
339   $cmd =~ s/(?<!\n)%.*//g;
340   my ($head, $tail) = split(/ /, $cmd, 2);
341   my %opts = {};
342   my ($body, $style) = split(/style:/, $tail, 2);
343   $opts{'style'} = $style;
344   $tail = $body;
345   foreach my $p (split(/ /, $tail)) {
346     my ($q, $t) = split(/:/, $p);
347     $opts{$q} = $t;
348   }
349   if (index($head, '\\') == 0) {
350     print $cmd;
351   } elsif (index($head, '->') >= 0) {
352     my $draw = '\draw[';
353     if (exists $opts{'pi'}) {
354       $draw = $draw . '<MB:phi-pi><F:draw=none>';
355       if (not exists $opts{'a'}) {
356         $opts{'a'} = '\pi';
357       }
358     }

```

```

359 if (exists $opts{'rho'}) and not(exists $opts{'bend'})) {
360   $draw = $draw . '<MB:,phi-rho>';
361 }
362 $draw = $draw . ',' . $opts{'style'} . ']';
363 my ($from, $to) = split (/>/, $head);
364 $draw = $draw . " ($from) ";
365 if (exists $opts{'bend'}) {
366   $draw = $draw . 'edge [<F:draw=none><MF:,bend right=' .
367     num($opts{'bend'}) . '>';
368   if (exists $opts{'rho'}) {
369     $draw = $draw . '<MB:,phi-rho>';
370   }
371   $draw = $draw . ']';
372 } else {
373   $draw = $draw . '--';
374 }
375 if (exists $opts{'a'}) {
376   my $a = $opts{'a'};
377   if (index($a, '$') == -1) {
378     $a = '$' . fmt($a) . '$';
379   } else {
380     $a = fmt($a);
381   }
382   $draw = $draw . '<MB: node [phi-attr] {' . $a . '>';
383 }
384 if (exists $opts{'break'}) {
385   $draw = $draw . '<F: coordinate [pos=' .
386     ($opts{'break'} / 100) . '] (break)>';
387 }
388 $draw = $draw . " (<MF:${to}><B:break-v>)" ;
389 if (exists $opts{'break'}) {
390   print tailor($draw, 'F') . ";\n";
391   print ' \node[outer sep=.1cm,inner sep=0cm] ' .
392     'at (break) (break-v) {' . vertex($to) .
393     '$};' . "\n";
394   print ' ' . tailor($draw, 'B');
395 } else {
396   print tailor($draw, 'M');
397 }
398 } elsif (index($head, '=>') >= 0) {
399   my ($from, $to) = split (/=>/, $head);
400   my $size = () = $head =~ /=/g;
401   if ($from eq '') {
402     print '\node [phi-arrow, left=' . ($size * 0.6) . 'cm of ' .
403       $to . '.center]';
404   } elsif ($to eq '') {
405     print '\node [phi-arrow, right=' . ($size * 0.6) . 'cm of ' .
406       $from . '.center]';
407   } else {
408     print '\node [phi-arrow] at ($(' .
409       $from . ')!0.5!(' . $to . ')$)';
410   }
411   print '{}';
412 } elsif (index($head, '!'') >= 0) {

```

```

413 my ($v, $marker) = split (/!+/, $head);
414 my $size = () = $head =~ /!/g;
415 print '\node [phi-marker, left=' .
416 ($size * 0.6) . 'cm of ' .
417 $v . '.center][' . ' . fmt($marker) . ']' ;
418 } elsif (index($head, '+') >= 0) {
419 my ($v, $suffix) = split (/!+/, $head);
420 my @friends = ($v);
421 foreach my $c (@cmds) {
422     $e = $c;
423     $e =~ s/^s+//g;
424     my $h = $e;
425     $h = substr($e, 0, index($e, ' ')) if index($e, ' ') >= 0;
426     foreach my $f (@friends) {
427         my $add = '';
428         if (index($h, $f . '->') >= 0) {
429             $add = substr($h, index($h, '->') + 2);
430         }
431         if ($h =~ /->\Q${f}\E/) {
432             $add = substr($h, 0, index($h, '->'));
433         }
434         if (index($e, ' xy:' . $f . ',') >= 0) {
435             $add = $h;
436         }
437         if (index($add, '+') == -1
438             and $add ne ''
439             and not(grep(/^Q${add}\E$/, @friends))) {
440             push(@friends, $add);
441         }
442     }
443 }
444 my @extra = ();
445 foreach my $e (@cmds) {
446     $m = $e;
447     if ($m =~ /^s*\Q${v}\E/s/) {
448         next;
449     }
450     if ($m =~ /^s*[^\s]+\s/ and not($m =~ /^s*\Q${head}\E/s/)) {
451         next;
452     }
453     foreach my $f (@friends) {
454         my $h = $f;
455         $h =~ s/[a-z]$//g;
456         if ($m =~ s/^(s*)\Q${f}\E+\Q${suffix}\E?s?/\1${h}${suffix} /g) {
457             last;
458         }
459         $m =~ s/^(s*)\Q${f}\E/s/\1${h}${suffix} /g;
460         $m =~ s/^(s*)\Q${f}\E->/\1${h}${suffix}->/g;
461         $m =~ s/\sxy:\Q${f}\E,/ xy:${h}${suffix},/g;
462         $m =~ s/->\Q${f}\E/s/->${h}${suffix} /g;
463     }
464     if ($m ne $e) {
465         push(@extra, ' ' . $m);
466     }

```

```

467 }
468 splice(@extra, 0, 0, @extra[-1]);
469 splice(@extra, -1, 1);
470 splice(@extra, 0, 0, '% clone of ' . $v . ' (' . $head .
471 '), friends: [' . join(', ', @friends) . '] in ' .
472 (0+@cmds) . ' lines');
473 splice(@cmds, $c, 1, @extra);
474 print '% cloned ' . $v . ' at line no.' . $c .
475 ' (+ ' . (0+@extra) . ' lines -> ' .
476 (0+@cmds) . ' lines total)';
477 } elsif ($head =~ /^v[0-9]+[a-z]?$/) {
478   print '\node[';
479   if (exists $opts{'xy'}) {
480     my ($v, $right, $down) = split(/,/ , $opts{'xy'});
481     my $loc = '';
482     if ($down > 0) {
483       $loc = 'below ';
484     } elsif ($down < 0) {
485       $loc = 'above ';
486     }
487     if ($right > 0) {
488       $loc = $loc . 'right';
489     } elsif ($right < 0) {
490       $loc = $loc . 'left';
491     }
492     print ', ' . $loc . '=';
493     print abs(num($down)) . 'cm and ' .
494     abs(num($right)) . 'cm of ' . $v . '.center';
495   }
496   if (exists $opts{'data'}) {
497     print ',phi-data';
498     if ($opts{'data'} ne '') {
499       my $d = $opts{'data'};
500       if (index($d, '|') == -1) {
501         $d = '$\Delta\phiDotted\text{' .
502         '\textnormal{\texttt{' . fmt($d) . '}}}'$';
503       } else {
504         $d = fmt($d);
505       }
506       $opts{'box'} = $d;
507     }
508   } elsif (exists $opts{'atom'}) {
509     print ',phi-atom';
510     if ($opts{'atom'} ne '') {
511       my $a = $opts{'atom'};
512       if (index($a, '$') == -1) {
513         $a = '$\lambda\phiDotted{' . fmt($a) . '$';
514       } else {
515         $a = fmt($a);
516       }
517       $opts{'box'} = $a;
518     }
519   } else {
520     print ',phi-object';

```

```

521 }
522 if (exists $opts{'edgeless'}) {
523     print ',draw=none';
524 }
525 print ',, . $opts{'style'} . ']' ;
526 print ' ( ' . $head . ')';
527 print ' {' ;
528 if (exists $opts{'tag'}) {
529     my $t = $opts{'tag'};
530     if (index($t, '$') == -1) {
531         $t = '$' . $t . '$';
532     } else {
533         $t = fmt($t);
534     }
535     print $t;
536 } else {
537     print '$' . vertex($head) . '$';
538 }
539 print '}' ;
540 if (exists $opts{'box'}) {
541     print ' node[phi-box] at (' ;
542     print $head, '.south east) {' ;
543     print $opts{'box'}, '}' ;
544 }
545 }
546 print ";\n";
547 }
548 print '\end{picture}%', "\n";
549 print "% --- after processing:\n%";
550 foreach my $c (@cmds) {
551     print '% ', $c, "\n";
552 }
553 print '% --- ( ' . (0+@cmds) . " lines)\n";
554 print '\endinput';
555 \end{VerbatimOut}
556 \message{eolang: File with Perl script
557   '\eolang@tmpdir/eolang-sodg.pl' saved^^J}%
558 \makeatother

```

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

```

559 \setcounter{FancyVerbLine}{0}

```

tikz Then, we include [tikz](#) package and its libraries:

```

560 \RequirePackage{tikz}
561 \usetikzlibrary{arrows}
562 \usetikzlibrary{shapes}
563 \usetikzlibrary{decorations}
564 \usetikzlibrary{decorations.pathmorphing}
565 \usetikzlibrary{decorations.pathreplacing}
566 \usetikzlibrary{positioning}
567 \usetikzlibrary{calc}
568 \usetikzlibrary{math}
569 \usetikzlibrary{arrows.meta}

```

picture Then, we define internal environment phicture:

```

570 \newenvironment{phicture}%
571 {\noindent\begin{tikzpicture}[
572   ->,>=stealth',node distance=0,thick,
573   pics/parallel arrow/.style={
574     code={\draw[-latex,phi-rho] (##1) -- (-##1);}}]}%
575 {\end{tikzpicture}}
576 \tikzstyle{phi-arrow} = [fill=white!80!black, single arrow,
577   minimum height=0.5cm, minimum width=0.5cm,
578   single arrow head extend=2mm]
579 \tikzstyle{phi-marker} = [inner sep=0pt, minimum height=1.4em,
580   minimum width=1.4em, font={\small\color{white}\ttfamily},
581   fill=gray]
582 \tikzstyle{phi-thing} = [thick,inner sep=0pt,minimum height=2.4em,
583   draw,font={\small}]
584 \tikzstyle{phi-object} = [phi-thing,circle]
585 \tikzstyle{phi-data} = [phi-thing,regular polygon,
586   regular polygon sides=8]
587 \tikzstyle{phi-empty} = [phi-object]
588 \tikzset{%
589   phi-rho/.style={
590     postaction={%
591       decoration={
592         show path construction,
593         curveto code={
594           \tikzmath{
595             coordinate \I, \F, \v;
596             \I = (\tikzinputsegmentfirst);
597             \F = (\tikzinputsegmentlast);
598             \v = ($(\I) -(\F)$);
599             real \d, \a, \r, \t;
600             \d = 0.8;
601             \t = atan2(\vy, \vx);
602             if \vx<0 then { \a = 90; } else { \a = -90; };
603             {
604               \draw[arrows={-latex}, decorate,
605                 decoration={%
606                   snake, amplitude=.4mm,
607                   segment length=2mm,
608                   post length=1mm
609                 }]
610               ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
611               -- ++(\t: 2*\d em);
612             };
613           }
614         },
615         lineto code={
616           \tikzmath{
617             coordinate \I, \F, \v;
618             \I = (\tikzinputsegmentfirst);
619             \F = (\tikzinputsegmentlast);
620             \v = ($(\I) -(\F)$);
621             real \d, \a, \r, \t;
622             \d = 0.8;

```

```

623         \t = atan2(\vy, \vx);
624         if \vx<0 then { \a = 90; } else { \a = -90; };
625         {
626             \draw[arrows={-latex}, decorate,
627                 decoration={%
628                     snake, amplitude=.4mm,
629                     segment length=2mm,
630                     post length=1mm}]
631                 ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
632                 -- ++(\t: 2*\d em);
633         };
634     }
635 }
636 },
637 decorate
638 }
639 }
640 }
641 \tikzstyle{phi-pi} = [draw,dotted]
642 \tikzstyle{phi-atom} = [phi-object,double]
643 \tikzstyle{phi-box} = [xshift=-5pt,yshift=3pt,draw,fill=white,
644     rectangle,thin,minimum width=1.2em,anchor=north west,
645     font={\scriptsize}]
646 \tikzstyle{phi-attr} = [midway,sloped,inner sep=0pt,
647     above=2pt,sloped/.append style={transform shape},
648     font={\scriptsize},color=black]

```

\sodgSaveTo Then, we define the \sodgSaveTo command to instruct the sodg environment that the output should not be sent to the document but saved to the file instead:

```

649 \makeatletter
650 \newcommand\sodgSaveTo[1]{\def\eolang@sodgSaveTo{#1}}
651 \makeatother

```

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

```

652 \makeatletter\newenvironment{sodg}%
653 {\catcode'\|=12 \VerbatimEnvironment%
654 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
655 \begin{VerbatimOut}
656   {\eolang@tmpdir/\jobname/sodg.tex}}
657 {\end{VerbatimOut}}%
658 \def\hash{\eolang@mdfive
659   {\eolang@tmpdir/\jobname/sodg.tex}-\the\inputlineno}%
660 \catcode'\$=3 %
661 \eolang@ifabsent
662   {\eolang@tmpdir/\jobname/\hash-sodg-post.tex}
663   {%
664     \iexec[null]{cp "\eolang@tmpdir/\jobname/sodg.tex"
665       "\eolang@tmpdir/\jobname/\hash.tex"}%
666     \message{eolang: Start parsing 'sodg' at line no. \the\inputlineno^^J}
667     \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-sodg-post.tex]{
668       perl "\eolang@tmpdir/eolang-sodg.pl"
669       "\eolang@tmpdir/\jobname/\hash.tex"
670       \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\n|$)//g'\fi
671       \ifdefined\eolang@sodgSaveTo > \eolang@sodgSaveTo\fi}%

```

```

672 }
673 \catcode'\$ \active%
674 \setcounter{FancyVerbLine}{\value{eolang@lineno}}%
675 \def\eolang@sodgSaveTo{\relax}%
676 }\makeatother

```

`\eoAnon` Then, we define a supplementary command to help us anonymize some content.

```

677 \RequirePackage{hyperref}
678 \pdfstringdefDisableCommands{
679   \def\({}%
680   \def\)}%
681   \def\alpha{alpha}%
682   \def\varphi{phi}%
683 }
684 \makeatletter
685 \NewExpandableDocumentCommand{\eoAnon}{O{ANONYMIZED}m}{%
686   \ifdefined\eolang@anonymous%
687     \textcolor{orange}{#1}%
688   \else%
689     #2%
690   \fi%
691 }\makeatother

```

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

```

692 \newcommand\eolang{%
693   \eoAnon[XYZ]{\sffamily EO}}

```

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

```

694 \newcommand\phic{%
695   \eoAnon[(\alpha)-cal-cu-lus]{(\varphi)-cal-cu-lus}}

```

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

```

696 \newcommand\xmirl{%
697   \eoAnon[XML{^+}]{XMIR}}

```

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

```

698 \newcommand\phiConst{%
699   \mathrel{\hspace{.15em}}%
700   \mapstochar\mathrel{\hspace{-.15em}}\mapsto}

```

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

```

701 \newcommand\phiWave{%
702   \mapstochar\mathrel{\mspace{0.45mu}}\leadsto}

```

`\phiSlot` Then, we define a command to render an arrow for a slot in a basket:

```

703 \newcommand\phiSlot[1]{%
704   \xrightarrow{\text{\sffamily\scshape #1}}}

```


`\phi0set` Then, we define two commands to position a text over and under an arrow, as suggested [here](#):

```

705 \makeatletter
706 \newcommand{\phi0set}[2]{%
707   \mathrel{\mathop{#2}\limits^{
708     \vbox to 0ex{\kern-2\ex@
709       \hbox{$\scriptscriptstyle#1$\vss}}}}
710 \newcommand{\phiUset}[2]{%
711   \mathrel{\mathop{#2}\limits_{
712     \vbox to 0ex{\kern-6.3\ex@
713       \hbox{$\scriptscriptstyle#1$\vss}}}}
714 \makeatother

```

`\phiMany` Then, we define a command for an arrow with iterating indecies:

```

715 \newcommand{\phiMany}[3]{%
716   \phi0set{#3}{\phiUset{#2}{#1}}

```

`\phiEOL` Then, we define a command for line breaks in formulas:

```

717 \newcommand{\phiEOL}{\[-4pt]}

```

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

```

718 \RequirePackage{trimclip}
719 \RequirePackage{amsfonts}
720 \makeatletter
721 \newcommand{\phiDotted}{%
722   \mapstochar\mathrel{\mathpalette\phiDotted@\relax}}
723 \newcommand{\phiDotted@}[2]{%
724   \begingroup%
725   \settowidth{\dimen\z@}{\m@th#1\rightarrow}%
726   \settoheight{\dimen\tw@}{\m@th#1\rightarrow}%
727   \sbox\z@{%
728     \makebox[\dimen\z@][s]{%
729       \clipbox{0 0 {0.4\width} 0}%
730       {\resizebox{\dimen\z@}{\height}%
731         {\m@th#1\dashrightarrow}}}%
732     \hss%
733     \clipbox{{0.69\width} {-0.1\height} 0
734       {-\height}}{\m@th#1\rightarrow}%
735   }%
736 }%
737 \ht\z@=\dimen\tw@ \dp\z@=\z@%
738 \box\z@%
739 \endgroup%
740 }
741 \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. | 10 | 0.12.1 | <code>eolang-sodg.pl</code> : The bug is fixed related to the formatting of indexes of vertices. | 16 |
| 0.0.2 | <code>sodg</code> : The environment <code>phigure</code> renamed to <code>sodg</code> for the sake of better semantic. The graph in the picture is solely a SODG graph, that's why the name <code>sodg</code> is better. | 23 | 0.13.0 | <code>eolang-phi.pl</code> : Parsing of <code>QQ</code> into <code>\dot{\Phi}</code> implemented. | 11 |
| | <code>eolang-phi.pl</code> : New symbol added for basket slots | 11 | 0.14.0 | <code>eolang-sodg.pl</code> : The <code>edgeless</code> tag of a vertex removes the border of it. | 16 |
| | Parsing of the symbols “@,” “^,” and “&” enabled (<code>\varphi</code> , <code>\rho</code> , and <code>\sigma</code>) | 11 | 0.15.0 | <code>eolang-sodg.pl</code> : The <code>style</code> tag of vertices and edges. | 16 |
| | The symbols “[” and “]” replaced with “[[” and “]]” for abstract object brackets, because they conflicted with normal square brackets | 11 | 0.16.0 | <code>phiquation</code> : The processing of <code>phiquation</code> data is done only if it's the first time processing, otherwise cache is used, thus making processing faster. | 15 |
| | <code>eolang-sodg.pl</code> : 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. | 16 | | <code>sodg</code> : The processing of <code>sodg</code> data is done only if it's the first time processing, otherwise cache is used, thus making processing faster. | 23 |
| | <code>\phiq</code> : Parsing of additional symbols enabled. | 16 | 0.17.0 | <code>\eolang@ifabsent</code> : A new supplementary <code>eolang@ifabsent</code> command added | 14 |
| 0.1.0 | General: Parsing of package options introduced. | 11 | 0.2.0 | <code>eolang-phi.pl</code> : Numbers automatically render as <code>\texttt</code> . No need to use vertical bars around them anymore. | 11 |
| | <code>\eolang</code> : New command <code>\eolang</code> added to print the name of the language in both normal and the anonymous mode of <code>acmart</code> | 24 | | <code>eolang-sodg.pl</code> : The content of the <code>atom</code> and the <code>data</code> boxes is parsed automatically as formulas and numbers, respectively. | 16 |
| | <code>\eolang@mdfive</code> : New supplementary command added to calculate MD5 sum of a file. | 11 | | <code>\xm</code> : New command <code>\xm</code> prints XMIR in both normal and the anonymous mode of <code>acmart</code> | 24 |
| | <code>eolang-phi.pl</code> : A new Perl script “ <code>eolang-phi.pl</code> ” added for parsing of <code>phi</code> expressions. | 11 | 0.3.0 | <code>\eolang@lineno</code> : New counter for protecting <code>lineno</code> | 11 |
| | <code>eolang-sodg.pl</code> : There are two Perl scripts now: one for <code>phiquation</code> , another one for <code>sodg</code> | 16 | | <code>eolang-phi.pl</code> : New arrow added, that looks like <code>\leadsto</code> | 11 |
| | <code>\phic</code> : New command <code>\phic</code> prints the name of φ -calculus in both normal and the anonymous mode of <code>acmart</code> | 24 | | <code>\phiWave</code> : New command <code>\phiWave</code> added to denote a link to a multi-layer attribute. | 24 |
| | <code>\phiConst</code> : New command <code>\phiConst</code> added to denote a link to a constant attribute. | 24 | 0.4.0 | <code>eolang-sodg.pl</code> : Labels on the edges are automatically printed as math | |
| | <code>\phiDotted</code> : New command <code>\phiDotted</code> added to denote a link to a special attribute. | 25 | | | |

| | | | |
|---|----|--|----|
| formulas. Also, boxes are prefixed with the <code>\Delta</code> and the <code>\lambda</code> commands. | 16 | <code>eolang-phi.pl</code> : New syntax sugar for Φ , just using capital “Q” is enough. | 11 |
| Relative positioning of vertices fixed. | 16 | Object names are automatically converted to <code>\texttt</code> , provided their names include two or more symbols. | 11 |
| 0.5.0 | | Text in quotes is automatically converted to <code>\texttt</code> | 11 |
| <code>eolang-phi.pl</code> : Automated formatting of TRUE and FALSE added. | 11 | 0.8.0 | |
| <code>eolang-sodg.pl</code> : It is possible to use TikZ commands inside the <code>sodg</code> environment. | 16 | General: The anonymous package option added. | 11 |
| New syntax introduced that allows to make clones of vertices and all their dependants. | 16 | <code>eolang-phi.pl</code> : Inside <code>phiquation</code> any text inside the <code>\text</code> macro is not processed. | 11 |
| Now edges may have the <code>break</code> attribute, to make them shorter. | 16 | <code>eolang-sodg.pl</code> : The <code>tag</code> attribute is introduced for changing labels inside a vertex circle. | 16 |
| <code>\phiMany</code> : New command <code>\phiMany</code> enables iterating over an arrow. | 25 | <code>\phiOset</code> : New commands <code>\phiOset</code> and <code>\phiUset</code> help position text over and under an arrow. | 25 |
| <code>\phiSlot</code> : New command <code>\phiSlot</code> added to denote a link to a slot in a basket. | 24 | <code>\phiSaveTo</code> : The output of the <code>phiquation</code> environment can be redirected to a file. | 14 |
| 0.6.0 | | <code>\sodgSaveTo</code> : The output of the <code>sodg</code> environment can be redirected to a file. | 23 |
| General: Package option <code>nocomments</code> added in order to enable comments suppression in temporary <code>.tex</code> files (may be pretty important for <code>.dtx</code> documents). | 11 | 0.9.0 | |
| <code>eolang-sodg.pl</code> : The <code>rrho</code> attribute is retired, now <code>rho</code> works just fine in all situations. | 16 | <code>\eoAnon</code> : New command <code>\eoAnon</code> added. | 24 |
| 0.7.0 | | <code>eolang-phi.pl</code> : Proper handling of the <code>matrix</code> environment. | 11 |
| <code>nodollar</code> : Now it is possible to use dollar sign instead of the <code>\phiq</code> command. | 16 | <code>\phiEOL</code> : New command <code>\phiEOL</code> added, instead of <code>\[-4pt]</code> | 25 |

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 | D | F |
|--|--|---|
| \\$ 147, 261, 274, 278, 281, 660, 673 | \d .. 168, 599, 600, 610, 611, 621, 622, 631, 632 | \F 595, 597, 598, 610, 617, 619, 620, 631 |
| \% 236, 272, 670 | \dashrightarrow ... 731 | \FancyVerbLine 559 |
| \(..... 679, 695, 697 | \def 204, 224, 240, 260, 650, 658, 675, 679, 680, 681, 682 | G |
| \) 680, 695, 697 | \Delta 501 | \gdef 279 |
| * 111, 113, 144 | \detokenize 259 | H |
| \+ 287, 419, 450, 456 | \dimen 725, 726, 728, 730, 737 | \hash 224, 227, 230, 232, 235, 260, 263, 268, 269, 271, 658, 662, 665, 667, 669 |
| \- 695 | \dp 737 | \hbox 709, 713 |
| \. .. 112, 114, 152, 163, 287 | \draw ... 352, 574, 604, 626 | \height 730, 733, 734 |
| \/ 110, 111 | E | \hspace 699, 700 |
| \? 155 | \E 144, 431, 439, 447, 450, 456, 459, 460, 461, 462 | \hss 732 |
| \[..... 110, 149 | \end 191, 193, 196, 199, 248, 255, 548, 555, 575, 657 | \ht 737 |
| \{ 49, 78, 101, 117, 118, 144, 148, 152, 168 | \endinginput 198, 554 | I |
| \} 49, 78, 117, 118, 144, 168 | \eoAnon . 677, 693, 695, 697 | \I 595, 596, 598, 610, 617, 618, 620, 631 |
| \] 110, 150 | \eolang 692 | \iexec ... 19, 229, 232, 265, 267, 269, 664, 667 |
| \^ 148 | \eolang-phi.pl 29 | \ifdefined 236, 237, 261, 272, 274, 276, 670, 671, 686 |
| \ 111, 166, 167, 244, 251, 292, 653 | \eolang@sodg.pl ... 283 | \IfFileExists 208 |
| Numbers | \eolang@anonymous 14, 686 | \ifluatex 12 |
| \2 110, 112, 114, 123, 152, 307 | \eolang@ifabsent 206, 226, 262, 661 | \ifnum 18, 214 |
| \3 112, 114 | \eolang@lineno 24 | \ifxetex 12 |
| \4 112 | \eolang@mdfive 25, 224, 658 | \input 212 |
| A | \eolang@nocomments ... 13, 236, 272, 670 | \inputlineno 225, 231, 260, 659, 666 |
| \a 599, 602, 610, 621, 624, 631 | \eolang@nodollar 261, 274, 276 | J |
| \active . 274, 278, 281, 673 | \eolang@phiSaveTo 204, 237, 240 | \jobname 19, 21, 225, 227, 229, 230, 232, 235, 247, 254, 263, 265, 267, 268, 269, 271, 656, 659, 662, 664, 665, 667, 669 |
| \alpha 681, 695 | \eolang@process 223, 248, 255 | K |
| \AtBeginDocument .. 281 | \eolang@sodgSaveTo 650, 671, 675 | \kern 708, 712 |
| B | \eolang@tmpdir .. 11, 19, 21, 30, 201, 225, 227, 229, 230, 232, 233, 235, 247, 254, 263, 265, 267, 268, 269, 270, 271, 284, 557, 656, 659, 662, 664, 665, 667, 668, 669 | L |
| \Bbbk 3 | \ex@ 708, 712 | \lambda 513 |
| \begin 30, 51, 172, 175, 177, 246, 253, 284, 335, 571, 655 | C | \leadsto 702 |
| \box 738 | \catcode 244, 251, 261, 274, 278, 281, 653, 660, 673 | |
| C | \clean 259, 260, 266 | |
| \clipbox 729, 733 | \color 580 | |

| | | | | | |
|--|---|--|---|---|--|
| <code>\limits</code> | 707, 711 | <code>\phiConst</code> | 698 | <code>\sxy</code> | 461 |
| M | | <code>\picture</code> | 570 | T | |
| <code>\m@th</code> | 725, 726, 731, 734 | <code>\phiDotted</code> | 501, 513, 718 | <code>\t</code> | 38, 313, 599, 601, 610, 611, 621, 623, 631, 632 |
| <code>\makeatletter</code> | 24, 26, 29, 203, 206, 223, 258, 283, 649, 652, 684, 705, 720 | <code>\phiDotted@</code> | 722, 723 | <code>\text</code> | 501, 704 |
| <code>\makeatother</code> | 24, 28, 202, 205, 222, 256, 275, 558, 651, 676, 691, 714, 741 | <code>\phiEOL</code> | 717 | <code>\textcolor</code> | 687 |
| <code>\makebox</code> | 728 | <code>\phiMany</code> | 715 | <code>\textnormal</code> | 502 |
| <code>\mapsto</code> | 700 | <code>\phiOset</code> | 705, 716 | <code>\texttt</code> | 502 |
| <code>\mapstochar</code> | 700, 702, 722 | <code>\phii</code> | 257, 279 | <code>\the</code> | 225, 231, 260, 659, 666 |
| <code>\mathop</code> | 707, 711 | <code>\phiiquation</code> | 223 | <code>\tikz</code> | 560 |
| <code>\mathpalette</code> | 722 | <code>\phiiSaveTo</code> | 203 | <code>\tikzinputsegmentfirst</code> | 596, 618 |
| <code>\mathrel</code> | 699, 700, 702, 707, 711, 722 | <code>\phiiSlot</code> | 703 | <code>\tikzinputsegmentlast</code> | 597, 619 |
| <code>\message</code> | 21, 200, 211, 215, 231, 556, 666 | <code>\phiiUset</code> | 710, 716 | <code>\tikzmath</code> | 594, 616 |
| <code>\mspace</code> | 702 | <code>\phiiWave</code> | 701 | <code>\tikzset</code> | 588 |
| N | | <code>\pi</code> | 356 | <code>\tikzstyle</code> | 576, 579, 582, 584, 585, 587, 641, 642, 643, 646 |
| <code>\newcommand</code> | 27, 204, 207, 223, 258, 650, 692, 694, 696, 698, 701, 703, 706, 710, 715, 717, 721, 723 | <code>\ProcessPgfPackageOptions</code> | 17 | <code>\ttfamily</code> | 580 |
| <code>\newcounter</code> | 24 | <code>\protected</code> | 279 | <code>\tw@</code> | 726, 737 |
| <code>\newenvironment</code> | 243, 250, 570, 652 | Q | | U | |
| <code>\NewExpandableDocumentCommand</code> | 685 | <code>\Q</code> | 144, 431, 439, 447, 450, 456, 459, 460, 461, 462 | <code>\usetikzlibrary</code> | 561, 562, 563, 564, 565, 566, 567, 568, 569 |
| <code>\node</code> | 391, 402, 405, 408, 415, 478 | R | | V | |
| <code>\nodollar</code> | 276 | <code>\relax</code> | 3, 240, 675, 722 | <code>\v</code> | 595, 598, 617, 620 |
| <code>\noindent</code> | 571 | <code>\RequirePackage</code> | 1, 2, 3, 4, 5, 6, 7, 8, 25, 257, 560, 677, 718, 719 | <code>\value</code> | 239, 245, 252, 654, 674 |
| P | | <code>\resizebox</code> | 730 | <code>\varphi</code> | 682, 695 |
| <code>\pdf@filemdfivesum</code> | 27 | <code>\rightarrow</code> | 725, 726, 734 | <code>\vbox</code> | 708, 712 |
| <code>\pdf@mdfivesum</code> | 260 | S | | <code>\VerbatimEnvironment</code> | 244, 251, 653 |
| <code>\pdfstringdefDisableCommands</code> | 678 | <code>\sbox</code> | 727 | <code>\vss</code> | 709, 713 |
| <code>\pgfkeys</code> | 9 | <code>\scriptscriptstyle</code> | 709, 713 | <code>\vx</code> | 601, 602, 623, 624 |
| <code>\Phi</code> | 298 | <code>\scriptsize</code> | 645, 648 | <code>\vy</code> | 601, 623 |
| <code>\phic</code> | 694 | <code>\scshape</code> | 704 | W | |
| P | | <code>\setcounter</code> | 239, 245, 252, 559, 654, 674 | <code>\width</code> | 729, 733 |
| <code>\pdf@filemdfivesum</code> | 27 | <code>\settoheight</code> | 726 | X | |
| <code>\pdf@mdfivesum</code> | 260 | <code>\settowidth</code> | 725 | <code>\xmir</code> | 696 |
| <code>\pdfstringdefDisableCommands</code> | 678 | <code>\sffamily</code> | 693, 704 | <code>\xrightarrow</code> | 704 |
| <code>\pgfkeys</code> | 9 | <code>\ShellEscapeStatus</code> | 259 | Z | |
| <code>\Phi</code> | 298 | <code>\small</code> | 580, 583 | <code>\z@</code> | 725, 727, 728, 730, 737, 738 |
| <code>\phic</code> | 694 | <code>\sodg</code> | 652 | | |
| P | | <code>\sodgSaveTo</code> | 649 | | |
| <code>\pdf@filemdfivesum</code> | 27 | <code>\StrSubstitute</code> | 259 | | |
| <code>\pdf@mdfivesum</code> | 260 | | | | |
| <code>\pdfstringdefDisableCommands</code> | 678 | | | | |
| <code>\pgfkeys</code> | 9 | | | | |
| <code>\Phi</code> | 298 | | | | |
| <code>\phic</code> | 694 | | | | |