XY-pic and Notation for Categorical Diagrams

K. H. Rose
DIKU, University of Copenhagen[*]

October 20, 1994

Abstract. This paper discusses textual notation for categorical diagrams based on my experience from developing the XY-pic package [13, 14] for the TeX typesetting system. First I give a short survey of some different such notations that have been used by drawing packages for TeX, categorising each as *positional*, *visual*, or *conceptual*. The last is new so I discuss the conceptual XY-pic 'graph' notation in some detail next. Finally I discuss how *orthogonal* combination of different notations makes very powerful yet intuitive notation for diagrams possible, and explain how this is realised by XY-pic such that the code below (to the left) will typeset a pullback diagram (shown to the right):

```
\xy\xygraph{
 !M{ X \times_Z Y \="xy" \:[r]_p \:[d]^q
                   & X \="X" \:[d]_f \\
     Y \="Y" \:[r]^g & Z }
 [ul]U   ( ? :@/^1ex/ ^x        "X" ,
           ? :@{-->}   |-{(x,y)} "xy" ,
           ? :@/_1ex/ _y        "Y" )
}\endxy
```

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

Note: This paper is written for the category theory community by a theoretical computer scientist. Therefore I have avoided colloquial 'we' in favor of personal 'you' and 'I'.

Key words: Category theory, diagrams, typesetting, TeX, LaTeX.

1 Introduction

In the last decade the typesetting programming systems TeX [8] has made quality typesetting available in practice to mathematical authors. The reason is a combination of two simple facts:

– TeX takes its input as plain text files with *markup* much in the same way as authors are used to type manuscripts, and

– TeX runs on all kinds of computers from the smallest personal ones to large mainframes.

The markup used for mathematics by TeX[1] has been an important factor in this success – in fact, TeX conventions serve as a generic language for electronic communication of mathematics (as plain text), *e.g.*, when authors of joint papers discuss

[*] E-mail: kris@diku.dk. Address: Universitetsparken 1, 2100 København Ø, Denmark.

[1] Inspired by the notation of the EQN 'preprocessor' [6] of the TROFF system [10].

details using electronic mail or TALK programs[2]. On the other hand one might very well argue that TEX is to blame for the lack of a truly standard WYSIWYG[3] math interchange format.

The crucial decision when designing markup notation is the distinction between what is 'contents' and what is 'markup'. For plain text this seems relatively easy: the actual words and punctuation constitute the contents and other aspects like emphasizing is markup. The distinction is not always clear, however: Consider quotation marks – in a manuscript the plain `"` character in text sometimes means “ and sometimes ”. The human typesetter rarely doubts which one is intended but in TEX, for example, the author has to type `` ` ` `` or `''` explicitly, in fact the `"` character is not used at all.

For mathematics the situations is complicated by the fact that a textual description of a formula contains more characters than the actual typeset version because mathematicians frequently use the same characters for many different purposes: aspects traditionally considered markup are significant, *e.g.*, X is usually *not* the same as $\mathcal{X}$. A good example in TEX is the use of `\left[` and `\right]` to denote brackets that match each other and thus should have the same size and preferably one sufficiently large to embrace the material in between (this particular redundancy is necessary because some authors write things like $[0;1[$ for the downwards closed interval from 0 to 1). I will call such markup for *logical markup*. Sometimes it is even worse, namely when the desired typesetting is independent of or even violates the logical structure. I will call such markup *physical markup*. The distinction between these is often a matter of taste.

For graphics the distinction between contents and markup is made even more difficult by the non-linear nature of pictures (that is what they are for, after all). This is the subject of this paper: In section 2 I will discuss three approaches to markup for graphics: the 'positional', the 'visual', and the 'conceptual'. Then the approach taken by XY-pic to conceptual notation is discussed in detail in section 3, and finally section 4 discusses how 'orthogonal' combination of different notations is used in XY-pictures to produce an intuitive as well as general notation for diagrams.

2 Markup principles

I will discuss three markup principles used by todays diagram drawing notations (including XY-pic): the 'positional', the 'visual', and the 'conceptual'. I will use the pullback diagram from the abstract to asses the notations; all the typeset pullbacks are shown in the appendix for comparison.

2.1 The 'positional' markup principle

This principle is based on the idea that *a diagram is described by describing each glob of ink used to build it and where it is located in the picture.* The abstraction

[2] TALK is one of several programs for direct interactive communication between two users.

[3] Abbreviates 'What You See Is What You Get'.

is simple but also very important: each picture element is specified using two parameters: its *location* and *shape*. It is used by both the rather simple LaTeX [9] 'picture environment' and the quite powerful PICTEX [17] drawing macros: they differ mainly in the available 'library shapes' and configurability of the coordinate system.

The general nature of positional notations make them quite capable of expressing categorical diagrams, *e.g.*, the code

```
\setlength\unitlength{0.3pc}
\newcommand{\Ob}[1]{\makebox(0,0){$#1$}}
\newcommand{\Lab}[2]{\makebox(0,0)[#1]{$\scriptstyle#2$}}
\begin{picture}(32,22)(-1,-1)
 \put(0,20){\Ob{U}}
  \bezier{222}(2,19.5)(17,19)(28,12)\put(28,12){\vector(3,-2){0}}
                                    \put(17,19){\Lab{l}{x}}
  \put(2,18.5){\vector(4,-3){8.5}}
  \bezier{222}(1,18)(5,6)(13,1)\put(13,1){\vector(3,-2){0}}
                                    \put(5,7){\Lab{r}{y}}
 \put(15,10){\Ob{X \times_Z Y}}
  \put(21,10){\vector(1,0){7}}      \put(22.5,10){\Lab{t}{p\mathstrut}}
  \put(15,8){\vector(0,-1){6}}      \put(15,5){\Lab{l}{\,q}}
 \put(30,10){\Ob{X}}
  \put(30,8){\vector(0,-1){6}}      \put(30,5){\Lab{r}{f\,}}
 \put(15,0){\Ob{Y}}
  \put(17,0){\vector(1,0){11}}      \put(22.5,0){\Lab{b}{g}}
 \put(30,0){\Ob{Z}}
\end{picture}
```

typesets the pullback diagram using a LaTeX picture environment (result on p.11). It was not difficult to compose this code, it just took time drawing it on graph paper and then typing it in (and the dashed line broken with (x, y) was simplified to a plain arrow). The most trouble was when I realised that the width of the central entry meant that the columns should be at x-coordinate 0, $1\frac{1}{2}$, and 3, rather than 0, 1, 2, and had to change all the individual coordinates. This demonstrates that it is not writing diagrams with this kind of notation that is difficult: it is changing and maintaining them. However, the importance of the abstraction should not be underestimated: All diagram packages that I know of have inherited the separation of location and object in some form or other.

Many other macro packages with positional diagram notation exist[4]. However, two interesting variants should be mentioned: the 'diagmac' macros by John Reynolds [11] adds a notion of *state* to the principle such that it is particularly easy to specify locations relative to the most recently typeset objects as well as to previously typeset objects (by name). The 'diagram' macros by Michael Barr [2] add a very large collection of preformatted 'elementary' diagrams (we discuss this further in 2.3 below).

---

[4] In fact even the TeXbook has a small 'dirty trick' macro `\point` [8, p.389] that allows objects to be placed at arbitrary locations.

The predecessor to the XY-pic macros [5] was build on top of PICTEX and thus based on the positional principle but had a 'library' of objects especially targeted for commutative diagrams, notably labeled arrows. The principle is still present in XY-pic version 3 as the 'kernel positioning language' which can be used to add arbitrary drawn material to diagrams at any point, manipulating a graphic state similar to that of Reynolds.

2.2 The 'visual' markup principle

A different approach is to conceive of a diagram by its *layout*, *i.e.*, by restricting the objects in the diagram to occur in a particular pattern. This *visual markup principle* is also the basis of several notations. The first was the `\CD` command of AMS-TEX by Spivak [15, first version in '82] but that is not sufficiently powerful to be discussed here.[5] By the way, all the notations I know of use a rectangular pattern.

Visual markup is used very succesfully by the 'Diagram' macros of Borceux [3, first version in '88] in which a diagram is expressed as a matrix where each object in the diagram is centered in one of the entries. This saves the user from specifying the coordinates of the objects absolutely because they are determined by their relative position in the matrix grid. For example,

```
\renewcommand{\S}{\scriptstyle}
\begin{diagram}[66]
U &&
  & \sedotar     & \Esear{\S x} &&
  & \sseaR{\S y} & X\times_Z Y  &\Ear{\S p} & X          &&
  &              & \saR  {\S q} &           & \Sar{\S f} &&
  &              & Y            &\Ear{\S g} & Z          &&
\end{diagram}
```

typesets our pullback example (although without the break and curving arrows as can be seen on p.11). Notice how each particular object type has a special name. The fact that labels are not placed in the matrix structure but rather relative to arrows means that they need special notation: as can be seen in the example, Borceux uses capitalisation of the first/last letter of commands to indicate that they take a super-/subscript label argument.

Taylor's 'diagrams' macros [16] use basically the same notation except that a more natural markup is used for labels (and many more options are available): our pullback example is typeset by

```
\newarrow{Dashto}....>
\begin{diagram}[height=2em,width=3em,tight,labelstyle=\scriptstyle]
U &                             &              &        &        \\
  & \rdTo(4,2)^x \rdDashto~{(x,y)} \rdTo(2,4)_y &&      &        \\
  &                             & X\times_Z Y  & \rTo_p & X      \\
  &                             & \dTo>q       &        & \dTo<f \\
```

[5] Exercise 18.46 in the TEXbook [8, p.182] also demonstrates the principle by using the plain TEX `\matrix` macro to typeset a simple commutative diagram.

```
  &                             & Y                & \rTo^g & Z       \\
\end{diagram}
```

(still without the curving arrows as reproduced on p.11).

In this authors opinion the above notations suffer from demanding to much manual alignment of the entries with arrows in them – even simple diagrams require many entries, and even though most are empty this is error prone. For example, the cube diagram in [3] uses a 7×7 matrix. The reason the number of entries explodes is that the abstraction does not exploit that arrows are always *between* two other entries.

For this reason XY-pic version 2 [12] introduced a different variant of matrix-based 'visual' markup where '(simple) objects' and 'connections' are distinct: the latter kind is always associated directly to a 'source' object and explicitly specify the 'target' object – this requires no more on the user's part because this information is essentially identical to the 'direction' information required for arrows in the traditional matrix setting without the distinction.

Using the XY-pic 'matrix' feature (*aka* 'v2 mode') the diagram is entered as follows (the result can be seen on p.11):

```
\xy\xymatrix{
 U \arrow @/^.5pc/ [drr] ^x
   \arrow @{-->} [dr] |-{(x,y)}
   \arrow @/_.5pc/ [ddr] _y                                \\
   & X \times_Z Y \arrow[d]^q \arrow[r]_p & X \arrow[d]_f  \\
   & Y \arrow[r]^g                        & Z              }\endxy
```

It has these components:

Contents: The 'ordinary math', *i.e.*, text with TEX math markup.

Logical markup: The `&` and `\\` delimiting the entries and rows, the type of the arrows (`@{-->}` for dashed and nothing for ordinary), the targets of the arrows, and the labels in either 'super' (`^`), 'sub' (`_`), or 'break' (`|`) position.

Physical markup: The 'curvature' of the curving arrows (`@/^.5pc/` for $\frac{1}{2}$pc towards the 'super' direction and similarly for `@/_.5pc/` towards the 'sub' direction), and the `-` in front of the (x, y) break to indicate placing it on the middle of the arrow rather than the default middle between the centers of the entries at the ends.

However, my experience with XY-pic user queries indicates that concrete implementations based on the 'visual' principle should still allow local 'doodling' in the form of excursions using the positional principle because quite often diagrams contain somehting that does not fit the matrix structure.

I return to how XY-pic achieves this in section 4.

2.3 The 'conceptual' markup principle

The 'visual' principle represents the first step towards supporting 'intuitive' notation for diagrams in the sense that fewer concrete considerations are required when typesetting a diagram. The most intuitive notation, however, would be one based on the inherent structure of the diagrams that are being depicted. I will call this the *conceptual markup principle*.

The 'diagram' macros[6] by Barr [2] approach commutative diagrams in this way by defining a number of 'special shapes' and a method of 'pasting' them together. While the pasting method is still positional in our terminology, this means that the input can be structured much like the diagram is conceptualised. Our pullback example is entered like this:

```
\renewcommand{\S}{\scriptstyle}
\bfig
\putsquare(300,0)[X \times_Z Y`X`Y`Z;\S p`\S q`\S f`\S g]
\putmorphism(-120,840)(3,-1)[```\S x]{1060}1a
\putmorphism(0,800)(1,-1)[U``]{300}1a
\putmorphism(0,800)(1,-3)[```\S y]{266}1a
\efig
```

As can be seen this diagram is much more compact and quite easy to edit though it still suffers from the problems discussed in 2.1, and the notation is restricted to commutative diagrams (even quite limited ones: the labels can not be adjusted and there are no breaks or curved lines as can be seen on the output on p.12).

However, until recently no notation more conceptual than this existed for typesetting categorical diagrams with TEX.[7] XY-pic has an experimental one which is described in the next section.

3 The XY-pic 'graph' notation

This spring I[8] have researched into how conceptual markup can be separated from the other two kinds such that they can (then) be cleanly combined as discussed in the next section. The remainder of this section discusses a particular result of this research, implemented in XY-pic.

The insight that made what follows possible may seem somewhat provocative to you but is not really surprising from a computer scientists point of view. It is *don't ask the experts – ask the teachers.*[9] This may seem weird but let us return

[6] If the reader is confused by the similarity of the *names* of the discussed macros then this is understandable. But they are different: 'diagram', 'diagrams', and 'Diagram' are all distinguishable by TEX – at least on computers where case is significant...

[7] For the TROFF system [10] the PIC [7] language has some conceptual capabilities and PostScript [1] also provides some but in both cases extensive programming is required.

[8] In collaboration with Ross Moore while visiting Macquarie University, Sydney, on a combined ARC/MURG grant.

[9] The sometimes rather heated discussion on the categories electronic mailing list seems to support this thesis ☺

to our canonical example: any category theorist I've asked has said 'but it is just a pullback!'. To learn how a pullback should be conceptualised I've had to observe *how* it is drawn when introduced to newcomers. The following is an example of a sequence of drawings introducing pullbacks; I have added the equivalent XY-pic 'graph' conceptual markup to the right of each drawing:

" Let $\mathcal{C}$ be a category and let

$$\begin{tikzcd} & X \arrow[d, "f"'] \\ Y \arrow[r, "g"] & Z \end{tikzcd}$$

```
\xy\xygraph{
 []Z ( [u]X :_f ? , [l]Y :^g ? )
}\endxy
```

be a given diagram of objects and morphisms in $\mathcal{C}$. Construct a category $\mathcal{K}$ as follows: for the objects of $\mathcal{K}$ take the commutativity squares $[X, Y, Z; V]$ of the form

$$\begin{tikzcd} V \arrow[r, "p"'] \arrow[d, "q"] & X \arrow[d, "f"'] \\ Y \arrow[r, "g"] & Z \end{tikzcd}$$

```
\xy\xygraph{
 []Z    ( [u]X :_f ? , [l]Y :^g ? )
 [ul]V ( ? :_p "X" , ? :^q "Y" )
}\endxy
```

and for the set of morphisms from $[X, Y, Z; V]$ to $[X, Y, Z; U]$ take the morphisms $v : V \to U$ of $\mathcal{C}$ such that

$$\begin{tikzcd} U \arrow[dr, "v"] \arrow[drr, "x"] \arrow[ddr, "y"'] & & \\ & V \arrow[r, "p"'] \arrow[d, "q"] & X \arrow[d, "f"'] \\ & Y \arrow[r, "g"] & Z \end{tikzcd}$$

```
\xy\xygraph{
 []Z    ( [u]X :_f ? , [l]Y :^g ? )
 [ul]V ( ? :_p "X" , ? :^q "Y" )
 [ul]U ( ? :^x "X" , ? :_y "Y" ,
          ? :|v "V" )
}\endxy
```

is commutative. A terminal object in $\mathcal{K}$ is called a *pullback* for f, g. "

This is implemented by the XY-pic 'graph' (*aka* 'native v3') feature. With it we can typeset our generic pullback as

```
\xy\xygraph{!{0;<5em,0pc>:<0pc,4em>::0}
 []   Z                   ( [u]X :_f ? , [l]Y :^g ? )
 [ul] {X \times_Z Y}="xy" ( ? :_p "X" , ? :^q "Y" )
 [ul] U  ( ? :@/^.5pc/ ^x "X" , ? :@/_.5pc/ _y "Y" ,
           ? :@{-->}   |-{(x,y)} "xy" )              }\endxy
```

using the following conventions:

Contents:

– Letters, `X`, `Y`, etc., stand for *node contents*, *i.e.*, themselves in math mode: X, Y, etc. Similarly for formulæ in braces `{`...`}`.

– A colon, `:`, denotes an *arrow* from one node to the next. It can have *labels* indicated by the usual TEX sub-/superscript notation (using `_` and `^`, respectively) as well as 'breaks' (indicated by a `|` character).

Logical markup:

– Bracketing with `[]` is used for *relative positioning* where the letters `dulr` denote down, up, left, and right, respectively.

– A previously typeset node is referred to by enclosing it in `"`...`"`, *e.g.*, `"X"` refers to the X node. Complex nodes can be given an 'alias' using `=` as it has been done for the `"xy"` node in the pullback.

– Parenthesis are used for *mapping*: each element of the list in `()` is build separately with `?` denoting the current node before the opening `(`. This crucial construction is used three times in the example corresponding to the three times where the diagram is extended with arrows to or from a particular new node.

– Dashed arrows are indicated by the notation `@{-->}` (solid arrows are the default).

Physical markup:

– Arrow curvature is indicated by `@/^`...`/` and `@/_`...`/` markup.

– The dimensions of the grid used for the relative positioning is indicated by the (rather cryptic) 'coordinate setup' `!`... at the beginning.

The notation requires that all nodes are given as a positioning (with `[]` just maintaining the current position) followed by the node contents. A graph is then simply a sequence of nodes and lists.

4 Orthogonality

It is an ordinary requirement in programming language design that the individual features of the language are independent in such a way that as many combinations as possible make sense. By analogy to vector spaces this is called *orthogonality* of the features. In this section we discuss XY-pic as an example of how the notation principles discussed in the previous sections can be combined thus.

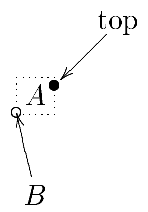
4.1 Embedding positional in visual and conceptual notation

The more ‘high-level’ notations like visual and conceptual ones draw their strength from the fact that they are based on some abstraction which restricts the possibilities and thus allows a more compact notation. This means, however, that sometimes ‘doodling’ in a diagram to add little remarks, emphasis, etc., is not easy because such material does not obey the restrictions.

XY-pic solves this problem by *embedding* the positional kernel language [14, part I] into both the visual ‘matrix’ notation and conceptual ‘graph’ notation (and others, *cf.* [14, part III]):

– In an XY-pic ‘matrix’ all targets may in turn be complete positional drawing sequences enclosed in `{}`s and a special `\save` ... `\restore` construction allows drawing even where a target is not expected.

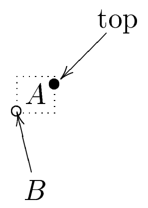
  For example, the diagram below to the left was typeset by the code to the right which uses two ‘escapes’ to change the target of the arrow and decorate the top entry:




```
\xy\xymatrix{
  A \save *\frm{..}!R!U(.5)*{\bullet}="*"
          +<2pc>*\hbox{top}
        \arrow "*" \restore
\\
  B \arrow [u]+DL*{\circ}+0
}\endxy
```

  The first escape adds a frame to A and a ‘handle’ called `"*"` to a bullet on the right edge; the word ‘top’ is then typeset at an appropriate distance and an arrow from it to the handle is added.

– Similarly: in an XY-pic ‘graph’ a kernel position can be used instead of a node using the `!` escape – here is the same figure as above:




```
\xy\xygraph{
  []A !{ *\frm{..}!R!U(.5)*{\bullet}="*"
         +<2pc>*\hbox{top} \arrow "*" }
  "A"[d]B : !{"A"+DL*{\circ}+0}
}\endxy
```

  A variant of this kind of escape was used to modify the graph grid earlier: it used that the graph grid simply uses the kernel reference coordinate system.

4.2 Embedding visual in conceptual notation

In the same way it can be useful to include 'visual elements' in a conceptually specified picture. You might, for example, think of the pullback example as a commuting square and a product, *i.e.*, *not* wish to include the conceptual structure of the square. This is accomplished in the XY-pic graph mode by allowing a special 'matrix escape' introduced by '`!M`' with the following properties:

- Within the matrix everything is specified as the matrix mode requires with the addition of `\=` and `\:` which function like the graph `=` and `:` markup.
- The entire matrix is a node in the graph with center as the top left entry.
- The grid of the graph is changed such that the relative moves `[r]` and `[d]` from the top left entry move on top of the appropriate neigbour entries.

The XY-pic graph feature (including this embedding of matrix markup) was used to typeset the pullback in the abstract.

5 Conclusion

This concludes the overview and categorisation of notations for categorical diagrams (that I know) with emphasis on XY-pic (that I must admit to know best). I hope that this will start a lively and fruitful discussion of how graphs and diagrams in general, categorical ones in particular, should be specified such that they are easy to maintain and communicate; in particular I am interested in feedback on the notion of *conceptual markup*. The talk given at ECCT-94 on this subject presents some additional issues; it is available on the World Wide Web with URL: `http://www.diku.dk/users/kris/ECCT-94/talk.html`.

Acknowledgements First of all I wish to thank the ECCT for inviting me and thus giving me the opportunity to present this material. Second, a huge thanks goes to Ross Moore for his work on XY-pic extensions without which the scope of XY-pic would be much narrower, and for numerous suggestions and criticisms that have helped in improving the consistency of XY-pic.

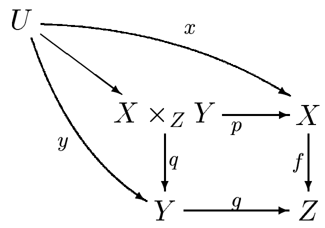
Appendix

A All Diagrams

The exact results of typesetting all diagrams from section 2 follow.

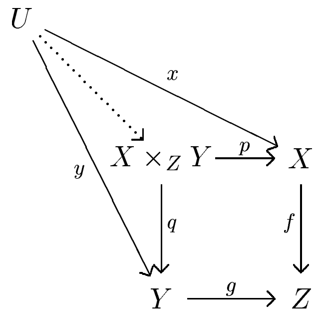
Note for typesetters: these should be typeset literally, *not* beautified!

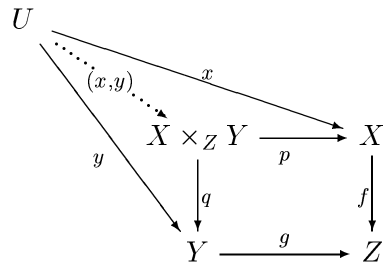
LATEX 'picture environment' pullback, code on p.3:



Typeset using LATEX [9] (using the 'bezier' package [4, §10.2.1]).

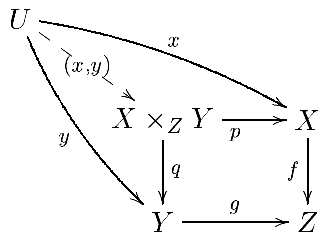
Borceux 'Diagram' pullback, code on p.4, and Taylor 'diagrams' pullback, code on p.5:





The left typeset using 'Diagram 3', the code based on manual example [3, p.6] but using `&` as the delimiter character; the right by 'diagrams' version 3.29, the code is slightly edited version of manual example [16, p.14]. Taylor's macros can, in fact, typeset true dashed lines but only when producing output for PostScript [1] printers.

XY-pic pullback using the 'matrix' feature, code on p.5:



Typeset using XY-pic 3.0+.

Barr 'diagram' pullback, code on p.6, and XY-pic 'graph' pullback, code on p.7:

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

The left was typeset using 'diagram.tex' version 'Last revised 91-05-04' in the style of the last manual example [2, p.10], the right using XY-pic 3.0+.

References

1. Adobe Systems Incorporated. *PostScript Language Reference Manual*, second edition, 1990.
2. Michael Barr. *The Diagram Macros.* Available by anonymous ftp from `triples.math.mcgill.ca:/pub/texmacros/diagdoc.tex`.
3. Francis Borceux. *User's Guide for Diagram 3.* Louvain-la-Neuve. Available from the author, borceux@agel.ucl.ac.be.
4. Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion.* Addison-Wesley, 1994.
5. Kristoffer H. Holm and Hans Dybkjær. Drawing graphs and diagrams. Documents the obsolete `graphs` macros, September 1990.
6. B. W. Kernighan and L. L. Cherry. *Typesetting Mathematics – User's Guide.* Bell labs.
7. Brian W. Kernighan. PIC—a language for typesetting graphics. *Software Practice and Experience*, 12(1):1–21, 1982.
8. Donald E. Knuth. *The TeXbook.* Addison-Wesley, 1984.
9. Leslie Lamport. *LaTeX—A Document Preparation System.* Addison-Wesley, 2nd edition, 1994.
10. J. F. Osanna. *NROFF/TROFF User's Manual.* Bell Laboratories, Murray Hill, N.J., 1979.
11. John Reynolds. *User's Manual for Diagram Macros*, December 1987. Available by anonymous ftp from `e.ergo.cs.cmu.edu`.
12. Kristoffer H. Rose. Typesetting diagrams with XY-pic: User's manual. In Jiří Zlatuška, editor, *EuroTeX '92—Proceedings of the 7th European TeX Conference*, pages 273–292, Prague, Czechoslovakia, September 1992. Czechoslovak TeX Users Group.
13. Kristoffer H. Rose. XY-pic user's guide. Mathematics Report 94–148, MPCE, Macquarie University, NSW 2109, Australia, June 1994. For version 2.10+. Latest version available by anonymous ftp in `ftp.diku.dk: /diku/users/kris/TeX/xyguide.ps.Z`.
14. Kristoffer H. Rose and Ross Moore. XY-pic reference manual. Mathematics Report 94–155, MPCE, Macquarie University, NSW 2109, Australia, June 1994. For version 2.10+. Latest version available by anonymous ftp in `ftp.diku.dk: /diku/users/kris/TeX/xyrefer.ps.Z`.
15. Michael D. Spivak. *The Joy of TeX—A Gourmet Guide to Typesetting with the AMS-TeX Macro Package.* American Mathematical Society, second edition, 1990.
16. Paul Taylor. *Commutative Diagrams in TeX (version 4).* Department of Computing, Imperial College, 180 Queens Gate, London SW7 2BZ, January 1993.
17. Michael J. Wichura. *The PiCTeX Manual.* University of Washington, 1987.