User's Manual for Diagram Macros

J. C. Reynolds

December, 1987

Revised for `diagmac2.sty`
by Bob Tennent
Version 2.1 May, 2009

Contents

1 Introduction

> These macros are in the public domain, and have not changed in many years. Acknowledgement of their usage is not necessary. However, neither I nor CMU accept any responsibility for the consequences of errors in these macros or their documentation. This is more than the usual disclaimer; TEX is a beastly language for programming anything complex, and I am not an expert in its use, so that there are probably errors lurking in the macros.
>
> John Reynolds[1]

The file `diagmac2.sty` contains TEX macros for producing various kinds of diagrams. It consists of two parts: a collection of general macros for producing a wide variety of diagrams, and a second collection of macros (which call upon the first) that are specifically oriented to category-theory diagrams.

`diagmac2.sty` is fully compatible with Reynolds's original `diagmac`, but takes advantage of the `pict2e` implementation of the `picture` environment to allow arbitrary slopes for edges and diameters for circles. In this manual, the sections of the original manual that refered to the limitations have been deleted. Sections 5 and 6 are new in Version 2.1.

The LATEX declarations `\thinlines` and `\thicklines` or the `pict2e` declaration `\linethickness` may be used to vary the thickness of lines, arrowheads, and circles.

---

[1] `ftp://ftp.cs.cmu.edu/user/jcr/README`

2 Programs and States

Certain parameters to these macros are "programs." A program is a TeX text that does not directly produce any output but causes state changes by calling macros. For example, in LaTeX, the text read in picture mode, i.e. the text between `\begin{picture}` and `\end{picture}` commands, is a program that causes state changes by calling the macro `\put`. (Internally, such macros cause state changes by assigning to hidden registers and redefining hidden control symbols. As a consequence, a program cannot call state-changing macros within a group.)

The diagram-producing macros use two kinds of program, called diagram programs and expression programs. The state manipulated by a diagram program, called a diagram state, is a plane containing symbols, lines, and circles. Locations on this plane are specified by an x, y-coordinate system, in which x specifies horizontal distance, with increasing values to the right, and y specifies vertical distance, with increasing values upwards. The diagram state also contains a "vertex list," which is a list of points (i.e. x, y-coordinate pairs) paired with polygonal regions called "shadows."

The diagram state may also contain a "current edge," which is a (perhaps invisible) directed line segment. When the current edge is defined, it is determined by four dimension registers:

- `\xstart`: the x-coordinate of the start point
- `\ystart`: the y-coordinate of the start point
- `\xend`: the x-coordinate of the end point
- `\yend`: the y-coordinate of the end point,

and two number registers:

- `\xslope`: the x-component of the slope
- `\yslope`: the y-component of the slope

giving the slope of the edge, reduced to lowest terms. A diagram program may refer to any of these quantities, and may also alter the dimension registers explicitly (as well as by calling diagram macros), providing this alteration preserves the slope of the edge.

The state manipulated by an expression program, called an expression state, is also a plane, containing an expression and other symbols, etc., upon which is imposed an x, y-coordinate system. This state contains an invisible "current rectangle," determined by the four dimension registers:

- `\lexpr`: the x-coordinate of the left side
- `\rexpr`: the x-coordinate of the right side
- `\texpr`: the y-coordinate of the top
- `\bexpr`: the y-coordinate of the bottom,

and a "center point," determined by the two dimension registers:

- `\xcenter`: the x-coordinate of the center point
- `\ycenter`: the y-coordinate of the center point.

An expression program may refer to or alter these six dimension registers explicitly (as well as by calling various macros).

The expression state may also contain a (perhaps invisible) polygon called the "current shadow," and a (perhaps invisible) circle called the "current circle." When the current circle is defined, it is determined by three dimension registers:

- `\dcircle`: the diameter
- `\xcircle`: the x-coordinate of the center
- `\ycircle`: the y-coordinate of the center.

An expression program may refer to or alter these three dimension registers explicitly (as well as by calling various macros).

The qualification "perhaps invisible" is meant to indicate that the position, shape, and size of edges, shadows, and circles are established by one group of macros (e.g. `\setedge`, `\rect`, `\octagon`, `\setcircle`), but that these entities are actually drawn, i.e. made to appear on the plane of the diagram or expression state, by another group of macros (e.g. `\drawsolidedge`, `\outline`, `\drawcircle`).

In calls of the diagram macros, a coordinate is sometimes specified by a dimension, but often it is specified by a number (i.e. integer) that gives the coordinate as a multiple of the dimension that is the meaning of the control symbol `\diagramunit`. This control symbol is defined to be 1pt, but the user may redefine it to be some other dimension, either in his main program or at the beginning of a diagram program.

In addition to the control symbols discussed in this description, this collection of macros defines a large number of control symbols that are normally no concern of the user. To avoid the accidental redefinition of these symbols by the user, they are all given names beginning with `\zz`.

3 The General Macros for Diagrams

We now describe the general macros for drawing diagrams. The main level macro is

```
\diagram{<diagram program>}
```

It executes the diagram program that is its only parameter, and then issues the final state produced by this program as a horizontal box whose height, width, and depth are just enough to enclose all of the symbols and lines in this state, plus the origin (0,0) of the coordinate system. The height (depth) will be the distance from the horizontal line $y = 0$ to the highest (lowest) extent of any symbol or line.

Within a diagram program, one can call the following macros:

```
\vertex<number:x-coord>,<number:y-coord>:
    {<balanced mathematical text>}{<expression program>}
```

`\vertex` sets the ¡balanced mathematical text¿ in math mode, with text style, and creates an expression state containing the resulting expression, with the current rectangle just enclosing the expression. The center point is placed midway between the left and right sides of the current rectangle, at a height above the baseline of the expression given by the control symbol `\centerheight`, which is defined to be 3pt. (The effect is to place the center point on the axis of the expression. However, the user may need to change the definition of `\centerheight` if he is using unusual fonts or script style.) The reference point of the expression will lie at the origin of the coordinate system.

Next, `\vertex` executes the `<expression program>` to modify the expression state. Then the material in the expression state is placed in the current diagram state, at a position so that the center point lies at the point `<number:x-coord>,<number:y-coord>`. Finally, if the expression state contains a current shadow, the point `<number:x-coord>,<number:y-coord>` is paired with the shadow and placed on the vertex list.

```
\place<number:x-coord>,<number:y-coord>:
    {<balanced mathematical text>}{<expression program>}

\placed{<dimen:x-coord>}{<dimen:y-coord>}
    {<balanced mathematical text>}{<expression program>}
```

`\place` behaves the same way as `\vertex`, except that nothing is placed on the vertex list. `\placed` behaves the same way as `\place`, except that the coordinates at which the center point is placed are expressed by dimensions rather than numbers.

```
\setedge<number:x-start-coord>,<number:y-start-coord>,
    <number:x-end-coord>,<number:y-end-coord>:
```

`\setedge` makes the current edge a directed line segment from the point “start” given by its first two parameters to the point “end” given by its last two parameters. This line segment is invisible (until it is drawn by one of the macros discussed below).

`\setedge` also examines the vertex list to obtain any shadows that have been associated with the start or end points by prior executions of `\vertex`.

```
\shiftedge{<dimen:length>}
```

`\shiftedge` displaces the current edge by a vector whose length is determined by the `<dimen:length>` parameter, and whose direction is obtained by rotating the current edge 90 degrees counterclockwise.

```
\shadeedge
```

`\shadeedge` changes the extent of the current edge, without displacing or rotating it, to exclude the portions of the edge lying within shadows associated with its start and end points. If the execution of `\setedge` that established the current edge found a shadow associated with the start point, then `\shadedge` will shorten (or conceivably lengthen) the current edge so that its start point lies on the boundary of the shadow. (If this is not possible, the start point will be adjusted to be as close as possible to the shadow.) The end point is adjusted similarly.

```
\drawsolidedge
```

`\drawsolidedge` draws the current edge as a solid line.

```
\drawdashedge{<dimen:length>}{<dimen:length>}{<number>}{<number>}
```

`\drawdashedge` draws the current edge as a dashed line. The dashed line will always begin and end with a dash. The number of dashes will be as large as possible subject to the constraint that, if one or more blanks occur, the dashes will be at least as long as the first parameter and the blanks will be at least as long as the second parameter. If one or more blanks occur, the excess length of the dashes and of the blanks will be proportional to the third and fourth parameters respectively. The first two parameters must be positive dimensions, and the last two parameters must be nonnegative numbers whose sum is positive.

```
\drawdotedge{<dimen:length>}{<1 or 0>}
```

`\drawdotedge` draws the current edge as a dotted line. The number of dots will be the largest number such that the distance between dots is at least as large as the first parameter, which must be a positive dimension. A dot will always appear at the start point, and will appear at the end point if the second parameter is 1. If the second parameter is 0 then the final dot will be omitted.

```
\drawedgehead{<number:0 to 100>}{<1 or 0>}{<1 or 0>}
```

`\drawedgehead` draws an arrowhead on the current edge at a distance from the start point of p times the length of the edge, where p is the first parameter divided by 100. The arrowhead will point to the end point if the second parameter is 1, or to the start point if the second parameter is 0. If the third parameter is 1, the arrowhead will be advanced towards its tip by the value of the control symbol `\edgeheaddisp`, which is defined to be 4pt, but may be redefined by the user.

```
\abutleft<number:y-coord>:
    {<balanced mathematical text>}{<expression program>}

\abutright<number:y-coord>:
    {<balanced mathematical text>}{<expression program>}

\abutbelow<number:x-coord>:
    {<balanced mathematical text>}{<expression program>}

\abutabove<number:x-coord>:
    {<balanced mathematical text>}{<expression program>}
```

Each of these macros uses the `<balanced mathematical text>` to initialize an expression state (in the same way as `\vertex`) and then executes the `<expression program>`, which must establish a shadow. The material in the expression state is then placed in the diagram state, at a location such that the shadow touches the current edge (or its extension as an infinite line), and lies to the left (or to the right, below, or above, as determined by the macro name). For `\abutleft` and `\abutright`, which must not be used when the current edge is horizontal, the first parameter gives the y-coordinate of the point at which the center point is to be located. For `\abutbelow` and `\abutabove`, which must not be used when the current edge is vertical, the first parameter gives the x-coordinate.

```
\abutleftd{<dimen:y-coord>}
    {<balanced mathematical text>}{<expression program>}

\abutrightd{<dimen:y-coord>}
    {<balanced mathematical text>}{<expression program>}

\abutbelowd{<dimen:x-coord>}
    {<balanced mathematical text>}{<expression program>}

\abutaboved{<dimen:x-coord>}
    {<balanced mathematical text>}{<expression program>}
```

Each of these macros behaves the same way as its cousin, described above, except that the first parameter is a dimension instead of a number.

Within an expression program, one can call the following macros:

```
\leftghost{<balanced mathematical text>}

\rightghost{<balanced mathematical text>}
```

These macros change `\xcenter` (the x-coordinate of the center point). The `<balanced mathematical text>` is set in an hbox, using math mode, text style, which is ignored except for its width. `\leftghost` sets `\xcenter` to the left of the current rectangle plus half the width of the hbox. `\rightghost` sets `\xcenter` to the right of the current rectangle minus half the width of the hbox. The effect is to place the "ghost expression" (invisibly) within the current rectangle at the left or right side, and to move the center point horizontally to the midpoint of the ghost expression.

```
\border{<dimen:x-length>}{<dimen:y-length>}

\borderto{<dimen:x-length>}{<dimen:y-length>}

\symmetrize
```

These macros enlarge the current rectangle. `\border` moves the left and right sides outwards by its first parameter, and raises the top and lowers the bottom by its second parameter. (If either parameter is negative, the rectangle will contract.) `\borderto` enlarges the current rectangle so that its width is at least the first parameter and its height (including depth) is at least the second parameter. (Equal amounts will be added at the left and right, and at the top and bottom.)

`\symmetrize` raises the top or lowers the bottom so that they are equally distant from the center point.

```
\place<number:x-coord>,<number:y-coord>:
    {<balanced mathematical text>}{<expression program>}

\placed{<dimen:x-coord>}{<dimen:y-coord>}
    {<balanced mathematical text>}{<expression program>}
```

These macros can be called from expression programs as well as diagram programs. They have no effect on the current rectangle or center point.

```
\rect
```

`\rect` defines the current shadow to be the current rectangle.

```
\hexagon
```

`\hexagon` defines the current shadow to be a hexagon with two horizontal sides identical with the top and bottom of the current rectangle, and four sides of slope (+ or - 1), (+ or - 2).

```
\octagon{<dimen:length>}
```

`\octagon` defines the current shadow to be an octagon inscribed in the current rectangle. The horizontal sides and vertical sides are shorter than those of the current rectangle by twice the parameter, and the remaining sides have slope (+ or - 1), (+ or - 1).

```
\diamond
```

`\diamond` defines the current shadow to be a square, just large enough to enclose the current rectangle, whose sides have slope (+ or - 1), (+ or - 1).

```
\rorect{<dimen:diameter>}{<1 or 0>}{<1 or 0>}
```

`\rorect defines` the current shadow to be a rectangle with rounded (i.e. quarter-circle) corners. The diameter of the corners is determined as follows.

1. Take the maximum of:
   (a) the first parameter;
   (b) if the second parameter is 1, then the width of the current rectangle, else 0;
   (c) if the third parameter is 1, then the height of the current rectangle, else 0.
2. Take the diameter of the smallest printable circle larger or equal to (1), or if no such printable circle exists, take the diameter of the largest printable circle.

The shadow is then the smallest rounded rectangle with corners of this diameter such that the corresponding true (unrounded) rectangle encloses the current rectangle.

The effect (if there is a sufficiently large printable circle) is to produce:

	if the 2nd parameter is	and the 3rd parameter is
a rounded rectangle	0	0
a vertical oblong	1	0
a horizontal oblong	0	1
a circle	1	1

If the shadow is drawn (using `\outline`, as described below) its shape will be the rounded rectangle just described. However, if the shadow is used to shade an edge or to abut an expression to an edge or circle, then a slight fudge occurs: the shadow is taken to be the smallest octagon (with the same shape as that produced by `\octagon`) enclosing the specified rounded rectangle.

```
\outline
```

`\outline` draws the current shadow.

```
\setcircle{<dimen:diameter>}{<dimen:x-coord>}{<dimen:y-coord>}
```

`\setcircle` defines the current circle to have a diameter given by the first parameter and a center defined by the second and third parameter.

```
\shiftcircle{<dimen:x-length>}{<dimen:y-length>}
```

`\shiftcircle` displaces the current circle by the vector described by its parameters.

```
\drawcircle<1 or 0:upper right quadrant><1 or 0:lower right quadrant>
    <1 or 0:lower left quadrant><1 or 0:upper left quadrant>
```

`\drawcircle` draws the current circle. More precisely, it draws those quadrants of the current circle for which the corresponding parameter is 1.

```
\drawcirclehead{<number:x-slope>}{<number:y-slope>}{<1 or 0>}
```

`\drawcirclehead` draws an arrowhead on the current circle, at the intersection with a directed line segment starting at the center with a slope determined by the first two parameters. If the third parameter is 1 (0) the arrowhead will point in a clockwise (counterclockwise) direction. The arrowhead will be advanced towards its tip by the distance `\circleheaddisp`. This control symbol is defined to be 2pt, but may be redefined by the user.

```
\abutcircleleft{<dimen:y-length>}
    {<balanced mathematical text>}{<expression program>}

\abutcircleright{<dimen:y-length>}
    {<balanced mathematical text>}{<expression program>}

\abutcirclebelow{<dimen:x-length>}
    {<balanced mathematical text>}{<expression program>}

\abutcircleabove{<dimen:x-length>}
    {<balanced mathematical text>}{<expression program>}
```

Each of these macros uses the `<balanced mathematical text>` to initialize an expression state (in the same way as `\vertex`) and then executes the `<expression program>`, which must establish a shadow. The material in the final expression state produced by this program is then placed in the expression state of the expression program containing the call of `\abutcircle...` , at a location such that shadow touches the current circle on the outside of this circle. For `\abutcircleleft` and `\abutcircleright` the first parameter gives the y-coordinate of the point at which the center is to be located. For `\abutcirclebelow` and `\abutcircleabove` the first parameter gives the x-coordinate.

Actually, the abutment is approximate. For `\abutcircleabove`, the shadow is abutted against three tangents to the current circle, that touch at the top of the circle and at the two points 45 degrees to the left and right of the top, and is then given the lowest of the three positions obtained by these abutments. The other three macros behave similarly.

An Example Consider the simple example in Figure 1. This call of `\diagram` contains a diagram program in which the four calls of `\vertex` place the expressions A, B, A', and B' at the four corners of a 100pt by 150pt rectangle. Then come four groups of five calls that draw edges along the sides of this rectangle and abut expressions to the middles of these edges.

In each group, `\setedge` determines the position of the edge, `\shadeedge` adjusts the end points to exclude the shadows of the expressions that have been placed at these points by `\vertex`, `\drawsolidedge` draws the edge as a solid line, and `\drawedgehead` places an arrowhead at the end of the edge. Then `\abut...` places an expression above, below, to the left, or to the right of the midpoint of the edge, so that its shadow touches the edge.

In the calls of `\vertex`, `{\border{3pt}{4pt}\rect}` is an expression program that enlarges the current rectangle by 3pt at the left and right and by 4pt at the top and bottom, and then establishes this expanded rectangle as the shadow. In the calls of `\abut...`, `{\border{2pt}{2pt}\octagon{3pt}}` is an expression program that enlarges the current rectangle by 2pt on each side and then defines the shadow to be an octagon inscribed in this expanded rectangle, with slanted edges of length 4.24pt.

The result is as follows:

$$\begin{tikzcd}
A \arrow[r, "c"] \arrow[d, "a"'] & B \arrow[d, "b"] \\
A' \arrow[r, "c'"'] & B'
\end{tikzcd}$$

```
\[
\diagram{
  \vertex 0,100:{A}{\border{3pt}{4pt}\rect}
  \vertex 150,100:{B}{\border{3pt}{4pt}\rect}
  \vertex 0,0:{A'}{\border{3pt}{4pt}\rect}
  \vertex 150,0:{B'}{\border{3pt}{4pt}\rect}

  \setedge 0,100,150,100:
  \shadeedge
  \drawsolidedge
  \drawedgehead{100}10
  \abutabove 75:{\textstyle c}{\border{2pt}{2pt}\octagon{3pt}}

  \setedge 0,0,150,0:
  \shadeedge
  \drawsolidedge
  \drawedgehead{100}10
  \abutbelow 75:{\textstyle c'}{\border{2pt}{2pt}\octagon{3pt}}

  \setedge 0,100,0,0:
  \shadeedge
  \drawsolidedge
  \drawedgehead{100}10
  \abutleft 50:{\textstyle a}{\border{2pt}{2pt}\octagon{3pt}}

  \setedge 150,100,150,0:
  \shadeedge
  \drawsolidedge
  \drawedgehead{100}10
  \abutright 50:{\textstyle b}{\border{2pt}{2pt}\octagon{3pt}}
}
\]
```

Figure 1: A Simple Example

4 The Macros for Category-Theory Diagrams

Now we describe the additional macros oriented towards category-theory diagrams. The main level program is

```
\ctdiagram{<diagram program>}
```

`\ctdiagram` is similar to `\diagram`, except that it executes `\ctsolid`, `\cthead`, and `\ctoutermid` (described below) before the `<diagram program>`, so that the category-theory macros for drawing edges will draw solid edges with arrowheads and will calculate midpoints of edges before shading or displacement.

Within a diagram program, one can call the following macros (in addition to the general macros described previously):

```
\ctvg<number:x-coord>,<number:y-coord>:
    {<balanced mathematical text>}{<expression program>}
```

```
\ctv<number:x-coord>,<number:y-coord>:{<balanced mathematical text>}
```

`\ctvg` is similar to `\vertex`, except that:

1. The `<balanced mathematical text>` is set in `\ctvertexstyle`. The control symbol `\ctvertexstyle` is defined to be `\displaystyle`, but may be redefined by the user.
2. The execution of the `<expression program>` is followed by a "standard expression program" that enlarges the current rectangle by `\ctvertexborderlr` on the left and right and by `\ctvertexbordertb` on the top and bottom, and then creates a rectangular shadow of the same size. The control symbols `\ctvertexborderlr` and `\ctvertexbordertb` are defined to be 3pt and 4pt respectively, but may be redefined by the user.

`\ctv` is similar to `\ctvg` except that only the standard expression program is executed.

```
\ctsolid
```

```
\ctdash
```

```
\ctdot
```

These macros cause subsequent executions of the edge-drawing macros described below to draw solid, dashed, or dotted edges respectively. Horizontal and vertical dashed edges are drawn by `\drawdashedge{7pt}{7pt}11`, but other dashed edges are drawn by `\drawdashedge{15pt}{7pt}01`. Dotted edges are drawn by `\drawdotedge{8pt}1`. (These conventions can be altered by redefining the macros `\zzctdrawdashedge` and `\zzctdrawdotedge`.)

```
\cthead
```

```
\ctnohead
```

`\cthead` (`\ctnohead`) causes subsequent executions of the edge-drawing macros described below to draw (not to draw) arrowheads.

```
\cten<number:x-start-coord>,<number:y-start-coord>,<number:x-end-coord>,
    <number:y-end-coord>:
```

`\cten` draws an edge from x-start to x-end, after shading the start and end points with any shadows associated with these points on the vertex list. The edge will be solid, dashed, or dotted depending upon whether `\ctsolid`, `\ctdash`, or `\ctdot` was called last. An arrowhead will or will not be placed at the end point depending upon whether `\cthead` or `\ctnohead` was called last.

```
\ctetg<number:x-start-coord>,<number:y-start-coord>,<number:x-end-coord>,
    <number:y-end-coord>;<number:x-coord>:{<balanced mathematical text>}

\ctebg<number:x-start-coord>,<number:y-start-coord>,<number:x-end-coord>,
    <number:y-end-coord>;<number:x-coord>:{<balanced mathematical text>}

\ctelg<number:x-start-coord>,<number:y-start-coord>,<number:x-end-coord>,
    <number:y-end-coord>;<number:y-coord>:{<balanced mathematical text>}

\cterg<number:x-start-coord>,<number:y-start-coord>,<number:x-end-coord>,
    <number:y-end-coord>;<number:y-coord>:{<balanced mathematical text>}
```

Each of these macros draws an edge in the same way as `\cten`, and then abuts the `<balanced mathematical text>` to the

top	for	`\ctetg`
bottom	for	`\ctebg`
left	for	`\ctelg`
right	for	`\cterg`

of the edge, with its center placed at the x-coordinate (for `\ctetg` or `\ctebg`) or y-coordinate (for `\ctelg` or `\cterg`) specified by the fifth parameter. The abutted expression is set in `\ctabutstyle`, with an octagonal shadow (of the shape produced by `\octagon`). This octagon will be inscribed in a rectangle obtained by bordering the expression by `\ctabutborderlr` on the left and right, and by `\ctabutbordertb` on the top and bottom; the length of the slanted sides of the octagon will be `\ctabutborderinset` times the square root of 2.

The relevant control symbols are defined to be:

`\ctabutstyle`	`\textstyle`
`\ctabutborderlr`	2pt
`\ctabutbordertb`	2pt
`\ctabutborderinset`	3pt

These symbols may be redefined by the user, but `\ctabutborderinsetdouble` must also be redefined so that its value is twice `\ctabutborderinset`.

`\ctetg` and `\ctebg` should not be used to draw a vertical edge; `\ctelg` and `\cterg` should not be used to draw a horizontal edge.

```
\ctetbg<number:x-start-coord>,<number:y-start-coord>,<number:x-end-coord>,
    <number:y-end-coord>;<number:x-coord>,<number:x-coord>:
    {<1 or 0>}{<1 or 0>}
    {<balanced mathematical text>}{<balanced mathematical text>}
```

`\ctetbg` draws a pair of edges in the same manner as `\cten` and then abuts the first `<balanced mathematical text>` above the pair, in the same manner as `\ctetg`, with its center placed at the x-coordinate specified by the fifth parameter, and abuts the second `<balanced mathematical text>` below the pair, in the same manner as `\ctebg`, with its center placed at the x-coordinate specified by the sixth parameter. If the seventh parameter is 1 (and `\cthead` has been called most recently), the arrowhead on the upper edge will occur at the end point; otherwise it will occur (pointing backwards) at the start point. The eighth parameter controls the arrowhead on the lower edge similarly. The distance between the edges will be twice the control symbol `\ctdoubleedgedisp`, which is defined to be 2pt, but may be redefined by the user.

`\ctetbg` should not be used to draw a vertical edge.

```
\ctelrg<number:x-start-coord>,<number:y-start-coord>,<number:x-end-coord>,
    <number:y-end-coord>;<number:y-coord>,<number:y-coord>:
    {<1 or 0>}{<1 or 0>}
    {<balanced mathematical text>}{<balanced mathematical text>}
```

`\ctelrg` draws a pair of edges in the same manner as `\cten` and then abuts the first `<balanced mathematical text>` to the left, in the same manner as `\ctetg`, with its center placed at the y-coordinate specified by the fifth parameter, and abuts the second `<balanced mathematical text>` to the right, in the same manner as `\ctebg`, with its center placed at the y-coordinate specified by the sixth parameter. If the seventh parameter is 1 (and `\cthead` has been called most recently), the arrowhead on the left edge will occur at the end point; otherwise it will occur (pointing backwards) at the start point. The eighth parameter controls the arrowhead on the right edge similarly. The distance between the edges will be twice the control symbol `\ctdoubleedgedisp`, which is defined to be 2pt, but may be redefined by the user.

`\ctelrg` should not be used to draw a horizontal edge.

```
\ctet<number:x-start-coord>,<number:y-start-coord>,<number:x-end-coord>,
    <number:y-end-coord>:{<balanced mathematical text>}

\cteb<number:x-start-coord>,<number:y-start-coord>,<number:x-end-coord>,
    <number:y-end-coord>:{<balanced mathematical text>}

\ctel<number:x-start-coord>,<number:y-start-coord>,<number:x-end-coord>,
    <number:y-end-coord>:{<balanced mathematical text>}

\cter<number:x-start-coord>,<number:y-start-coord>,<number:x-end-coord>,
    <number:y-end-coord>:{<balanced mathematical text>}

\ctetb<number:x-start-coord>,<number:y-start-coord>,<number:x-end-coord>,
    <number:y-end-coord>:{<1 or 0>}{<1 or 0>}
    {<balanced mathematical text>}{<balanced mathematical text>}

\ctelr<number:x-start-coord>,<number:y-start-coord>,<number:x-end-coord>,
    <number:y-end-coord>:{<1 or 0>}{<1 or 0>}
    {<balanced mathematical text>}{<balanced mathematical text>}
```

These macros behave similarly to their cousins described above, except that the fifth parameter (and also the sixth parameter in the case of `\ctetb` and `\ctelr`) is omitted. In its place, these macros use the x- or y-coordinate of the midpoint between the start and end points of the edge. If `\ctoutermid` (described below) has been called most recently, then the midpoint will be calculated from the start and end coordinates given as parameters to the macros. If `\ctinnermid` (described below) has been called most recently, then the midpoint will be computed after displacement and shading, so that it will be the midpoint of the actual line segment that is printed. (In the case of `\ctetb` and `\ctelr`, this midpoint will be calculated separately for the two edges that are printed.)

```
\ctoutermid
```

```
\ctinnermid
```

These macros control the calculation of edge midpoints as described above.

Within a expression program, one can call the following macros (in addition to the general macros described previously):

```
\ctgl{<balanced mathematical text>}
```

```
\ctgr{<balanced mathematical text>}
```

These macros are similar to `\leftghost` and `\rightghost` except that the `<balanced mathematical text>` is set in `\ctvertexstyle`.

```
\ctlptl{<balanced mathematical text>}
```

```
\ctlptr{<balanced mathematical text>}
```

```
\ctlpbr{<balanced mathematical text>}
```

```
\ctlpbl{<balanced mathematical text>}
```

These macros print a loop (three quarters of a circle) of diameter `\ctloopdiameter` on the exterior of the current rectangle, with its center at the

top left	for	`\ctlptl`
top right	for	`\ctlptr`
bottom right	for	`\ctlpbr`
bottom left	for	`\ctlpbl`

corner of the current rectangle, and with a clockwise arrowhead at the clockwise end of the loop. Then the `<balanced mathematical text>` is abutted to the

left	for	`\ctlptl`
right	for	`\ctlptr`
right	for	`\ctlpbr`
left	for	`\ctlpbl`

of the loop, with its center

above	for	`\ctlptl`
above	for	`\ctlptr`
below	for	`\ctlpbr`
below	for	`\ctlpbl`

the center of the loop by the distance 5pt.

The control symbols `\ctloopdiameter` and `\ctabutcircledisp` are defined to be 20pt and 5pt respectively, but may be redefined by the user.

The current rectangle is expanded by `\ctvertexborderlr` at the left and right and by `\ctvertexbordertb` at the top and bottom before the loop center is determined, and is contracted to its original size afterwards. Thus the loop center will lie at a corner of the shadow that will be produced by the "standard expression program" executed by `\ctvg`. (Actually, the loop center is displaced by `\circleheaddisp`, so that the tip of the arrowhead will just touch the shadow.) The arrowhead is always printed, regardless of the use of `\cthead` and `\ctnohead`.

The `<balanced mathematical text>` is set in `\ctabutstyle`, and is given an octagonal shadow in the same manner as by `\ctetg`. The abutment to the loop is similar to that performed by `\abutcircleleft` or `\abutcircleright`.

```
\ctlptlcc{<balanced mathematical text>}

\ctlptrcc{<balanced mathematical text>}

\ctlpbrcc{<balanced mathematical text>}

\ctlpblcc{<balanced mathematical text>}
```

These macros are similar to their cousins described above, except that a counterclockwise arrowhead is placed at the counterclockwise end of the loop.

An Example The following program produces the same display as the previous example.

```
\[
\ctdiagram{
  \ctv 0,100:{A}
  \ctv 150,100:{B}
  \ctv 0,0:{A’}
  \ctv 150,0:{B’}
  \ctet 0,100,150,100:{c}
  \cteb 0,0,150,0:{c’}
  \ctel 0,100,0,0:{a}
  \cter 150,100,150,0:{b}
}
\]
```

Less trivial examples of the usage of these macros are found in Section 6 and in `diagmactest.tex`.

5 Extensions

This section describes two macros that have been added to `diagmac2.sty` by Bob Tennent (and are not in Reynolds's original `diagmac`).

```
\drawedgebar
```

`\drawedgebar` draws a bar across the end of the current edge.

```
\ctec<number:x-start-coord>,<number:y-start-coord>,
     <number:x-end-coord>,<number:y-end-coord>,
     <number:x-ctrl-coord>,<number:y-ctrl-coord>:{<balanced mathematical text>}
```

`\ctec` draws a bezier-curve edge from the start point to the end point, using the third point as a control point. The `<balanced mathematical text>` is centered at the control point.

6 Examples

$$\begin{tikzcd}
a \arrow[r, mapsto, "f"] \arrow[d, mapsto, "g"'] & f(a) \arrow[d, mapsto, "g"] \\
g(a) \arrow[r, mapsto, "f"'] & f(g(a)) = g(f(a))
\end{tikzcd}$$

```
\[
\ctdiagram{
  \def\diagramunit{0.8pt}
  \ctinnermid
  \ctv 0,100:{a}
  \ctv 150,100:{f(a)}
  \ctv 0,0:{g(a)}
  \ctv 150,0:{f(g(a))=g(f(a))}
  \ctet 0,100,150,100:{f}
  \drawedgebar
  \cteb 0,0,150,0:{f}
  \drawedgebar
  \ctel 0,100,0,0:{g}
  \drawedgebar
  \cter 150,100,150,0:{g}
  \drawedgebar
}
\]
```

$$\begin{tikzcd}
I \otimes (I \otimes v) \arrow[r, "\alpha^{-1}"'] \arrow[rr, bend left, "\mathsf{id}"] \arrow[rdd, "I \otimes \lambda_v"'] & (I \otimes I) \otimes v \arrow[r, "\alpha^{-1}"'] \arrow[d, dash] & I \otimes (I \otimes v) \arrow[ldd, "\lambda_I \otimes v"] \\
 & \rho_I \otimes v = \lambda_I \otimes v \arrow[d] & \\
 & I \otimes v &
\end{tikzcd}$$

```
\[
\ctdiagram{
  \ctv 0,0: {I\otimes v}
  \ctv 0,60: {(I\otimes I)\otimes v}
  \ctv -80,60: {I\otimes (I\otimes v)}
  \ctv 80,60: {I\otimes (I\otimes v)}
  \ctelg -80,60,0,0;25:{I\otimes\lambda_v\!}
  \cterg 80,60,0,0;25:{\,\lambda{_I\otimes v}}
  \ctec -80,60,80,60,0,80:{\mathsf{id}}
  \cteb -80,60,0,60:{\alpha^{-1}}
  \cteb 0,60,80,60:{\alpha^{-1}}
  \ctv 0,36:{\rho_I\otimes v =
            \lambda_I\otimes v}
  \ctnohead
  \cten 0,60,0,36:
  \cthead
  \cten 0,36,0,0:
}
\]
```

$$\begin{tikzcd}
V \arrow[r, dash] & I \arrow[r] & E \arrow[ll, bend right, "\delta_0"'] \arrow[ll, bend left, "\delta_1"]
\end{tikzcd}$$

```
\[
\ctdiagram{
  \ctv -60,0:{V}
  \ctv 60,0:{E}
  \ctv 0,0: {I}
  \cten 0,0,60,0:
  \ctnohead
  \cten -60,0,0,0:
  \ctet -50,20,50,20:{\delta_0}
  \setcircle{20pt}{\xstart}{\ystart}
  \shiftcircle{0pt}{-10pt}
  \drawcircle0001
  \drawcirclehead{-1}{0}{0}
  \setcircle{20pt}{\xend}{\yend}
  \shiftcircle{0pt}{-10pt}
  \drawcircle1000
  \cteb -50,-18,50,-18:{\delta_1}
  \setcircle{20pt}{\xstart}{\ystart}
  \shiftcircle{0pt}{10pt}
  \drawcircle0010
  \drawcirclehead{-1}{0}{1}
  \setcircle{20pt}{\xend}{\yend}
  \shiftcircle{0pt}{10pt}
  \drawcircle0100
}
\]
```

$$\begin{tikzcd}
\widetilde{E} \arrow[drr, bend left, "p"] \arrow[dr, dotted, "\hat{p}"] \arrow[ddr, bend right, "\tilde{q}"'] & & \\
& E_b \arrow[r, "\overline{b}"'] \arrow[d, "q_b"] \arrow[dr, phantom, "\lrcorner", very near start] & E \arrow[d, "q"] \\
& \widetilde{B} \arrow[r, "b"'] & B
\end{tikzcd}$$

```
\[
\ctdiagram{
  \ctv 0,0:{\widetilde{B}}
  \ctv 60,0:{B}
  \ctv 0,60:{E_b}
  \ctv 60,60:{E}
  \ctinnermid
  \cteb 0,0,60,0:{b}
  \cter 0,60,0,0:{q_b}
  \cteb 0,60,60,60:{\overline{b}}
  \cter 60,60,60,0:{q}
  \ctv -40,100: {\widetilde{E}}
  \ctec -40,100,0,0,-40,50:{\tilde{q}}
  \ctec -40,100,60,60,10,100:{p}
  \def\ctvertexborderlr{1pt}
  \def\ctvertexbordertb{1pt}
  \ctv -20,80:{\hat{p}}
  \ctnohead\ctdot
  \def\zzctdrawdotedge{\drawdotedge{2.5pt}1}
  \cten -40,100,-20,80:
  \cthead
  \def\zzctdrawdotedge{\drawdotedge{2.5pt}0}
  \cten -20,80,-6,66:
  \ctv 5,55:{\mbox{\Large$\lrcorner$}}
}
\]
```

$$\begin{tikzcd}
 & I\otimes((v\otimes I)\otimes w) \arrow[rrr, "I\otimes\alpha^{-1}"] \arrow[drr, dash] & & & I\otimes(v\otimes(I\otimes w)) \arrow[drr, "\alpha^{-1}"] \arrow[dd, dash] & & \\
I\otimes((I\otimes v)\otimes w) \arrow[ur, "I\otimes\sigma_{I,v}\otimes w"] \arrow[ddrr, dash] & & & I\otimes(\rho_v\otimes w) \arrow[ddddr] & & & (I\otimes v)\otimes(I\otimes w) \arrow[dd, "(I\otimes v)\otimes\lambda_w"] \\
 & & & & I\otimes(v\otimes\lambda_w) \arrow[ddd] & & \\
I\otimes(I\otimes(v\otimes w)) \arrow[uu, "I\otimes\alpha^{-1}"] \arrow[drr, dash] & & I\otimes(\lambda_v\otimes w) \arrow[ddrr] & & & & (I\otimes v)\otimes w \arrow[dd, "\lambda_v\otimes w"] \\
 & & I\otimes\lambda_{v\otimes w} \arrow[drr] & & & \alpha^{-1} \arrow[ur] & \\
(I\otimes I)\otimes(v\otimes w) \arrow[uu, "\alpha^{-1}"] \arrow[rrrr, "\rho_I\otimes(v\otimes v)"] & & & & I\otimes(v\otimes w) \arrow[rr, "\lambda_{v\otimes w}"] \arrow[ur, dash] & & v\otimes w
\end{tikzcd}$$

```
\[
\ctdiagram{\ctinnermid
  \ctv 0,0: {(I\otimes I) \otimes(v\otimes w)}
  \ctv 0,60: {I\otimes \bigl(I\otimes(v\otimes w)\bigr)}
  \ctv 0,120: {I\otimes \bigl((I\otimes v)\otimes w\bigr)}
  \ctv 80,180: {I\otimes\bigl((v\otimes I)\otimes w\bigr)}
  \ctv 200,0: {I\otimes (v\otimes w)}
  \ctv 200,180:{I\otimes \bigl(v\otimes(I\otimes w)\bigr)}
  \ctv 280,0:{v\otimes w}
  \ctv 280,60:{(I\otimes v)\otimes w}
  \ctv 280,120: { (I\otimes v)\otimes(I\otimes w)}
  \ctel 0,0,0,60:{\alpha^{-1}}
  \ctel 0,60,0,120:{I\otimes \alpha^{-1}}
  \ctet 0,0,200,0:{\rho_I\otimes(v\otimes v)}
  \ctet 200,0,280,0:{\lambda_{v\otimes w}}
  \ctelg 0,120,80,180;155:{I\otimes\sigma_{I,v}\otimes w}
  \ctet 80,180,200,180:{I\otimes \alpha^{-1}}
  \cterg 200,180,280,120;155:{\alpha^{-1}}
  \cter 280,120,280,60:{(I\otimes v)\otimes \lambda_w}
  \cter 280,60,280,0: {\lambda_v\otimes w}
  \ctv 240,30:{\alpha^{-1}}
  \ctnohead\cten 200,0,240,30:
  \cthead\cten 240,30,280,60:
  \ctv 100,30:{I\otimes\lambda_{v\otimes w}}
  \ctnohead\cten 0,60,100,30:
  \cthead\cten 100,30,200,0:
  \ctv 100,60:{I\otimes(\lambda_v\otimes w)}
  \ctnohead\cten 0,120,100,60:
  \cthead \cten 100,60,200,0:
  \ctv 120,120:{I\otimes (\rho_v\otimes w)}
  \ctnohead\cten 80,180,120,120:
  \cthead\cten 120,120,200,0:
  \ctv 200,90:{I\otimes (v\otimes \lambda_w)}
  \ctnohead\cten 200,180,200,90:
  \cthead\cten 200,90,200,0:
}
\]
```

$$\begin{tikzcd}
RW^{\mathsf{op}} \arrow[r, shift left=2, "\widetilde{F}"] \arrow[r, shift right=2, "\widetilde{G}"'] \arrow[r, phantom, "\Downarrow\widetilde{\eta}"] \arrow[d, "rw^{\mathsf{op}}"'] & RS \arrow[d, "rs"] \\
W^{\mathsf{op}} \times W^{\mathsf{op}} \arrow[r, shift left=2, "F_0 \times F_1"] \arrow[r, shift right=2, "G_0 \times G_1"'] \arrow[r, phantom, "\Downarrow\eta_0 \times \eta_1"] & S \times S
\end{tikzcd}$$

```
\newcommand{\op}{\mathsf{op}}
\newcommand{\vnat}{\Downarrow\mskip-\medmuskip}
\[
\ctdiagram{
  \ctinnermid
  \ctv 0,0:{W^\op\times W^\op}
  \ctv 180,0:{S\times S}
  \ctv 0,60:{RW^\op}
  \ctv 180,60:{RS}
  \ctel 0,60,0,0:{\mathit{rw}^\op}
  \cter 180,60,180,0:{\mathit{rs}}
  \def\ctdoubleedgedisp{6.5pt}
  \ctetb 0,60,180,60:11{\widetilde{F}}{\widetilde{G}}
  \ctv 98,60:{\vnat\widetilde{\eta}}
  \ctetb 0,0,180,0:11{F_0\times F_1}{G_0\times G_1}
  \ctv 112,0:{\vnat\eta_0\times\eta_1}
}
\]
```

See also the examples in Reynolds's `diagmactest`.