TR 2002/001 ISSN 0874-338X

Logical Specification of Commutative Diagrams in a (La)TEX Document

Pedro Quaresma

Centre for Informatics and Systems of the University of Coimbra

Logical Specification of Commutative Diagrams in a (La)TEX Document

Pedro Quaresma*
CISUC
Departamento de Matemática, Universidade de Coimbra
3001-454 COIMBRA, PORTUGAL
e-mail: pedro@mat.uc.pt phone: +351-239 791 170 fax: +351-239 832 568

February 26, 2008

Abstract

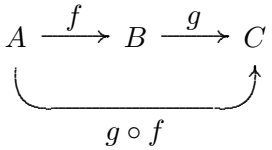
DCpic is a package of TEX macros for drawing Commutative Diagrams in a (La)TEX or ConTEXt document. Its distinguishing features are: the use of PICTEX a powerful graphical engine, and a simple specification syntax. A commutative diagram is described in terms of its objects and its arrows. The objects are textual elements and the arrows can have various straight or curved forms.

We describe the syntax and semantics of the user's commands, and present several examples of their use.

keywords: Commutative Diagrams, (La)TEX, PICTEX

1 Introduction

Commutative Diagrams (Diagramas Comutativos, in Portuguese), are a kind of graphs which are widely used in Category Theory [4, 7, 9], not only as a concise and convenient notation but also for "arrow chasing", a powerful tool for mathematical thought. For example, the fact that in a Category we have arrow composition is easily expressed by the following commutative diagram.



*This work was partially supported by the Portuguese Ministry of Science and Technology (MCT), under the programme PRAXIS XXI.

The word commutative means that the result from going throught the path f plus g is equal to the result from going throught the path $g \circ f$. Most of the graphs used in Category Theory are labelled digraphs which we can specify in terms of its objects, and its arrows.

The (La)TEX approach to typesetting can be characterized as "logical design" [5, 6], but commutative diagrams are pieces of "visual design", and that, in our opinion is the *piece de resistance* of commutative diagrams package implementation in (La)TEX. In a commutative diagrams package a user seeks the simplest notation, a logical notation, with the most powerful graphical engine possible, the visual part. The DCpic package has the simplest notation off all the commutative diagrams packages available [3]. In terms of graphical capabilities the PICTEX [12] package provides us with the best TEX-graphics engine, that is, without going to *Postscript* specials.

The DCpic package depends only of PICTEX and TEX, which means that you can use it in all formats that are based on these two. We have tested DCpic with LATEX, TEX plain, pdfLATEX, pdfTEX [11], and ConTEXt [8]; we are confident that it can be used under many other formats.

The present version (3.1) of DCpic package is available in CTAN and in the author's Web-page[1], the version described in this paper (3.2) will be putted in CTAN briefly.

2 The Logical Specification of Commutative Diagrams

As said the commutative diagrams are labelled digraphs, so we have a set of objects, a set of arrows (morphisms), and two functions from the set of arrows to the set of objects saying to each arrow what is its domain and its codomain. Trying to specify that in a LATEX/PICTEX picture amounts to say where to place the objects, and its labels, and then specify the arrows accordingly. In version 3.1 we have to specify the objects placement, and also the arrows placement, e.g.:

```
\begindc
\obj(1,1){$A$}
\obj(3,1){$B$}
\mor(1,1)(3,1){$f$}
\enddc
```

will produce the following diagram

$$A \xrightarrow{f} B$$

[1] http://www.mat.uc.pt/~pedro/LaTeX/

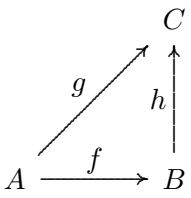
For most of the commutative diagrams we should not need to specify the arrows placement, so, beginning in version 3.2, we introduced a new variant. The object has an optional argument:

```
\obj(1,1){$A$}[objA]
\obj(1,1){$B$}[objB]
```

where the identifiers `objA`, and `objB` are choosen by the user (they should be unique). Then we can specify the arrows in the following form

```
\mor{objA}{objB}{$f$}
```

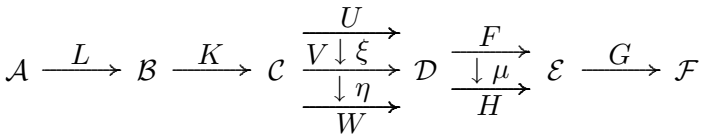
using this specification syntax we can write




```
\begindc
\obj(1,1){$A$}[A]
\obj(3,1){$B$}[B]
\obj(3,3){$C$}[C]
\mor{A}{B}{$f$}
\mor{A}{C}{$g$}
\mor{B}{C}{$h$}
\enddc
```

Notice that, with this notation, whenever we choose to change the objects actual placement, the arrows will change acordingly, in the "old" syntax we had to change the arrows placement whenever we change the objects placement.

It is still possible to specify the arrows using the "old" syntax, this is nedeed when we want to draw diagrams with diferent kinds of arrows, for example to specify multiple parallel arrows like in the Godement's "five" rules [4] diagram.




```
\begindc[7]
\obj(12,10){$\mathcal{A}$}[A]
\obj(19,10){$\mathcal{B}$}[B]
\obj(26,10){$\mathcal{C}$}[C]
\obj(34,10){$\mathcal{D}$}[D]
\obj(41,10){$\mathcal{E}$}[E]
\obj(48,10){$\mathcal{F}$}[F]
\mor{A}{B}{$L$}
```

```
\mor{B}{C}{$K$}
\mor{C}{D}{$V\qquad\ $}
\mor(26,12)(34,12){$U$}
\mor(26,12)(34,12){$\downarrow\xi$}[\atright,\solidarrow]
\mor(26,8)(34,8){$\downarrow\eta$}
\mor(26,8)(34,8){$W$}[\atright,\solidarrow]
\mor(34,11)(41,11){$F$}
\mor(34,9)(41,9){$\downarrow\mu$}
\mor(34,9)(41,9){$H$}[\atright,\solidarrow]
\mor{E}{F}{$G$}
\enddc
```

Some of the arrows may be specified in terms of its objects, but whenever we want to draw an arrow not going (exactly) from an object centre point to another object centre point, we must specify the arrow placement in absolute terms.

The opcional argument in the `begindc` environment is a magnification factor, a large number (default value is 30) gives us a coarse diagram, small numbers allows us to specify a "fine grain" diagram allowing the drawing of multiple parallel arrows and other types of "fine grain" diagrams.

3 Constructing Commutative Diagrams

DCpic depends on PiCTeX, thus you must include an apropriate command to load PiCTeX and DCpic in your document, e.g. in a LaTeX document we will add "`\usepackage{dcpic,pictex}`", to the preamble.

A commutative diagram in DCpic is a "picture" in PiCTeX, in which we place our *objects* and *morphisms* (arrows). The user's commands in DCpic are: `begindc` and `enddc` which established the coordinate system where the objects will be placed; `obj`, the command which defines the place and the contents of each object; `mor`, and `cmor`, the commands which define the morphisms, linear and curved arrows, and its labels.

Now we will describe each of these commands in greater detail.

3.1 The Diagram Environment

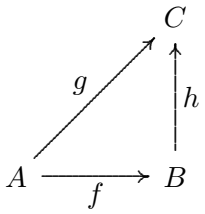
The command `begindc`, establishes a Cartesian coordinate system with 1pt units,

`\begindc[`*<magnification factor>*`] ... \enddc`

such a small unit gives us a good control over the placement of the graphical objects, but in most of the diagrams not involving curved arrows such a "fine grain" is not desirable, so the optional argument specifies a magnifying factor

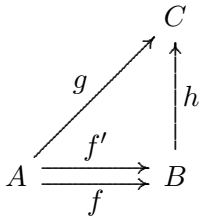
$m \in \mathbb{N}$, with a default value of 30. The advantage of this decision is twofold: we can define the "grain" of the diagram, and we can adjust the size of the diagram to the available space.

- a "course grain" diagram is specified almost as a table, with the numbers giving us the lines and the columns were the objects will be placed, the following diagram has the default magnification factor:



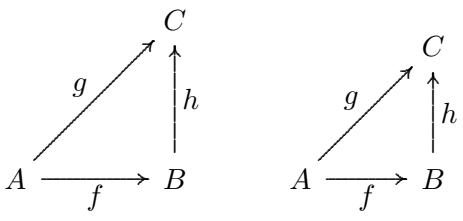

```
\begindc
\obj(1,1){$A$}[A]
\obj(3,1){$B$}[B]
\obj(3,3){$C$}[C]
\mor{A}{B}{$f$}[\atright,\solidarrow]
\mor{A}{B}{$g$}
\mor{B}{C}{$h$}[\atright,\solidarrow]
\enddc
```

- a "fine grain" diagram is a bit harder to design but it gives us a better control over the objects placement, the following diagram has a magnification factor of three, this gives us the capability of drawing the arrows f and f' very close together:




```
\begindc[3]
\obj(10,10){$A$}[A]
\obj(30,10){$B$}[B]
\obj(30,30){$C$}[C]
\mor(10,9)(30,9){$f$}[\atright,\solidarrow]
\mor(10,11)(30,11){$f^\prime$}
\mor{A}{C}{$g$}
\mor{B}{C}{$h$}[\atright,\solidarrow]
\enddc
```

- the magnification factor gives us the capability of adapting the size of the diagram to the available space, without having to redesign the diagram, for example the specification of the next two diagrams differs only in the magnification factor: 30 for the first; and 25 for the second.



Note that the magnification factor does not interfere with the size of the objects, but only with the size of the diagram as a whole.

After establishing our "drawing board" we can begin placing our "objects" on it, we have three commands to do so, the `obj`, `mor`, and `cmor`, for objects, morphisms, and "curved" morphisms respectively.

3.2 Objects

Each object has a place, a content, and an optional label.

```
\obj(<x>,<y>){<contents>}[<label>]
```

the x and y, integer values, will be multiplied by the magnifying factor. The *contents* will be put in the centre of an "hbox" expanding to both sides of $(m \times x, m \times y)$. The optional argument *label* is used to pass the object's placement, and the sizes of the "hbox" that contains the object's *contents*, to the arrows drawing commands.

3.3 Linear Arrows

The command to draw linear arrow has two variants.

A: As mandatory arguments a pair of objects (domain, and codomain), and a label

```
\mor(<domain>)(<codomain>){<label>}[<label placement>,<arrow type>]
```

the other arguments are optional.

The mandatory arguments *domain*, and *codomain* refer to the corresponding objects, and obtain from them the necessary information to place the arrow in its actual position.

The distance from the centre of the domain object to the actual beginning of the arrow is adjusted to the actual size of the "hbox" that contains the object. The same things happens with the arrow actual ending.

B: As mandatory arguments two pairs of coordinates, the beginning and the ending points, and a label.

```
\mor(<x1>,<y1>)(<x2>,<y2>)[<d1>,<d2>]{<label>}[<label placement>,<arrow type>]
```

the other arguments are optional.

In this variant we explicitly specify the arrows starting and ending points. The distance from the point $(m \times x_1, m \times y_1)$ to the actual beginning of the arrow may be modified by the user with the specification of d_1, the same thing happens for the arrow actual ending in which case the user-value will

be d_2. The specification of d_1 and d_2 is optional. The two pairs of coordinates should coincide with the coordinates of two objects in the diagram, but no verification of this fact is made.

The actual construction of the arrow and the other (optional) arguments are common to both variants. The line connecting the two points is constructed in the following way: the beginning is given by a point 10pt away from the point $(m \times x_1, m \times y_1)$, likewise the end point is 10pt away from $(m \times x_2, m \times y_2)$. If the "arrow type" specifies that, a tail, and a pointer (arrow) will be added. If the arrow is horizontal (vertical) the label is placed in a "hbox" with centre point, (x_l, y_l), at a distance of 10pt plus a correction factor depending of the "hbox" width (height) from the middle point of the arrow. If the arrow is obliquos the point (x_l, y_l), at a distance of 10pt from the middle point of the arrow, will be the bottom-right corner or the top-left corner of the "hbox" containing the label, depending of the angle of the arrow, and the label placement. In all cases the position of the "hbox" is such that its contents will not interfere with the line.

The placement of the label, to the left (default value), or to the right, and the type of the arrow: a solid arrow (default value), a dashed arrow, a line, an injection arrow, or an application arrow, are the last optional arguments of this command.

3.4 Quadratic Arrows

The command that draws curved lines in DCpic uses the `setquadratic` command of PiCTeX, this will imply a quadratic curve specified by an odd-number of points,

`\cmor(`*<list of points>*`)␣`*<arrow direction>*`(`*<x>*`,`*<y>*`){`*<label>*`}[`*<arrow type>*`]`

the space after the list of points is mandatory. After drawing the curved line we must put the tip of the arrow on it, at present it is only possible to choose from: up, down, left, or right pointing arrow, and we must explicitly specify what type we want. The next thing to draw it is the arrow label, the placement of that label is determined by the x, and y values which give us the coordinates, after being magnified, of the centre of the "hbox" that will contain the label itself.

The arrow type is an optional argument, its default value is a solid arrow, the other possible values are a dashed arrow and a line, in this last case the arrow tip is omitted. The arrow type values are a subset of those of the `mor` command.

A rectangular curve with rounded corners is easy to specify and should cater for most needs, with this in mind we give the following tip to the user: to specify a rectangular, with rounded corners, curve we choose the points which give us the *expanded chess-horse movement*, that is, (x, y), $(x \pm 4, y \mp 1)$, $(x \mp 1, y \pm 4)$, or (x, y),$(x \pm 1, y \mp 4)$, $(x \mp 4, y \pm 1)$, those sets

of points will give us the four corners of the rectangle; to form the whole line it is only necessary to add an odd number of points joining the two (or more) corners.

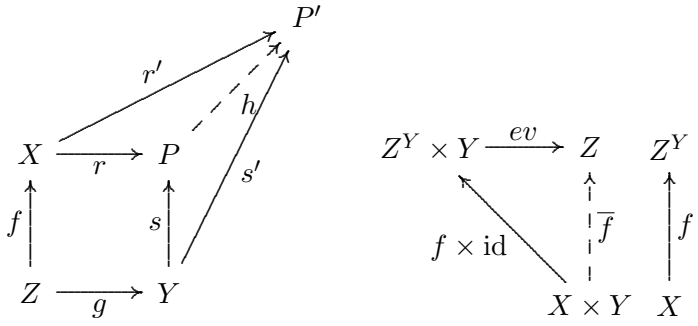
4 Examples

We now present some examples that give an idea of the DCpic package capabilities. We will present here the diagrams, and in the appendix the code which produced such diagrams.

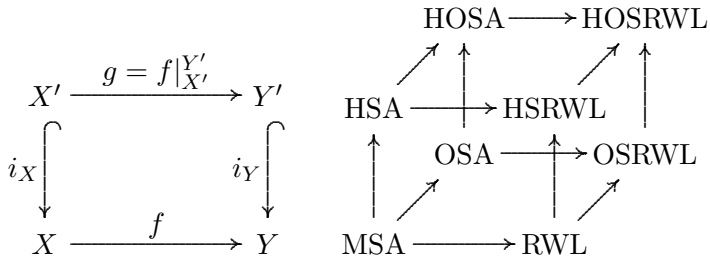
4.1 The Easy Ones

The diagrams presented in this section are very easy to specify in the DCpic syntax, just a couple of objects and the arrows joining them.

Push-out and Exponentials:



Function Restriction and the *CafeOBJ* Cube [2]

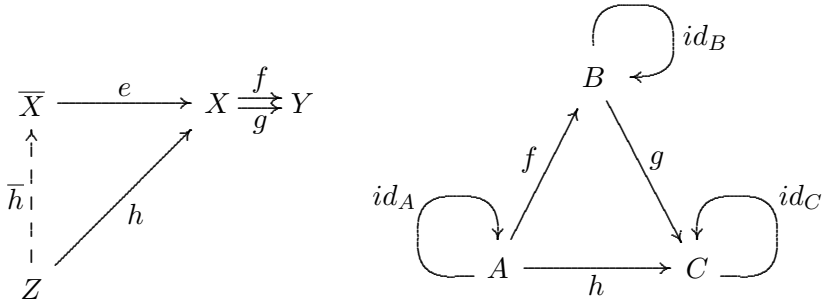


4.2 The Not so Easy

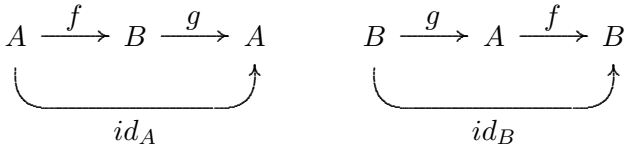
The diagrams presented in this section are a bit harder to specify. We have curved arrows, and also double arrows. The construction of the former was already explained. The double arrow (and triple, and ...) is made with two distinct arrows drawn close to each other in a diagram with a very "fine grain", that is, using a magnifying factor of just 2 or 3.

All the diagrams were made completely within DCpic.

Equaliser, and a 3-Category:



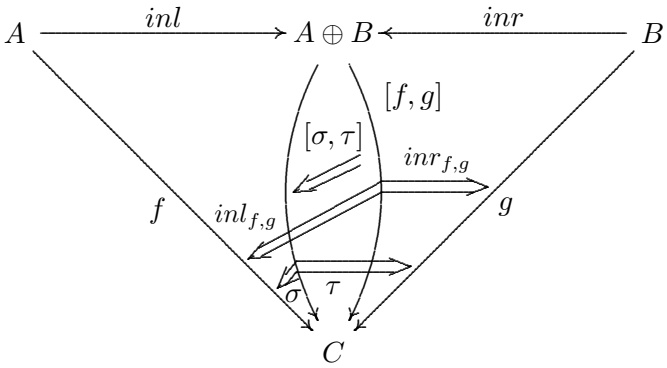
Isomorfisms:



4.3 The others ...

It was already stated that some kinds of arrows are not supported in DCpic, e.g., $\Rightarrow$, but we can put a PICTEX command inside a DCpic diagram, so we can produce a diagram like the one that we will show now. Its complete specification within DCpic is not possible, at least for the moment.

Lax coproduct [1]



5 DCpic compared

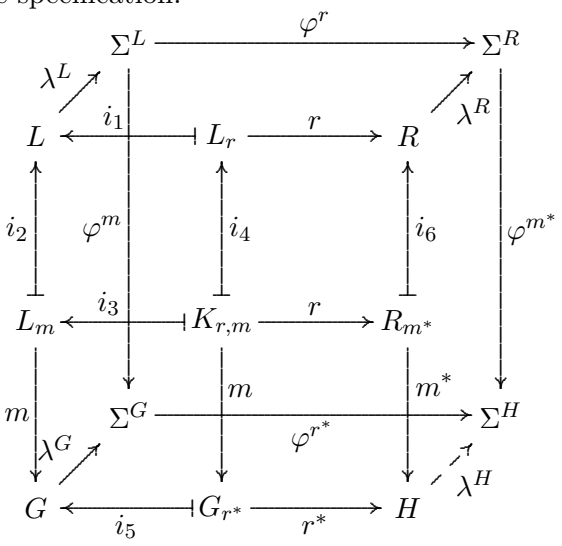
If one took the Feruglio article [3] about typesetting commutative diagrams in (La)TEX we can say that:

- the graphical capabilities of DCpic are among the best. Excluding packages which use Postscript specials the DCpic package is the best among available packages.

- the specification syntax is one of the simplest, the package by John Reynolds [10] has a syntax similar to the "arrows specified with two pairs of coordinates" syntax.

We did not try to take any measure of computational performance.

The following diagram is one of the test-diagrams used by Feruglio, as we can see DCpic performs very well, drawing the complete diagram based on a very simple specification.



6 Conclusions

We think that DCpic performs well in the "commutative diagrams arena", it is easy to use, with its commands we can produce the most usual types of commutative diagrams, and if we accept the use of PiCTeX commands, we are capable of producing any kind of diagram. It is also a (La)TeX-only package, that is, the file produced by DCpic does not contain any Postscript special, neither any special font, which in terms of portability is an advantage.

The author and his colleagues in the Mathematics Department of Coimbra University have been using the (now) old version (2.1) of DCpic for some time with much success, some of the missing capabilities of the older version were incorporated in the new version (3.2), and the missing capabilities of the new version will be taken care in future versions.

Acknowledgements

We wish to thank to all the participants in EuroTeX2001 for some very fruitful discussions, specially Michal Marvan for his ideas about the type of arrows, and about the different types of morphism specification.

References

[1] S. Abramsky, Dov Gabbay, and T. Maibaum, editors. *Handbook of Logic in Computer Science*, volume 1 of *Oxford Science Publications*. Claredon Press, Oxford, 1992.

[2] Rãzvan Diaconescu and Kokichi Futatsugi. *CafeOBJ Report: The Language, Proof Techniques, and Methodologies for Object-Oriented Algebraic Specification*, volume 6 of *AMAST series in Computing*. World Scientific, 1998.

[3] Gabriel Valiente Feruglio. Typesetting commutative diagrams. *TUGboat*, 15(4):466–484, 1994.

[4] Horst Herrlich and George Strecker. *Category Theory*. Allyn and Bacon Inc., 1973.

[5] Donald E. Knuth. *The TeXbook*. Addison-Wesley Publishing Company, Reading,Massachusetts, 1986.

[6] Leslie Lamport. *LaTeX: A Document Preparation System*. Addison-Wesley Publishing Company, Reading, Massachusetts, 2nd edition, 1994.

[7] S. MacLane. *Categories for the Working Mathematician*. Springer-Verlag, New York, 1971.

[8] Ton Otten and Hans Hagen. *ConTeXt an excursion*. Pragma ADE, Hasselt, 1999.

[9] Benjamin Pierce. *Basic Category Theory for Computer Scientists*. Foundations of Computing. The MIT Press, London, England, 1998.

[10] John Reynolds. *User's Manual for Diagram Macros*. http://www.cs.cmu.edu/~jcr/, 1987. `diagmac.doc`.

[11] Hàn Thế Thành, Sebastian Rahtz, and Hans Hagen. *The pdfTeX manual*, 1999.

[12] Michael Wichura. *The PiCTeX Manual*. Personal TEX Inc, 12 Madrona Avenue, Mill Valley, CA 94941 - USA, 3rd edition, March 1992

7 Appendix: The DCpic Specifications

Push-out:

```
\begindc[26]
\obj(1,1){$Z$}[Z]
\obj(1,3){$X$}[X]
\obj(3,1){$Y$}[Y]
\obj(3,3){$P$}[P]
\obj(5,5){$P^\prime$}[Pp]
\mor{Z}{X}{$f$}
\mor{Z}{Y}{$g$}[\atright,\solidarrow]
\mor{X}{P}{$r$}[\atright,\solidarrow]
\mor{Y}{P}{$s$}
\mor{X}{Pp}{$r^\prime$}
\mor{Y}{Pp}{$s^\prime$}[\atright,\solidarrow]
\mor{P}{Pp}{$h$}[\atright,\dasharrow]
\enddc
```

Exponentials:

```
\begindc
\obj(1,3){$Z^Y\times Y$}[ZYY]
\obj(3,3){$Z$}[Z]
\obj(3,1){$X\times{}Y$}[XxY]
\obj(4,1){$X$}[X]
\obj(4,3){$Z^Y$}[ZY]
\mor{ZYY}{Z}{$ev$}
\mor{XxY}{ZYY}{$f\times{}\mathrm{id}$}
\mor{XxY}{Z}{$\overline{f}$}[\atright,\dasharrow]
\mor{X}{ZY}{$f$}[\atright,\solidarrow]
\enddc
```

Function Restriction:

```
\begindc[28]
\obj(1,1){$X$}[X]
\obj(1,3){$X^\prime$}[Xp]
\obj(3,1){$Y$}[Y]
\obj(3,3){$Y^\prime$}[Yp]
\mor{X}{Y}{$f$}
\mor{Xp}{X}{}[\atright,\injectionarrow]
\mor{Yp}{Y}{}[\atright,\injectionarrow]
\mor{Xp}{Yp}{$g=f|^{Y^\prime}_{X^\prime}$}
\enddc
```

***CafeOBJ* Cube:**

```
\begindc[17]
\obj(1,1){MSA}[MSA]
\obj(5,1){RWL}[RWL]
\obj(3,3){OSA}[OSA]
\obj(7,3){OSRWL}[OSRWL]
\obj(1,4){HSA}[HSA]
\obj(5,4){HSRWL}[HSRWL]
```

```
\obj(3,6){HOSA}[HOSA]
\obj(7,6){HOSRWL}[HOSRWL]
\mor{MSA}{RWL}{}
\mor{MSA}{HSA}{}
\mor{MSA}{OSA}{}
\mor{RWL}{HSRWL}{}
\mor{RWL}{OSRWL}{}
\mor{OSA}{HOSA}{}
\mor{OSA}{OSRWL}{}
\mor{OSRWL}{HOSRWL}{}
\mor{HSA}{HSRWL}{}
\mor{HSA}{HOSA}{}
\mor{HOSA}{HOSRWL}{}
\mor{HSRWL}{HOSRWL}{}
\enddc
```

Equaliser:

```
\begindc[2]
\obj(1,1){$Z$}[Z]
\obj(1,36){$\overline{ X}$}[oX]
\obj(36,36){$X$}[X]
\obj(52,36){$Y$}[Y]
\mor{Z}{oX}{$\overline{ h}$}[\atleft,\dasharrow]
\mor{Z}{X}{$h$}[\atright,\solidarrow]
\mor{oX}{X}{$e$}
\mor(36,37)(52,37)[8,8]{$f$}
\mor(36,35)(52,35)[8,8]{$g$}[\atright,\solidarrow]
\enddc
```

A 3-Category:

```
\begindc[3]
\obj(14,11){$A$}[A]
\obj(39,11){$C$}[C]
\obj(26,35){$B$}[B]
\mor{A}{C}{$h$}[\atright,\solidarrow]
\mor{A}{B}{$f$}
\mor{B}{C}{$g$}
\cmor((11,10)(10,10)(9,10)(5,11)(4,15)(5,19)(9,20)(13,19)(14,15))
 \pdown(1,20){$id_A$}
\cmor((42,10)(43,10)(44,10)(48,11)(49,15)(48,19)(44,20)(40,19)(39,15))
 \pdown(52,20){$id_C$}
\cmor((26,39)(27,43)(31,44)(35,43)(36,39)(35,36)(31,35)) \pleft(40,40){$id_B$}
\enddc
```

Isomorfisms:

```
\begindc[3]
\obj(10,15){$A$}[A]
\obj(40,15){$A$}[Aa]
```

```
\obj(25,15){$B$}[B]
\mor{A}{B}{$f$}
\mor{B}{Aa}{$g$}
\cmor((10,11)(11,7)(15,6)(25,6)(35,6)(39,7)(40,11)) \pup(25,3){$id_A$}
\obj(55,15){$B$}[Bb]
\obj(85,15){$B$}[Bc]
\obj(70,15){$A$}[Ab]
\mor{Bb}{Ab}{$g$}
\mor{Ab}{Bc}{$f$}
\cmor((55,11)(56,7)(60,6)(70,6)(80,6)(84,7)(85,11)) \pup(70,3){$id_B$}
\enddc
```

Lax coproduct: Guess how.

DCpic and the others:

```
\begindc[35]
\obj(1,1){$G$}[Gr]
\obj(3,1){$G_{r^*}$}[Grstar]
\obj(5,1){$H$}[H]
\obj(2,2){$\Sigma^G$}[SigmaG]
\obj(6,2){$\Sigma^H$}[SigmaH]
\obj(1,3){$L_m$}[Lm]
\obj(3,3){$K_{r,m}$}[Krm]
\obj(5,3){$R_{m^*}$}[Rmstar]
\obj(1,5){$L$}[L]
\obj(3,5){$L_r$}[Lr]
\obj(5,5){$R$}[R]
\obj(2,6){$\Sigma^L$}[SigmaL]
\obj(6,6){$\Sigma^R$}[SigmaR]
\mor{Gr}{SigmaG}{$\lambda^G$}
\mor{Grstar}{Gr}{$i_5$}[\atleft,\aplicationarrow]
\mor{Grstar}{H}{$r^*$}[\atright,\solidarrow]
\mor{H}{SigmaH}{$\lambda^H$}[\atright,\dasharrow]
\mor{SigmaG}{SigmaH}{$\varphi^{r^*}$}[\atright,\solidarrow]
\mor{Lm}{Gr}{$m$}[\atright,\solidarrow]
\mor{Lm}{L}{$i_2$}[\atleft,\aplicationarrow]
\mor{Krm}{Lm}{$i_3\quad$}[\atright,\aplicationarrow]
\mor{Krm}{Rmstar}{$r$}
\mor{Krm}{Lr}{$i_4$}[\atright,\aplicationarrow]
\mor{Krm}{Grstar}{$\stackrel{\displaystyle m}{\barraB}$}
\mor{Rmstar}{R}{$i_6$}[\atright,\aplicationarrow]
\mor{Rmstar}{H}{$\stackrel{\displaystyle m^*}{\barraA}$}
\mor{L}{SigmaL}{$\lambda^L$}
\mor{Lr}{L}{$i_1\quad$}[\atright,\aplicationarrow]
\mor{Lr}{R}{$r$}
\mor{R}{SigmaR}{$\lambda^R$}[\atright,\solidarrow]
\mor{SigmaL}{SigmaG}{$\varphi^m$}[\atright,\solidarrow]
\mor{SigmaL}{SigmaR}{$\varphi^r$}
\mor{SigmaR}{SigmaH}{$\varphi^{m^*}$}
\enddc
```