Maclane pentagon is some comonadic descent

Christopher Mary
EGITOR.NET, https://github.com/mozert

March 20, 2015

With the recent versions of LyX and with the `preview`-style installed in the LaTeX-System, the graph drawing package XY-pic can be conveniently used inside LyX. Diagrams can be edited and displayed inside the main LyX editing window. Here, we shall describe how to use the `\xymatrix` command from `xypic` inside LyX in order to draw, to edit and to preview diagrams as typically used in category theory, algebra, and related fields.

Contents

1 Contents **2**

2 Preparation **3**

3 Commutative diagrams **4**

3.1 The matrix layout of diagrams 4
3.2 Arrows 5
3.3 Labels 5
3.4 Arrow modification 6
3.4.1 Arrow design 6
3.4.2 Designing your own arrows 7
3.5 Arrow positioning 8
3.5.1 Inline or centered diagrams 8
3.6 Bending arrows 8
3.6.1 Raising the shaft 8
3.6.2 Specifying exit- and entrance directions 8
3.7 Modifying vertices 9
3.7.1 Framing objects 9
3.7.2 Framing rectangles 10

4 Using LyX's math editor **10**
4.1 Caveat - how to enter braces 10
4.2 Setting up the matrix 10
4.3 Entering nodes, arrows and labels 11
4.4 Modifying arrows 11
4.5 What if something goes wrong 11

5 Hacks **12**
5.1 Horizontal and vertical scaling 12
5.2 Label positioning 12
5.3 Invisible stretched arrows 13
5.4 Further `Xy`-tricks 13

1 Contents

This text responds to Gross *Coq Categories Experience* [1] and Chlipala *Compositional Computational Reflection* [2] of ITP 2014. "Compositional" is synonymous for functional/functorial; "Computational Reflection" is synonymous for monadic semantics; and this text attempts some comonadic descent along functorial semantics : Dosen semiassociative coherence covers Maclane associative coherence [3].

Categories [4] [5] study the interaction between reflections and limits. The basic configuration for reflections is :

$$\begin{tikzcd}
\text{CoMod} \arrow[r, "\text{reflection}"] \arrow[d, "\text{identity}"'] \arrow[r, phantom, shift right=6, "\overset{\text{unit}}{\Longrightarrow}"] & \text{Mod} \arrow[dl, "\text{coreflection}"] \\
\text{CoMod} \arrow[d, "\text{reification}"'] & \\
\text{Modos} &
\end{tikzcd}$$

where, for all reification functor into any Modos category, the map
$(\_ \star \text{reflection}) \circ (\text{reification} \star \text{unit})$ is bijective, or same, for all object M' in CoMod, the map $(\text{coreflection}\ \_\ ) \circ \text{unit}_{M'}$ is bijective; and it is said that the unit natural/polymorphic/commuting transformation is the unit of the reflection and the reflective pair $(\text{reification} \circ \text{coreflection}, \text{reification} \star \text{unit})$ is some coreflective ("Kan") extension functor of the reification functor along the reflection functor.

The `xypic`-package has long served as a convenient tool for easily constructing graphs and diagrams in LaTeX. Unfortunately, its use in LyX had long been restricted to the infamous TeX code boxes, meaning that the LyX editor could only display the

LaTeX-source and not the finished diagram. The new `preview`-style of LaTeX which is part of the AUCTeX project[?], finally enables the editing and displaying of `xypic`-diagrams, constructed, displayed and interactively edited inside LyX.

In this note, we describe how XY-pic can be used from inside LyX, how diagrams can be created and edited. We have tested the following using LyX versions 1.3.7 up to 1.6, running under Windows XP and under Windows Vista.

There are two modes of operations: For a start, and for some first tests, it may be easiest to first enter the XY-pic code inside the LyX-window, select it all and convert it to a graphical representation by pressing Ctrl-m or Ctrl-M. If you use XY-pic more frequently, or if you want to modify your initial figure, you will want to assemble and modify your figures using LyX's math editor.

Once the cursor is moved over a diagram, this is displayed as an array of nodes and arrow-commands. These can be changed interactively. When the cursor leaves the editing area, the diagram reappears.

In the first two sections of this documentation, we explain how to use LyX in the first mentioned mode and we introduce all XY-pic features that might be of use for drawing commutative diagrams, graphs or automata. Section 4 explains how to use the XY-pic commands inside a math-editing area.

It is not our intention to write another introduction to XY-pic, rather our motivation is to give an introduction how the most important commands work inside LyX, since the keystrokes as explained in the XY-pic manual[?] will not always function correctly inside LyX.

2 Preparation

The following requires that the LaTeX-packages `xypic` and `preview` are installed in the LaTeX system. They are available from CTAN, see at [?], resp. at [?]. After freshly installing them, it may be necessary, to run Tools ▷ Reconfigure from the main LyX menu. The steps to a first diagram output in LyX are then:

1. Activate and test `preview`
   a) Open LyX, choose Tools ▷ Preferences ▷ Look and Feel ▷ Display and turn *Instant Preview* on.
   b) In Document ▷ Settings ▷ Math Options, uncheck *Use AMS Math package automatically* and check *Use AMS Math package.*
   c) Test, if `instant-preview` works by opening a LyX-document and entering any math-formula, e. g. $a + b = c$.
   d) Move the cursor out of the formula, and watch it change its appearance to look just like in the finished DVI- or PostScript document.

2. Activate and test XY-pic:

   a) Inside your LYX-Document, enter the text
   `\xymatrix{A \ar[r] & B}` .

   b) Select the whole text and choose Insert ▷ Math ▷ Display Formula, or use the corresponding keyboard shortcut Ctrl-M.

   c) Move the mouse cursor out of the editing box and wait for a split second to see an arrow appear: $A \longrightarrow B$.

3 Commutative diagrams

The following diagram, which is taken from the documentation of XY-pic[?] by its creator Kristoffer H. Rose, will provide an example for many of the features available with that package. Its source code is:

```
\xymatrix{
  U     \ar@/_{1pc}/[ddr]_\psi\ar@/^{1pc}/[drr]^\varphi
\ar@{.>}[dr]|-{(x,y)}\\
    & X \times_Z Y \ar[d]^q \ar[r]_p  &  X \ar[d]_f\\
    & Y \ar[r]^g                      &  Z }
```

Again, to turn this code into a graphical output, select it all at once starting from the `\xymatrix{` ... up to the closing brace ... `}` and turn it into display-math as explained above. A moment after the cursor leaves the math-area, you should see the diagram in its full graphical glory as shown below.

$$\begin{tikzcd}
U \arrow[rrd, bend left, "\varphi"] \arrow[rdd, bend right, "\psi"'] \arrow[rd, dotted, "{(x,y)}" description] & & \\
& X \times_Z Y \arrow[r, "p"'] \arrow[d, "q"] & X \arrow[d, "f"'] \\
& Y \arrow[r, "g"] & Z
\end{tikzcd}$$

3.1 The matrix layout of diagrams

`xymatrix` uses a matrix to define the layout of the vertices of a diagram. For the above example, we need a 3×3-matrix of which 5 entries are used for the vertices U, $X \times_Z Y$, X, Y, Z, the other positions remaining empty. In this case, the following matrix determines the layout:

```
\xymatrix{
    U                      \\
       & X\times_Z Y   & X  \\
       & Y             & Z    }
```

The pattern should be familiar from LaTeX: We see three rows, the first two being terminated by the end-of-line-marker `\\` . Each line consists of entries, separated by the ampersand `&`.

3.2 Arrows

Having entered the vertices, we add arrows between them. The basic `xypic`-command to produce an arrow is `\ar` , it is entered into the cell of the matrix where the arrow is to start. The target of the arrow is defined by direction commands `u` (up) `d` (down) `l` (left), or `r` (right). These can be combined to a path and enclosed in square brackets. As an example, the arrows from the vertex U in the upper left corner down and right to the vertices $X \times_Z Y$, Y, and X are, respectively, defined as `\ar[dr]`, `\ar[ddr]` and `\ar[drr]`. Thus the above diagram with all arrows added becomes:

```
\xymatrix{
    U \ar[ddr] \ar[drr] \ar[dr]\\
    & X \times_Z Y \ar[d] \ar[r]& X \ar[d]\\
    & Y \ar[r] & Z }
```

$$\begin{tikzcd}
U \arrow[dr] \arrow[ddr] \arrow[drr] & & \\
& X \times_Z Y \arrow[r] \arrow[d] & X \arrow[d] \\
& Y \arrow[r] & Z
\end{tikzcd}$$

3.3 Labels

Labels are attached to arrows by affixing them as upper or lower indices to the `\ar`-command. Thus, `\ar[drr]^\varphi` defines an arrow going one cell down, two to the right and having the label φ attached above. To attach a label below the arrow, make it a lower index as in `\ar[ddr]_\psi`. This explanation is correct only for arrows pointing to the right. More precisely, imagine looking along the arrow in the direction it is pointing. Then an upper index places a label to the left and a lower index places it to the right. Consequently, an arrow pointing from right to left, such as `\ar[l]^\alpha_\beta` will have label α below and label β above the arrow, e. g. $\xleftarrow[\alpha]{\beta}$. Using the character | instead of ^ or _ , it is even possible to place the label right onto the arrow, obscuring part of its shaft.

Normally, a label is placed halfway between an arrow's start and target objects. In the first diagram, the central arrow starting in U has the label (x, y) in the middle of the arrow's shaft, rather than in the middle between the two objects it connects. This is achieved by prefixing the label with a minus sign, here: `\ar[dr]|-{(x,y)}`.

```
\xymatrix{
    U \ar[ddr]_\psi \ar[drr]^\varphi\ar[dr]|-{(x,y)}\\
    & X \times_Z Y \ar[d]^q \ar[r]_p& X \ar[d]_f\\
    & Y \ar[r]^g & Z }
```

$$\begin{tikzcd}
U \arrow[ddr, "\psi"'] \arrow[drr, "\varphi"] \arrow[dr, "{(x,y)}" description] & & \\
& X \times_Z Y \arrow[r, "p"'] \arrow[d, "q"] & X \arrow[d, "f"'] \\
& Y \arrow[r, "g"] & Z
\end{tikzcd}$$

XY-pic normally permits labels to be shifted towards the tip or towards the start of an arrow by prefixing the label with a ratio, such as e. g. `(.3)`. In LyX this works only for labels which are placed on top of the arrow, such as `\ar[r]|(0.3){\phi}`.

For labels placed to the left or to the right of the arrow this does not work. The corresponding XY-pic code such as e. g. `\ar[r]^(.3)\phi` or `\ar[r]_(.3)\psi` is not correctly interpreted by LyX's math editor. Two workarounds are suggested in the last section of this note.

3.4 Arrow modification

Modification of the design, the form or the positioning of arrows are introduced by the `@`-character. This is followed by a pair of matching brackets, where the form of the bracket pair, `{ }` or `< >` or `/ /` indicates, whether we want to modify the design, the or the curvature of the arrow. Various modifications can be applied to an arrow at the same time.

3.4.1 Arrow design

Various designs such as *solid*, *dotted*, *dashed* or *double* are possible for the shaft of an arrow. These can be combined with various ends and various tips. In general, the design of an arrow is described by following the command `\ar` immediately by an `@`-sign and a pair of braces $\{\dots\}$ containing characters describing the end, the shaft and the tip of the arrow. These characters are chosen to give some form of ASCII-rendering of the real thing. For instance `\ar@{>..>>}` produces an arrow with split end, a dotted shaft and double head. A number of other arrow designs is given in the table below. Note that the ends of embedding arrows $A \hookrightarrow B$ are described by raising or lowering opening parentheses, such as in `\ar@{^(->}[r]` .

Result	Source code in LyX
$\longrightarrow$	`\ar`
$- - - - - - \rightarrow$	`\ar@{-->}`
$\cdots\cdots\cdots\rightarrow$	`\ar@{..>}`
$\rightsquigarrow$	`\ar@{~>}`
$\longrightarrow\!\!\!\!\twoheadrightarrow$	`\ar@{->>}`
$- - - - - - \twoheadrightarrow$	`\ar@{-->>}`
$\rightarrowtail\!\!\!\longrightarrow\!\!\!\!\twoheadrightarrow$	`\ar@{>->>}`
$\lhook\joinrel\longrightarrow$	`\ar@{_(->}`
$\hookrightarrow$	`\ar@{^(->}`
$\vdash\!\!\!\longrightarrow\!\!\!\dashv$	`\ar@{|-|}`

Following the @-character by either a 2, 3, _ or a ^, we can produce arrows with double, triple shaft or arrows showing only the lower or upper half of their tips and ends. Arrows need not have tips nor ends, as the last example shows:

Result	Source code for LyX
$\Longrightarrow$	`\ar@2`
$\Rrightarrow$	`\ar@3`
$\rightharpoondown$	`\ar@_{->}`
$\rightharpoonup$	`\ar@^{->}`
$\rightharpoonup\!\!\!\longrightarrow\!\!\!\!\rightharpoonup$	`\ar@^{>>->>}`
$\leftharpoonup\!\!\!\longrightarrow\!\!\!\rightharpoondown$	`\ar@{^<-_>}`
$= = = = = =$	`\ar@2{--}`

3.4.2 Designing your own arrows

Within certain limits there is even a way to design your own arrows. Using some the characters `><|ox+/()[` one can even design one's own arrow tips using the `\newdir` command in the preamble. For explanations, we refer to the XY-manual, from which we take the example:

```
  \newdir{|>}{!/4.5pt/@{|}*:(1,-.2)@^{>}*:(1,+.2)@_{>}}.
```

This defines a new arrow tip, referred to as `|>` in `\ar@{-|>}[r]` and which displays correctly in LyX as:

$$A \longrightarrow\!\triangleright\, B$$

3.5 Arrow positioning

Arrows are shifted sideways with the modifier `@<`. . . `>` where the ellipsis is replaced by a positive or negative measure. For instance, to design a pair of mutually opposing arrows between two nodes, we shift them to see them apart. Note that the direction of the shift (positive) is to the left if one looks along the arrow. Thus

```
  \xymatrix{\circ \ar@<1ex>[r]& \circ \ar@<1ex>[l]}
```

produces

$$\begin{tikzcd} \circ \arrow[r, shift left] & \circ \arrow[l, shift left] \end{tikzcd}$$

3.5.1 Inline or centered diagrams

Arrows and diagrams can be used inline, such as this one: $\begin{tikzcd} \circ \arrow[r, shift left] & \circ \arrow[l, shift left] \end{tikzcd}$. When their code is written inside LyX as above, select it and choose either Ctrl-m for inline appearance or Ctrl-M for displaystyle. Diagrams constructed inline can later be centered, or, conversely, centered diagrams can be changed to inline formulas with Edit ▷ Math ▷ Change Formula Type.

3.6 Bending arrows

There are two simple methods to make arrows bend. The first is giving an explicit value by which the midpoint of the arrow's shaft is raised or depressed, the other is by forcing the arrow to leave its origin in a prescribed compass direction and to make him enter the target at another direction. The necessary bending of the arrow is determined automatically. We describe both methods.

3.6.1 Raising the shaft

For bending arrows we use the modifier @/. . ./ . The ellipsis stands for a TEX-measure which needs to be entered as a lower or upper index. Whereas in `xypic`, we could simply write, e. g. `\ar@/_1pc/` for an arrow bending `1pc` downwards, this cannot directly be done in LyX. It is necessary, to enclose the measure in a pair of braces, such as e. g. `\ar@/_{1pc}/`. As an example, here are two opposing arrows between A and B, each bending by .5 pica, given by the following source code:

```
  \xymatrix{A \ar@/_{.5pc}/[r] & B \ar@/_{.5pc}/[l]}
```

$$\begin{tikzcd} A \arrow[r, bend right] & B \arrow[l, bend right] \end{tikzcd}$$

3.6.2 Specifying exit- and entrance directions

An alternative for making arrows bend is by specifying their compass direction as they are leaving their source and their direction from which they enter their target. Instead of north, north-east, east, etc., the directions are named `u`, `ur`, `r`, `dr`, `d`, `dl`, `l`, `ul`, standing for up, up-right, right, down-right, etc.. A direction is specified as @(*out*,*in*) where *out* stands for the direction the first object is left and *in* stands for the direction from which the target is entered. As an example, we show some bending arrows and a loop, which arises when we do not specify a target for an arrow, just its incoming and outgoing direction:

$$\begin{tikzcd}
A \arrow[r, bend right] \arrow[rr, bend right] & B \arrow[dl, bend left] & C \\
E \arrow[loop above]
\end{tikzcd}$$

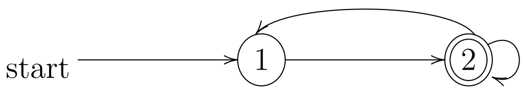
```
\xymatrix{A \ar@(dr,dl)[r]\ar@(dr,dl)[rr]
                  &B\ar@(d,r)[dl]  &C \\
           E \ar@(ul,ur)}
```

3.7 Modifying vertices

The above example is reminiscent of an automata diagram, except that in such a diagram states would be enclosed in small circles, with double circles denoting final states.

3.7.1 Framing objects

With XY-pic, entries can obtain a single or a double frame, such as $\boxed{A}$ or $\boxed{\boxed{B}}$ by prefixing an entry with `*[F-]` or `*[F=]` and enclosing the portion of the entry to be framed in braces. Normally, the frame will be very tight so that it must be widened by prefixing with `+` or with `++`. Round frames, such as $\textcircled{A}$ and $\textcircled{\textcircled{B}}$ are obtained by specifying the shape as `[o]`. So the latter figure was constructed as *++[o][F=]{B}. This way, the following automaton



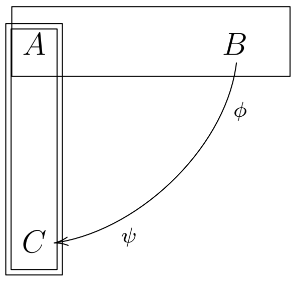
can be typeset as

```
\xymatrix{\txt{start}\ar[r]
           & *++[o][F]{1}\ar[r]
           & *++[o][F=]{2}\ar@(ur,dr)\ar@(ur,ul)[l]
          }.
```

The LaTeX command `\entrymodifiers={...}` will make a certain entry style the default, that can, of course be overridden for individual entries. Thus, after `\entrymodifiers={++[o][F-]`
all following entries inside XY-matrices would be encircled.

3.7.2 Framing rectangles

Framing a whole rectangle inside an xymatrix is done with the macro pair `\save ... \restore`. The dimension of the rectangle is given as a dotted pair $P_0.P_1$ of points denoting the top left and lower right corners of the rectangle. Each point, in turn,is given as a doubly quoted comma separated pair "x, y" specifying row x and column y. These are followed by the framing commands, to produce figures such as the following:



Here, the code `\save"1,1"."2,1"*+[F=]\frm{}\restore` produces the doubly framed rectangle, and the code for the extra wide horizontal rectangle is `\save"1,1"."1,2"*++[F]\frm{}\`
This code can be placed in arbitrary cells of the xymatrix.

4 Using LyX's math editor

As an alternative to writing the `xypic` code, then transforming it into a math-editing environment by marking it and applying Ctrl-m, or Ctrl-M, one may construct and modify the whole xypic-diagram inside LyX's math editor. We describe the editing steps for a figure just like the one above.

4.1 Caveat - how to enter braces

Recall that in LyX's math-editor any pair of braces `{` and `}` that are to enclose a macro-parameter must be entered by typing just `\{` . The closing brace is automatically supplied and in between a box into which to the parameter is entered. In connection with XY-diagrams, this applies in particular to arrow modifications that are normally given in the form `@{ ... }` with the ellipsis standing for the description of end, shaft and tip of the arrow. Inside the math-editor, enter just `@\{` and let LyX provide the closing brace and the box into which to enter the description of the arrow.

Braces that are entered without the backslash `\` will just appear as typed, but cannot be used to receive a macro parameter. They are useful, for instance to denote sets, e. g. `{x\in X \mid x\notin x}` will display as $\{x \in X \mid x \notin x\}$.

4.2 Setting up the matrix

With Ctrl-m or Ctrl-M open a formula environment and enter: `\xymatrix`. This produces a 1×1- XY-matrix. Add extra rows by typing Ctrl-Enter and add columns by typing Alt-m c i.

At any time, further rows or columns can be entered or deleted using commands available from Edit ▷ Math, resp. their shortcuts, beginning with Alt-m c for the column commands or Alt-m w for the row commands. A more direct way uses the icons in the math toolbar once it has been activated via View ▷ Toolbars.

4.3 Entering nodes, arrows and labels

Type the nodes into the correct positions of the matrix. If you move the cursor out of the matrix, you should see a first rendering of the node layout. Next, add the arrows at the nodes from where they should emanate by typing `\ar[`*p*`]`, where *p* can be any path made up from the characters `u`, `d`, `l`, `r`. Make sure that the path indeed leads to an existing node within the matrix. Otherwise, the figure will not display when the cursor leaves the editing area.

Next, label the arrows by attaching a label text as upper or lower indices to the end of the arrow's path. As always in LyX's math editor, an underscore `_` opens a box for a lower index and a `^` followed by a space opens a box for an upper index. You can enter any LaTeX-code as a label.

4.4 Modifying arrows

Finally, you can modify the appearance of the arrows by entering @-modifiers `@{...}`, `@<...>`, `@(...,...)` or `@/.../`. The above caveat applies to the first form only. It must be entered as `@\{` with the arrow description entered inside the LyX-supplied box. If this box remains empty, you have specified an empty arrow. This is a useful construction, too, as you will see in the next section.

The other modifiers, `@<...>`, `@(...,...)` and `@/.../` are typed as shown with the arrow description replacing the ellipsis. The code for bending arrows, which in `xypic` is `@/_`*measure*`/` or `@/^`*measure*`/` where *measure* is any valid TeX-measure that should be entered as upper or lower index to the first slash /. Make sure that the ending slash does not end up being part of the upper or lower index.

4.5 What if something goes wrong

When constructing a diagram, you should at times check it by just moving the cursor out of the editing area to see whether instant preview can successfully convert it into graphical output. If this does not happen, it may either be that instant preview for some reason is not aware that it should retranslate the graphics. Moving the cursor into the editing area and out again sometimes wakes up instant preview.

A more serious reason could be a syntactical error in your input (in that case, the math edition area disappears completely). If necessary, undo the last editing steps, using Ctrl+Z, or try to translate the LyX-file into DVI using Ctrl+D or View ▷ DVI. There should be some error generated, which hopefully gives you a hint as to the source of the mistake.

5 Hacks

Certain things do not work correctly inside LyX. The ones that we (used to) miss most are the horizontal and vertical scaling of diagrams, and the correct positioning of arrows. There are some workarounds that we are explaining here.

5.1 Horizontal and vertical scaling

It is often convenient to stretch the horizontal or the vertical dimensions of an entire diagram by using spacing commands for rows and/or columns. According to the XY-manual, [?], for instance, `\xymatrix@R=1pc{...}` defines an XY-matrix with row spacing of 1 pica. Similarly, `\xymatrix@C=...{...}` allows to modify the space between columns. Unfortunately, one cannot type "`\xymatrix@R=1pc`" into a formula to get the desired matrix because the @-character is interpreted by LyX as ending the `\xymatrix` command. But there is another way of getting the desired matrix:

XY stores the values for row-spacing and column-spacing in the variables `\xymatrixrowsep@` and `\xymatrixrowsep@`. So one can add the following macro to the preamble (Layout ▷ Document ▷ Preamble):

```
\newcommand{\xyR}[1]{%
\xydef@\xymatrixrowsep@{#1}}
```

A macro `\xyC` can be defined correspondingly by replacing `\xymatrixrowsep@` with `\xymatrixcolsep@`. Now, a figure can be scaled by entering \xyR{ into the formula before the XY-matrix. Place the cursor inside the matrix, just before the first entry. Then enter `\xyR\{` or `\xyC\{` or both. Don't forget the backslashes and remember, that the closing brace is automatically supplied by LyX. Inside the braces enter the dimensions.

Note: These commands affect all following XY-matrices. Therefore don't forget to reset the values to default after the usage of `\xyR` etc..

The default is 2pc. Here you see a diagram which is squashed vertically and stretched horizontally with `\xyR{0.5pc}\xyC{9pc}`:

$$\begin{tikzcd}[row sep=small, column sep=huge] A \arrow[r, "\text{horizontal stretch}"'] \arrow[d] & B \\ C & \end{tikzcd}$$

5.2 Label positioning

Another useful XY-command allows the correct positioning of labels along the shaft of arrows. This feature is sometimes necessary, when the default position of a label would otherwise clutter the picture, or would even coincide with other items, such as the intersection of the arrows in the figure below. In order to shift a label position alongside the shaft of an arrow, XY allows to prefix the label by a decimal number in parentheses, specifying the fraction of distance alongside the arrow where the label is to be placed. For instance, the code `\ar[r]|(0.3)\varphi`, will place the label on the shaft, but only about one third of the way.

Doing the same with labels above or below arrows as `\ar[dr]^(0.3)\varphi`, respectively `\ar[ur]_(0.3)\phi`, does not work from inside LyX. Instead, one has to replace the hat-symbol ^, resp. the underscore _ , by the macros `\sp`, resp. `\sb`, obtaining `\ar[dr]\sp(0.3)\varphi` and `\ar[ur]\sb(0.3)\phi`. It is, in fact, possible to enter several labels this way and those labels are placed correctly, even if the arrow bends. In the following figure, the bending arrow with its four labels has been produced with `\ar@(r,r)[d] \sp(0.2){\phi_{1}} \sp(0.4){\phi_{2}} \sb(0.6){\phi_{3}} \sp(0.8){\phi_{4}}`.

$$\begin{tikzcd}
\bullet \arrow[ddr, "\varphi" pos=0.3] & \circ \arrow[dd, bend left=90, "\phi_1" pos=0.2, "\phi_2" pos=0.4, "\phi_3"' pos=0.6, "\phi_4" pos=0.8] \\
& \vdots \\
\circ \arrow[uur, "\phi"' pos=0.3] & \bullet
\end{tikzcd}$$

5.3 Invisible stretched arrows

A more general trick uses invisible arrows to place any object almost anywhere inside a diagram. Produce an invisible arrow, shorten (or prolong) it past its goal by adding a decimal stretching ratio, e. g. `(0.6)` or `(1.4)` to its path. Attach a label to this invisible arrow.

Thus, the down pointing arrow with its label φ at (0.3) of its way along the shaft might as well have been produced by adding to the regular arrow `\ar[dr]` an invisible φ-labelled arrow `\ar@{}[dr(0.6)]^\varphi`, reaching only 0.6 of the way. Its label will now appear at 0.3 of the way of the original visible arrow.

This workaround has two minor drawbacks: First, it does not work with bending arrows. Secondly, prolonging an invisible arrow beyond the normal dimension of the figure will invisibly extend the figure box, and thereby cause too much vertical space between the figure and the preceding or the following paragraph.

Nevertheless, invisible arrows are an important tool, since they can, in principle, be used to place information at any chosen place in a diagram. In the above figure, for instance, we have used an invisible arrow to carry the `\vdots` as label and at the earlier figure we had used an invisible arrow to carry the text "`horizontal stretch`" into the center of the figure.

5.4 Further Xy-tricks

Here we have focused only on the `\xymatrix` command, which is just one of the features available in Kris Rose's amazing Xy-package. The Xy-pic manual [?] demonstrates many of the advanced possibilities of that package. Beware, that its style is rather terse and you will likely need a lot of experimenting and modifying the many worked examples. Lauda [?] explains how to use Xy-pic to make braids, cobordism, string diagrams, and much more. You might want to look on `inset_preview.lyx` example shipped with LyX, which demonstrates how to use more advanced XY-commands via `ERT` and `Instant preview` insets.

References

[1] Jason Gross, Adam Chlipala, David I. Spivak. “Experience Implementing a Performant Category-Theory Library in Coq”. In: Interactive Theorem Proving. Springer, 2014.

[2] Gregory Malecha, Adam Chlipala, Thomas Braibant. “Compositional Computational Reflection”. In: Interactive Theorem Proving. Springer, 2014.

[3] Kosta Dosen, Zoran Petric. “Proof-Theoretical Coherence”. http://www.mi.sanu.ac.rs/~kosta/coh.pdf , 2007.

[4] Francis Borceux, George Janelidze. “Handbook of categorical algebra. Volumes 1 2 3.”. Cambridge University Press, 2001.

[5] Francis Borceux. “Handbook of categorical algebra. Volumes 1 2 3.”. Cambridge University Press, 1994.