Brackets around anything

Timothy Hall

Introduction

It is often the case when labeling terms in a figure that the label itself cannot be placed so that it refers to an unambiguous term in the figure. For example, consider the label h in Figure 1.

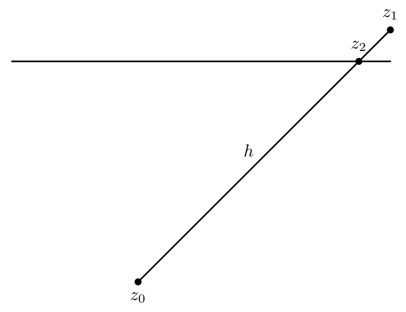

Figure 1: Ambiguous label h

Does the h refer to the segment between points z_0 and z_1, or only between points z_0 and z_2? It would be clearer if there were a bracket explicitly delineating the term in the figure corresponding to h, such as that found in Figure 2.

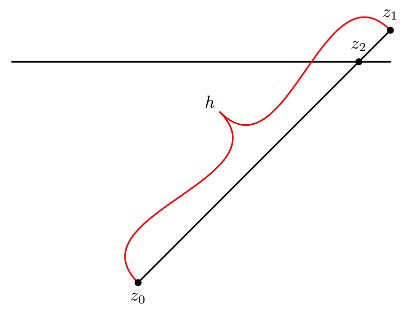

Figure 2: Clear label h

The placement of a bracket around terms of interest in a figure may be accomplished through the use of the following flexible and easy to use general purpose METAPOST[1] definition.

```
def sbrack(suffix r,s,d) text c=
   save a,v;
   numeric a;
   pair v;
   a=angle(z.r-z.s);
   v=0.5[z.r,z.s]+((d*pt,0)
                   rotated (a+90));
   draw z.r{dir (a+90)}
      ..{dir (a+90)}v{dir (a-90)}
      ..{dir (a-90)}z.s
      withcolor c;
enddef;
```

Its syntax is

`sbrack(#,#,size)color`

where the first # is the bracket's starting point, the second # is the bracket's ending point, and *size* is the distance (in standard METAPOST points) from the midpoint between the starting and ending points to the "vertex" of the bracket (the distance being measured perpendicular to the line segment[2] between the two points). The *color* refers to the drawing color[3] of the bracket, and may be specified by a standard name, such as "red," or by an RGB triple, such as $(1, 0, 1)$ for magenta, or by any other representation recognized by the METAPOST modifier `withcolor`.

Note that the order of the points listed in the definition is important. The `sbrack` utility always draws the bracket *to the right* as it progresses from the starting point to the ending point. So if the bracket in Figure 2 were drawn by

`sbrack(1,0,36)red;`

then the use of

`sbrack(0,1,36)red;`

would draw the bracket on the other side of the line (see Figure 3).

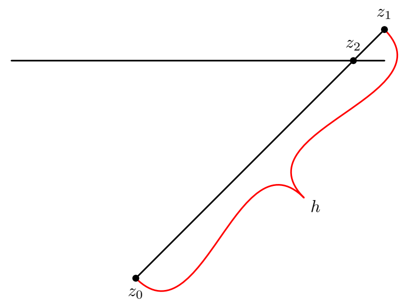

Figure 3: Bracket on other side

[1] Note that, except for the `withcolor` modifier, this definition could also be a METAFONT definition, should the need ever arise.

[2] A line segment need not be part of the figure for the midpoint to be calculated; see Figure 5.

[3] For black and white printers and displays, the `withcolor` modifier will result as a "shade of gray" rendering of the corresponding color.

Placement of labels

The label h in Figure 2 was placed by

```
label.lft(%
   btex $h$ etex,
   0.5[z0,z1]
   +((40pt,-6*sind(angle(z1-z0))*pt)
   rotated(angle(z1-z0)+90)));
```

Since the vertex of the bracket was placed 36 points from the line (in this particular case), this label is placed to the left of the vertex by an additional 4 points (for a total of 40 points). The term involving the sine (in degrees) ensures that the label will be "pointed to" by the curvature of the path around the vertex of the bracket.

The label h in Figure 3 was placed by

```
label.lft(%
   btex $h$ etex,
   0.5[z0,z1]
   +((40pt,4*sind(angle(z0-z1))*pt)
   rotated(angle(z0-z1)-90)));
```

which only differs from before in the order of the points (z_0 and z_1 are interchanged), and that positive 4 is used instead of negative 6, and the rotation angle is 180 degrees from the previous form. The choice of 4 or 6 is subjective and depends on the particular circumstances where it is used[4] (and on the preferences of the user). However, the positive and negative signs, as well as the +90 degrees and −90 degrees choice in the rotation, must be used correctly depending on which side the bracket is drawn. Using the wrong values will result in (a) the label being on the opposite side as the bracket, or (b) the label being significantly misaligned, and therefore distracting.

The label itself may be rotated to align consistently with the bracket itself. For example, in Figure 4, the label was produced by

```
label.lft(%
   btex $\left\|\rho\right\|$ etex,z3)
   rotatedabout(z3,angle(z0-z1)-90);
```

where z_3 is the vertex of the bracket.[5] The complete METAPOST code for this example is instructional, as it combines several aspects of displaying a bracketed figure.

[4] In the example given, the nature of the ascender on the left side of the letter "h" is the primary reason for using 4 rather than 6.

[5] Note `rotatedabout` or `rotatedaround` must be used, rather than `rotated`.

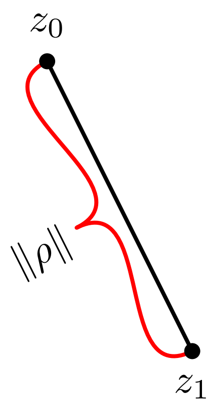

Figure 4: TEX Label

```
beginfig(4);
   pickup pencircle scaled1pt;

   z0=(0,0);
   z1-z0=(0.50in,-1.00in);
   z3=0.5[z1,z0]
     +((12pt,2*sind(angle(z0-z1))*pt)
      rotated(angle(z0-z1)+90));

   draw z0..0.5[z0,z1]..z1;
   sbrack(0,1,12)red;

   pickup pencircle scaled4pt;
   drawdot z0;
   drawdot z1;

   label.top(btex $z_0$ etex,z0);
   label.bot(btex $z_1$ etex,z1);
   label.lft(%
      btex $\left\|\rho\right\|$etex,z3)
      rotatedabout(z3,angle(z0-z1)-90);
endfig;
```

Straight line segments are not required for `sbrack` to work properly, since its definition only requires non-identical points, a spacing measure, and a drawing color (see Figure 5). None of these parameters have a default value. Note that it is up to the individual user to place the brackets and labels drawn by `sbrack` so that they do not interfere with any other terms in the figure.

Summary example

Figure 6 summarizes the flexibility of the `sbrack` METAPOST definition. It is drawn by the following program:

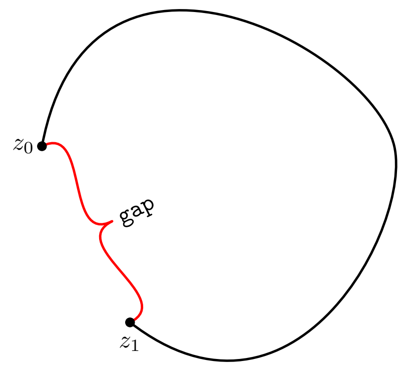

Figure 5: Indicator Label

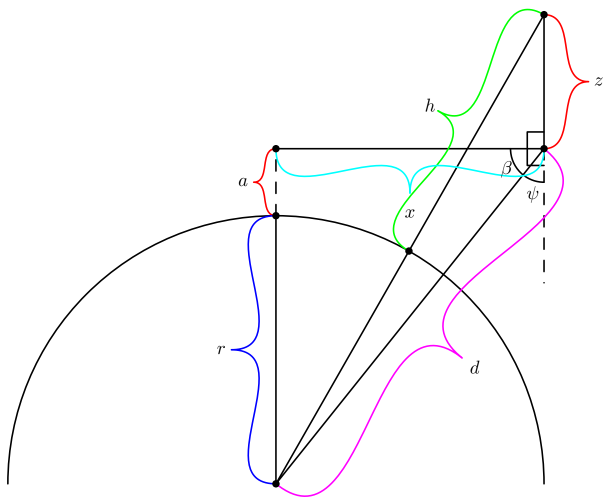

Figure 6: Summary Example

```
beginfig(6);
   path pth[];
   pickup pencircle scaled1pt;
   pth[1]=halfcircle scaled4in;

   z0=(0,0); z1-z0=(0,2in);
   z1'-z1=(0,0.5in); z1''=0.5[z1,z1'];
   z2-z1'=(2in,0);
   pth[2]=quartercircle scaled1/2in
          rotated180 shifted z2;
   z2'-z2=(0,-1in); z3-z2=(0,1in);
   z4=pth[1] intersectionpoint (z0--z3);
   z5'-z2=(-1/8in,0); z5''-z2=(0,1/8in);
   z5'''-z2=(0,-1/8in);
   z6=pth[2] intersectiontimes (z0--z2);
   pth[3]=subpath (0,x6) of pth[2];
   pth[4]=subpath (x6,2) of pth[2];

   draw halfcircle scaled 4in;
   draw z1'--z2; draw z0--z1;
   draw z0--z2; draw z2--z3;
   draw z0--z3;
   draw z2--z2' dashed evenly scaled2;
   draw z1--z1' dashed evenly scaled2;
   draw z5'--(x5',y5'')--z5'';
   draw z5'--(x5',y5''')--z5''';
   draw pth[3]; draw pth[4];
   sbrack(1',1,12)red;
   sbrack(3,4,24)green;
   sbrack(0,2,36)(1,0,1);
   sbrack(1',2,24)(0,1,1);
   sbrack(2,3,24)red;
   sbrack(1,0,24)blue;

   pickup pencircle scaled4pt;
   drawdot z0; drawdot z1; drawdot z1';
   drawdot z2; drawdot z3; drawdot z4;

   label.top(btex $x$ etex,
             0.5[z1',z2]+((40pt,
             -6*sind(angle(z1'-z2))*pt)
             rotated(angle(z1'-z2)+90)));
   label.rt(btex $d$ etex,
            0.5[z0,z2]+((40pt,
            4*sind(angle(z0-z2))*pt)
            rotated(angle(z0-z2)+90)));
   label.rt(btex $z$ etex,
            0.5[z2,z3]+(24pt,0));
   label.lft(btex $r$ etex,
             0.5[z0,z1]-(24pt,0));
   label.lft(btex $a$ etex,
             z1''-(12pt,0));
   label.lft(btex $h$ etex,
             0.5[z3,z4]+((24pt,
             -4*sind(angle(z3-z4))*pt)
             rotated(angle(z3-z4)+90)));
   label.lft(btex $\beta$ etex,
             point 3*x6/4 of pth[2]);
   label.bot(btex $\psi$ etex,
             point (1+x6/2) of pth[2]);
endfig;
```

Technical note

The term d in the sbrack definition is actually not a **suffix** as it is labeled. It is a **text** argument; indeed, a numeric that is joined with `*pt` to form the x-value of a **pair**. However, the syntax of sbrack usage is simplified, i.e., made to be of shortest character length, by placing d with the suffixes, rather than as a separate text or expression. To wit:

```
    def sbrack(suffix r,s)(text d)text c =
```

is cumbersome, not only for the redundant use of **text**, but also because the syntax would then be

```
    sbrack(0,1)(24)red;
```

and for

```
    def sbrack(suffix r,s)(text d,c) =
```

the calling syntax would be

```
    sbrack(0,1)(24)(red);
```

both of which require more characters than the given definition. In fact, using **expr** in place of **text** would be inappropriate, since neither d nor c should be evaluated. However, if a further development or enhancement were made to the `sbrack` definition to allow d to be a variable, then the use of **expr**, in this context, would be not only appropriate, but required.

Further development and enhancements

A user may, of course, change any of the parameters found in the definition of `sbrack`, such as changing the relative position of the vertex (perhaps 75% of the way from the starting point to the ending point).

Another possibility is redefining the position of the label (relative to the vertex) to be on the opposite side of the vertex as is now given. In this way, the label would appear between the bracket and a line segment between the defining points.

Yet another value-added enhancement would be for the macro to calculate a default bracket spacing value, perhaps in such a way that the bracket does not “collide” with the term it is enclosing, and does not interfere with any other terms in the figure.

Some intriguing results come from adding *tension* statements to the curves generated by the *draw* command, as well as explicitly defining the control points of a curve in terms of the label position. Any such further developments and enhancements may be incorporated into the `sbrack` definition to meet any particular METAPOST (or METAFONT) need.

◇ Timothy Hall
PQI Consulting
P. O. Box 425616
Cambridge, MA 02142-0012
`http://www.pqic.com/TUG/baa.htm`