

# Using X<sub>Y</sub>-pic in L<sub>Y</sub>X

H. Peter Gumm

October 22, 2008

## Abstract

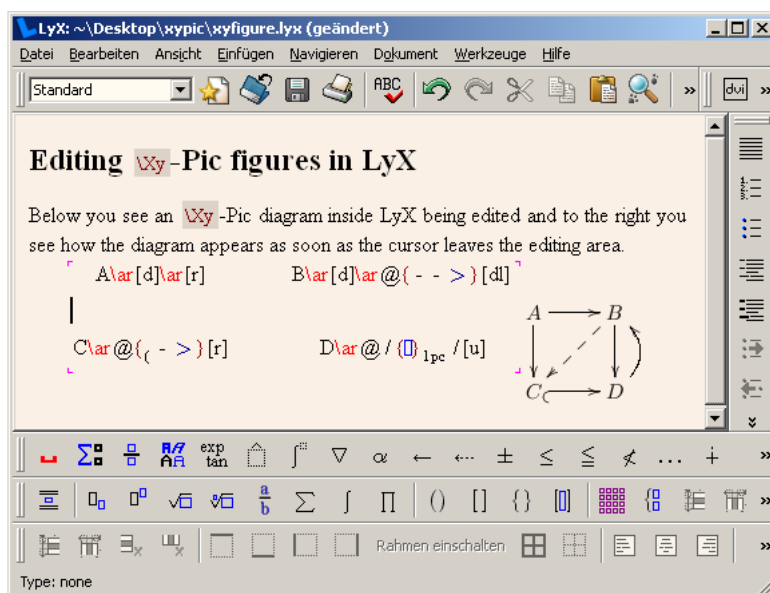
With the recent versions of L<sub>Y</sub>X and with the `preview`-style installed in the L<sup>A</sup>T<sub>E</sub>X-System, the graph drawing package X<sub>Y</sub>-Pic can be conveniently used inside L<sub>Y</sub>X. Diagrams can be edited and displayed inside the main L<sub>Y</sub>X editing window. Here, we shall describe how to use the `\xymatrix` command from `xypic` inside L<sub>Y</sub>X in order to draw, to edit and to preview diagrams as typically used in category theory, algebra, and related fields.

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Preparation</b>	<b>3</b>
<b>3</b>	<b>Commutative diagrams</b>	<b>3</b>
3.1	The matrix layout of diagrams . . . . .	4
3.2	Arrows . . . . .	4
3.3	Labels . . . . .	4
3.4	Arrow modification . . . . .	5
3.4.1	Arrow design . . . . .	5
3.4.2	Designing your own arrows . . . . .	6
3.5	Arrow positioning . . . . .	6
3.5.1	Inline or centered diagrams . . . . .	7
3.6	Bending arrows . . . . .	7
3.6.1	Raising the shaft . . . . .	7
3.6.2	Specifying exit- and entrance directions. . . . .	7
3.7	Modifying vertices . . . . .	8
3.7.1	Framing objects . . . . .	8
3.7.2	Framing rectangles . . . . .	8
<b>4</b>	<b>Using L<sub>Y</sub>X's math editor</b>	<b>9</b>
4.1	Caveat - how to enter braces . . . . .	9
4.2	Setting up the matrix . . . . .	9
4.3	Entering nodes, arrows and labels . . . . .	9
4.4	Modifying arrows . . . . .	9
4.5	What if something goes wrong . . . . .	10
<b>5</b>	<b>Hacks</b>	<b>10</b>
5.1	Horizontal and vertical scaling . . . . .	10
5.2	Label positioning . . . . .	11
5.3	Invisible stretched arrows . . . . .	11

# 1 Introduction

The `xypic`-package has long served as a convenient tool for easily constructing graphs and diagrams in  $\text{\LaTeX}$ . Unfortunately, its use in  $\text{LyX}$  had long been restricted to the infamous ERT-boxes, meaning that the  $\text{LyX}$  editor could only display the  $\text{\LaTeX}$ -source and not the finished diagram. The new `preview`-style of  $\text{\LaTeX}$  which is part of the `AUCTeX` project[4], finally enables the editing and displaying of `xypic`-diagrams, constructed, displayed and interactively edited inside  $\text{LyX}$ .



In this note, we describe how  $\text{Xy-pic}$  can be used from inside  $\text{LyX}$ , how diagrams can be created and edited. We have tested the following using  $\text{LyX}$  versions 1.3.7 up to 1.6, running under WindowsXP 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  $\text{Xy-Pic}$  code inside the  $\text{LyX}$ -window, select it all and convert it to a graphical representation by pressing `Ctrl-m` or `Ctrl-M`. If you use  $\text{Xy-Pic}$  more frequently, or if you want to modify your initial figure, you will want to assemble and modify your figures using  $\text{LyX}$ '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  $\text{LyX}$  in the first mentioned mode and we introduce all  $\text{Xy-Pic}$  features that might be of use for drawing commutative diagrams, graphs or automata. Section 3 explains how to use the  $\text{Xy-Pic}$  commands inside a math-editing area.

It is not our intention to write another introduction to  $\text{Xy-Pic}$ , rather our motivation is to give an introduction how the most important commands work inside  $\text{LyX}$ , since the keystrokes as explained in the  $\text{Xy-Pic}$  manual[1] will not always function correctly inside  $\text{LyX}$ .

## 2 Preparation

The following requires that the  $\text{\LaTeX}$ -packages `xypic` and `preview` are installed in the  $\text{\LaTeX}$  system. They are available from CTAN, see at [2], resp. at [3]. After freshly installing them, it may be necessary, to run `Edit`  $\triangleright$  `Reconfigure` from the main  $\text{\LaTeX}$  menu. The steps to a first diagram output in  $\text{\LaTeX}$  are then:

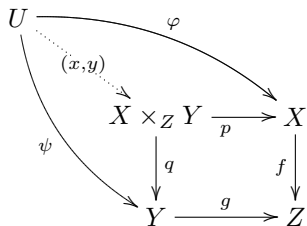
1. Activate and test `preview`
  - (a) Open  $\text{\LaTeX}$ , choose `Edit`  $\triangleright$  `Preferences`  $\triangleright$  `Look and Feel`  $\triangleright$  `Graphics` and place a check-mark at *Instant Preview*.
  - (b) Test, if `instant-preview` works by opening a  $\text{\LaTeX}$ -document and entering any math-formula, e.g.  $a + b = c$ .
  - (c) 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 `xypic`
  - (a) Choose `Layout`  $\triangleright$  `Document`  $\triangleright$  `Preamble` and type `\usepackage[all]{xy}`.
  - (b) Inside your  $\text{\LaTeX}$ -Document, enter the text `\xymatrix{A \ar[r] & B}`.
  - (c) Select the whole text and choose `Insert`  $\triangleright$  `Math`  $\triangleright$  `Display Formula`, or use the corresponding keyboard shortcut `Ctrl-M`.
  - (d) 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`[1] 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[r]_p & X \ar[d]_f \\
  & \downarrow q & & \downarrow 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.



### 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  $3 \times 3$ -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 }

```

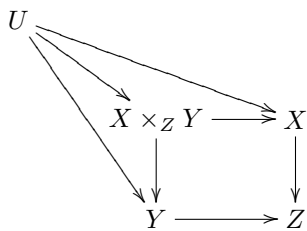
The pattern should be familiar from L<sup>A</sup>T<sub>E</sub>X: We see three rows, the first two being terminated by the end-of-line-marker `\\`. Each line consists of entries, separated by the ampersand `&`.

### 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[dr]`. Thus the above diagram with all arrows added becomes:

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

```



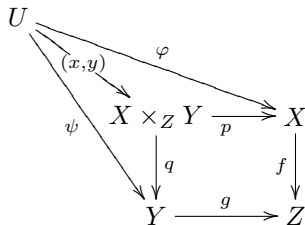
### 3.3 Labels

Labels are attached to arrows by affixing them as upper or lower indices to the `\ar`-command. Thus, `\ar[ddr]^{\varphi}` defines an arrow going one cell down, two to the right and having the label  $\varphi$  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

$\alpha$  below and label  $\beta$  above the arrow, i.e.  $\xleftarrow[\alpha]{\beta}$ . 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[dr]^{(x,y)} \ar[dr]_{\varphi} & & \\
  & X \times_Z Y \ar[r]_p & X \\
  & \downarrow q & \downarrow f \\
  & Y \ar[r]_g & Z
```



$\Xy-pic$  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  $\text{LyX}$  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  $\text{LyX}$ 's math editor. Two workarounds are suggested in the last section of this note.



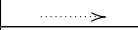
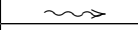
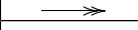
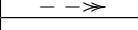
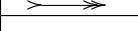
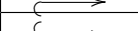
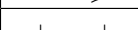
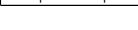
### 3.4 Arrow modification

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.

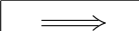
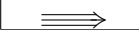
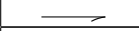
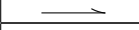
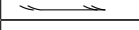
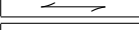

#### 3.4.1 Arrow design

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 `{...}` 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  $A \overset{\curvearrowright}{\longrightarrow} B$  are described by raising or lowering opening parentheses,

such as in `\ar@{^(->)}[r]`.

Result	Source code in LyX
	<code>\ar</code>
	<code>\ar@{--&gt;}</code>
	<code>\ar@{..&gt;}</code>
	<code>\ar@{~&gt;}</code>
	<code>\ar@{-&gt;&gt;}</code>
	<code>\ar@{--&gt;&gt;}</code>
	<code>\ar@{&gt;-&gt;&gt;}</code>
	<code>\ar@{_(-&gt;)}</code>
	<code>\ar@{^(-&gt;)}</code>
	<code>\ar@{ - }</code>

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
	<code>\ar@2</code>
	<code>\ar@3</code>
	<code>\ar@_{-&gt;}</code>
	<code>\ar@^{-&gt;}</code>
	<code>\ar@^{&gt;-&gt;&gt;}</code>
	<code>\ar@{^&lt;-_-&gt;}</code>
	<code>\ar@2{--}</code>

### 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 ~~X~~ 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:

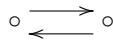
$$A \text{ --- } \triangleright B$$

## 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



### 3.5.1 Inline or centered diagrams

Arrows and diagrams can be used inline, such as this one:  $\circ \rightleftarrows \circ$ . When their code is written inside L<sub>Y</sub>X 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 > Alignment.

## 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

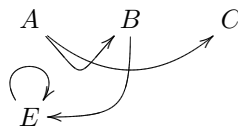
For bending arrows we use the modifier @/.../. The ellipsis stands for a T<sub>E</sub>X-measure which needs to be entered as a lower or upper index. Whereas in xypic, we could simply write, e.g. \ar@/\_1pc/ for an arrow bending 1pc downwards, this cannot directly be done in L<sub>Y</sub>X. It is necessary, 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]}
```



### 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:



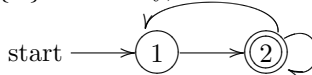
```
\xymatrix{A \ar@(dr,dl)[r]\ar@(dr,dl)[rr] \\ & & B \ar@(d,r)[dl] & C \\ E \ar@(ul,ur)}
```

### 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  $\text{X}\text{Y}$ -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



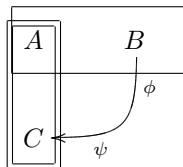
can be typeset as

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

The  $\text{L}\text{A}\text{T}\text{E}\text{X}$  command  $\text{\entrymodifiers}=\{\dots\}$  will make a certain entry style the default, that can, of course be overridden for individual entries. Thus, after  $\text{\entrymodifiers}=\{++[o][F-]\}$ , all following entries inside  $\text{X}\text{Y}$ -matrices would be encircled.

#### 3.7.2 Framing rectangles

Framing a whole rectangle inside an  $\text{xymatrix}$  is done with the macro pair  $\text{\save} \dots \text{\restore}$ . 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  $\text{\save"1,1"."2,1"}*++[F=]\text{\frm{}}\text{\restore}$  produces the doubly framed rectangle, and the code for the extra wide horizontal rectangle is  $\text{\save"1,1"."1,2"}*++[F]\text{\frm{}}\text{\restore}$ . This code can be placed in arbitrary cells of the  $\text{xymatrix}$ .

## 4 Using L<sup>A</sup>T<sub>E</sub>X'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 L<sup>A</sup>T<sub>E</sub>X'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 L<sup>A</sup>T<sub>E</sub>X'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 L<sup>A</sup>T<sub>E</sub>X 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  $1 \times 1$ -`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.

### 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 L<sup>A</sup>T<sub>E</sub>X'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 L<sup>A</sup>T<sub>E</sub>X-code as a label.

### 4.4 Modifying arrows

Finally, 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 L<sup>A</sup>T<sub>E</sub>X-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, 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. If necessary, redo the last editing steps, using `Ctrl-z`, or try to translate the `LATEX`-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 `LATEX`. 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

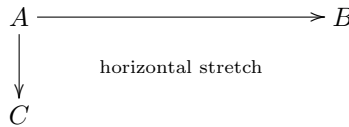
It is often convenient to stretch the horizontal or the vertical dimensions of an entire diagram by using spacing commands for rows and/or columns. According to the `XY`-manual, for instance, `\xymatrix@R=1pc{...}` defines an `XY`-matrix with row spacing of 1 pica. Similarly, `\xymatrix@C=...{...}` allows to modify the space between columns. Unfortunately, these commands do currently not work inside `LATEX`, as the `@`-character is interpreted by `LATEX` as ending the `XY`-matrix-macro.

Knowing that `XY` stores the values for row-spacing and column-spacing in the variables `\xymatrixrowsep@` and `\xymatrixcolsep@`, add the following macro to the preamble (`Layout▷Document▷Preamble`)

```
\makeatletter
\newcommand{\xyR}[1]{%
\makeatletter
\xydef@\xymatrixrowsep@{#1}
\makeatother
} % end of \xyR
```

A macro `\xyC` can be defined correspondingly by replacing `\xymatrixrowsep@` with `\xymatrixcolsep@`. Now, a figure can be scaled by entering `\xyR{...}` into 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

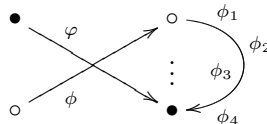
remember, that the closing brace is automatically supplied by LyX. Inside the braces enter the dimensions. The default is `2pt`. Here you see a diagram which is squashed vertically and stretched horizontally with `\xyR{9pc}\xyC{.5pc}`:



## 5.2 Label positioning

Another useful X<sub>Y</sub>-vcommand 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, X<sub>Y</sub> 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  $\hat{\phantom{x}}$ , 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}`.



## 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  $\varphi$  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  $\varphi$ -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.

## References

- [1] Kristoffer H. Rose: *X<sub>Y</sub>-Pic User’s Guide*. Version 3.7, Feb. 16, 1999. Available as part of the `xypic` L<sup>A</sup>T<sub>E</sub>X package.
- [2] <http://www.tug.org/tex-archive/help/Catalogue/entries/preview-latex.html>
- [3] <http://www.tug.org/tex-archive/help/Catalogue/entries/xypic.html>
- [4] <http://www.gnu.org/software/auctex/>