Typesetting diagrams with kuvio.tex

Anders G. S. Svensson <svensson@math.ubc.ca>

The macros in `kuvio.tex` use PostScript code embedded in `\special` control sequences to perform rotations and reflections and to set graylevels. The `\special` syntax used is that of Tomas Rokicki's excellent `dvips`. Be forewarned that this is likely to confuse many `dvi` (not PostScript) previewers.

A summary contains details on the syntax of control sequences as well as discussion of several points not covered in the main sections of this manual.

Contents

Revision 2.5.12 for patchlevel 216. June 9, 1996.

1. Cell Macros

Let's start with a simple diagram.

```
\Diagram
P' \aTo (2,-4) <{b'} \aTo (4,-2) >{a'}                  \\
          &  \rdTo                                     \\
          &        &  P       &  \rTo ^a &  A      \\
          &        &  \dTo <b &          &  \dTo   \\
          &        &  B       &  \rTo    &  X      \\
\endDiagram
```

$$\begin{tikzcd}
P' \arrow[dr] \arrow[drr, "a'"] \arrow[ddr, "b'"'] & & \\
& P \arrow[r, "a"] \arrow[d, "b"'] & A \arrow[d] \\
& B \arrow[r] & X
\end{tikzcd}$$

The control sequences `\rTo`, `\dTo`, `\rdTo` and `\aTo` in this example each specify an arrow in the diagram. These control sequences are called *cell macros* and, in this case, each is associated with the *cell type* `To`. Note that, except for `\aTo`, the direction of the arrow is built into the name of the cell macro: `\rTo` for a rightward `To` cell, `\dTo` for a downward `To` cell, etc. In the case of `\aTo`, the arrow extends from the alignment entry in which the macro was found to the alignment entry whose relative position is specified by a parenthesized pair of integers. In our example, the arrow specified by

```
\aTo (2,-4)
```

points to the alignment entry 2 columns to the right and 4 rows below the entry in which this code is located.

We shall soon see that there are many cell types predefined in `kuvio.tex` and that other types can easily be defined by the user using the control sequence `\newcell`. The full list of cell macros associated with a cell type *type* is as follows.

`\a`*type*	*type* cell originating at an alignment entry
`\b`*type*	*type* cell terminating at an alignment entry
`\ru`*type*	rightward and upward *type* cell
`\r`*type*	rightward *type* cell
`\rd`*type*	rightward and downward *type* cell
`\lu`*type*	leftward and upward *type* cell
`\l`*type*	leftward *type* cell
`\ld`*type*	leftward and downward *type* cell
`\u`*type*	upward *type* cell
`\d`*type*	downward *type* cell
`\`*type*	*type* cell typeset as a *modification*

The last of these differs from the others in that it it not used in alignments. This will be discussed in §4. The `\b`*type* cell macro is similar to `\a`*type* except that the required coordinate pair specifies the relative position of the tail (rather than the head) of the arrow.

```
\Diagram
A \aTo (1,-1) &               & A \\
              & B \bTo (1,1)     \\
\endDiagram
```

$$\begin{tikzcd} A \arrow[dr] & & A \arrow[dl] \\ & B & \end{tikzcd}$$

Let us refer to cell macros which can be used in alignments as *alignment cell macros* and all others as *modification cell macros*. Thus, associated with any cell type we have ten alignment cell macros and one modification cell macro.

Labels are tied to arrows using `<`, `>`, `_` and `^` for labels to the left, right, below and above an arrow respectively. Many characteristics of an arrow and its labels can be modified using certain characters and control sequences, which we will refer to as *modifiers*, following an occurrence of a cell macro. The `:` modifier can be used to tie labels to a point other than the midpoint of an arrow.

```
\Diagram
A & \rTo ^f :{.75} & B \\
\endDiagram
```

$$A \xrightarrow{\qquad f} B$$

The argument passed to `:` is used to interpolate linearly between the tail and head of an arrow, 0 being the tail and 1 being the head. One can also use `:` to move labels in a direction perpendicular to that of an arrow by supplying it with a dimension rather than a real number.

```
\Diagram
A & \rTo ^f :{5pt} & B \\
\endDiagram
```

$$A \xrightarrow{f} B$$

To use both functions of the `:` modifier, separate two arguments (in either order) by a comma or semicolon.

```
\Diagram
A & \rTo ^f :{.75,5pt} & B \\
\endDiagram
```

$$A \xrightarrow{\qquad f} B$$

Except for the `\a`*type* and `\b`*type* variants, arrows specified by alignment cell macros, *alignment cells*, stretch to meet non-empty alignment entries.

```
\Diagram
A & \rTo & B        \\
C &      & \rTo & D \\
\endDiagram
```

$$\begin{tikzcd} A \arrow[r] & B & \\ C \arrow[rr] & & D \end{tikzcd}$$

The control sequence `\stop` can be used to stop an alignment cell at an otherwise empty alignment entry.

```
\Diagram
  & \rTo & \stop & \rTo & \stop          \\
  &      & \stop & \rTo & \stop & \rTo & \\
\endDiagram
```

$$\begin{tikzcd} {} \arrow[r] & {} \arrow[r] & {} & {} \\ {} & {} \arrow[r] & {} \arrow[r] & {} \end{tikzcd}$$

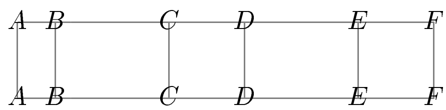
2. Grid Lines

Each diagram is constructed on a grid, vertical grid lines corresponding to columns in an alignment and horizontal grid lines to rows. Tokens preceding the first occurrence of a cell macro in an alignment entry are typeset at the intersection point of two grid lines.

There are two kinds of grid: *rigid* and *flexible*. The grid lines of a rigid grid will be uniformly spaced unless explicitly moved while those of a flexible grid will be positioned using knowledge of the particular vertices and arrows occurring in a diagram. The pair `\Diagram` `\endDiagram` uses a rigid grid by default. Flexible grids are enabled using the control sequence `\flexible`. These will be discussed in §12.

Grid lines can be moved using any of the control sequences `\mx`, `\my`, `\dx`, `\dy`, `\ml`, `\mr`, `\dl` and `\dr`. `\mx` adjusts the position of the vertical grid line associated with the column in which this control sequence occurs while `\dx` simultaneously adjusts the position of all grid lines associated with columns lying to the right of the one in which it occurs. The row in which either `\mx` or `\dx` occurs is immaterial.

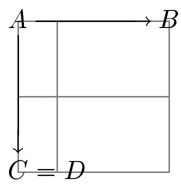
```
\gridlines
\Diagram
A & B \mx{-5mm} & C & D & E          & F \\
A & B           & C & D & E \dx{5mm} & F \\
\endDiagram
```



The behaviour of `\my` and `\dy` on horizontal grid lines corresponding to alignment rows is analogous, `\dy` moving grid lines associated with rows lying on or above a particular row.

The control sequence `\ml` moves a vertical grid line leftward so that the alignment entry in which this control sequence occurs abuts the next entry typesetting as a box of non-zero dimensions.

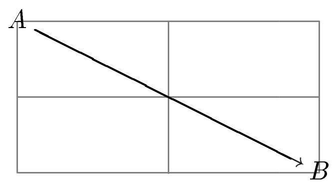
```
\gridlines
\Diagram
A    & \rTo       & B \\
\dTo                \\
C    & {} = D \ml   \\
\endDiagram
```



Similarly, `\mr` moves a vertical grid line rightward. `\dl` and `\dr` have the same relation to `\ml` and `\mr` that `\dx` has to `\mx`.

The default spacing between grid lines can be adjusted in two ways: By scaling the current values using `\xscale` and `\yscale` or by explicitly setting new values using `\xgrid` and `\ygrid`.

```
\gridlines \xgrid=2cm
\Diagram
A              \\
  & \rdTo      \\
  &       & B  \\
\endDiagram
```

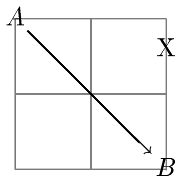


The control sequence `\scale` can be used to set both `\xscale` and `\yscale` to the same value. Similarly, `\grid` can be used to set both `\xgrid` and `\ygrid`.

3. Coordinates and Modifications

Grid lines provide us with a coordinate system in which points with non-integral coordinates are obtained by linear interpolation between grid lines. These coordinates can be used to typeset *modifications* at arbitrary locations on (or outside) a diagram, following an occurrence of the control sequence `\Modify`.

```
\gridlines
\Diagram
A            \\
  & \rdTo    \\
  &       & B \\
\Modify
\Text{X} (2,1.5)
\endDiagram
```

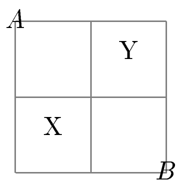


Here the control sequence `\Text` takes a single argument, the material to be typeset, and must also be supplied with a parenthesized coordinate pair indicating a position on the diagram.

Besides `\Text` there are control sequences `\Txt`, `\Math`, `\Vertex` and `\Label`. The latter three typeset their argument in math mode: `\Math` in `\textstyle`, and `\Vertex` and `\Label` in the current values of `\vertexstyle` and `\labelstyle`, the defaults being `\displaystyle` and `\scriptstyle` respectively. These are the same styles in which vertices and labels in the alignment part of a diagram are typeset.

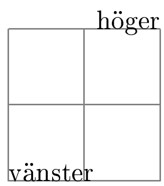
Control sequences which typeset modifications behave very much like cell macros in that they are also affected by modifiers. The `:` modifier can be used to specify a position to `\Text`, `\Txt`, `\Math`, `\Vertex` and `\Label` as a fractional distance from a first coordinate to a second. The fraction .5 is assumed if none is specified explicitly.

```
\gridlines
\Diagram
A       \\
        \\
  & & B \\
\Modify
\Text{X} (0,0) (1,1)
\Text{Y} (0,0) (2,2) :{.75}
\endDiagram
```



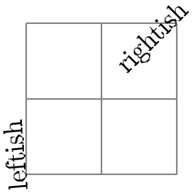
The position indicated to `\Text` and its siblings are those of the *tag point*: The point on the typeset material which is placed at the position in question. For `\Text` the default tag point lies at the midpoint of the baseline while for `\Txt`, `\Math`, `\Vertex` and `\Label` it lies at the midpoint of the math axis. The location of the tag point along the length of the typeset material can be changed using the `*` modifier.

```
\gridlines
\Diagram
     \\
 & & \\
     \\
\Modify
\Text{v\"anster}  (0,0) *0
\Text{h\"oger}    (2,2) *1
\endDiagram
```



Modifications can be rotated using the `/` modifier.

```
\gridlines
\Diagram
     \\
 & & \\
     \\
\Modify
\Text{leftish}  (0,0)  *{.2}  /{95}
\Text{rightish} (2,2)  *{.8}  /{45}
\endDiagram
```



4. Modification Cells I

Modification cell macros typeset arrows as modifications. We refer to such arrows as *modification cells*.

```
\Diagram
xF & \rTo ^{fF} & yF             \\
\dTo <{x\eta} & & \dTo >{y\eta} \\
xG & \rTo _{fG} & yG             \\
\Modify
\Para (1.5,.5) <{f\eta} /{-135}
\endDiagram
```

$$\begin{tikzcd} xF \arrow[r, "fF"] \arrow[d, "x\eta"'] \arrow[dr, phantom, "f\eta"] & yF \arrow[d, "y\eta"] \\ xG \arrow[r, "fG"'] & yG \end{tikzcd}$$

The length of a modification cell can be specified explicitly using the `|` modifier.

```
\Diagram
xF & \rTo ^{fF} & yF             \\
\dTo <{x\eta} & & \dTo >{y\eta} \\
xG & \rTo _{fG} & yG             \\
\Modify
\Para (1.3,.7) <{f\eta} /{-135} |{14mm}
\endDiagram
```

$$\begin{tikzcd} xF \arrow[r, "fF"] \arrow[d, "x\eta"'] \arrow[dr, phantom, "f\eta"] & yF \arrow[d, "y\eta"] \\ xG \arrow[r, "fG"'] & yG \end{tikzcd}$$

The following alternative to the previous two examples will become clear in §10.

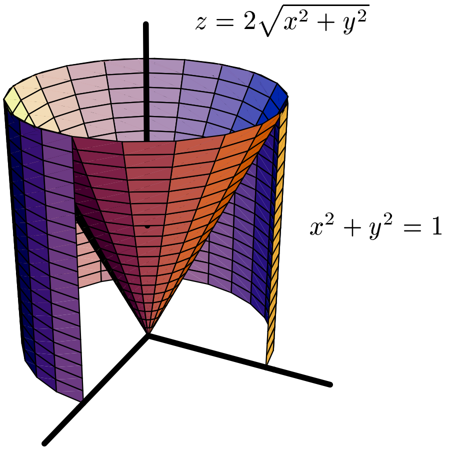
```
\ptpush+=3pt
\Diagram
xF            & \rTo ^{fF}        & yF                   \\
\dTo <{x\eta} &                   & \dTo >{y\eta} \pt{t} \\
xG            & \rTo _{fG} \pt{h} & yG                   \\
\Modify
\Para \pt{t} \pt{h} <{f\eta}
\endDiagram
```

$$\begin{tikzcd} xF \arrow[r, "fF"] \arrow[d, "x\eta"'] \arrow[dr, phantom, "f\eta"] & yF \arrow[d, "y\eta"] \\ xG \arrow[r, "fG"'] & yG \end{tikzcd}$$

5. Figures and Graphs

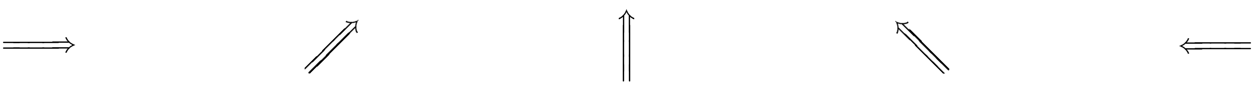
Before continuing our discussion of modification cells, let us first mention two variants of the `\Diagram` `\endDiagram` pair. The first, `\Figure` `\endFigure`, can be used to typeset modifications on top of any TEX box. One use for this is for typesetting TEX labels on top of included PostScript figures.

```
\input epsf  %  from the dvips distribution

\Figure
\epsfxsize=3in
\epsffile{plot.eps}
\Modify
\Math{x^2 + y^2 = 1}            (.8,.55)   *0
\Math{z = 2\sqrt{x^2 + y^2}}    (.6,.9)    *0
\endFigure
```



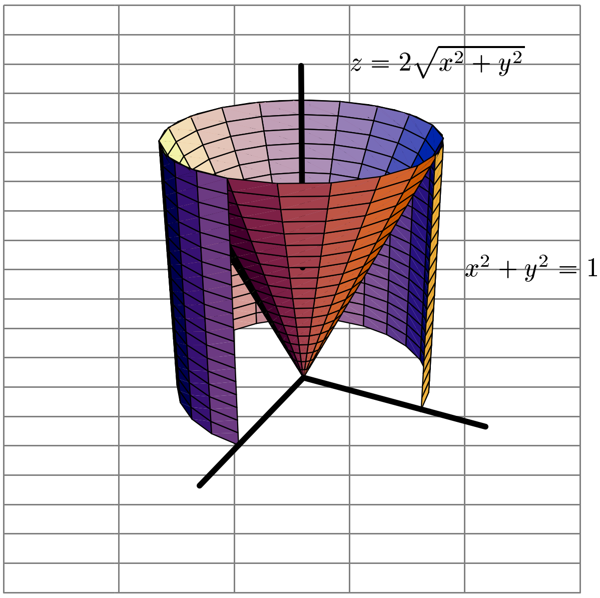
The second, `\Graph` `\endGraph`, uses `\Figure` to typeset modifications on top of an empty box whose dimensions we specify.

```
\Graph{\hsize}{1.5cm}
\Two          (0,.5)
\Two /{45}    (.25,.5)
\Two /{90}    (.5,.5)
\Two /{135}   (.75,.5)
\Two /{180}   (1,.5)
\endGraph
```



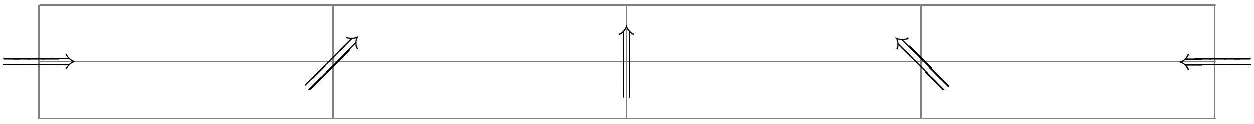
The coordinate system used for typesetting modifications on a `\Figure` or `\Graph` has its origin at the bottom left corner of the box in question. There are two ways of specifying units of length in the coordinate directions of this coordinate system, the default units being chosen so that the top right corner of the box has coordinates (1,1). First, the control sequences `\xrange`, `\yrange` and `\range` can be used to divide the default units by any integer value.

```
\xrange=5 \yrange=20 \gridlines
\Figure
\epsfxsize=3in
\epsffile{plot.eps}
\Modify
\Math{x^2 + y^2 = 1}            (4,11)  *0
\Math{z = 2\sqrt{x^2 + y^2}}    (3,18)  *0
\endFigure
```



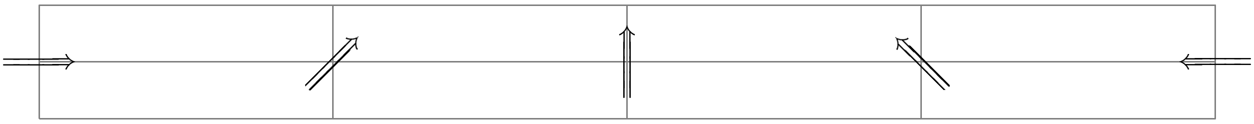
Second, one can use `\xunit`, `\yunit` and `\unit` to explicitly specify units of length.

```
\xunit=.25\hsize \yrange=2 \gridlines
\Graph{\hsize}{1.5cm}
\Two (0,1)   \Two /{45} (1,1)   \Two /{90} (2,1)   \Two /{135} (3,1)   \Two /{180} (4,1)
\endGraph
```



For any individual `\Graph`, unit and range specifications can be incorporated into the argument list.

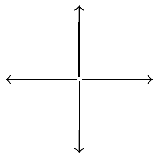
```
\gridlines
\Graph{\hsize,.25\hsize}{1.5cm,2}
\Two (0,1)   \Two /{45} (1,1)   \Two /{90} (2,1)   \Two /{135} (3,1)   \Two /{180}  (4,1)
\endGraph
```



6. Modification Cells II

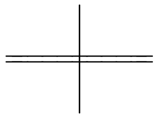
The modification cells shown above place the midpoint of an arrow at a specified location on a diagram. Just as with ordinary modifications, we can change the tag point using the * modifier.

```
\range=2
\Graph{2cm}{2cm}
\To (1,1) *0        \To (1,1) *0 /{180}
\To (1,1) *0 /{90}  \To (1,1) *0 /{270}
\endGraph
```



There is another variant of modification cell in which we specify the coordinates of the tail and head of an arrow rather the position of the tag point.

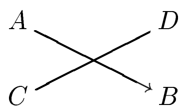
```
\Graph{2cm,2}{15mm,2}
\Line  (1,0)  (1,2)
\Bar   (0,1)  (2,1)
\endGraph
```



We will refer to modification cells as *type 1* or *type 2* depending on whether they have been specified with one or two coordinate pairs.

Vertices can be attached to the tail and head of such a modification cell using the modifiers `\tl` and `\hd`. In the case of `\Diagram` `\endDiagram`, if the coordinates of either end of a type 2 modification cell are integral then an implicit `\tl` or `\hd` is performed with the contents of the appropriate alignment entry whenever a tail or head hasn't been explicitely attached.

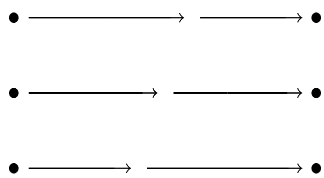
```
\Diagram
A &  &   \\
  &  & B \\
\Modify
\To   (0,1) (2,0)
\Line (0,0) (2,1) \tl{C} \hd{D}
\endDiagram
```



7. Fudges and Moves

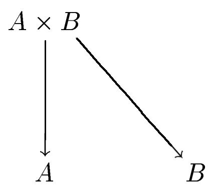
The length of individual arrows can be adjusted using the modifiers `\dt` and `\dh` for the tail and head of an arrow respectively. Passing a single dimension to one of these control sequences lengthens the appropriate end of an arrow by the specified amount.

```
\let\b\bullet
\Diagram
\b & \rTo \dh{10pt}  & \stop & \rTo \dt{-10pt} & \b \\
\b & \rTo            & \stop & \rTo            & \b \\
\b & \rTo \dh{-10pt} & \stop & \rTo \dt{10pt}  & \b \\
\endDiagram
```



`\tx`, `\ty`, `\hx` and `\hy` can be used to make an arrow point from or to a point other than the midpoint of a vertex.

```
\Diagram
A\times B                    \\
\dTo       & \rdTo \tx{7pt}     \\
A          &                & B \\
\endDiagram
```



Length and positional adjustments with the aforementioned six control sequences are referred to as *fudges*.

The length of a positionally fudged arrow is adjusted to accommodate the vertices at either end of the arrow.

```
\Diagram
X                                       \\
  & \rdTo \hy{-20pt}
    \rdTo
    \rdTo \hx{20pt}                     \\
  &                  & A\times B\times C \\
\endDiagram
```

$$\begin{tikzcd} X \arrow[dr, shift right=2] \arrow[dr] \arrow[dr, shift left=2] & \\ & A \times B \times C \end{tikzcd}$$

Arrows and modifications can be *moved* using the modifiers `\up` and `\rt`.

```
\Diagram
X & \rTo \up{3pt}
    \lTo \up{-2pt} & A \\
\endDiagram
```

$$\begin{tikzcd} X \arrow[r, shift left=1] & A \arrow[l, shift left=1] \end{tikzcd}$$

Moves differ from fudges in that an entire arrow is literally just moved: No length adjustments are made. Using the modifiers `\ru` and `\rr`, moves can be made relative to a coordinate system which has been rotated from the horizontal along with a (rightward pointing) arrow or modification.

```
\Diagram
A                       \\
  & \rdTo ^f \ru{5pt}
    \rdTo _g \ru{-5pt}  \\
  &                   & B \\
\endDiagram
```

$$\begin{tikzcd} A \arrow[dr, shift left=2, "f"] \arrow[dr, shift right=2, "g"'] & \\ & B \end{tikzcd}$$

8. Defining New Cell Types

The control sequence `\newcell` is used to define cell types. For example, near the end of `kuvio.tex` we find the following two lines.

```
\newcell{To}
\let\Tofill\rightarrowfill
```

Expansion of `\newcell{`*type*`}` causes several control sequences to be defined, one being the *box macro* `\`*type*`box`.

```
\def\typebox#1{\hbox to #1{\typefill}}
```

This definition tells the macros in `kuvio.tex` how to typeset a rightward arrow of cell type *type*. As we can see, before using a cell macro associated with this cell type we were required to either define the *fill macro* `\`*type*`fill` or else redefine `\`*type*`box`.

```
\newcell{Eql}
\def\Eqlbox#1{\vcenter{\offinterlineskip
   \hbox to #1{\rightarrowfill} \vskip-2pt \hbox to #1{\rightarrowfill}}}

\Diagram
A & \rEql & B \\
\endDiagram
```

$$A \rightrightarrows B$$

9. Pads and Pushes

Let us try placing labels on our newly defined `Eql` cells.

```
\Diagram
X & \rEql ^{f_1} _{f_2} & A \\
\endDiagram
```

$$X \overset{f_1}{\underset{f_2}{\rightrightarrows}} A$$

These labels are probably a little too close to the arrows for most people's tastes. There are three *label pads* associated with every arrow in a diagram. As we have already seen, a pad which applies to only a single arrow can be specified with the `:` modifier. Global and type-specific label pads can be assigned using `\labelpad`. For any individual arrow, these three label pads are summed to determine the distance at which a label is placed from line joining its endpoints.

```
\labelpad=4pt \labelpad{Eql}=2.5pt
\Diagram
X & \rEql ^{f_1} _{f_2} & A & \rTo ^g & B \\
\endDiagram
```

$$X \overset{f_1}{\underset{f_2}{\rightrightarrows}} A \xrightarrow{\quad g \quad} B$$

Another kind of padding determines how much space will be placed around the sides of a diagram. Assignments can be made to `\lpad` (left), `\rpad` (right), `\bpad` (bottom) and `\tpad` (top). Assignments can also be made to `\Diagrampad`, `\Figurepad` and `\Graphpad` to add space to all four sides of a diagram.

```
\framed
\Diagram A & \rTo & B \\ \endDiagram
\quad\Diagrampad=10pt
\Diagram A & \rTo & B \\ \endDiagram
\quad\lpad=-5pt
\Diagram A & \rTo & B \\ \endDiagram
```

$$\boxed{A \longrightarrow B} \quad \boxed{\;A \longrightarrow B\;} \quad \boxed{\;A \longrightarrow B\;}$$

Cell pushes determine the spacing between an arrow and the vertices at its tail and head. As with label pads, cell pushes are additive. Global and type-specific cell pushes are assigned using `\cellpush`, just as with `\labelpad`. Length fudges with `\dt` and `\dh` can be thought of as arrow-specific pushes, although a positive fudge corresponds to a negative push.

```
\cellpush{Eql}=3pt
\Diagram
X & \rEql & A & \rTo & B \\
\endDiagram
```

$$X \;\rightrightarrows\; A \longrightarrow B$$

10. Naming Points and Attaching Arrows

Modification cells can be *attached* to points on other arrows by *naming* these points using the `\pt` modifier. Once a point has been named it can be referenced as a coordinate.

```
\Diagram
X & \rTo \up{10pt}  \pt{a1,.2} \pt{b1,.8}
    \rTo \up{-10pt} \pt{a2,.2} \pt{b2,.8} & Y \\
\Modify
\Two \pt{a1} \pt{a2}
\Two \pt{b1} \pt{b2}
\endDiagram
```

$$\begin{tikzcd} X \arrow[r, shift left=2ex] \arrow[r, shift right=2ex] \arrow[r, phantom, "\Downarrow\qquad\Downarrow"] & Y \end{tikzcd}$$

A point must be named before it is referenced. Note that cell macros are processed in order from left to right and top to bottom.

The form `\pt{`*name*`,`*number*`}` can be shortened to `\pt{`*name*`}` if *name* has not already been named or if another coordinate is not expected. In this case the default position is chosen. Global and type-specific defaults can be set using `\ptpoint`.

```
\ptpoint=.75 \ptpoint{One}=.5
\Diagram
X & \rTo \up{10pt}  \pt1
    \rTo \up{-10pt} \pt2 & Y & \rOne \up{10pt}  \pt3
                               \rOne \up{-10pt} \pt4 & Z \\
\Modify
\Two \pt1 \pt2
\Two \pt3 \pt4
\endDiagram
```

$$\begin{tikzcd} X \arrow[r, shift left=2] \arrow[r, shift right=2] \arrow[r, phantom, "\Downarrow"] & Y \arrow[r, shift left=2] \arrow[r, shift right=2] \arrow[r, phantom, "\Downarrow"] & Z \end{tikzcd}$$

A type-specific `\ptpoint` can be unset, causing the global value to be used, by making an empty assignment of the form `\ptpoint{`*type*`}={}`.

Global and type-specific assignments can also be made with `\ptpush` and `\atpush`. The ends of an arrow of cell type *type* attached to another arrow are shortened by the sum of the global and type-specific `\ptpush`. The ends of an arrow attached to an arrow of cell type *type* are shortened by the sum of the global and type-specific `\atpush`.

11. Defining New Diagram Types

Before actually typesetting a diagram, `\Diagram` begins a group and expands `\everyDiagram`. Similarly there a control sequences `\everyFigure` and `\everyGraph`. The control sequences `\newDiagram`, `\newFigure` and `\newGraph` allow one to further customize the setup for a named class of diagrams.

Expansion of `\newDiagram{`*type*`}` is roughly equivalent to the following.

```
\def\type{\bgroup\everyDiagram\everytype\D@Diagram}
\def\endtype{\D@endDiagram\egroup}
\let\everytype\relax
```

Here `\D@Diagram` and `\D@endDiagram` are a pair of inaccessible control sequences in terms of which `\Diagram` and `\endDiagram` are similarly defined. Thus, the setup for diagrams of type *type* is customized by redefining `\every`*type*.

```
\newDiagram{Diag}
\def\everyDiag{\flexible \xgrid=0pt}

\Diag
\cdots & \rTo & H_n A & \rTo & H_n X & \rTo & H_n(X,A) & \rTo & H_{n-1}A & \rTo & \cdots \\
\endDiag
```

$$\cdots \longrightarrow H_nA \longrightarrow H_nX \longrightarrow H_n(X,A) \longrightarrow H_{n-1}A \longrightarrow \cdots$$

The diagram type `Diag` is predefined in `kuvio.tex`. The other predefined diagram types are `Dg` and `Long`, both of which have `\flexible` and `\xgrid=0pt` in addition to adjusting the values of certain parameters.

By a *Diagram* (capitalized) we will mean any diagram typeset using `\Diagram \endDiagram` or any pair associated with a diagram type defined using `\newDiagram`. We will similarly use the terms *Figure* and *Graph*.

Rather than make global changes to the parameters available in `kuvio.tex`, it is probably always best to identify the various kinds of diagrams one is interested in and then define diagram types for each of these. The predefined types have been set up with the author's interests and tastes in mind but a little experimentation should yield other useful possibilities.

The following example uses *incremental assignment*, which is available for all dimension assignments made available by `kuvio.tex`.

```
\newDiagram{Cat}
\def\everyCat{\everyDiag\cellwidth+=25pt\ygrid+=-1mm}

\Cat
 &         & \stop & \rTo \pt t & \stop            \\
 & \rTo _f & \stop & \rTo \pt m & \stop & \rTo _g & \\
 &         & \stop & \rTo \pt b & \stop            \\
\Modify
\Two \pt t \pt m <\alpha
\Two \pt m \pt b <\beta
\endCat
```

$$\begin{tikzcd}
 & {} \arrow[r] \arrow[d, phantom, "\alpha\Downarrow"] & {} & \\
{} \arrow[r, "f"'] & {} \arrow[r] \arrow[d, phantom, "\beta\Downarrow"] & {} \arrow[r, "g"'] & {} \\
 & {} \arrow[r] & {} &
\end{tikzcd}$$

12. Flexible Grids

Expansion of the control sequence `\flexible` alters the way in which the macros in `kuvio.tex` position vertical grid lines in a Diagram. Rather than simply spacing them uniformly, they are instead placed using the knowledge of the particular vertices and arrows occuring in the Diagram.

The vertical grid lines in a flexible grid gravitate toward a single column, the *gravitating column*. By default this will be the centermost column, or columns in case of an even number, but the gravitating column can also be set explicitly using the control sequence `\grav`. For many diagrams though, especially simpler ones, the actual column will be immaterial.

Columns gravitate in such a way as to ensure that the distance between adjacent vertices which do not otherwise have a horizontal arrow between them are at least `\xgrid` apart. When `\xgrid=0pt` this behaviour can be used to make adjacent entries abut.

```
\def\xc#1{#1\cdots#1}

\Dg
v_1\xc\otimes v_k & \rMapsto & v_k\xc\otimes v_1 \\
T(V)              & \rTo     & T(V)              \\
\dTo <\theta      &          & \dTo >\theta      \\
C(V,q)            & \rTo     & C(V,q)            \\
v_1\cdots v_k     & \rMapsto & v_k\cdots v_1 & {} \equiv (v_1\cdots v_k)^* \\
\endDg
```

$$\begin{tikzcd}
v_1 \otimes \cdots \otimes v_k \arrow[r, mapsto] & v_k \otimes \cdots \otimes v_1 & \\
T(V) \arrow[r] \arrow[d, "\theta"'] & T(V) \arrow[d, "\theta"] & \\
C(V,q) \arrow[r] & C(V,q) & \\
v_1 \cdots v_k \arrow[r, mapsto] & v_k \cdots v_1 & {} \equiv (v_1 \cdots v_k)^*
\end{tikzcd}$$

Alignment entries are always centered on the column in which they occur. Notice how the two rightmost entries in the following example lie in different columns of the alignment.

```
\Dg
A         &         & SO(4)        & {} \approx S^3\times SO(3) \\
\dMapsto  &         & \dTo                                     \\
[Ae_1]    & \qquad  & SO(4)/SO(3)  &        & {} \approx S^3     \\
\endDg
```

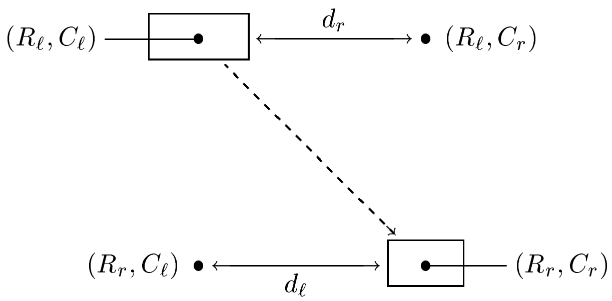
$$\begin{tikzcd}
A \arrow[d, mapsto] & & SO(4) \approx S^3 \times SO(3) \arrow[d] \\
{[Ae_1]} & & SO(4)/SO(3) \approx S^3
\end{tikzcd}$$

The parameter `\cellwidth` controls the minimum distance between vertices the ends of a horizontal arrow. Other components of a particular diagram may, of course, cause this distance to increase but for Diagrams consisting of a single row arrows will be of a uniform length.

```
\cellwidth=25pt
\Diag
\cdots & \rTo &
\pi_j S^k & \rTo & \pi_j V(n,k) & \rTo & \pi_j V(n-1,k+1) & \rTo & \pi_{j-1} S^k &
\rTo & \cdots \\
\endDiag
```

$$\cdots \longrightarrow \pi_j S^k \longrightarrow \pi_j V(n,k) \longrightarrow \pi_j V(n-1,k+1) \longrightarrow \pi_{j-1} S^k \longrightarrow \cdots$$

There are two parameters which control the distance between columns containing the vertices at the ends of a diagonal arrow. Unless one of the the `\lb`, `\rb` or `\db` modifiers is specified, the distance between the centers of the columns containing the two vertices at either end of such an arrow will be at least `\columndist`. If one of these modifiers is specified, `\bracewidth` will instead be used to position columns in the following manner.



Suppose the vertices at the two ends of a diagonal arrow have row and column coordinates (R_ℓ, C_ℓ) and (R_r, C_r) with $C_\ell < C_r$. If `\lb` is specified then the distance d_ℓ from the vertex at (R_r, C_ℓ) to the vertex at (R_r, C_r) will be at least `\bracewidth`. If `\rb` is specified then the distance d_r from the vertex at (R_ℓ, C_ℓ) to the vertex at (R_ℓ, C_r) will be at least `\bracewidth`. If `\db` is specified then both requirements will be met.

Note that specifying `\db` is not equivalent to specifying both `\lb` and `\rb`: The last modifier specified will take precedence.

We will refer to a diagonal arrow as being either *braced* or *unbraced* depending on whether `\bracewidth` or `\columndist` has been used in positioning the columns containing its head and tail vertices. We will also distinguish between *fully braced*, *left braced* and *right braced* diagonal arrows, the latter two having been specified using the `\lb` and `\rb` modifiers respectively.

The control sequence `\braced` causes all diagonal arrows to be fully braced by default. It also inverts the action of `\db` so that diagonal arrows will be unbraced only when this modifier is specified. The control sequence `\loose` resets the behaviour to the default and, additionally, sets `\columndist=0pt`. This may be appropriate for certain triangular diagrams.

The control sequence `\bb` in the following (and other) examples invokes the beautiful bbm family of fonts.

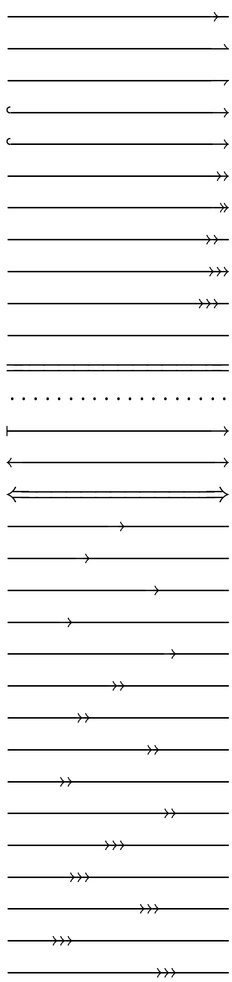
```
\loose
\Dg
       &        & {\bb R}^2             \\
       & \ldTo  &       & \rdTo         \\
C({\bb R}^2,-g)&& \rTo &       & {\bb H} \\
\endDg
```

$$\begin{tikzcd} & \mathbb{R}^2 \arrow[dl] \arrow[dr] & \\ C(\mathbb{R}^2,-g) \arrow[rr] & & \mathbb{H} \end{tikzcd}$$

13. Predefined Cell Types

There are a number of fill macros defined in `kuvio.tex`.

```
\lrightarrowfill
\rightharpoonupfill
\rightharpoondownfill
\hookrightarrowfill
\rightintofill
\rrightarrowfill
\rightepifill
\lrrightarrowfill
\rrrightarrowfill
\lrrrightarrowfill
\linefill
\barfill
\cdotfill
\mapstofill
\leftrightarrowfill
\Leftrightarrowfill
\rightarrowxfill
\rightarrowxxfill
\xrightarrowxfill
\rightarrowxxxfill
\xxrightarrowxfill
\rrightarrowxfill
\rrightarrowxxfill
\xrrightarrowxfill
\rrightarrowxxxfill
\xxrrightarrowxfill
\rrrightarrowxfill
\rrrightarrowxxfill
\xrrrightarrowxfill
\rrrightarrowxxxfill
\xxrrrightarrowxfill
```



Several of these also have leftward variants. Of course, all can be used in diagrams, the nicest way being to define a new cell type using `\newcell`.

The following cell types are predefined in `kuvio.tex`.

```
To
Mapsto
Into
Epi
Line
Bij
```

`Nul`	
`Two`	$\Longrightarrow$
`Bar`	$=\!=\!=$
`Eq`	$=$
`Null`	
`Dots`	....

The difference between a `Nul` cell and a `Null` cell is the value of their respective label pads. The former acts like a phantom `To` cell, the latter like a phantom `Two` cell. Phantom cells can also be realized using the `\nl` modifier. The type `One` can be used synonymously with `To`. The type `Impl` can be used synonymously with `Two`.

14. Another Font: arrsy10

Expanding the control sequence `\arrsy` causes the author's arrsy10 font to be loaded.† The following fills then become available.

`\Rightharpoonupfill`	$\Rightarrow$ (harpoon up)
`\Rightharpoondownfill`	$\Rightarrow$ (harpoon down)
`\Rrightarrowfill`	$\Rrightarrow$
`\equivfill`	$\equiv$
`\dashfill`	- - - - - - - - - - - - - - - -
`\rightdashfill`	- - - - - - - - - - - - - - - $\rightarrow$
`\rightepifill`	$\twoheadrightarrow$
`\rightmonofill`	$\rightarrowtail$
`\rightmonotailfill`	$\succ$———
`\rightisofill`	$\succ\!\!\twoheadrightarrow$
`\rightdashmonofill`	$\succ$- - - - - - - - - - - - - - $\rightarrow$
`\rightdashmonotailfill`	$\succ$- - - - - - - - - - - - - - -
`\rightdashepifill`	- - - - - - - - - - - - - - $\twoheadrightarrow$
`\rightdashisofill`	$\succ$- - - - - - - - - - - - - $\twoheadrightarrow$
`\squigglefill`	$\sim\sim\sim\sim$

The following cell types also become available.

`Allo`	$\Rightarrow$ (harpoon up)
`Para`	$\Rightarrow$ (harpoon down)
`Three`	$\Rrightarrow$
`Equiv`	$\equiv$
`Nulll`	
`Dash`	- - - - - - - - - - - - - - - -
`Dashto`	- - - - - - - - - - - - - - - $\rightarrow$
`Mono`	$\rightarrowtail$
`Tail`	$\succ$———
`Iso`	$\succ\!\!\twoheadrightarrow$

† Use of this font is entirely optional and `\arrsy` will result in an error if arrsy10 has not been installed. Also, users of LaTeX should use the `arrsy` option to the `kuvio` package in place of the control sequence `\arrsy`, as described in §23.

15. Fill Cells and Box Cells

Both fill and box macros can be used to specify arrows in a diagram without defining a new cell type. This (especially the former) may be useful for seldomly used arrows. The macros in question are used exactly as if we had defined cell types `Fillcell` and `Boxcell` except that the cell macros associated with these types accept a single argument, a fill or box macro.

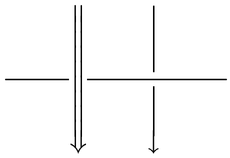
```
\Diagram
A & \rFillcell\lrightarrowfill ^f & B \\
\endDiagram
```

$$A \xrightarrow{f} B$$

16. Breaking Arrows

White rules can be used to break arrows in a more general way than is provided for by `\stop`. The modifier `\br` causes an arrow to be typeset on top of a white rule.

```
\Graph{3cm,3}{2cm,2}
\To   (2,2) (2,0)
\Line (0,1) (3,1) \br
\Two  (1,2) (1,0) \br
\endGraph
```



The order in which arrows are typeset here is significant: Cell macros are processed from left to right and top to bottom. The `\pp` modifier can be used to change the order in which alignment cells are typeset.

The modifier `\rl` causes a white rule to be typeset in place of an arrow.

Global and type-specific assignments made with `\breakpad` determine the width of the rule used with `\br` or `\rl`. The width used with an arrow of cell type *type* is twice the sum of an optional dimension passed as an argument to `\br` or `\rl` and the global and type-specific values of `\breakpad`.

The cell type `Rule` can also used to typeset rules as arrows in a diagram. In this case the `\rw` modifier can be used to set the width of the rule. The default width can be set using `\Rulewidth`.

17. Shades of Gray

The pair `\gr \endgr` can be used to typeset material in shades of gray.

```
\gr{.5}\vrule width 1cm height 12pt\endgr
```

The *graylevel* should lie between 0 (black) and 1 (white) inclusive. There are also pairs `\gray \endgray` for a graylevel which can be set using `\graygray` (the predefined value being .5), `\black \endblack` for graylevel 0 and `\white \endwhite` for graylevel 1. Arrows and modifications can be typeset in shades of gray using the first control sequence of one of the above pairs as a modifier.

```
\Diagram
A & \rTo \gray & B \\
\endDiagram
```

$$A \longrightarrow B$$

Shades of gray may print unsatisfactorily on some printers.

18. Frames and Shades

The pair `\frame \endframe` can be used to put a frame around the enclosed material. This material should not contain any vertical commands (unless enclosed in a box).

```
\input calc  % my calculus macros

Laplacian:\quad
\frame$\displaystyle
   \Delta f = \p\wrt{x^i}\left( g^{ki} \p f\wrt{x^k} \right)
              + g^{kj} \p f\wrt{x^k} \Gamma^i_{ij}
$\endframe
```

Laplacian: $\boxed{\displaystyle \Delta f = \frac{\partial}{\partial x^i}\left(g^{ki} \frac{\partial f}{\partial x^k} \right) + g^{kj} \frac{\partial f}{\partial x^k} \Gamma^i_{ij}}$

The thickness of the rule can be changed by making an assignment to `\framerulewidth` while the padding around the formula can be changed by making an assignment to `\framepad`. Padding can also be applied selectively to the sides of the material being framed using `\lpad`, `\rpad`, `\bpad` and `\tpad`. The graylevel of the frame can be changed using `\framegray`. One can put a frame around a diagram using `\framed`.

The pair `\shade \endshade` places a shadow behind a frame. The offsets of the bottom left and top right corners of the shadow must be specified as two coordinate pairs.

```
\framepad=1cm \shadegray=.5
\shade{10pt,-10pt}{5pt,-5pt}\vbox{\hsize=.8\hsize
``Om man l\"aser svenska k\"arleks visor fr\aa n 1600--talet \"ar det
sl\aa ende hur vanligt det \"ar att de \"alskande drar sig undan och m\"ots i
den gr\"ona naturen. F\"or de allra flesta var det endast d\"ar som man kunde
uppn\aa\ n\aa gon sann avskildhet; under denna epok var sex en
syssels\"attning som i h\"og grad idkades under \"oppen himmel.''\par
Peter Englund, {\it F\"orflutenhetens Landskap}.}\endshade
```

"Om man läser svenska kärleks visor från 1600–talet är det slående hur vanligt det är att de älskande drar sig undan och möts i den gröna naturen. För de allra flesta var det endast där som man kunde uppnå någon sann avskildhet; under denna epok var sex en sysselsättning som i hög grad idkades under öppen himmel."

Peter Englund, *Förflutenhetens Landskap*.

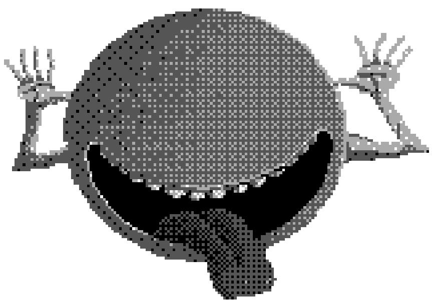
Frames and solid boxes can be typeset as modifications using `\Frame` and `\Box`.

19. Rotation and Reflection

There are several control sequence pairs in `kuvio.tex` for performing rotations and reflections. The material enclosed by these pairs is enclosed in an `\hbox` and should not contain any vertical commands unless enclosed in a `\vbox`.

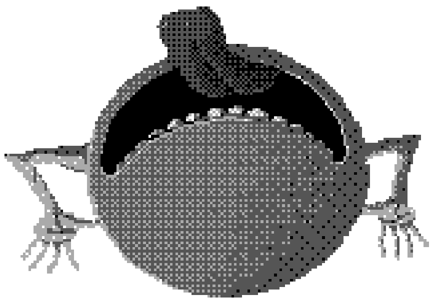
Let us play with the following `.eps` file.

```
\epsffile{face.eps}
```



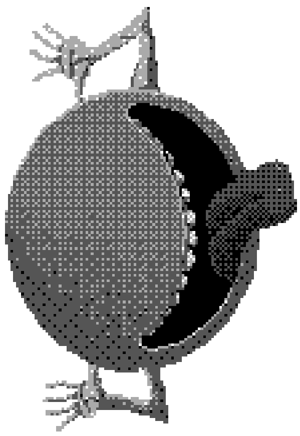
The pair `\flip` `\endflip` rotates material by 180 degrees.

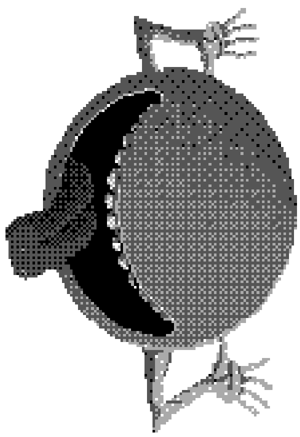
```
\flip\epsffile{face.eps}\endflip
```



The pair `\land` `\endland` rotates material by 90 degrees counterclockwise. Similarly, the pair `\opland` `\endopland` rotates material by 90 degrees clockwise.

```
\land\epsffile{face.eps}\endland \opland\epsffile{face.eps}\endopland
```





A diagram can be rotated using `\landscape` or `\oplandscape`.

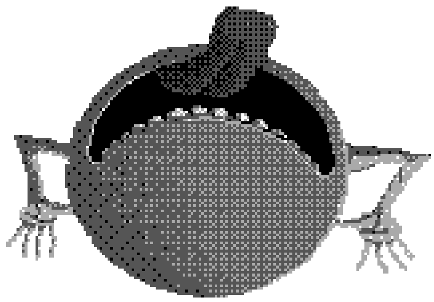
```
{\bf George W. Whitehead}, {\it Elements of Homotopy Theory}, page 592.
\bigskip
$$
\landscape \let\labelstyle\textstyle \cellpad=10pt
\Diagram
blah, blah, blah ...
\endDiagram
$$
```

George W. Whitehead, *Elements of Homotopy Theory*, page 592.

$$
\begin{tikzcd}
 & H_n(X^+ \cup \mathbf{T}A^+, X^+) \arrow[d, "H_n q_1"] & & H_n(\mathbf{T}A^+, A^+) \arrow[ll, "H_n(k_1)"'] \arrow[dll, "H_n(p_1)"] \arrow[ddr, "\partial_1"] & \\
 & H_n(\mathbf{S}A^+, P) & & * & \\
H_n(X^+ \cup \mathbf{T}A^+, P) \arrow[uur, "H_n(j_1)"] \arrow[ur, "H_n(q_4)"'] \arrow[dr, "H_n(q_2)"] \arrow[ddr, "H_n(j_2)"'] & & & H_{n-1}(A) \arrow[r, "H_{n-1}(i_A)"] & H_{n-1}(A^+, P) \\
 & H_n(X^+/A^+, P) \arrow[uu, "H_n(t)"] & H_n(X, A) \arrow[l, "P_n(X,A)"'] \arrow[ur] \arrow[dr, "H_n(i_X)"] & & \\
 & H_n(X^+ \cup \mathbf{T}A^+, \mathbf{T}A^+) \arrow[u, "H_n(q_3)"] & & H_n(X^+, A^+) \arrow[ll, "H_n(k_2)"] \arrow[ull, "H_n(p_X)"] \arrow[uur, "\partial_2"'] &
\end{tikzcd}
$$

Material can also be reflected across horizontal or vertical mirrors.

```
\hmir\epsffile{face.eps}\endhmir
```

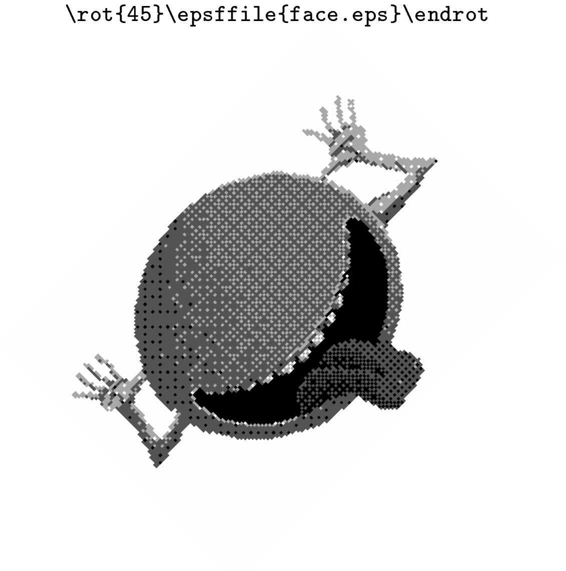


```
\vmir Who me, get carried away?!\endvmir
```

Who me, get carried away?!

One can also rotate material by an arbitrary angle using `\rot \endrot` but be aware that this pair does not modify the bounding box of the rotated material.

```
\rot{45}\epsffile{face.eps}\endrot
```



20. Compilation

Due to the large number of computations being performed, typesetting large diagrams can be rather time-consuming. The control sequence `\compileto` can be used to compile a diagram into a form which the macros in `kuvio.tex` can typeset more quickly. The information contained in a diagram is then written to two files having the extensions `.kdg` and `.kuv`.

```
\compileto{stable}

\Long
0            &      &            &      &              &      & 0          \\
\dEq         &      &            &      &              &      & \dEq       \\
\pi_{n+1} S^k & \rTo & \pi_n O(k) & \rTo & \pi_n O(k+1) & \rTo & \pi_n S^k \\
\endLong
```

$$\begin{array}{ccccccc} 0 & & & & & & 0 \\ \| & & & & & & \| \\ \pi_{n+1}S^k & \longrightarrow & \pi_n O(k) & \longrightarrow & \pi_n O(k+1) & \longrightarrow & \pi_n S^k \end{array}$$

Expansion of `\compileto{`*name*`}` causes the next diagram in the input (at a given level of grouping) to be compiled to, or read from, the files *name*`.kdg` and *name*`.kuv`. If these files already exist, then the text of the diagram is compared with the contents of *name*`.kdg` to determine whether or not the diagram should be recompiled or read from *name*`.kuv`.

To force the recompilation of a diagram, `\recompileto` can be used in place of `\compileto`. The control sequence `\recompile` can also be used to force recompilation at a given level of grouping. The control sequence `\nocompile` can be used to turn off compilation at a given level of grouping. Note that `\global\recompile` and `\global\nocompile` behave as expected.

Compilation can be automated using `\autocompileto`. Expansion of `\autocompileto{`*name*`}` allocates a count register, `\autocompilecounter`, and causes the tokens

```
\global\advance\autocompilecounter by 1
\compileto{name-\the\autocompilecounter}
```

to be expanded just prior to each expansion of `\everyDiagram`, `\everyFigure` or `\everyGraph`. Each subsequent expansion of `\autocompileto` resets `\autocompilecounter` to zero. Note that `\autocompilecounter` is incremented even if `\nocompile` has been specified.

The control sequence `\autocompile` is an abbreviation for `\autocompileto{.\jobname}`.

```
\input kuvio \autocompile
```

blah, blah, blah...

Be aware that compilation hard codes much information into `.kuv` files. While changes to diagram input are detected during future TEXings, changes to many parameters which affect the layout of components of a diagram are not. It may be necessary to use `\recompile` to have such changes propagate to compiled files. Note that whether or not `\gridlines` was specified when a diagram was compiled will be detected by the macros. Also, parameters and control sequences which affect only the box containing a diagram (`\Diagrampad`, `\lpad`, `\landscape`, `\deep`, etc) can be safely applied to a diagram which has already been compiled.

21. Making Definitions

TEX associates a category code with each character of input and once a category code has been assigned it cannot be changed. Unfortunately, since several category codes are changed between diagram delimiters like `\Diagram \endDiagram`, an ordinary `\def` having diagram macros in the replacement text may not behave as expected. In particular, the following characters require special consideration.

```
(    *    /    :    <    >    ^    _    |
```

Each of these should be preceded by a backslash when used as a modifier in a macro definition.

```
\def\ReallySimpleRightwardArrow#1#2#3{%
   \Dg #1 & \rTo \^{#2} & #3 \\ \endDg}

\ReallySimpleRightwardArrow{X^I}{e_0}X
```

$X^I \xrightarrow{e_0} X$

22. Name Clashes

Macros are loaded safely from `kuvio.tex`. If a name conflict is detected for an accessible control sequence (except for those having the string "kuvio" as part of their names), a message announces the conflict. The control sequence is then either made available through the control sequence `\kuviocs`, or it is redefined and the old definition is made available through the control sequence `\nonkuviocs`.

For example, LaTeX defines `\center`. This control sequence is not redefined by `kuvio.tex`: One must use `\kuviocs\center` to access the control sequence documented on page 31. Also, `plain.tex` defines `\land`. This control sequence *is* redefined by `kuvio.tex`: The version defined in `plain.tex` can be accessed as `\nonkuviocs\land`. (It is also available as `\wedge` since `plain.tex` make this equivalent to `\land`.)

If a control sequence has not been defined by `kuvio.tex`, then prefixing it by `\kuviocs` or `\nonkuviocs` does nothing.

The control sequence `\kuvioforce` can be used to force redefinition of a control sequence which `kuvio.tex` makes available through `\kuviocs`.

23. Using kuvio.tex with LaTeX

Users of LaTeX† can load `kuvio.tex` via the package file `kuvio.sty` using the `\usepackage{kuvio}` declaration. This package provides the option `arrsy` which should be used in place of the control sequence `\arrsy`. This option causes the LaTeX package file `arrsy.sty` to be loaded in place of its `plain` TeX counterpart `arrsy.tex`.

```
\usepackage[arrsy]{kuvio}
```

There are also options `autocompile`, `gridlines`, `kuviofmt`, `nocompile`, `overgrid` and `recompile` which can be used in place of the corresponding control sequences to affect global changes on a document, although none of these options offer any additional functionality.

Diagrams should still be input using the `\`*type* `\end`*type* syntax and *not* the LaTeX environment syntax, `\begin{`*type*`}` `\end{`*type*`}`. The latter may appear to work but will fail when diagrams are being compiled.

Several environments in the `amsmath` package of $\mathcal{AMS}$-LaTeX read their bodies as the argument of a delimited macro. This has the side effect of setting the category codes of all the characters in the body before its expansion. This is causes trouble for `kuvio.tex` since it wants to change several catcodes between a pair of diagram delimiters. To get around this the control sequence `\forcekdg` will cause diagrams at the same level of grouping that are not already being compiled to be written to and then read from the file `\jobname.kdg`, thereby circumventing any catcode assignments already made between a pair of diagram delimiters. This can be affected globally in LaTeX by loading the `kuvio` package with the `forcekdg` option.

`\forcekdg` is expanded automatically if the `amsmath` package is being used. If you don't want this, load `kuvio` with the `unforcekdg` option.

The LaTeX user should also aware of the following section when placing diagrams in an alignment-like environment.

† LaTeX 2_ε, not LaTeX 2.09.

24. Specifying Diagrams in Alignments

Due to the nature in which TEX processes alignments, it is an unfortunate necessity that a diagram specified inside an alignment (`\halign`, etc) be enclosed in a pair of braces.

```
$$
\eqalign{
  {\rm equalizer:}\qquad&   {\Dg X & \rTo & Y & \rTo \up{3pt}
                                                 \rTo \up{-3pt} & Z \\ \endDg}\cr
  \noalign{\medskip}
  {\rm coequalizer:}\qquad& {\Dg X & \rTo \up{3pt}
                                      \rTo \up{-3pt} & Y & \rTo & Z \\ \endDg}\cr
}
$$
```

$$\begin{aligned} \text{equalizer :}\qquad & X \longrightarrow Y \rightrightarrows Z \\ \text{coequalizer :}\qquad & X \rightrightarrows Y \longrightarrow Z \end{aligned}$$

The same consideration should be bourn in mind by LATEX and $\mathcal{AMS}$-LATEX users when using any `array`-like environment.

25. Summary

Brief descriptions (plus some examples) of the available control sequences and modifiers, including several not mentioned thus far. Modifiers are indicated by a • in the left margin. Default or predefined values, where appropriate, are listed in the right margin.

• `^{`*math*`}` `_{`*math*`}`
`<{`*math*`}` `>{`*math*`}`

Place labels on an arrow in `\labelstyle`.

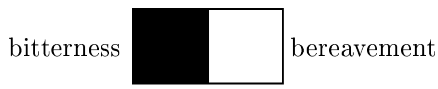
• `(`*integer*`,`*integer*`)`

Specify coordinates to an `\a`*type* or `\b`*type* cell macro.

• `(`*number*`,`*number*`)`

Specify coordinates to a modification. Both `\Box` and `\Frame` modifications require two sets of coordinates while others can accept either one or two sets.

```
\Graph{2cm,2}{1cm}
\Txt{bitterness} (2,.5) *{1.1}
\Txt{bereavement} (3,.5) (4,.5) :{1.1} *0
\Box (2,0) (3,1)  \Frame (3,1) (4,0)
\endGraph
```



For modifications on a Figure or Graph, a coordinate may be given as a dimension.

• `*{`*number*`}` .5

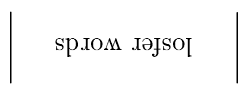
Set the tag point of a modification. Has no effect on type 2 modification cells, `\Box` modifications or `\Frame` modifications.

```
\Graph{1cm}{1cm}
\To (1,.5) *0   \Line (1,.5) *{1.5}
\Text{chaos} (1,.5) *{-2}
\endGraph
```

• `/{`*integer*`}` 0

Rotate a modification by *integer* degrees counterclockwise. Has no effect on type 2 modification cells, `\Box` modifications or `\Frame` modifications.

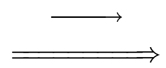
```
\Graph{3cm}{1cm}
\Line (0,.5) /{90}  \Line (1,.5) /{90}
\Txt{losfer words} (.5,.5) /{180}
\endGraph
```



- `|{`*dimen*`}`

  Set the length of a type 1 modification cell. See `\celllength`.

```
\Graph{2cm}{5mm}
\To  (.5,1)  \Two (.5,0) |{2cm}
\endGraph
```



- `:{`*number*`}` .5

  `:{`*dimen*`}` 0pt

  `:{`*number*`,`*dimen*`}` or `:{`*dimen*`,`*number*`}`

  Adjust the positioning of a label tied to an arrow, *number* determining a point along the line joining the endpoints of the arrow and *dimen* a point on a line perpendicular to the arrow.

```
\Diagram A & \rTo ^f :{.75} & B & \rTo ^g :{.75,5pt} & C & \rTo ^h :{5pt} & D \\
\endDiagram
```

$$A \xrightarrow{f} B \xrightarrow{g} C \xrightarrow{h} D$$

A semicolon can also be used as the argument separator. In fact, in this case we can replace the label pad *dimen* by a comma-separated pair of dimensions, the first moving a label parallel to the direction of an arrow and a second being a label pad. That is, the forms `:{`*dimen*`,`*dimen*`}`, `:{`*number*`;`*dimen*`,`*dimen*`}` and `:{`*dimen*`,`*dimen*`;`*number*`}` are also legal.

```
\Diagram A & \rTo ^f :{-10pt,0pt} & B & \rTo ^g :{0;10pt,0pt} & C \\
\endDiagram
```

$$A \xrightarrow{f} B \xrightarrow{g} C$$

Label positioning with `:` applies to all labels placed on an arrow. For any individual label, this can be overridden by supplying a positioning specification to `^`, `_`, `<` or `>`, delimited by square braces. This may be useful for cell types whose arrows are composed of many pieces, although the same results can be obtained using null arrows (see `\nl`).

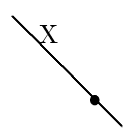
```
\newcell{Adj}
\def\Adjbox#1{\vcenter{\offinterlineskip\dimen0=#1
   \hbox to #1{\leftarrowfill\hskip.2\dimen0} \hbox to #1{\hskip.2\dimen0\rightarrowfill}}}
\labelpad{Adj}=2.5pt

\Diagram A & \rAdj ^[.6]{f_1} _[.4]{f_2} & B \\
\endDiagram
```

$$A \underset{f_2}{\overset{f_1}{\leftrightarrows}} B$$

- `:{`*number*`}` .5

  For `\Label`, `\Math`, `\Text`, `\Txt` and `\Vertex` modifications, specify a coordinate as a fractional distance from one point to another.

```
\Graph{15mm}{15mm}
\Line (0,1) (1,0)
\Text{X} (0,1) (1,0) :{.25} *0
\Math\bullet (0,1) (1,0) :{.75}
\endGraph
```



`\al`

Move the column in which this control sequence occurs. Like `\dl` except that, instead of also moving columns to the right of the given column, recursively moves those which are *bound* to it.

A column is bound to another in the following situations.

(1) If an arrow goes between two entries in columns $C_\ell < C_r$ and both columns are on or to the left of the gravitating column then C_ℓ is bound to C_r.

(2) If two entries in the same row of columns $C_\ell < C_r$ are adjacent (in the sense that these entries typeset material with non-zero dimensions and there are no other such entries in columns strictly between C_ℓ and C_r) and both columns are on or to the left of the `\gravitating column` then C_ℓ is bound to C_r.

(3) If an arrow goes between two entries in columns $C_\ell < C_r$ and both columns are on or to the right of the gravitating column then C_r is bound to C_ℓ.

(4) If two entries in the same row of columns $C_\ell < C_r$ are adjacent and both columns are on or to the right of the gravitating column then C_r is bound to C_ℓ.

The control sequence `\displaybindings` causes a message of the form `(B:`C-C'`)` to be displayed whenever column C' is bound to column C.

`\ar`

Move the column in which this control sequence occurs. Like `\dr` except that, instead of also moving columns to the left of the given column, recursively moves those which are bound to it. See `\al`.

`\arrsy`

Load the font `arrsy10` and define several control sequences and cell types using this font. Of course, the font has to have been installed on your system for this to work.

```
\input kuvio \arrsy  % plain TeX          \usepackage[arrsy]{kuvio}  % LaTeX
```

blah, blah, blah...

`\atpush=`*dimen* 3pt
`\atpush+=`*dimen*
`\atpush{`*type*`}=`*dimen*   `\atpush{`*type*`}+=`*dimen*

Global and type-specific amounts by which the end of an arrow attached to another arrow (using `\pt`) is shortened whenever a vertex has not been attached using `\hd` or `\tl`. The type-specific quantity applies to ends attached to an arrow of cell type *type*. Spaces in *type* are ignored.

`\autocompile`

Abbreviation for `\autocompileto{.\jobname}`.

`\autocompilecounter`

Count register used with `\autocompileto`.

`\autocompileto{`*name*`}`

Cause all diagrams to be compiled unless otherwise specified using `\nocompile`. The files *name*-`\the\autocompilecounter.kdg` and *name*-`\the\autocompilecounter.kuv` are either written or read, the count register `\autocompilecounter` being incremented with each diagram being compiled. Spaced in *name* are ignored.

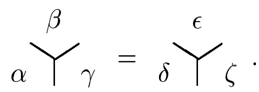
`\ax{`*dimen*`}`

Move the column in which this control sequence occurs rightward by *dimen*. Like `\dx` except that, instead of also moving columns to the right of the given column, recursively moves those which are bound to it. See `\al`.

`\base`

Make the baseline of a Diagram coincide with that of a row in its alignment. Takes precedence over any baseline specification given outside the alignment (using `\tall`, `\deep`, etc).

```
\newDiagram{Ydiag}
\def\everyYdiag{\flexible \xgrid=0pt \ygrid=4pt \joined \columndist=9pt}

\def\Y#1#2#3{%
   \Ydiag
    &         & #2                          \\
                     \\                     \\
    & \rdLine &                & \ldLine &  \\
    & \base   & \stop\my{2pt}               \\
    & #1      & \qquad\dLine   & #3         \\
                                            \\
   \endYdiag}

Thus, $\Y\alpha\beta\gamma = \Y\delta\epsilon\zeta$.
```

Thus, 

`\black`

Typeset an arrow or modification with graylevel 0.

`\black \endblack`

Typeset the enclosed material with graylevel 0.

`\Box`

Typeset a box as a modification. The coordinates of two diagonally opposite corners must be specified.

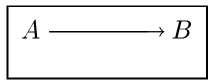
`\Boxcell{`*box macro*`}`

Typeset an arrow as a modification without naming a cell type. Possibly the most useless control sequence of the lot! Has ten equally useless siblings for use in the alignment part of a Diagram: `\aBoxcell`, `\bBoxcell`, `\rBoxcell`, `\rdBoxcell`, `\dBoxcell`, `\ldBoxcell`, `\lBoxcell`, `\luBoxcell`, `\uBoxcell` and `\ruBoxcell`.

\bpad=*dimen* 0pt
\bpad+=*dimen*

Padding added to the bottom of a diagram or framed material. Does not alter the baseline.

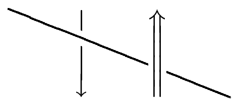
```
\framed \bpad=10pt
\Diagram
A & \rTo & B \\
\endDiagram
```



• \br\br{*dimen*}

Typeset an arrow on top of a white rule.

```
\Graph{3cm,3}{12mm,2}
\To (1,2) (1,0)
\Line (0,2) (3,0) \br  \Two (2,0) (2,2) \br
\endGraph
```



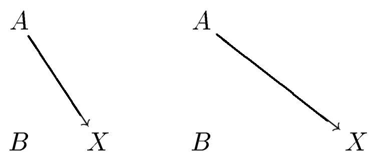
\braced

With flexible grids, make all diagonal arrows fully braced by default and invert the action of `\db`: An arrow will be unbraced when `\db` is specified.

\bracewidth=*dimen* 1cm
\bracewidth+=*dimen*

With flexible grids, width of the *brace* used with braced diagonal arrows.

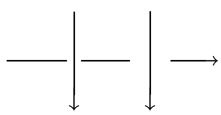
```
\braced
\Dg
A \\ & \rdTo \\ B & & X \\
\endDg
\qquad\bracewidth+=1cm
\Dg
A \\ & \rdTo \\ B & & X \\
\endDg
```



\breakpad=*dimen* 2.5pt
\breakpad+=*dimen*
\breakpad{*type*}=*dimen* \breakpad{*type*}+=*dimen*

Global and type-specific pads to determine the rule width used with `\br` and `\rl`. Spaces in *type* are ignored.

```
\breakpad{One}=5pt \yscale=.75
\Diagram
\\ & \dTo \br & \rTo \pp \dOne \br & \\ \\
\endDiagram
```



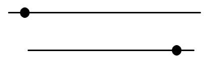
`\celllength=`*dimen* 1cm
`\celllength+=`*dimen*

Default length of a type 1.

`\cellpush=`*dimen* 2pt
`\cellpush+=`*dimen*
`\cellpush{`*type*`}=`*dimen* `\cellpush{`*type*`}+=`*dimen*

Global and type-specific cell pushes. Spaces in *type* are ignored.

```
\cellpush=-10pt
\Graph{2cm}{5mm,2}
\Line (0,2) (1,2) \tl{\bullet} \hd{}
\Line (0,0) (1,0)              \hd{\bullet}
\endGraph
```



Note the use of `\tl` and `\hd` in this example. Cell pushes are applied to a modification cell only when a vertex is attached using one of these two.

`\cellwidth=`*dimen* 1cm
`\cellwidth+=`*dimen*

With flexible grids, the minimum distance between vertices at the ends of horizontal alignment cells.

```
\Dg A & \rTo & B & {} = C \\ \endDg
\qquad\cellwidth+=1cm
\Dg A & \rTo & B & {} = C \\ \endDg
```

$A \longrightarrow B = C \qquad A \xrightarrow{\hspace{3em}} B = C$

`\center` or `\centre`

Center diagrams on the math axis. With the exception of Diagrams consisting of a single row, this is the default in math modes.

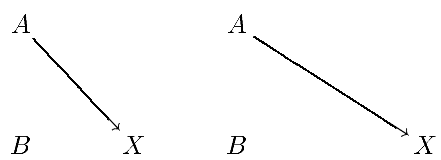
`\centermath` or `\centremath`

Center diagrams in math modes only. This is the default behaviour.

`\columndist=`*dimen* 15mm
`\columndist+=`*dimen*

With flexible grids, the minimum distance between the centers of columns at either end of an unbraced diagonal arrow.

```
\Dg
A \\ & \rdTo \\ B & & X \\
\endDg
\qquad\columndist+=1cm
\Dg
A \\ & \rdTo \\ B & & X \\
\endDg
```



`\compileto{`*name*`}`

Compile a diagram to, or read a diagram from, the files *name*`.kdg` and *name*`.kuv`. Spaces in *name* are ignored.

• `\db`

Make a diagonal arrow fully braced unless `\braced` has been specified, in which case the arrow is made unbraced.

`\deep`

Make the baseline of a Diagram coincide with that of its topmost row and the baseline of a Figure or Graph coincide with that of its upper edge. This is the default for Diagrams in non-math modes.

`Dg`

Diagram type having `\flexible`, `\xgrid=0pt`, `\ygrid+=-2mm`, `\cellwidth+=3mm` and `\bracewidth+=-2.5mm`.

```
\Dg
\eta_k:\qquad & {\bb R} & \rInto & X_k & {} =
  \{\,(A,\vect v)\in W_k\times{\bb R}^n \mid A\vect v = \lambda_k\vect v\,\} \\
              &         &        & \dTo \\  % \vect is from arrsy10
              &         &        & W_k  \\
\endDg
```

$$\begin{tikzcd} \eta_k : & \mathbb{R} \arrow[r, hook] & X_k \arrow[d] & = \{\,(A,\vec{v}) \in W_k \times \mathbb{R}^n \mid A\vec{v} = \lambda_k \vec{v}\,\} \\ & & W_k & \end{tikzcd}$$

• `\dh{`*dimen*`}`

Fudge the length of an arrow at its head.

```
\Diagram
A & \rTo \dh{-10pt} & B \\
\endDiagram
```

$$A \longrightarrow \quad B$$

• `\dh{`*dimen* or *number* , *dimen*`}`

Fudge the positioning of an arrow at its head. Equivalent to `\hx{`*dimen* or *number*`}` `\hy{`*dimen*`}`.

• `\dh{`*dimen*1 or *number* , *dimen*2 ; *dimen*3`}`
`\dh{`*dimen*3 ; *dimen*1 or *number* , *dimen*2`}`

Fudge both the length and positioning at the head of an arrow. Equivalent to `\hx{`*dimen*1 or *number*`}` `\hy{`*dimen*2`}` `\dh{`*dimen*3`}`.

`Diag`

Diagram type having `\flexible` and `\xgrid=0pt`.

```
\ygrid=7mm
\Diag
\cdots& \rTo& H_n A& \rTo& H_n X& \rTo& H_n(X,A)& \rTo& H_{n-1}A& \rTo& \cdots \\
      &     & \dTo &     & \dTo &     & \dTo    &     & \dTo                   \\
\cdots& \rTo& H_n B& \rTo& H_n Y& \rTo& H_n(Y,B)& \rTo& H_{n-1}B& \rTo& \cdots \\
\endDiag
```

$$\begin{tikzcd}
\cdots \arrow[r] & H_nA \arrow[r] \arrow[d] & H_nX \arrow[r] \arrow[d] & H_n(X,A) \arrow[r] \arrow[d] & H_{n-1}A \arrow[r] \arrow[d] & \cdots \\
\cdots \arrow[r] & H_nB \arrow[r] & H_nY \arrow[r] & H_n(Y,B) \arrow[r] & H_{n-1}B \arrow[r] & \cdots
\end{tikzcd}$$

`\Diagram \endDiagram`

Diagram delimiters. Uses a rigid grid unless `\flexible` is specified.

`\diagramdot` — `\bullet`

Token used with `\dotted` and `\Dot`.

```
\let\diagramdot\circ \dotted
\Diagram
 & \\
   \\
\endDiagram
```

$$\begin{array}{cc} \circ & \circ \\ \\ \circ & \circ \end{array}$$

`\Diagrampad=`*dimen* — `5pt`
`\Diagrampad+=`*dimen*

Padding added to all sides of a Diagram.

```
\framed
\Diagram A & \rTo & B \\ \endDiagram
\Diagrampad=10pt \quad
\Diagram A & \rTo & B \\ \endDiagram
```

$\boxed{A \longrightarrow B} \quad \boxed{\;\; A \longrightarrow B \;\;}$

`\displayall`

Abbreviation for `\displaybindings` `\displaymovements` `\displaystretches`, each of which announce certain computations.

`\Displaybox`

Control sequence, taking a single parameter (a box register), which displays a diagram.

\dl

Move the column in which this control sequence occurs so that the alignment entry in which it occurs abuts it nearest neighbour to the left. Additionally, move each column to the right of the given column by the same amount. Many of the same effects can be obtained simply by using a flexible grid with `\xgrid=0pt`.

\Dot

Abbreviation for `\Math{\diagramdot}`.

\dotted

Place a `\diagramdot` in every empty alignment entry in a Diagram unless prevented from doing so by `\nodot`.

```
\dotted
\Diagram
  & \nodot \\
A          \\
\endDiagram
```

$$\begin{array}{cc} \bullet & \\ A & \bullet \end{array}$$

\dr

Move the column in which this control sequence occurs so that the alignment entry in which it occurs abuts it nearest neighbour to the right. Additionally, move each column to the left of the given column by the same amount. Many of the same effects can be obtained simply by using a flexible grid with `\xgrid=0pt`.

• \dt{*dimen*}

Fudge the length of an arrow at its tail.

```
\Diagram
A & \rTo \dt{-10pt} & B \\
\endDiagram
```

$$A \quad \longrightarrow B$$

• \dt{*dimen*1 or *number* , *dimen*2}

Fudge the positioning of an arrow at its tail. Equivalent to `\tx{`*dimen*1 or *number*`}` `\ty{`*dimen*2`}`.

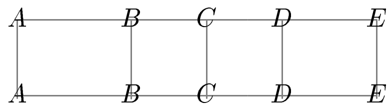
• \dt{*dimen*1 or *number* , *dimen*2 ; *dimen*3}
\dt{*dimen*3 ; *dimen*1 or *number* , *dimen*2}

Fudge both the length and positioning at the tail of an arrow. Equivalent to `\tx{`*dimen*1 or *number*`}` `\ty{`*dimen*2`}` `\dt{`*dimen*3`}`.

\dx{*number* or *dimen*}

Move the column in which this control sequence occurs. `\dx{`*dimen*`}` moves the column rightward by *dimen* and also moves all columns to the right of the given column by the same amount. If the distance to the next column (to the right) is currently d, `\dx{`*number*`}` moves the column rightward by $number \cdot d$ and also moves all columns to the left of the given column by the same amount.

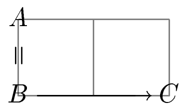
```
\gridlines
\Diagram
A & B \dx{5mm} & C & D            & E \\
A & B          & C & D \dx{-.25} & E \\
\endDiagram
```



\dy{*number* or *dimen*}

Move the row in which this control sequence occurs. `\dy{`*dimen*`}` moves the row upward by *dimen* and also moves all rows above the given row by the same amount. If the distance to the next row (above) is currently d, `\dy{`*number*`}` moves the row upward by $number \cdot d$ and also moves all rows below the given row by the same amount.

```
\gridlines
\Diagram
A                  \\
\dEq & \dy{1}      \\
B    & \rTo    & C \\
\endDiagram
```



\epi

An epimorphism arrow, as in $A \twoheadrightarrow B$. The `arrsy10` font must be loaded for this to work. See `\arrsy`.

• \fd{*fudge*}

Abbreviation for `\dt{`*fudge*`}` `\dh{`*fudge*`}`.

\Figure \endFigure

Figure delimiters.

\Figurepad=*dimen* 0pt
\Figurepad+=*dimen*

Padding added to all sides of a Figure.

\Fillcell{*fill macro*}

Like, `\Boxcell`, typeset an arrow as a modification without naming a cell type. Has ten siblings for use in the alignment part of a Diagram: `\aFillcell`, `\bFillcell`, `\rFillcell`, `\rdFillcell`, `\dFillcell`, `\ldFillcell`, `\lFillcell`, `\luFillcell`, `\uFillcell` and `\ruFillcell`.

```
\Diagram
A & \rFillcell\lrrrightarrowfill & B \\
\Modify
\Fillcell\rrightarrowxfill (3,0) /{90}
\endDiagram
```

$A \longrightarrow B$

`\flexible`

Enable the flexible grid.

`\flip \endflip`

Rotate the enclosed material by 180 degrees. The material cannot contain vertical commands (unless enclosed in a box).

`\forcekdg`

Force diagrams to be read from `\jobname.kdg` if not otherwise being compiled. See §24.

• `\fr{`*dimen*`}`

Abbreviation for `\tr{`*dimen*`}` `\hr{`*dimen*`}`.

`\Frame`

Typeset a frame as a modification. The coordinates of two diagonally opposite corners must be specified.

`\frame \endframe`

Put a frame around the enclosed material. The material cannot contain vertical commands (unless enclosed in a box).

```
\frame\bf N\"ara skjuter ingen hare.\endframe
```

Nära skjuter ingen hare.

`\framed`

Place diagrams in a frame.

```
\framed
\Long
H_n X\otimes G & \rMono & H_n(X;G) & \rEpi & {\rm Tor}(H_{n-1}X,G) \\
\endLong
```

$$\boxed{H_nX \otimes G \rightarrowtail H_n(X;G) \twoheadrightarrow \mathrm{Tor}(H_{n-1}X,G)}$$

`\framegray=`*number* or `\framegrey=`*number* 0

Set the graylevel of the frame used with `\frame \endframe`.

\framepad=*dimen* 5pt
\framepad+=*dimen*

Padding around material framed with `\frame \endframe`.

```
\frame \TeX \endframe
\framepad=10pt \frame \TeX \endframe
```

$\boxed{\TeX}\boxed{\;\TeX\;}$

\Framerulewidth=*dimen* .4pt
\Framerulewidth+=*dimen*

Default thickness of the rule used when typesetting `\Frame` modifications.

\framerulewidth=*dimen* .4pt
\framerulewidth+=*dimen*

Thickness of the rule used when typesetting frames with `\framed` or `\frame \endframe`.

```
\framerulewidth=2mm
\frame Which way to the home world?\endframe
```

$\boxed{\text{Which way to the home world?}}$

- ### \fs{*dimen* or *number*}

Abbreviation for `\ts{`*dimen* or *number*`}` `\hs{`*dimen* or *number*`}`.

The following trick with `\fs` can be used to typeset an arrow which is parallel to another but offset by some distance.

```
\Dg
M &       & \rTo ^\theta  &       & \beta^{-1}(W) \\
  & \rdTo &               & \ldTo               \\
  &       & \beta^{-1}(F)                       \\
\Modify
\Mapsto (0,2) (2,0) \fs{-50pt} \tl{x} \hd{R(x)}
\endDg
```

$$\begin{tikzcd}
x \arrow[dr, mapsto] & & M \arrow[rr, "\theta"] \arrow[dr] & & \beta^{-1}(W) \arrow[dl] \\
& R(x) & & \beta^{-1}(F) &
\end{tikzcd}$$

- ### \fx{*dimen* or *number*}

Abbreviation for `\tx{`*dimen* or *number*`}` `\hx{`*dimen* or *number*`}`.

- ### \fy{*dimen*}

Abbreviation for `\ty{`*dimen*`}` `\hy{`*dimen*`}`.

• \gr{*number*}

Typeset an arrow or modification with the specified graylevel. Probably none too pretty on many an output device.

```
\Graph{1cm}{1cm}
\Box (0,0) (1,1) \gr{.25}
\endGraph
```

\gr{*number*} \endgr

Typeset the enclosed material with the specified graylevel.

```
\gr{.25}Anyone seen my bat'leth?\endgr
```

Anyone seen my bat'leth?

\Graph \endGraph
\Graph{*dimen*}{*dimen*} \endGraph
\Graph{*dimen*, *dimen* or *integer*}{*dimen*} \endGraph
\Graph{*dimen*}{*dimen*, *dimen* or *integer*} \endGraph
\Graph{*dimen*, *dimen* or *integer*}{*dimen*, *dimen* or *integer*} \endGraph

Graph delimiters.

\Graphpad=*dimen* 0pt
\Graphpad+=*dimen*

Padding added to all sides of a Graph.

\grav

Explicitly set the gravitating column. Columns at the ends of an arrow which passes through the gravitating column will tend to be centered on it.

```
\loose
\Dg
p^{-1}(U) &           & \rTo    &       & U\times F & {} \approx U\times {\bb R}P^2 \\
          & \rdTo <p &         & \ldTo >{\pi_1}    \\
          &           & U \grav &       &           \\
\endDg
```

$$\begin{tikzcd} p^{-1}(U) \arrow[rr] \arrow[dr, "p"'] & & U \times F \approx U \times \mathbb{R}P^2 \arrow[dl, "\pi_1"] \\ & U & \end{tikzcd}$$

\gravitateleft

Set the default location of the gravitating column to the first column of an alignment.

`\gravitateright`

Set the default location of the gravitating column to the last column of an alignment.

• `\gray` or `\grey`

Typeset an arrow or modification with the graylevel set by `\graygray`.

`\gray \endgray` or `\grey \endgrey`

Typeset the enclosed material with the graylevel set by `\graygray`.

`\graygray=`*number* or `\greygrey=`*number* .5

Set the graylevel used with `\gray`.

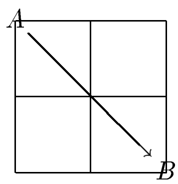
`\grid=`*dimen*
`\grid+=`*dimen*

Abbreviation for `\xgrid=`*dimen* `\ygrid=`*dimen* or `\xgrid+=`*dimen* `\ygrid+=`*dimen*.

`\gridgray=`*number* or `\gridgrey=`*number* .5

Set the graylevel used with `\gridlines`.

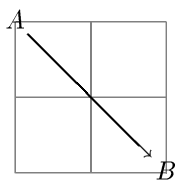
```
\gridlines \gridgray=0
\Diagram
A              \\
  & \rdTo      \\
  &       & B \\
\endDiagram
```



`\gridlines`

Display gridlines.

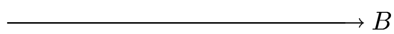
```
\gridlines
\Diagram
A              \\
  & \rdTo      \\
  &       & B \\
\endDiagram
```



• `\hd{`*math*`}`

Attach math material to the head of a type 2 modification cell.

```
\Graph{5cm}{1cm}
\To (0,.5) (1,.5) \hd{B}
\endGraph
```



`\hmir` `\endhmir`

Reflect the enclosed material across a horizontal mirror lying on the baseline. The material cannot contain vertical commands (unless enclosed in a box).

`\hpad=`*dimen*
`\hpad+=`*dimen*

Abbreviation for `\lpad=`*dimen*`\rpad=`*dimen* or `\lpad+=`*dimen*`\rpad+=`*dimen*.

• `\hr{`*dimen*`}`

Raise the vertex attached to the head of a type 2 modification cell.

• `\hs{`*dimen* or *number*`}`

Slide the vertex attached to the head of a type 2 modification cell horizontally.

• `\hx{`*dimen* or *number*`}`

Fudge the positioning of an arrow at its head. Makes the head of an arrow point to a point which is offset horizontally from the center of the math axis of the head vertex (*number* $= 0 \leftrightarrow$ left edge of head vertex, *number* $= 1 \leftrightarrow$ right edge of head vertex). Has no effect on type 1 modification cells.

```
\Diagram
X                          \\
  & \rdTo \hx{.2}          \\
  &               & X\times X \\
\endDiagram
```

$$\begin{tikzcd} X \arrow[ddrr] & & \\ & & \\ & & X \times X \end{tikzcd}$$

• `\hy{`*dimen*`}`

Fudge the positioning of an arrow at its head. Makes the head of an arrow point to a point which is offset vertically from the math axis of the head vertex. Has no effect on type 1 modification cells.

`\into`

Abbreviation for `\lhook\joinrel\rightarrow`, as in $A \hookrightarrow X$.

• `\jh`

Join the head of an arrow to the point at the coordinate of its head vertex by applying `\joinpush` rather than `\cellpush` (or `\ptpush` and `\atpush`) and ignoring the dimensions of the vertex.

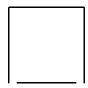
```
\Diagram
\bullet & \rLine \jh & \bullet \\
\endDiagram
```

$$\begin{tikzcd} \bullet \arrow[r, dash] & \bullet \end{tikzcd}$$

\joined

Invert the action of the `\jh`, `\jn` and `\jt` modifiers: Ends of arrows are joined unless one of these is specified.

```
\joined \scale=.5
\Diagram
       & \rLine     &        \\
\dLine &            & \dLine \\
       & \rLine \jn &        \\
\endDiagram
```



\joinpush=*dimen* -1pt

\joinpush+=*dimen*

\joinpush{*type*}=*dimen* \joinpush{*type*}+=*dimen*

Global and type specific pushes used with `\jt`, `\jh` and `\jn`. Spaces in *type* are ignored.

• \jn

Abbreviation for `\jt \jh`.

• \jt

Join the tail of an arrow to the point at the coordinate of its tail vertex.

```
\Diagram
\bullet & \rLine \jt & \bullet \\
\endDiagram
```


\kuvio

Abbreviation for `{\tt kuvio.tex}`.

\kuviocs*name*

Access a control sequence which `kuvio.tex` makes available but does not define explicitly due to a name conflict with an existing control sequence.

\kuviodate

Expands to the date of the current patchlevel of `kuvio.tex`.

\kuviofmt

Load all macros. This should be expanded when using `kuvio.tex` to build a format file.

\kuvioforce*name*

Force redefinition of *name* when `kuvio.tex` makes it available only through `\kuviocs`.

`\kuviopatchlevel`

Expands to the current patchlevel of `kuvio.tex`.

`\kuviorequire{`*integer*`}`

A warning of forthcoming disaster will be issued if `\kuviopatchlevel` is not at least *integer*.

`\kuviorevision{`*text*`}`

With compilation, cause the string *text* to be written to `.kuv` files. A diagram is recompiled if this string doesn't match the one found in its `.kuv` file.

`\kuviospecial\`*name*

Set the `\special` syntax. Not a good thing to do unless you know what you're doing. The following two lines in `kuvio.tex` set the syntax for `dvips`.

```
\def\dvipsspecial#1{\special{ps:#1}}
\kuviospecial\dvipsspecial
```

Experimentation with other `dvi` to PostScript translators is possible using `\kuviospecial`.

`\Label{`*math*`}`

Typeset math material as a modification in `\labelstyle`.

`\labelpad=`*dimen* 3pt
`\labelpad+=`*dimen*
`\labelpad{`*type*`}=`*dimen* `\labelpad{`*type*`}+=`*dimen*

Global and type-specific label pads. Spaces in *type* are ignored.

```
\labelpad{One}=5pt
\Diagram A & \rTo ^f & B \\ \endDiagram
\quad
\Diagram A & \rOne ^f & B \\ \endDiagram
```

$A \xrightarrow{f} B \qquad A \xrightarrow{f} B$

`\labelpoint=`*number* .5
`\labelpoint{`*type*`}=`*number*

Set the default location of labels tied to an arrow. Spaces in *type* are ignored.

```
\labelpoint{One}=.75
\Diagram A & \rTo ^f & B \\ \endDiagram
\quad
\Diagram A & \rOne ^f & B \\ \endDiagram
```

$A \xrightarrow{f} B \qquad A \xrightarrow{f} B$

Once a type-specific default has been set it can be unset, causing the global default to be used, by making an empty assignment of the form `\labelpoint{`*type*`}={}`. All predefined cell types in `kuvio.tex` have their `\labelpoint` unset.

`\labelstyle` `\scriptstyle`

Style in which labels of a diagram are typeset.

```
\let\labelstyle\textstyle
\Diagram
A & \rTo ^f & B \\
\endDiagram
```

$$A \xrightarrow{f} B$$

`\labelwidthpad=`*dimen* 5pt
`\labelwidthpad+=`*dimen*
`\labelwidthpad{`*type*`}=`*dimen* `\labelwidthpad{`*type*`}+=`*dimen*

With a flexible grid, `kuvio.tex` will stretch a horizontal arrow (specified using a `\l`*type* or `\r`*type* cell macro) so that a label centered along its length isn't wider than the arrow is long. The value of `\labelwidthpad` determines the minimum allowable distance between the edges of the label and the ends of the arrow. Note that the comparison of the arrow length and label width occurs before the application of cell pushes specified using `\dt` and `\dh`.

Spaces in *type* are ignored.

`\labelwidthpad` has no effect on labels which are not centered and no attempt is made to ensure that these "fit" on the arrow to which they are attached. Also, `\labelwidthpad` may not work well with arrows that are attached to other arrows and are thus also affected by `\atpush` and `\ptpush`. See also `\lw`.

```
\let\labelstyle\textstyle
\labelwidthpad{One}+=5pt
\leavevmode
\Dg
X(s)\times |u| & & \rTo ^{(f_{s\to t},\pi_2)}  &  & X(t)\times |u| \\
        & \rdTo <{f_{s\to u}} &      & \ldTo >{f_{t\to u}}         \\
        &                     & X(u)                               \\
\endDg\qquad
\Dg
X(s)\times |u| & & \rOne ^{(f_{s\to t},\pi_2)} &  & X(t)\times |u| \\
        & \rdTo <{f_{s\to u}} &      & \ldTo >{f_{t\to u}}         \\
        &                     & X(u)                               \\
\endDg
```

$$\begin{tikzcd} X(s)\times|u| \arrow[rr, "{(f_{s\to t},\pi_2)}"] \arrow[dr, "f_{s\to u}"'] & & X(t)\times|u| \arrow[dl, "f_{t\to u}"] \\ & X(u) & \end{tikzcd} \qquad \begin{tikzcd} X(s)\times|u| \arrow[rr, "{(f_{s\to t},\pi_2)}"] \arrow[dr, "f_{s\to u}"'] & & X(t)\times|u| \arrow[dl, "f_{t\to u}"] \\ & X(u) & \end{tikzcd}$$

`\land` `\endland`

Rotate the enclosed material by 90 degrees counterclockwise. The material cannot contain vertical commands (unless enclosed in a box).

`\landscape`

Rotate a diagram 90 degrees counterclockwise.

\latexTo

Load the font `line10`, define `\sleftarrowfill` and `\srightarrowfill` using the arrowhead character from this font and do `\let\Tofill\srightarrowfill`.

```
\latexTo
\Dg
A & \rTo & B & \rOne & C \\
\endDg
```

$$A \longrightarrow B \longrightarrow C$$

\lb

Make an arrow left braced.

Long

Diagram type having `\flexible`, `\xgrid=0pt` and `\ygrid+=-5mm`.

```
\Long
0          &      &               &      &                   &      & 0             \\
\dEq       &      &               &      &                   &      & \dEq          \\
\pi_j S^k & \rTo & \pi_j V(n,k) & \rTo & \pi_j V(n-1,k+1) & \rTo & \pi_{j-1} S^k \\
\endLong
```

$$\begin{tikzcd}
0 \arrow[d, equal] & & & 0 \arrow[d, equal] \\
\pi_j S^k \arrow[r] & \pi_j V(n,k) \arrow[r] & \pi_j V(n-1,k+1) \arrow[r] & \pi_{j-1} S^k
\end{tikzcd}$$

\loose

Make all diagonal alignment cells unbraced by default. Additionally, sets `\columndist=0pt`.

\lpad=*dimen* 0pt
\lpad+=*dimen*

Padding added to the left of a diagram or framed material.

```
\framed \lpad=10pt
\Diagram
A & \rTo & B \\
\endDiagram
```

$$\boxed{\quad A \longrightarrow B}$$

\lw or \lw{*dimen*}

In the first case, don't stretch a horizontal arrow even if a centered label is wider than the arrow is long. In the second case, increase the effective width of the label by twice *dimen*. See `\labelwidthpad`.

\Math{*math*}

Typeset math material as a modification in `\textstyle`.

`\ml`

Move the column in which this control sequence occurs so that the alignment entry in which it occurs abuts it nearest neighbour to the left. Many of the same effects can be obtained simply by using a flexible grid with `\xgrid=0pt`.

`\Modify`

Modifications are specified following an occurrence of this control sequence.

`\mono`

A monomorphism arrow, as in $A \rightarrowtail B$. The `arrsy10` font must be loaded for this to work. See `\arrsy`.

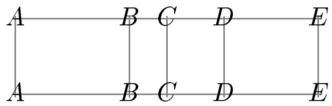
`\mr`

Move the column in which this control sequence occurs so that the alignment entry in which it occurs abuts it nearest neighbour to the right. Many of the same effects can be obtained simply by using a flexible grid with `\xgrid=0pt`.

`\mx{`*number* or *dimen*`}`

Move the column in which this control sequence occurs. `\mx{`*dimen*`}` moves the column rightward by *dimen*. If the distance to the next column (to the right) is currently d, `\mx{`*number*`}` moves the column rightward by $number \cdot d$.

```
\gridlines
\Diagram
A & B \mx{5mm} & C & D           & E \\
A & B          & C & D \mx{-.25} & E \\
\endDiagram
```



`\my{`*number* or *dimen*`}`

Move the row in which this control sequence occurs. `\my{`*dimen*`}` moves the row upward by *dimen*. If the distance to the next row (above) is currently d, `\my{`*number*`}` moves the row upward by $number \cdot d$.

• `\mv{`*dimen*1`,`*dimen*2`}`

Abbreviation for `\rt{`*dimen*1`}` `\up{`*dimen*2`}`.

`\newcell{`*type*`}`

Define a new cell type. One must define either the fill macro `\`*type*`fill` or redefine the box macro `\`*type*`box` before using a cell macro associated with this cell type. Spaces in *type* are ignored.

```
\newcell{Br}
\def\Brfill{\linefill\ \rightarrowfill}

\Diagram
A & \rBr & B \\
\endDiagram
```

A —— ——→ B

\newDiagram{*type*}

Define a new Diagram type. The macro pair `\`*type* `\end`*type* delimits Diagrams of this type. The macro `\every`*type* can be used to customize their setup. Spaces in *type* are ignored.

```
\newDiagram{Framed}
\def\everyFramed{\Diagrampad=10pt \framed}

\Framed
A & \rTo & B \\
\endFramed
```

$$\boxed{A \longrightarrow B}$$

\newFigure{*type*}

Define a new Figure type. The macro pair `\`*type* `\end`*type* delimits Figures of this type. The macro `\every`*type* can be used to customize their setup. Spaces in *type* are ignored.

\newfill{*name*}{*tail tokens*}{*mid tokens*}{*head tokens*}

Define a fill macro, which can then be used to construct arrows for diagrams. Spaces in *name* are ignored.

```
\font\tenln=line10
\def\ltxah{\smash{\hbox{\kern-4pt\raise\fontdimen22\textfont2
   \hbox{\tenln\char"2D}\hskip.5pt}}}
\newfill{srightarrow}--\ltxah

\hbox to 3cm{\srightarrowfill}
```

$$\xrightarrow{\hspace{3cm}}$$

\newGraph{*type*}

Define a new Graph type. The macro pair `\`*type* `\end`*type* delimits Graphs of this type. The macro `\every`*type* can be used to customize their setup. Spaces in *type* are ignored.

\nonkuviocs*name*

Access the original definition of a control sequence which `kuvio.tex` has redefined.

• ## \nl

Typeset an arrow as a phantom.

```
\newcell{Adj}
\def\Adjbox#1{\vcenter{\offinterlineskip\dimen0=#1
   \hbox to #1{\hskip.2\dimen0\rightarrowfill}
   \hbox to #1{\leftarrowfill\hskip.2\dimen0}}}

\labelpad{Adj}=2.5pt
\Diagram
A & \rAdj ^{f_1} :{.6} \rAdj \nl _{f_2} :{.4} & B \\
\endDiagram
```

$$A \underset{f_2}{\overset{f_1}{\rightleftarrows}} B$$

\nocompile

Equivalent (almost) to `\def\compileto#1{}`. Note that `\global\nocompile` behaves as expected.

\nodot

Inhibit the typesetting of a `\diagramdot` in an alignment entry of a Diagram for which `\dotted` has been specified. See `\dotted`.

• \nw

Ignore a horizontal alignment cell as far as positioning the columns at its head and tail.

\opland \endopland

Rotate the enclosed material by 90 degrees clockwise. The material cannot contain vertical commands (unless enclosed in a box).

\oplandscape

Rotate a diagram 90 degrees clockwise.

\overgrid

Typeset a grid on top of a Figure. For Diagrams and Graphs, synonymous with `\gridlines`.

• \pd{*math*}

Abbreviation for `\hd{\phantom{`*math*`}}`.

• \pl{*math*}

Abbreviation for `\tl{\phantom{`*math*`}}`.

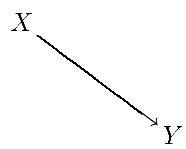
• \pp

Typeset an alignment cell before those corresponding to cell macros preceding it (above or to the left). Usually alignment cells are typeset in the same order in which their corresponding cell macros were processed (top to bottom, left to right) but order is important when breaking arrows with `\br` or `\rl`. Modification cells can be used for even greater control over the order in which arrows are typeset.

\Pt{*name*}

Name a point which can then be referenced using `\pt`. Spaces in *name* are significant.

```
\Graph{2cm}{15mm}
\Pt{tail} (0,1)
\Pt{head} (1,0)
\To \pt{tail} \tl{X} \pt{head} \hd{Y}
\endGraph
```



• `\pt{`*name*`,`*number*`}`
`\pt{`*name*`}`

Name, or reference as a coordinate, a point on an arrow. The string *name* must be enclosed in parentheses if it contains a comma. Also, spaces in *name* are significant. For type 1 modification cells `\pt` is somewhat stunted: A defined point is always just the specified coordinate.

`\ptpoint=`*number* .5
`\ptpoint{`*type*`}=`*number*

Set the default location of points defined on arrows. Behaves like `\labelpoint`. Spaces in *type* are ignored.

`\ptpush=`*dimen* 0pt
`\ptpush+=`*dimen*
`\ptpush{`*type*`}=`*dimen* `\ptpush{`*type*`}+=`*dimen*

Global and type-specific amounts by which the end of an arrow attached to another arrow (using `\pt`) is shortened whenever a vertex has not been attached using `\hd` or `\tl`. The type-specific quantity applies to ends of arrows of cell type *type*. Spaces in *type* are ignored.

`\range=`*integer*

Set both `\xrange` and `\yrange`.

• `\rb`

Make an arrow right braced.

`\recompile`

Equivalent (almost) to `\let\compileto\recompileto`. Note that `\global\recompile` behaves as expected.

`\recompileto{`*name*`}`

Compile a diagram to the files *name*`.kdg` and *name*`.kuv`. Spaces in *name* are ignored.

• `\rl\rl{`*dimen*`}`

Like `\br` except a white rule is typeset in place of an arrow.

• `\rm{`*dimen*`1,`*dimen*`2}`

Abbreviation for `\rr{`*dimen*1`}` `\ru{`*dimen*2`}`. This meaning is only taken when a modifier is expected so does not conflict with the usual meaning of `\rm`. This is also true of all other modifiers.

• `\ro`

Rotate labels on an arrow.

```
\Diagram
  &                 & A                                 \\
  & \ldTo <L >R \ro & \dTo <L >R \ro & \rdTo <L >R \ro \\
X &                 & X              &               & X \\
\endDiagram
```

$$\begin{tikzcd} & & A \arrow[dll, "L", "R"'] \arrow[d, "L"', "R"] \arrow[drr, "R", "L"'] & & \\ X & & X & & X \end{tikzcd}$$

If `\rotatedlabels` is specified then `\ro` inhibits rotation.

`\rot \endrot\rot{`*number*`} \endrot`

Rotate the enclosed material by *number* degrees counterclockwise. The material cannot contain vertical commands (unless enclosed in a box).

`\rotatedlabels`

Invert the action of the `\ro` modifier: Labels are rotated unless `\ro` is specified.

`\rpad=`*dimen* `0pt`
`\rpad+=`*dimen*

Padding added to the right of a diagram or framed material.

```
\framed \rpad=10pt
\Diagram
A & \rTo & B \\
\endDiagram
```

$$\boxed{A \longrightarrow B}$$

• `\rr{`*dimen*`}`

Move an arrow or modification to the right relative to a rotated coordinate system.

```
\Diagram
A & & B & \lTo ^f \rr{2\xgrid} & C \\
\endDiagram
```

$$A \xleftarrow{f} B \qquad\qquad C$$

Note that `\xgrid` is not actually a dimension register, although in the context of a Diagram it can be used as such. Ditto for `\ygrid`. For a Figure or Graph one can use `\xunit` and `\yunit` in the same way.

• `\rt{`*dimen*`}`

Move an arrow or modification to the right.

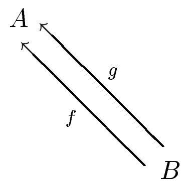
```
\Diagram
A & & B & \rTo ^f \rt{-2\xgrid} & C \\
\endDiagram
```

$$A \xrightarrow{f} B \qquad\qquad C$$

• `\ru{`*dimen*`}`

Move an arrow or modification upwards relative to a rotated coordinate system.

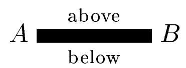
```
\Diagram
A                                          \\
  & \luTo _f \ru{5pt} \luTo ^g \ru{-5pt}   \\
  &                                    & B \\
\endDiagram
```



`Rule`

Cell type whose arrows are rules. The rule width can be set using `\rw`.

```
\Diagram
A & \rRule ^{\rm above} _{\rm below} & B \\
\endDiagram
```



`\Rulewidth=`*dimen* 5pt
`\Rulewidth+=`*dimen*

Default rule width used with `Rule` cells.

• `\rw{`*dimen*`}` .4pt

Set the rule width for a `Rule` cell or `\Frame` modification.

```
\Graph{.25\hsize}{5mm,2}
\Frame (0,0) (.45,2) \rw{1mm}
\Rule  (.55,1) (1,1) \rw{3mm}
\endGraph
```

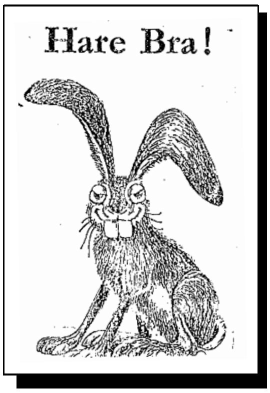
`\scale=`*number*

Abbreviation for `\xscale=`*number* `\yscale=`*number*.

`\shade \endshade`
`\shade{`*dimen*, *dimen*`}{`*dimen*, *dimen*`} \endshade`

Place a shadow behind a frame by specifying its offsets from the bottom left and top right corners of the enclosed material. The values of `\framegray`, `\framepad` and `\framerulewidth` apply.

```
\shade{5pt,-5pt}{3pt,-3pt}\epsfxsize=3cm \epsffile{harebra.eps}\endshade
```



`\shadegray=`*number* or `\shadegrey=`*number* 0

Set the graylevel of the shadow produced by `\shade` `\endshade`.

`\stop`

Force an arrow to stop at an alignment entry. Has no effect on cell macros in the same alignment entry.

```
\Diagram
A & \rTo \up{-2pt} & \stop \rTo \up{2pt} & B \\
\endDiagram
```

$$A \rightrightarrows B$$

`\squash`

Like `\smash` from `plain.tex` except typesets the following parenthesized material in a box having no height, depth or width. The typeset material is centered horizontally.

`\squish`

Like `\smash` from `plain.tex` except typesets the following parenthesized material in a box having the same height and depth but no width. The typeset material is centered horizontally.

`\tall`

Make the baseline of a Diagram coincide with that of its bottommost row and the baseline of a Figure or Graph coincide with that of its lower edge. This is the default for Figures and Graphs in non-math modes.

`\Text{`*text*`}`

Typeset *text* as a modification.

• `\tl{`*math*`}`

Attach math material to the tail of a type 2 modification cell.

```
\Graph{5cm}{5mm}
\To (0,.5) (1,.5) \tl{A}
\endGraph
```

$A \longrightarrow$

`\tpad=`*dimen* `0pt`
`\tpad+=`*dimen*

Padding added to the top of a diagram or framed material. Does not alter the baseline.

```
\framed \tpad=10pt
\Diagram
A & \rTo & B \\
\endDiagram
```

$\boxed{A \longrightarrow B}$

• `\tr{`*dimen*`}`

Raise the vertex attached to the tail of a type 2 modification cell.

• `\ts{`*dimen* or *number*`}`

Slide the vertex attached to the tail of a type 2 modification cell horizontally.

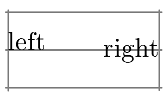
• `\tx{`*dimen* or *number*`}`

Fudge the positioning of an arrow at its tail. See `\hx`.

`\Txt{`*text*`}`

Typeset *text* as a modification, centered vertically on its math axis.

```
\gridlines
\Graph{2cm}{1cm,2}
\Text{left}   (0,1) *0  \Txt{right} (1,1) *1
\endGraph
```



• `\ty{`*dimen*`}`

Fudge the positioning of an arrow at its tail.

`\`*type*

Typeset an arrow of cell type *type* as a modification. Has ten siblings for use in the alignment part of a Diagram: `\a`*type*, `\b`*type*, `\r`*type*, `\rd`*type*, `\d`*type*, `\ld`*type*, `\l`*type*, `\lu`*type*, `\u`*type* and `\ru`*type*.

Non-empty alignment entries in a Diagram are implicitly attached (a la `\pl` and `\pd`) to the end of any type 2 modification cell having integral (no decimal point) coordinates.

`\unit=`*dimen*
`\unit+=`*dimen*

Abbreviation for `\xunit=`*dimen* `\yunit=`*dimen* or `\xunit+=`*dimen* `\yunit+=`*dimen*.

• `\up{`*dimen*`}`

Move an arrow or modification upwards.

```
\Diagram
A & \rTo ^f \up{-\ygrid} & B \\
A &                      & B \\
\endDiagram
```

$$\begin{array}{ccc} A & & B \\ \\ A & \xrightarrow{\quad f \quad} & B \end{array}$$

`\Vertex{`*math*`}`

Typeset math material as a modification in `\vertexstyle`.

`\vertexstyle` `\displaystyle`

Style in which vertices of a diagram are typeset.

```
\let\vertexstyle\scriptstyle
\Diagram
A & B \\
C & D \\
\endDiagram
```

$$\begin{array}{cc} \scriptstyle A & \scriptstyle B \\ \\ \scriptstyle C & \scriptstyle D \end{array}$$

`\vmir` `\endvmir`

Reflect the enclosed material across a vertical mirror located horizontally at the midpoint of the material. The material cannot contain vertical commands (unless enclosed in a box).

`\vpad=`*dimen*
`\vpad+=`*dimen*

Abbreviation for `\bpad=`*dimen* `\tpad=`*dimen* or `\bpad+=`*dimen* `\tpad+=`*dimen*.

• `\white`

Typeset an arrow or modification with graylevel 1.

```
\font\big=cmss17
\Graph{.25\hsize}{1cm}
\Box (0,0) (1,1)
\Txt{\big \"Ovningsk\"orning} (.5,.5) \white
\endGraph
```

Övningskörning

`\white \endwhite`

Typeset the enclosed material with graylevel 1.

`\xgrid=`*dimen* `1cm`
`\xgrid+=`*dimen*

Set the default distance between the vertical gridlines of a Diagram. Sets `\xscale=1`.

```
\xgrid=2cm
\Diagram
A & B \\
C & D \\
\endDiagram
```

$$\begin{array}{cc} A & B \\ C & D \end{array}$$

`\xrange=`*integer* 1

Set the horizontal units for the coordinate system used when typesetting modifications on a Figure or Graph by dividing the default unit by *integer*.

```
\xrange=2
\Graph{1cm}{5mm}
\Dot (0,.5) \Dot (1,.5) \Dot (2,.5)
\endGraph
```


`\xscale=`*number* 1

Scale `\xgrid` by *number*.

```
\xscale=1.5
\Diagram
A & B \\
C & D \\
\endDiagram
```

$$\begin{array}{cc} A & B \\ C & D \end{array}$$

\xunit=*dimen*
\xunit+=*dimen*

Set the horizontal units for the coordinate system used when typesetting modifications on a Figure or Graph.

```
\xunit=5mm
\Graph{1cm}{5mm}
\Dot (0,.5) \Dot (1,.5) \Dot (2,.5)
\endGraph
```


\ygrid=*dimen*
\ygrid+=*dimen*

1cm

Set the default distance between the horizontal gridlines of a Diagram. Sets `\yscale=1`.

\yrange=*integer*

1

Set the vertical units for the coordinate system used when typesetting modifications on a Figure or Graph by dividing the default unit by *integer*. See `\xrange`.

\yscale=*number*

1

Scale `\ygrid` by *number*. See `\xscale`.

\yunit=*dimen*
\yunit+=*dimen*

Set the vertical units for the coordinate system used when typesetting modifications on a Figure or Graph. See `\xunit`.

26. Examples

The `Dash`, `Dashto` and `Mono` cell types in some of these examples use the `arrsy10` font: See §14 and `\arrsy` in §25. Also, `\bb` invokes the bbm family of fonts.

```
$$
\grid=.75cm
\Diagram
A'          &       & \rTo ^{f'} &       & B'                         \\
            & \rdTo <a \dh{2pt}& &       &          & \rdTo _b         \\
\dTo <{g'} &       & A          &       & \dTo <{h'} :{.25}
                                          \rTo ^f :{.25} \br
                                                   &          & B      \\
                                                                         \\
C'          &       & \rTo ^{k'} :{.75} \dTo <g :{.25} \br
                                   &       & D'     &          & \dTo >h \\
            & \rdTo _c \dh{.5pt}&&       &          & \rdTo _d         \\
            &       & C          &       & \rTo ^k &          & D      \\
\endDiagram
$$
```

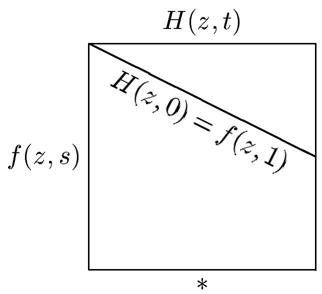
$$\begin{tikzcd}
A' \arrow[rr, "f'"] \arrow[dr, "a"'] \arrow[dd, "g'"'] & & B' \arrow[dr, "b"] \arrow[dd, "h'"' near start] & \\
& A \arrow[rr, "f" near start] \arrow[dd, "g"' near start] & & B \arrow[dd, "h"] \\
C' \arrow[rr, "k'" near end] \arrow[dr, "c"'] & & D' \arrow[dr, "d"] & \\
& C \arrow[rr, "k"] & & D
\end{tikzcd}$$

G. Bredon, *Introduction to Compact Transformation Groups*, page 75.

```
$$
\Dg
F\times K\times U      & \rTo ^{\rm proj}      &             & K\times U         \\
\dTo                   &                       &             & \dTo >{\rm proj}  \\
F\times_K (K\times U)  & {} \approx F\times U  & \rTo ^{\rm proj}& U             \\
\endDg
$$
```

$$\begin{tikzcd}
F \times K \times U \arrow[rr, "\mathrm{proj}"] \arrow[d] & & K \times U \arrow[d, "\mathrm{proj}"] \\
F \times_K (K \times U) & \approx F \times U \arrow[r, "\mathrm{proj}"] & U
\end{tikzcd}$$

```
$$
\joined \let\labelstyle\textstyle
\Graph{3cm,2}{3cm,4}
\Line (0,0) (0,4) <{f(z,s)}
\Line (0,4) (2,4) ^{H(z,t)}
\Line (0,0) (2,0) _{*}
\Line (2,0) (2,4)
\Line (0,4) (2,2) _{\;\;H(z,0) = f(z,1)} \ro
\endGraph
\qquad\qquad
H'(z,t,s) =
\cases{f(z,2s/(2-t))& if $0\le 2s\le 2 - t$\cr
       \cr
       H(z,2s - 2 + t)& if $2 - t\le 2s\le 2$\cr}
$$
```



$$H'(z,t,s) = \begin{cases} f(z,2s/(2-t)) & \text{if } 0 \le 2s \le 2-t \\ \\ H(z,2s-2+t) & \text{if } 2-t \le 2s \le 2 \end{cases}$$

```
$$
\def\mod{\mathrel{\big/}}
\Dg
{\rm Spin}(3) &        &       &       & SU(2)       \\
              & \rdTo  &       & \ldTo               \\
              &        & SO(3)                       \\
              &        & \dTo                        \\
& O(3)\mod O(1)^3 = {} & SO(3)\mod(O(1)^3\cap SO(3)) \\
\endDg
$$
```

$$\begin{tikzcd} \mathrm{Spin}(3) \arrow[dr] & & SU(2) \arrow[dl] \\ & SO(3) \arrow[d] & \\ & O(3) \mathrel{\big/} O(1)^3 = SO(3) \mathrel{\big/} (O(1)^3 \cap SO(3)) & \end{tikzcd}$$

S. Mac Lane, *Categories for the Working Mathematician*, page 109.

```
$$
\def\cod{{\rm cod}\,} \def\dom{{\rm dom}\,}
\cellwidth+=2mm
\Dg
& F & _{\cod u} & \rEq        & F & _{\cod u} &       &              & F_i                  \\
& \uTo >{p_u} & &             & \uTo &        &       & \ruTo <{p_i} \tx{-5pt} &
                                                                        \uDashto >{\mu_i} \\
{\mit\Pi}_{u:j\to k}F_k = {\mit\Pi}_u &
  F & _{\cod u} & \lTo ^f \up{3pt} \lTo _g \up{-3pt}
                              & {\mit\Pi}_i & & F_i & \lDashto ^e & d                \\
& \dTo >{p_u} & &             & \dTo >{p_{\dom u}}                                   \\
& F & _{\cod u} & \lTo ^{Fu} & F & _{\dom u}                                          \\
\endDg
$$
```

$$
\begin{tikzcd}
 & F_{\mathrm{cod}\,u} \arrow[r, phantom, "="] & F_{\mathrm{cod}\,u} & F_i \\
\mathit{\Pi}_{u:j\to k}F_k = \mathit{\Pi}_u F_{\mathrm{cod}\,u} \arrow[u, "p_u"'] \arrow[d, "p_u"'] & & \mathit{\Pi}_i F_i \arrow[ll, shift left, "f"'] \arrow[ll, shift right, "g"] \arrow[u] \arrow[ur, "p_i"] \arrow[d, "p_{\mathrm{dom}\,u}"] & d \arrow[l, dashed, "e"'] \arrow[u, dashed, "\mu_i"'] \\
 & F_{\mathrm{cod}\,u} & F_{\mathrm{dom}\,u} \arrow[l, "Fu"'] &
\end{tikzcd}
$$

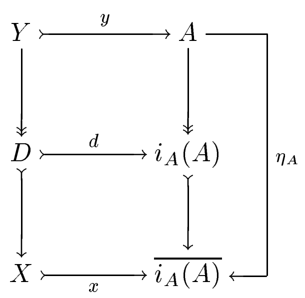
S. Mac Lane, *Categories for the Working Mathematician*, page 117.

```
$$
\Dg
u            & \rTo ^{e_1}    & v        & & \rTo \up{3pt} ^f \rTo \up{-3pt} _g & d    \\
\uDashto >s &                 & \dTo >e &                   &                  & \uTo \\
w            & \rTo ^{ee_1s} & w        & {} = {\mit\Pi}k_i & \rTo             & k_i  \\
\endDg
$$
```

$$
\begin{tikzcd}
u \arrow[r, "e_1"] & v \arrow[rr, shift left, "f"] \arrow[rr, shift right, "g"'] \arrow[d, "e"] & & d \\
w \arrow[u, dashed, "s"'] \arrow[r, "ee_1 s"] & w = \mathit{\Pi} k_i \arrow[rr] & & k_i \arrow[u]
\end{tikzcd}
$$

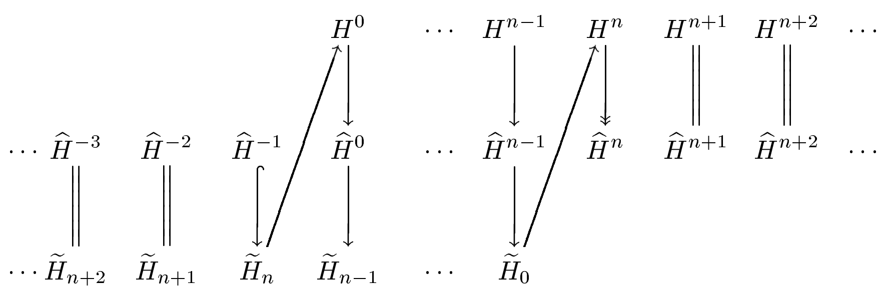
S. Mac Lane and I. Moerdijk, *Sheaves In Geometry and Logic*, page 232.

```
$$
\cellwidth+=3mm
\Dg
Y      & \rMono ^y & A                  & \rLine \jh & \stop                \\
\dEpi  &           & \dEpi              &            & \mx{-1cm}            \\
D      & \rMono ^d & i_A(A)             &            & \dLine >{\eta_A} \jn \\
\dMono &           & \dMono                                                 \\
X      & \rMono _x & \overline{i_A(A)} & \lTo \jt   & \stop                \\
\endDg
$$
```



K. S. Brown, *Cohomology of Groups*, page 280.

```
$$
\let\^\widehat \let\~\widetilde
\flexible \columndist=1cm
\xgrid+=2mm \ygrid+=-2mm
\Diagram
      &         &         &        &H^0 \bTo(-1,-4) \hx{-2pt}
                                          &\cdots&H^{n-1}  &H^n  &H^{n+1}&H^{n+2} &\cdots   \\
\mx{5mm}&       &         &        &\dTo  &      &\dTo     &\dEpi&\dBar  &\dBar   &\mx{-2mm}\\
\cdots&\^H^{-3} &\^H^{-2} &\^H^{-1}&\^H^0 &\cdots&\^H^{n-1}&\^H^n&\^H^{n+1}&\^H^{n+2}&\cdots\\
      &\dBar    &\dBar    &\dInto  &\dTo  &      &\dTo \dx{-2mm}                            \\
\cdots&\~H_{n+2}&\~H_{n+1}&\~H_n&\~H_{n-1}&\cdots&\~H_0 \aTo(1,4) \hx{-2pt}                 \\
\endDiagram
$$
```



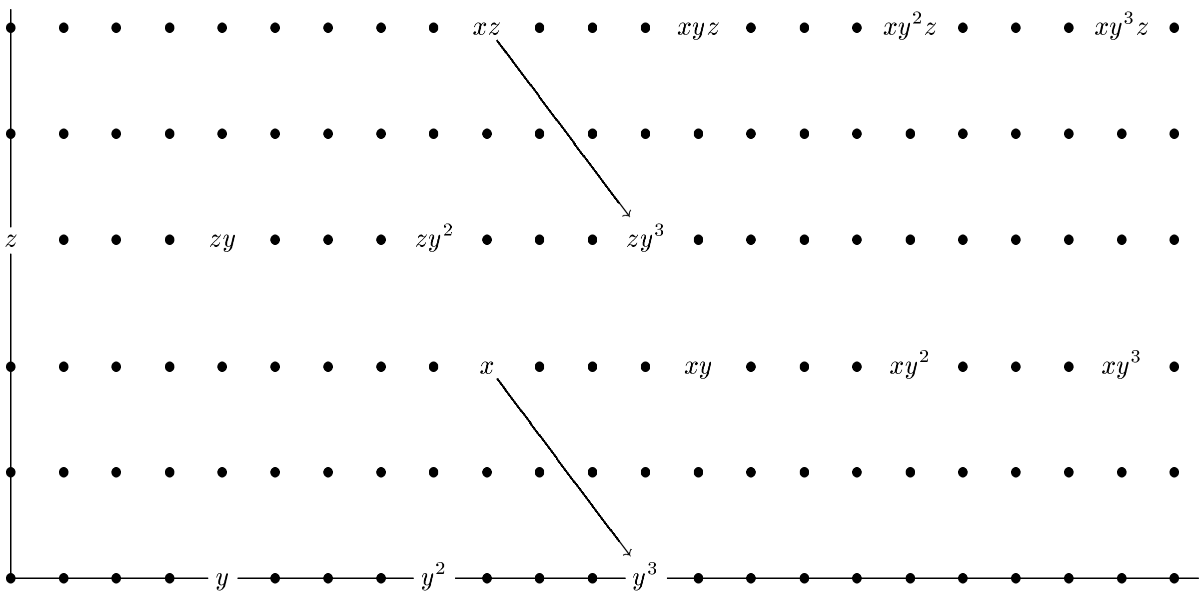
George W. Whitehead, *Elements of Homotopy Theory*, page 592.

```
$$
\def\T{{\bf T}} \def\S{{\bf S}}
\landscape \let\labelstyle\textstyle
\cellpush=10pt \xscale=2.2
\Diagram
       &                 & H_n(X^+\cup\T A^+, X^+) &
   \lTo ^{H_n(k_1)} &        & H_n(\T A^+,A^+)
                                \aTo (-3,-2) _{H_n(p_1)}
                                \aTo (3,-4) ^{\partial_1}                     \\
       &                 & \dTo >{H_n{q_1}}                                   \\
       &                 & H_n(\S A^+,P)           &
                    &        & * \dx{.7}                                      \\
       & \ruTo _{H_n(q_4)} \tx{15pt}
                         & \uTo >{H_n(t)}                                     \\
H_n(X^+\cup\T A^+,P) \aTo (2,4)  ^{H_n(j_1)} \tx{15pt}
                     \aTo (2,-4) _{H_n(j_2)} \tx{15pt}
       &   &   &    &        &   & H_{n-1}(A) &
                                      \rTo ^{H_{n-1}(i_A)} & H_{n-1}(A^+,P) \\
       & \rdTo ^{H_n(q_2)} \tx{15pt}
                         &                         & & & \ruTo             \\
       &                 & H_n(X^+/A^+,P)          &
  \lTo ^{P_n(X,A)}  & H_n(X,A)
                      \aTo (1,-2) >{H_n(i_X)}                                 \\
       &                 & \uTo >{H_n(q_3)}                                   \\
       &                 & H_n(X^+\cup\T A^+,\T A^+)&
  \lTo _{H_n(k_2)} & & H_n(X^+,A^+)
                       \aTo (3,4) _{\partial_2}
                       \aTo (-3,2) _{H_n(p_X)}                                \\
\endDiagram
$$
```

$$
\begin{tikzcd}
 & H_n(X^+\cup \mathbf{T}A^+, X^+) \arrow[d, "H_n q_1"] & & H_n(\mathbf{T}A^+, A^+) \arrow[ll, "H_n(k_1)"'] \arrow[dll, "H_n(p_1)"] \arrow[ddr, "\partial_1"] & \\
 & H_n(\mathbf{S}A^+, P) & & * & \\
H_n(X^+\cup \mathbf{T}A^+, P) \arrow[uur, "H_n(j_1)"] \arrow[ur, "H_n(q_4)"'] \arrow[dr, "H_n(q_2)"] \arrow[ddr, "H_n(j_2)"'] & & & H_{n-1}(A) \arrow[r, "H_{n-1}(i_A)"] & H_{n-1}(A^+, P) \\
 & H_n(X^+/A^+, P) \arrow[uu, "H_n(t)"'] & H_n(X, A) \arrow[l, "P_n(X,A)"'] \arrow[ur] \arrow[dr, "H_n(i_X)"] & & \\
 & H_n(X^+\cup \mathbf{T}A^+, \mathbf{T}A^+) \arrow[u, "H_n(q_3)"'] & & H_n(X^+, A^+) \arrow[ll, "H_n(k_2)"] \arrow[ull, "H_n(p_X)"] \arrow[uur, "\partial_2"'] &
\end{tikzcd}
$$

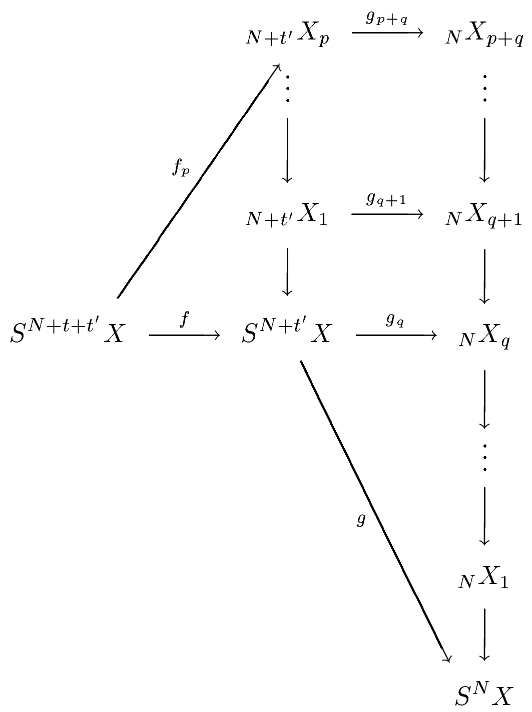
J. McCleary, *A User's to Spectral Sequences*, page 13.

```
$$
\dotted
\grid=7mm
\yscale=2
\Diagrampad=0pt
\Diagram
&&&&&&&&&xz&&&&xyz&&&&xy^2z&&&&xy^3z&\\
\\
z&&&&zy&&&&zy^2&&&&zy^3\\
\dy{-.2}
&&&&&&&&&x&&&&xy&&&&xy^2&&&&xy^3\\
\\
&&&&y&&&&y^2&&&&y^3\\
\Modify
\Line (0,0) (4,0) \dt{1pt}
\Line (4,0) (8,0)
\Line (8,0) (12,0)
\Line (12,0) (22.5,0)
\Line (0,0) (0,3) \dt{1pt}
\Line (0,3) (0,5.2)
\To (9,2) (12,0)
\To (9,5) (12,3)
\endDiagram
$$
```



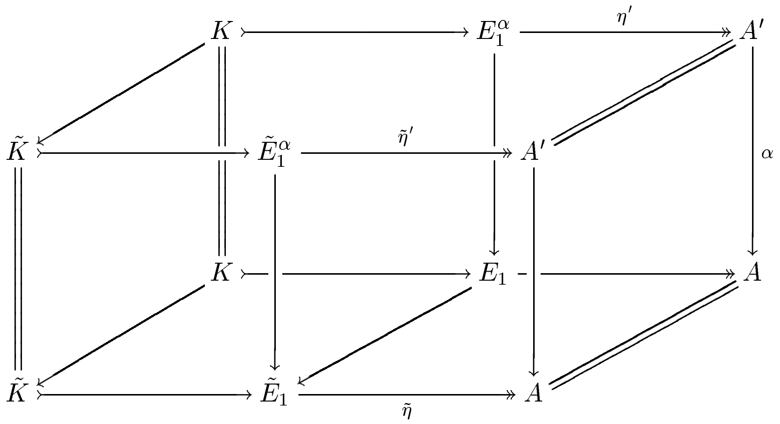
J. McCleary, *A User's to Spectral Sequences*, page 346.

```
$$
\cellpush+=5pt
\cellwidth+=2mm
\Dg
&&_{N+t'}X_p  & \rTo ^{g_{p+q}} & _N X_{p+q}  \\
&& \dFillcell{\;\cdots\;\rightarrowfill} &
                                & \dFillcell{\;\cdots\;\rightarrowfill} \\
                                            \\
&& _{N+t'}X_1 & \rTo ^{g_{q+1}} & _N X_{q+1}  \\
&& \dTo       &                 & \dTo        \\
S^{N+t+t'}X \aTo (2,5) ^{f_p} \tx{.7} \hx{5pt} & \rTo ^f &
   S^{N+t'}X  & \rTo ^{g_q}     & _N X_q      \\
                                            \\
&&            &                 & \dFillcell{\rightarrowfill\;\cdots\;\rightarrowfill} \\
                                            \\
&&            &                 & _N X_1      \\
&&            &                 & \dTo        \\
&&            &                 & S^N X \bTo (-2,6) <g \hx{.2} \\
\endDg
$$
```



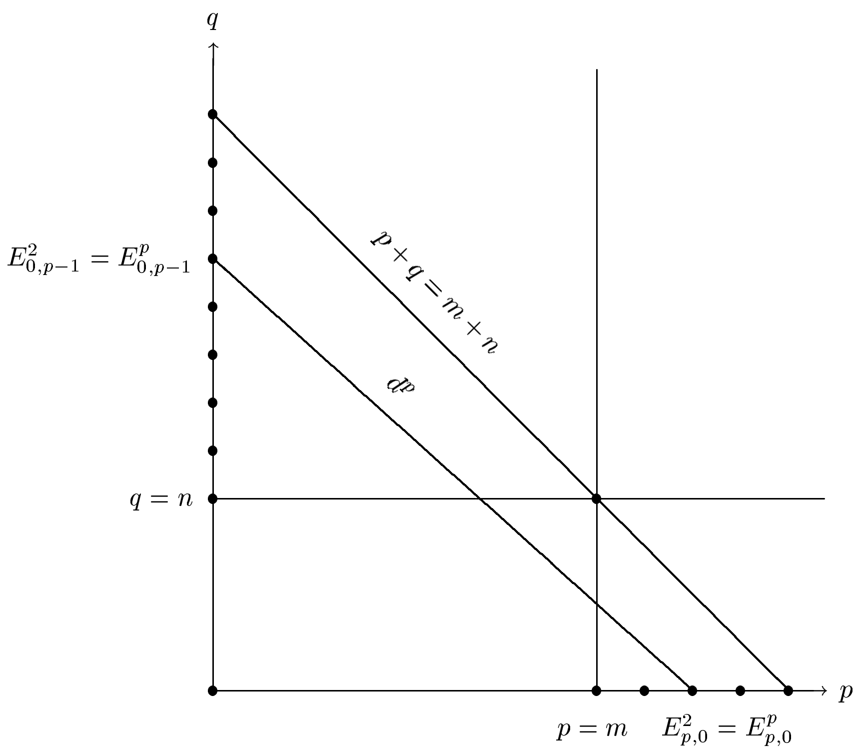
P. J. Hilton and U. Stammbach, *A Course in Homological Algebra*, page 149.

```
$$
\let~\tilde
\cellwidth+=17mm \columndist+=12mm
\gravitateleft
\Dg
  &      & K              & \rMono      && E_1^\alpha& \rEpi^{\eta'}&& A'       \\
  & \ldTo&                &             &&           &      & \ruBar          \\
~K&      & \dBar\rMono\br& ~E_1^\alpha&& \dTo\rEpi^{~\eta'}\br&
                                                    A'                       \\
\dBar&   &                &             &&           &      &    & \dTo>\alpha \\
  &      & K              & \rMono\dTo\br&& E_1\ax{2mm}& \rEpi\dTo\br&& A     \\
  & \ldTo&                &      & \ldTo&           &      & \ldBar          \\
~K&      & \rMono         & ~E_1 & \rEpi_{~\eta}&   & A                       \\
\endDg
$$
```

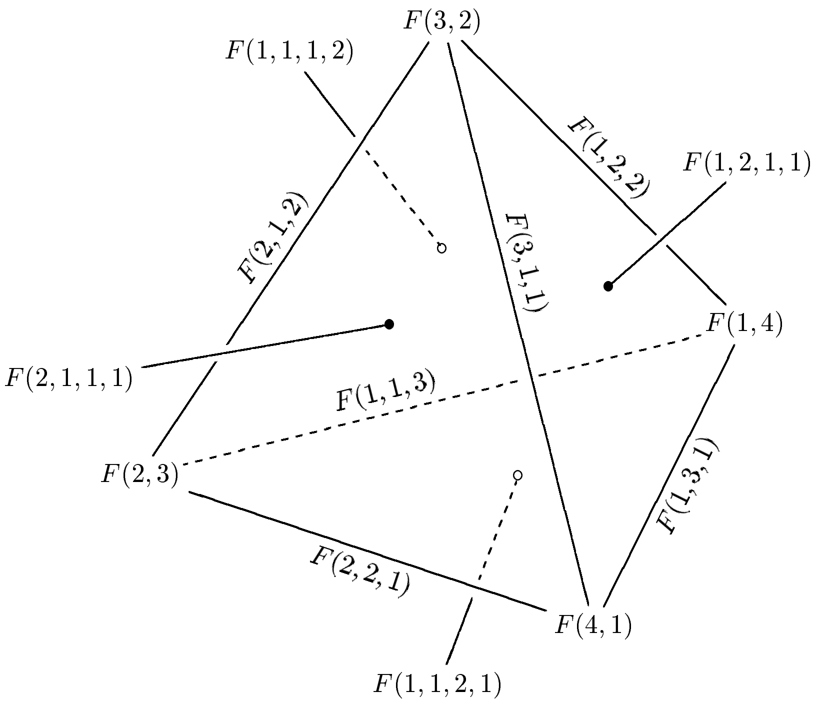


George W. Whitehead, *Elements of Homotopy Theory*, page 650.

```
$$
\cellpush=4pt
\let\labelstyle\textstyle
\lpad=1in \bpad=1cm
\Graph{4in,4}{3.5in,1in}
\To (0,0) (3.3,0)    \hd{p}
\To (0,0) (0,3.5)    \hd{q}
\Line (2,0) (2,3.25) \Line (0,1) (3.2,1) \Dot (2,1)
\Dot (0,0)
%
\Dot (2,0) \Dot (2.25,0) \Dot (2.5,0) \Dot (2.75,0) \Dot (3,0)
%
\Dot (0,1) \Dot (0,1.25) \Dot (0,1.5) \Dot (0,1.75)
\Dot (0,2) \Dot (0,2.25) \Dot (0,2.5) \Dot (0,2.75) \Dot (0,3)
%
\Line (0,2.25) (2.5,0)  ^{d^p} :{.35,5pt} \ro
\Line (0,3) (3,0)  ^{p + q = m + n} :{.35,5pt} \ro
%
\Math{q = n} (-.1,1) *1     \Math{E_{0,p-1}^2 = E_{0,p-1}^p} (-.1,2.25) *1
\Math{p = m} (2,-.2) *{.55} \Math{E_{p,0}^2 = E_{p,0}^p} (2.75,-.2) *{.6}
\endGraph
$$
```



```
$$
\rotatedlabels \let\labelstyle\textstyle \scale=2
\Diagram
 & & F(3,2) \aLine (1,-4) >{F(3,1,1)} :{.4} \br  \\
 & &        & \rdLine >{F(1,2,2)} \pt{right,.75} \\
 & &        &         & F(1,4)                  \\
F(2,3) \aDash (4,1) ^{F(1,1,3)} :{.4} \pp        \\
 & &        & F(4,1) \aLine (1,2) >{F(1,3,1)}    \\
\Modify
\Pt{top} (2,4) \Pt{b1} (0,1) \Pt{b2} (3,0)
\Pt{LeftF} (1.65,2) \Pt{BackF} (2,2.5)
\Pt{BottomF} (2.5,1) \Pt{RightF} (3.1,2.25)
\Nul  \pt{b1} \pt{b2}               \pt{bottom,.8}
\Nul  \pt{BottomF} \pt{bottom}      \pt{BottomL,2}
\Dash \pt{BottomF} \pt{bottom}      \tl\circ \dt{3pt}
\Line \pt{bottom} \pt{BottomL}      \hd{F(1,1,2,1)}
\Line \pt{b1} \pt{b2} _{F(2,2,1)}   \br
\Nul  \pt{b1} \pt{top}              \pt{left,.25} \pt{back,.75}
\Nul  \pt{BackF} \pt{back}          \pt{BackL,2}
\Nul  \pt{LeftF} \pt{left}          \pt{LeftL,2}
\Dash \pt{BackF} \pt{back}          \tl\circ \dt{3pt}
\Line \pt{back} \pt{BackL}          \hd{F(1,1,1,2)}
\Line \pt{b1} \pt{top} <{F(2,1,2)} \br
\Line \pt{LeftF} \pt{LeftL}         \tl\bullet \hd{F(2,1,1,1)} \jt \br
\Nul  \pt{RightF} \pt{right}        \pt{RightL,3}
\Line \pt{RightF} \pt{RightL}       \tl\bullet \hd{F(1,2,1,1)} \jt \br
\endDiagram
$$
```



```
$$
\def\y#1#2{Y(#1,#2)} \def\RP{{\bb R}P}
\gravitateright
\Dg
&&&\RP^1 = {}& \y11 & \rInto & \y21 & \rInto & \y31 & \rInto & \y41 & \rInto & \cdots \\
&&&          &      &        & \dTo &        & \dTo &        & \dTo &                 \\
&&\RP^2 = {}&&      &        & \y12 & \rInto & \y22 & \rInto & \y32 & \rInto & \cdots \\
&&          &&      &        &      &        & \dTo &        & \dTo &                 \\
&\RP^3 = {}&&&      &        &      &        & \y13 & \rInto & \y23 & \rInto & \cdots \\
&          &&&      &        &      &        &      &        & \dTo &                 \\
\RP^4 = {}&&&&      &        &      &        &      &        & \y14 & \rInto & \cdots \\
\endDg
$$
```

$$
\begin{tikzcd}
\mathbb{R}P^1 = Y(1,1) \arrow[r, hook] & Y(2,1) \arrow[r, hook] \arrow[d] & Y(3,1) \arrow[r, hook] \arrow[d] & Y(4,1) \arrow[r, hook] \arrow[d] & \cdots \\
 & \mathbb{R}P^2 = Y(1,2) \arrow[r, hook] & Y(2,2) \arrow[r, hook] \arrow[d] & Y(3,2) \arrow[r, hook] \arrow[d] & \cdots \\
 & & \mathbb{R}P^3 = Y(1,3) \arrow[r, hook] & Y(2,3) \arrow[r, hook] \arrow[d] & \cdots \\
 & & & \mathbb{R}P^4 = Y(1,4) \arrow[r, hook] & \cdots
\end{tikzcd}
$$

```
$$
\def\M{{\cal M}\dot} \def\inv{^{-1}} \def\orlz#1{\langle#1\rangle}
\let\labelstyle\textstyle
\Dg
[q(g,b),t] \aNul (1,-1) \aMapsto (2,-4) &&& \rMapsto &&&
                 q(g,\phi(b,t)) \aNul (-1,-1) \aMapsto (-2,-4) \mx{12pt} \\
       & \M R_s &       & \rTo    &        & \beta\inv(W_s)   \\
       &        & \rdTo &         & \ldTo >{R_s}              \\
       &        &       & \beta\inv\orlz{s}                   \\
       &  & q(g,r_s(b)) & {} = {} & q(g,r_s(\phi(b,t)))                 \\
\endDg
$$
```

$$
\begin{tikzcd}
{[q(g,b),t]} \arrow[rrrr, mapsto] \arrow[rrddd, mapsto] & & & & q(g,\phi(b,t)) \arrow[llddd, mapsto] \\
 & \mathcal{M}\dot{R}_s \arrow[rr] \arrow[dr] & & \beta^{-1}(W_s) \arrow[dl, "R_s"] & \\
 & & \beta^{-1}\langle s\rangle & & \\
 & & q(g,r_s(b)) = q(g,r_s(\phi(b,t))) & &
\end{tikzcd}
$$

27. Availability

Both `kuvio.tex` and `arrsy10` are available by anonymous ftp from the following location.

```
ftp.math.ubc.ca:/pub/svensson
```

They can also be found in ⟨*tex-archive*⟩`/macros/generic/diagrams/kuvio` on any of the following CTAN hosts.

```
ftp.dante.de    (Deutschland)
ftp.tex.ac.uk   (England)
ftp.cdrom.com   (USA)
```

28. Copyright

The file `kuvio.tex` may be freely distributed and used, with the following restrictions.

1. Substantial use of `kuvio.tex` in any published work should be suitably acknowledged.

2. No changes or modifications are to be made to `kuvio.tex`.

3. The documentation, *Typesetting diagrams with kuvio.tex*, must be distributed with `kuvio.tex`.

I would enjoy receiving a copy of any book, journal article or thesis prepared using `kuvio.tex`.

Thank you.

29. Final Thoughts

This document was written using `plain.tex`, `epsf.tex` (from the `dvips` distribution) and homegrown macros.

Remember to use grouping and diagram types to keep changes local to a particular level (as was done with the examples in this manual).

As noted earlier, `\special` may wreak havoc with your `dvi` file previewer. On the NeXT, Rokicki's TEXview has no trouble. However, older versions of TEXview may display rotated arrows improperly. The version from February 1994 (last seen at `labrea.stanford.edu:/pub/texview.tar.Z`) has fixed this problem.

If you have trouble previewing a `dvi` file and can preview a PostScript file, the command

```
dvips name.dvi -o
```

generates the PostScript file `name.ps` from the `dvi` file `name.dvi`. Save trees. Ghostview does an excellent job of previewing PostScript under X.

Two notable limitations of `kuvio.tex`: (1) Diagrams cannot be nested and (2) there are no helpful error messages if something goes wrong. This will probably not change anytime soon, if ever.

30. Index

The page number of a Summary entry is set in italics.