Graphics

Diagxy, a Lego-like diagram package

Michael Barr

1 Overview

This is an introduction to my diagxy package that sets commutative (usually) diagrams using an interface that patches pieces together, Lego style. Since it comes with full documentation/ tutorial, I will just hit the high points here and refer to the package for more details.

There are a number of good packages for making commutative diagrams. Mostly, they are based on \halign. This has advantages and disadvantages. The advantages are that the syntax is mostly familiar and the underlying TeX engine does all the work in its native mode, at a considerable advantage in speed, not to mention relative ease in programming. The main disadvantage from my point of view is lack of flexibility. TeX makes all the spacing decisions and the user has few options. I first became aware of this when I used XY-pic to make a "W" shaped diagram and discovered that, owing to a mismatch between the various nodes of the diagram, the "W" was leaning to one side. Another problem was that a fixed distance was used for arrows and if the label on an arrow was larger than that distance, it was hard to force it to be larger and the increments available (by adding nodes) were large.

In the 1980s I had designed a diagram package built around LaTeX's picture mode that was based on defining shapes that fit together like Lego blocks. The sizes of the shapes were under user control. This violates the LaTeX principle of logical markup, but I do not feel that diagrams can be done using logical markup, at least not at present. The original package had defined shapes such as squares (really rectangles, but we call them squares anyway) and triangles at various orientations that would fit together. I was limited by the directions available in the picture mode (an arrow could go in only one of 48 different directions, at a resolution that varies between 3 and 14 degrees). The package was not entirely satisfactory for other reasons.

One real advantage of XY-pic is that it has a full 256-character font of line segments in different directions and two fonts of arrowheads that allow arrows in any of 512 directions, which is as many as one could wish for. It was also possible to make macros to replace the plain TeX arrowheads with those of XY-pic, giving a more uniform look to papers, especially ones using many arrows. Therefore I had the idea a few years ago of reimplementing my original package making use of the XY-pic arrows. When I sat down to do this I discovered that, underlying the XY-pic matrix package was a drawing engine of great generality. In fact, it turned out to be much easier to use this engine as the back end to my reimplementation than to just use the arrows. As a result, it is possible to mix my diagram package with native XY-pic code with excellent results. Also, the present diagxy works with both LaTeX and plain TeX. Except for one incompatibility described below, diagxy appears to be compatible with AMS-LaTeX as well, although I have not tested that extensively.

There are two kinds of people: those who can read and write syntax diagrams and those who learn by example. I am far out on the latter limb of what is really a continuum. Accordingly, most of the documentation of diagxy is in the form of a tutorial, diaxydoc.tex, that gives many examples. Even though I designed and implemented the package, I myself still refer to the tutorial all the time.

2 Simple diagrams

The most elementary is

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, "g"'] & B \arrow[d, "h"] \\
C \arrow[r, "k"'] & D
\end{tikzcd}$$

which is produced using the code

```
$$\bfig\square[A`B`C`D;f`g`h`k]\efig$$
```

Here I used the ` symbol to separate the nodes. It seemed to be the symbol that was least likely to actually be used. If it is, it must be enclosed in braces to avoid being interpreted by TeX as a delimiter. Similarly, the ; was chosen as a character little used in mathematics (although unfortunately used a lot in computer programs) that separates the nodes from the arrow labels. Any of the entries can be left empty. Although the default is to place all four arrows, optional arguments allow omission of arrows too. Incidentally, square brackets are used internally as delimiters, so any square brackets should also be enclosed in braces. Sometimes they are unnecessary, but they can never hurt and I have not actually worked out what situations require their use.

Suppose the square is not large enough. If you want the diagram

$$\begin{tikzcd}
\mathrm{Hom}(A,B) \arrow[r, "\mathrm{Hom}(f,B)"] \arrow[d, "\mathrm{Hom}(A,g)"'] & \mathrm{Hom}(A',B) \arrow[d, "\mathrm{Hom}(A',g)"] \\
\mathrm{Hom}(A,B') \arrow[r, "\mathrm{Hom}(f,B')"'] & \mathrm{Hom}(A',B')
\end{tikzcd}$$

you use size parameters set off by < and > as in

```
 $$\bfig\square<1000,500>[\Hom(A,B)‘
\Hom(A’,B) ‘\Hom(A,B’)‘\Hom(A’,B’);
 \Hom(f,B)‘\Hom(A,g)‘ \Hom(A’,g)
‘\Hom(f,B’)]\efig$$
```

Here the numbers 1000 and 500 refer to the width and height of the square in units of .01 em. This is doubtless too fine; I hardly ever use a distance not a multiple of 10 (or, occasionally, 5), but I feel that now I am stuck with that choice. The default is <500,500> and will be supplied if not specified.

3 Lego blocks

Although simple diagrams are the most common, the real power of diagxy is the ease of putting blocks together to make more complicated diagrams. In order to do this, there are parameters, enclosed in (), that tell where to place a block in the given coordinate system. For example:

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, "g"'] & B \arrow[d, "h"] \\
C \arrow[r, "k"'] \arrow[d, "l"'] & D \arrow[d, "m"] \\
E \arrow[r, "n"'] & F
\end{tikzcd}$$

This is produced by

```
 $$\bfig\square(0,500)[A‘B‘C‘D;f‘g‘h‘k]
 \square[C‘D‘E‘F;‘l‘m‘n]\efig$$
```

This is an example of the Lego block approach. Note that C and D are repeated. This is necessary in order that the arrows be the correct lengths. On the other hand, there is no need to repeat k and, if you did, it would by default appear both above and below the middle arrow.

There are other optional parameters to tell where the arrow labels should go and to change the arrow style. The latter can get very complicated, as complicated as XY-pic allows. Arbitrary XY-pic arrow styles, including arrows that follow arbitrary Bezier curves, are allowed.

There are a number of basic triangle shapes, which are named according to the letter in the alphabet that most neatly fits in the triangle. For example, a `\ptriangle` is a right triangle whose hypotenuse goes from east to south and an `\Atriangle` is an isoceles triangle whose base is horizontal. This example

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, "h"'] & B \arrow[d, "k"] \arrow[dl, "g"] \\
C \arrow[r, "l"'] & D
\end{tikzcd}$$

is produced by

```
 $$\bfig\ptriangle[A‘B‘C;f‘h‘g]
  \dtriangle[B‘C‘D;‘k‘l]\efig$$
```

An alternate way of making the same diagram is given by

```
 $$\bfig\square[A‘B‘C‘D;f‘h‘k‘l]
  \morphism(500,500)<-500,-500>[B‘C;g]
  \efig$$
```

where `\morphism` creates the arrow from B to C.

Incidentally, the effect of `\bfig`...`\efig` in a displayed diagram is nearly the same as `\xy` ... `\endxy` except that it has the effect of enclosing the whole in a `\vcenter` box so that equation numbers, if any, are vertically centred. In case you are curious about the names `\bfig` and `\efig`, I used an NROFF-type system briefly in the very early 1980s (in the paleolithic era before TEX) and figures were set off using `.BFIG` and `.EFIG`.

4 In-line arrows

One of my minor gripes about LATEX is that the arrows used in text look quite different from those used in diagrams. So one of the things I have done is define a macro `\to` that works somewhat like `\rightarrow` but uses the XY-pic arrowheads. It has several other features, including various options, but the most important is that it grows to accommodate long labels. An example is $A \xrightarrow{\text{a long label}} B$, for which the source is

```
    $A\to^{\mbox{\rm a long label}}B$.
```

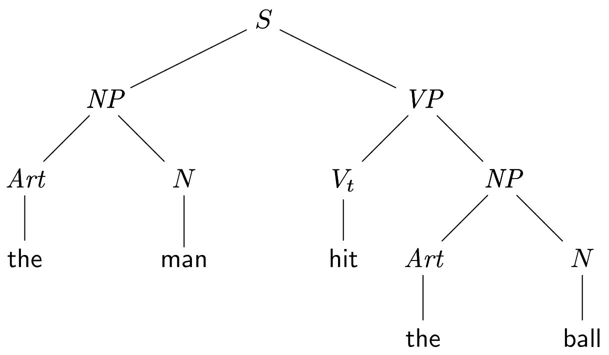

Figure 1: A tree diagram from mathematical linguistics

There are other macros, such as `\two` and `three`, which make double and triple arrows. The directions of the arrows (right or left) can be independently set, using standard Xy-pic controls.

5 Compatibility

There is nothing in this code (or in Xy-pic) that uses anything but plain TeX. I have used it extensively under LaTeX and never unearthed any incompatibility. I have not used it much with AMS-LaTeX but there is one known incompatibility. One of the AMS symbol fonts makes a little box which is named `\square`. A simple fix is to load the `amsfonts` first. If you actually require that character, then before loading diagxy, simply say `\let\Box\square`, to call it `\Box`. Incidentally, the `@` sign is used in the Xy-pic syntax, along with just about every other non-alphanumeric character. Thus you cannot change the catcodes of any of them, which means there are no private control sequences. Thus one must be careful not to redefine any of the internal sequences used.

6 An alternate syntax

A couple years after this package was released, I received a note from a graduate student named Gerd Zeibig who suggested an alternate syntax in which you specify the placement of nodes in the coordinate system and then draw arrows between them. He had also implemented this in a way that used two counters for each node. Given the shortage of counters in standard LaTeX, I decided to reimplement it using macro definitions in place of these counters. So it is now possible to describe diagrams as illustrated below. The diagram in Figure 1 appears in a set of notes on mathematical linguistics. While there is certainly no problem producing it using the earlier code, it is much more systematic to describe trees in this way:

```
 $$\bfig
\newcommand{\NP}{\hbox{\mit NP}}
\newcommand{\VP}{\hbox{\mit VP}}
\newcommand{\Art}{\hbox{\mit Art}}
 \node 1a(0,0)[S]
 \node 2a(-600,-300)[\NP]
 \node 2b(600,-300)[\VP]
 \arrow/-/[1a`2a;]
 \arrow/-/[1a`2b;]
 \node 3a(-900,-600)[\Art]
 \node 3b(-300,-600)[N]
 \node 3c(300,-600)[V_t]
 \node 3d(900,-600)[\NP]
 \arrow/-/[2a`3a;]
 \arrow/-/[2a`3b;]
 \arrow/-/[2b`3c;]
 \arrow/-/[2b`3d;]
 \node 4a(-900,-900)[{\sf the}]
 \node 4b(-300,-900)[{\sf man}]
 \node 4c(300,-900)[{\sf hit}]
 \node 4d(600,-900)[\Art]
 \node 4e(1200,-900)[N]
 \arrow/-/[3a`4a;]
 \arrow/-/[3b`4b;]
 \arrow/-/[3c`4c;]
 \arrow/-/[3d`4d;]
 \arrow/-/[3d`4e;]
 \node 5a(600,-1200)[{\sf the}]
 \node 5b(1200,-1200)[{\sf ball}]
 \arrow/-/[4d`5a;]
 \arrow/-/[4e`5b;]
 \efig$$
```

First you define the nodes and then draw arrows between them. In this case, we wanted only lines, whence the `/-/` specification on the arrows. The labels on the nodes are almost completely arbitrary (limited only by what `\csname` . . . `\endcsname` allows).

$$\begin{tikzcd}
TT \arrow[rr, "\mu"] \arrow[rrdd, "TT\eta'"'] \arrow[dddd, "T\eta'T"'] & & T \arrow[rrdd, "T\eta"] & & & & \\
& & 1 & & & & \\
& 2 & TTT' \arrow[rr, shift left, "\mu T'"] \arrow[rr, shift right, "T\sigma"'] \arrow[dd, "T\eta'TT'"'] & & TT' \arrow[dd, "T\eta'T'"'] \arrow[rrdd, "\mathrm{id}"] & & \\
& & & 3 & & 6 & \\
TT'T \arrow[rr, "TT'T\eta'"] \arrow[dd, "\sigma T"'] & & TT'TT' \arrow[rr, "TT'\sigma"] \arrow[dd, "\sigma TT'"'] & & TT'T' \arrow[rr, "T\mu'"] \arrow[dd, "\sigma T'"'] & & TT' \arrow[dd, "\sigma"] \\
& 4 & & 5 & & 7 & \\
T'T \arrow[rr, "T'T\eta'"'] & & T'TT' \arrow[rr, "T'\sigma T'"'] & & T'T' \arrow[rr, "\mu'"'] & & T'
\end{tikzcd}$$

Figure 2: A larger diagram

7 A large diagram

Figure 2 shows a large diagram that is taken directly from the book *Toposes, Triples and Theories* by Michael Barr and Charles Wells, published by Springer Verlag in 1984. It may well have been the very first book produced using LaTeX, which was not released until 1985. Using diagxy, it can be produced with the following code.

```
 $$\bfig
\scalefactor{1.4}
\qtriangle(0,1000)/>`>`/[TT`T`TTT'
  ;\mu`TT\eta'`]
\btriangle(500,1000)/`>`@<-14\ul>/%
  [T`TTT'`TT';`T\eta`T\sigma]
\morphism(0,1500)|l|/>/<0,-1000>%
  [TT`TT'T;T\eta'T]
\square(500,500)|ammx|/@<14\ul>`>
  `>`/[TTT'`TT'`TT'TT'`TT'T';%
   \mu T'`T\eta'TT'`T\eta'T'`]
\morphism(1000,1000)|r|/>/<500,
  -500>[TT'`TT';\hbox{\rm id}]
\square/>`>``>/[TT'T`TT'TT'`T'T`T'
  TT';TT'T\eta'`\sigma T``T'T\eta']
\square(500,0)|ammb|[TT'TT'`TT'T'`
  T'TT'`T'T'; TT'\sigma`\sigma TT'
  `\sigma T'`T'\sigma T']
\square(1000,0)/>``>`>/[TT'T'`TT'
  `T'T'`T';T\mu'``\sigma`\mu']
\place(500,1250)[1]
\place(215,1000)[2]
\place(750,750)[3]
\place(215,250)[4]
\place(750,250)[5]
\place(1140,750)[6]
\place(1250,250)[7]
 \efig$$
```

8 Availability

diagxy can be found on CTAN in the directory `macros/generic/diagrams/barr/`, and at my own ftp site at `ftp.math.mcgill.ca/pub/barr/diagxy.zip`.

◇ Michael Barr
Dept. of Math. and Stats.
McGill University
805 Sherbrooke St. W
Montreal, QC H3A 2K6
Canada
mbarr (at) barrs.org