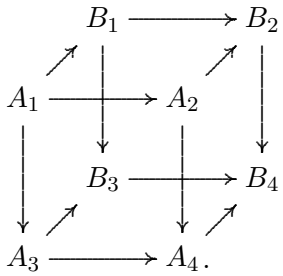
Making Graphs with the dcpic Package

The dcpic package was not originally written to typeset discrete graphs. It was created to facilitate the making of *commutative diagrams*, which are standard objects in algebra and topology. For example, here's one kind of commutative diagram:



Note that this has essentially the same kind of structure as a graph: we need only make the symbols into vertices and the arrows into edges. Fortunately, the dcpic package[1] allows for these options.

Getting Started

Obviously, you'll need to acquire the dcpic package. Newer versions of MiKTEX know about dcpic, so the MiKTEX package manager should be able to install it for you. Also, running dcpic makes use of two auxiliary packages: graphicx and pictex. In the old days you had to acquire these separately, but they might be so standard by now that you may already have them. Check and see.

Don't forget: once you install lots of new stuff, you need to tell MiKTEX that the new packages are there and ready to be used. In your MiKTEX options dialog box, choose to "refresh the file name database."

When you are composing a document that uses dcpic, you must alert your TEX compiler that you will be using these additional packages. Thus you need to add dcpic, pictex, and graphicx to the list of packages in the `\usepackage` line in your header. My standard `\usepackage` line looks like

```
\usepackage{amsmath, amsthm, amsfonts, amssymb,
            latexsym, graphicx, dcpic, pictex}
```

Basic Structure

In the language of modern algebra and topology, vertices are known as *objects* and edges are known as *morphisms*. This is also the terminology that dcpic uses.

The coding for a graph or commutative diagram is very logical. You assign coordinates to your vertices (i.e., objects) and dcpic will set the vertices according to their coordinate location. You also give each vertex its own shortcut reference label. You then specify where you want your edges (i.e., morphisms) to run by referring to these shortcut labels.

The opening and closing of a graph will always look like

```
\[
\begindc{2}[3]
...graph coding goes here...
\enddc
\]
```

[1] "dc" for *diagramas comutativos*, as the package was written by a Portuguese mathematician.

You shouldn't worry too much about the parameters in the second line, following `\begindc`. Essentially, those specify arrow types and magnification factors. For instance, if I wanted my morphisms to be actual arrows, I would instead use `\begindc{0}[3]`.

To create and place a vertex, the syntax is of the form `\obj(x,y)[label]{ }`. Here, `(x,y)` are the Cartesian coordinates of the vertex and `label` is a shortcut label. Usually you would have to specify what you want LaTeX to typeset for the actual object. However, in this setting the default is a solid bullet •, so you don't have to do anything. This is the reason that the above set of curly braces is empty.

On Choosing Coordinates: It does not matter how you choose the "origin," since dcpic will create the graph and center it in the page as a *single object*. I think you can save lots of time and stress by doing a little planning beforehand. Here's what I do:

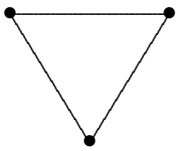
- Sketch my graph on a sheet of paper
- Locate the physical center of the graph (there may not be a vertex there—that's fine)
- Rig up coordinates so that each edge will be roughly 16 units long.
- TeX it and see what happens
- Make adjustments if necessary.

For example, if I am setting a square graph, the corners will be $(\pm 8, \pm 8)$.

Once you have your vertices defined and placed, it's time to put in your edges. The basic syntax for an edge is `\mor{a}{b}{$n$}` where `a` is the shortcut label for the initial vertex, `b` is the shortcut label for the terminal vertex, and `n` is the symbol to be typeset over the edge. If you do not have labelled edges, just use empty braces, as in `\mor{a}{b}{ }`.

Note: By default, dcpic will typeset the edge label to the *left* of the edge, where "left" is specified with respect to the orientation of the edge from `a` to `b`. If you ever need to change this, just reverse the roles of `a` and `b`!

Examples
Let's look at a simple graph, like this 3-cycle:



This graph was created by the following code:

```
\[
\begindc{2}[3]
\obj(-10,8)[1]{ }
\obj(10,8)[2]{ }
\obj(0,-8)[3]{ }
\mor{1}{2}{ }
\mor{2}{3}{ }
\mor{1}{3}{ }
```

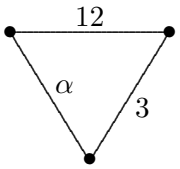
```
\enddc
\]
```

I started out with my standard ± 8 coordinate difference, but it made the top edge too short. So I just enlarged the x-coordinates of the top two vertices to stretch it out a bit. Sometimes you have to play around a little to make it look as nice as possible. Luckily, you only have to change the coordinates of the vertices, as the edges automatically refer back to them.

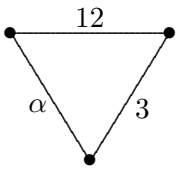
Now let's take that graph and assign weights to the edges. Easy enough:

```
\[
\begindc{2}[3]
\obj(-10,8)[1]{ }
\obj(10,8)[2]{ }
\obj(0,-8)[3]{ }
\mor{1}{2}{$12$}
\mor{2}{3}{$3$}
\mor{1}{3}{$\alpha$}
\enddc
\]
```

produces



Oops! The edge-weight α is on the "wrong" side since I ran that edge *from* vertex 1 *to* vertex 3. Let's reverse this so that α appears on the "right" side. I'll just change the `\mor{1}{3}{$\alpha$}` line to `\mor{3}{1}{$\alpha$}`, producing



Aha! I win!

A Final Word on Final Words

Sometimes, a graph is the last "word" in your sentence. Therefore, it should be followed with a period. (Yes, I know I have violated this rule above, but I did this for the sake of simplicity in my explanations.) In this case, you will have to make the period an object in the dcpic environment. Something as simple as `\obj(x,y)[p]{.}` suffices.