Fonts

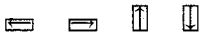
Arrows for Technical Drawings

David Salomon

Introduction

A general note: Square brackets are used throughout this article to refer to *The TEXbook*. Thus [437] refers to page 437.

Arrows, both vertical and horizontal, are common in technical diagrams. Unfortunately, the arrows available in TEX (`\rightarrow`, `\Rightarrow`, `\longrightarrow` [437], & `\rightarrowfill` [226]) are of limited use. The arrowheads are available in only one size and style, they already have short rules attached and, as the diagrams below show, they are inconveniently positioned in their bounding boxes.



The left and right arrows are shorter than their bounding boxes, and are not vertically centered in the boxes. The up and down arrows are narrower than their boxes, and have depths (of approximately 1.944pt). As a result, the simple construct

```
\def\vrulefill{\leaders\vrule\vfill}
\vbox to25pt{\offinterlineskip
 \hbox{$\uparrow$}\vrulefill
 \hbox{$\downarrow$}}
```

creates

and something more complicated is needed to align the rule with the arrows. Also the vertical size of the construction above is 26.944pt, not 25pt, because of the depth of the downarrow. Similarly, the result of

```
\hbox to25pt{$\leftarrow$%
 \hrulefill$\rightarrow$}
```

is

Also, since each arrow is about 8pt long, very short double arrows are impossible to create. Something such as

```
\hbox to10pt{$\leftarrow$%
 \hrulefill$\rightarrow$}
```

causes an 'overfull box'.

LATEX offers more arrows in its `line` font. They can point in quite a few directions, but there is only one style.

Description of the arrowheads

To satisfy my personal needs, I decided to develop a font for arrowheads that will be well documented and easy to use, yet general enough to produce arrowheads of many shapes. An important requirement was that the arrowheads be easy to place at the tips of rules. Since TEX does not have diagonal rules, only horizontal and vertical arrowheads were developed. The methods used here, however, can easily be extended for diagonal arrowheads.

The discussion below assumes a right-pointing arrowhead, but the results can easily be applied to the three other directions. A general arrowhead (see Figure 1) is defined by five points, $z_1 \ldots z_5$, of which z_4 is the origin, and z_5 is a reflection of z_2 (about the horizontal line at height `.5ruledim`). The two front edges are curved, the two back ones are straight, and there is a flat area at the base, to attach a rule. The exact shape of the arrowhead depends on the following parameters:

- `wd` is the distance from the tip to the base of the arrowhead. The bounding box has width `wd`.
- `tail` is the distance from the base to the ends of the wings. The total width of the arrowhead is, therefore, `wd+tail`, but only `wd` units are included in the bounding box; the rest sticks out of it. Negative values of `tail` produce arrowheads shaped like ◆, and large positive values (>`wd`) create arrowheads shaped like >. `tail`, therefore, should normally vary in the narrow range 0...`wd`.
- `ht` is the (approximate) total height of the arrowhead. The bounding box has height `.5ht` (and zero depth). Very tall arrowheads, such as ▶, are rarely used, so `ht` should normally be less than the total width of the arrowhead. Because of the special way arrows are used (see below), the bounding box has no depth. As a result, the left- and right-pointing arrowheads (and, normally, the upward one as well) stick below their boxes.
- The height of a standard `\hrule` is 0.4pt, so it makes sense to center the arrowhead 0.2pt above the baseline. However, to allow for any size rule, there is a parameter, `ruledim`, whose value should be the height of the rule to which the arrowhead is going to be attached. The arrowhead is centered `.5ruledim` above the baseline.

Points z_3, z_4 guarantee that, regardless of the shape of the arrowhead, there will be a flat area of size `ruledim` at the base of the arrowhead, so it can be smoothly connected to the rule. A close look at the code shows that the height of the arrowhead ($z_2 - z_5$) is `ht-ruledim` so, in order to end up with something that looks like an arrowhead, `ht` should

be greater than `ruledim`. (In rare cases, such as the 'pacman' arrowhead below, a large negative value of `curv` can increase the height of the arrowhead above this value.)

- The `curv` parameter can be used to curve the front edge of the arrowhead. Its value (in degrees) is added to the direction of the top front edge, and subtracted from that of the bottom front edge. Thus positive values of `curv` result in arrowheads looking like ➤, and negative values, in arrowheads like ➤. As a result, negative values of `curv` would rarely be used. The maximum value of `curv` (see discussion below) depends on the size and shape of the arrowhead, and is typically between 20° and 30°.
- `outlin` is a boolean parameter. If it is *true*, the arrowhead is drawn as an outline ➢, using the procedure suggested in [Ex. 13.23]; otherwise, the arrowhead is solid. For high resolution output devices, Doug Henderson's methods (ref. 1) create better results.

The source code

In a complete arrowhead font, all the characters are arrowheads, differing only in orientation and parameters. It is therefore natural to define the arrowheads in terms of procedures. I have found it convenient to use two procedures, one for left/right arrowheads and the other for up/down ones. The METAFONT code of the procedures is as follows.

```
path outerr;
def outlne = % Outlining, see Ex. 13.23
cull currentpicture keeping (1,infinity);
picture v; v:=currentpicture;
cull currentpicture keeping (1,1)
 withweight 3;
addto currentpicture also v
 - v shifted right
 - v shifted left
 - v shifted up
 - v shifted down;
cull currentpicture keeping (1,4)
enddef;

% procedure for right left arrowheads
def lr_head(text lr) =
R:=floor ruledim; if not odd R: R:=R+1; fi;
z1=(w,.5R); z2=(-tail,h);
z3=(0,2y1); z4=origin; z5=(x2,R-y2);
sAngle:=angle(z2-z1)+curv;
eAngle:=angle(z1-z5)-curv;
outerr:=z1{dir sAngle}..z2--z3--
 z4--z5..{dir eAngle}cycle;
if lr="r":
 fill outerr; if outlin: outlne; fi
elseif  lr="l":
 fill outerr
  reflectedabout((0,0),(0,1))
  shifted(w,0);
 if outlin: outlne; fi
```

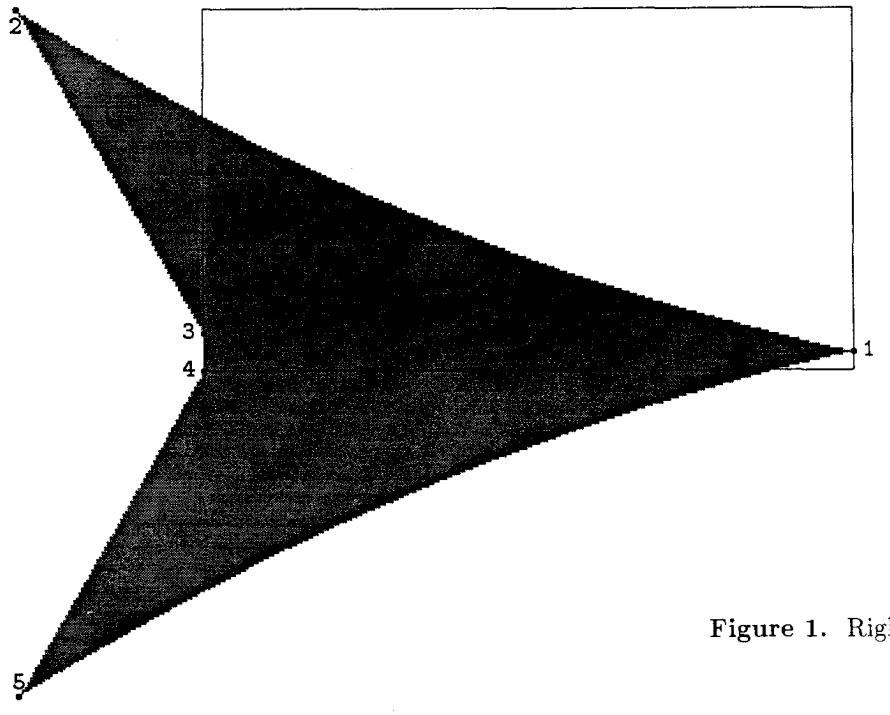


Figure 1. Right arrowhead

```
else: errmessage("wrong parameter,
 should be 'l' or 'r'");
fi
enddef;

% procedure for up down arrowheads
def ud_head(text ud) =
R:=floor ruledim; if not odd R: R:=R+1; fi;
z1=(.5R,h); z2=(w,-tail);
z3=(2x1,0); z4=(0,0); z5=(R-x2,y2);
sAngle:=angle(z2-z1)-curv;
eAngle:=angle(z1-z5)+curv;
outerr:=z1{dir sAngle}..z2--z3--
 z4--z5..{dir eAngle}cycle;
if ud="u":
 fill outerr; if outlin: outlne; fi
elseif  ud="d":
 fill outerr
  reflectedabout((0,0),(1,0))
  shifted(0,h);
 if outlin: outlne; fi
else: errmessage("! Wrong parameter,
 should be 'u' or 'd'");
fi
%
```

Following this, arrowheads can be created and placed in the font by, e.g.:

```
ruledim#:=.4pt#; outlin:=false;
ht#:=8pt#; wd#:=7pt#; tail#:=-2pt#; curv:=0;
define_pixels(ht,wd,tail,ruledim);

beginchar("R",wd#,.5ht#,0); "right";
lr_head("r");
endchar;

beginchar("L",wd#,.5ht#,0); "left";
lr_head("l");
endchar;

beginchar("U",.5ht#,wd#,0); "up";
ud_head("u");
endchar;

beginchar("D",.5ht#, wd#,0); "down";
ud_head("d");
endchar;
```

Note that it is also possible to create a hollow arrowhead by:

1. Drawing it with a 2-pixel wide pen. This may give better results in low resolution output devices.

```
if outlin: pickup pensquare scaled 2;
 draw outerr; fi
```

2. After creating the arrowhead, a smaller arrowhead is erased inside. By changing the scale and shift amounts, special shapes can be created.

```
path iner
fill outerr
iner:=outerr;
if outlin: erase fill iner scaled .8
 shifted(.1x1,.2y1); fi
```

An an example, the values

```
ht#:=8pt#; wd#:=5pt#; tail#:=2pt#;
curv:=9; ruledim#:=.4pt#;
```

produce ➤ when `outlin:=false;` and ➢ when `outlin:=true;`.

Improving the digitization

The only subtle point about the procedures above is the equation for z_1. Originally this equation was '`z1=(w,.5ruledim);`' but this resulted in arrowheads with flat tips, two pixels tall. To get a sharp, one-pixel tip, the y_1 coordinate should be an integer plus 1/2 (see Ex. 24.7 in *The METAFONTbook*). This was obtained by computing '`R:=floor ruledim; if not odd R: R:=R+1; fi;`' and setting '`z1=(w,.5R);`'. (`R` is the odd integer closest to `ruledim`, so `.5R` is an integer plus 1/2.) To end up with a symmetric arrowhead, the equation for z_5 was also changed from '`z5=(x2,ruledim-y2);`' to '`z5=(x2,R-y2);`'. Notice that the base of the arrowhead (the distance between points z_3, z_4) is now `R` and not `ruledim`, but the difference is at most one pixel, and is not noticeable. Notice also that the whole thing may not be necessary in a high-resolution output device, but in a 300dpi laser printer it significantly improves the appearance of the arrowhead (see Fig. 2).

Special cases

The top front edge of the arrowhead should go in the general direction of the top left point. If that direction is changed too much (by a large value of `curv`), funny results — and, sometimes, error messages — are obtained. As an example, the following set of parameters

```
ht#:=10pt#; wd#:=10pt#; tail#:=10pt#;
curv:=20; ruledim#:=.4pt#;
```

produces ➢ —.

Some combinations of the parameters create interesting (and, possibly, even useful) shapes, even though they don't look like arrows. A pacman ◖ is a left arrowhead created by:

```
ruledim#:=0pt#; outlin:=false; ht#:=2pt#;
wd#:=4pt#; tail#:=4pt#; curv:=-85;
```

A square diamond ◆, is created by: `ht#:=10pt#; wd#:=10pt#; tail#:=-5pt#; curv:=0;`

A circular wedge ◀, is the result of: `ht#:=10pt#; wd#:=10pt#; tail#:=-7pt#; curv:=-30;`

The examples shown here make it clear that the arrowheads are not meant to be used with text. Specifically, they don't have any depth, which interferes with the normal interline spacing, and they stick out of their boxes, which messes up the interword spacing.

Using the arrowheads

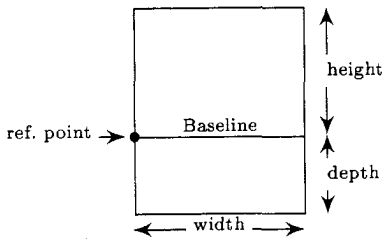
The arrowheads are meant to be used in diagrams, stuck at the ends of rules. A horizontal double arrow of size .5in ⟷ is obtained by `\hbox to.5in{\ar l\hrulefill r}`. A vertical arrow is also easy to create by means of vertical leaders. The ones shown here:

have been produced by

```
\def\vrulefill{\leaders\vrule\vfill}
\vbox to30pt{\offinterlineskip
 \hbox{\ar u}\vrulefill\hbox{\ar d}}
\def\vRulefill{\leaders\vrule width1pt\vfill}
\vbox to30pt{\offinterlineskip
 \hbox{\ar U}\vRulefill\hbox{\ar D}}
```

where positions 'u', 'd' of font `\ar` are occupied by up and down arrowheads with a base 0.4pt wide, and positions 'U', 'D' have similar arrowheads with 1pt wide bases.

As an example, the well known diagram [63] is duplicated here, using our arrowheads.



Bibliography

1. Henderson, D. *Outline fonts with METAFONT*, *TUGboat* 10, no. 1, 36–38, April 1989.

◇ David Salomon
California State University,
Northridge
Computer Science Department
Northridge, CA 91330
dxs@ms.secs.csun.edu

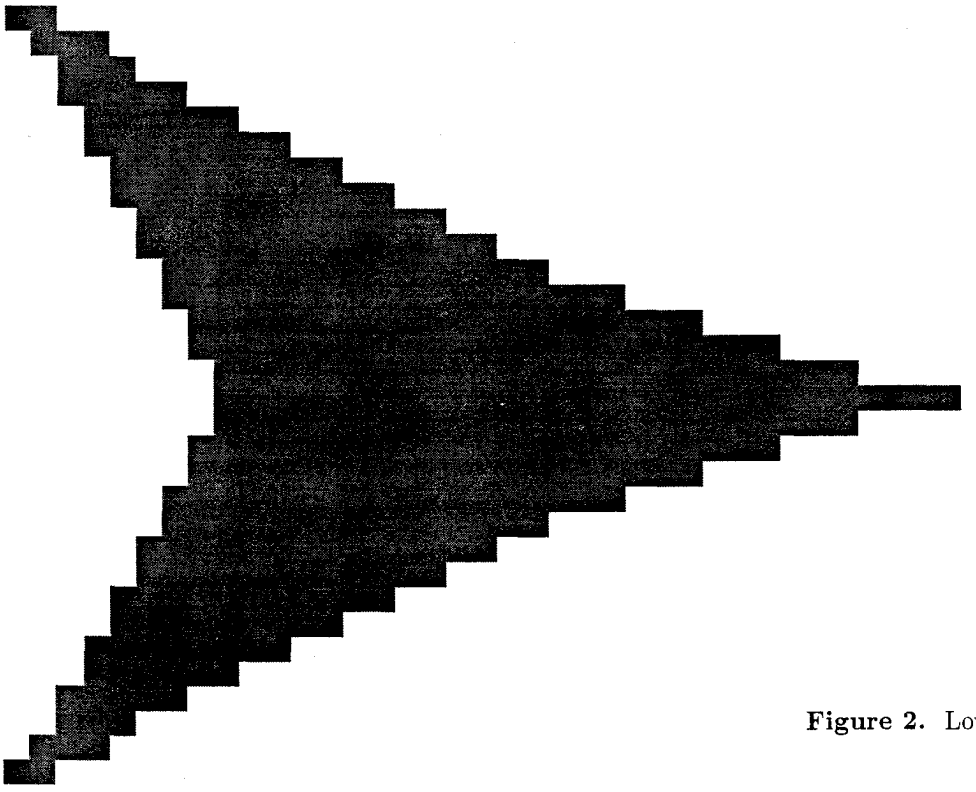


Figure 2. Lowres simulation