Control glyphs for arrows

Alan Jeffrey

1 Introduction

This document looks at suitable control glyphs for implementing math arrows. In particular, we look at ways that extensible and negated arrows can be produced.

2 Vertical arrows

Vertical arrows can be accessed using the growing delimiter mechanism. The non-negated arrows can be built with a 'top', 'bot' and 'rep' section, and the negated arrows can be built with a 'top', 'bot', 'med' and 'rep' section, where the 'med' section includes the negation.

3 Horizontal arrows

Horizontal arrows can be produced using the advanced ligature mechanism introduced in TEX 3. An *arrow* is defined by:

$$\mathit{arrow} ::= \mathit{leftarrow}\ \mathit{rightarrow}\ \mathit{extarrow}^*$$

A *leftarrow* is the leftmost part of an arrow, that is the tail of a rightwards-pointing arrow or the head of a leftwards-pointing arrow. For example, the *leftarrow* of '$\mapsto$' is ⟨*leftarrowmapsto*⟩ and the *leftarrow* of '$\leftrightarrow$' is ⟨*leftarowhead*⟩. A *leftarrow* is one of:

- ⟨*leftarrownone*⟩ There is nothing on the left of the arrow, for example '$\rightarrow$' or '$\rightrightarrows$'.
- ⟨*leftarrowhead*⟩ There is a single arrowhead on the left of this arrow, for example '$\leftrightarrow$' or '$\leftarrow$'.
- ⟨*leftarrowdblhead*⟩ There is a double arrowhead on the left of this arrow, for example '$\Leftarrow$' or '$\Leftrightarrow$'.
- ⟨*leftarrowtrplhead*⟩ There is a triple arrowhead on the left of this arrow, for example '$\Lleftarrow$' or '$\Longleftrightarrow$'.
- ⟨*leftarrowmapsto*⟩ There is a flat bar on the left of this arrow, for example '$\mapsto$'.
- ⟨*leftarrowhook*⟩ There is a hook on the left of this arrow, for example '$\hookrightarrow$'.
- ⟨*leftarrowharpoonup*⟩ There is an upwards harpoon on the left of this arrow, for example '$\leftharpoonup$'.
- ⟨*leftarrowharpoondown*⟩ There is a downwards harpoon on the left of this arrow, for example '$\leftharpoondown$'.
- ⟨*leftarrowharpoondblup*⟩ There is an upwards harpoon on the left of this double arrow, for example '$\leftrightharpoons$'.
- ⟨*leftarrowharpoondbldown*⟩ There is a downwards harpoon on the left of this double arrow, for example '$\rightleftharpoons$'.
- ⟨*leftarrowheaddbl*⟩ There are two arrowheads on the left of this single arrow, for example '$\twoheadleftarrow$'.
- ⟨*leftarrowdblheaddbl*⟩ There are two arrowheads on the left of this double arrow, for example '$\Lleftarrow$'.
- ⟨*leftarrowdblheadup*⟩ There is an arrowhead on the top left of this double arrow, for example '$\leftrightarrows$'.
- ⟨*leftarrowdblheaddown*⟩ There is an arrowhead on the bottom left of this double arrow, for example '$\rightleftarrows$'.
- ⟨*leftarrowrighthead*⟩ There is a rightwards arrowhead on the left of this arrow, for example '$\rightarrowtail$'.
- ⟨*leftarrowcurlyhead*⟩ There is an arrowhead with a curly line leading out of it on the left of this arrow, for example '$\rightsquigarrow$'
- ⟨*leftarrowturn*⟩ There is a turn on the left of this arrow, for example '$\looparrowright$'.
- ⟨*leftarrowcirc*⟩ There is an open circle on the left of this arrow, for example '$\circ\!\!-$'.
- ⟨*leftarrowbullet*⟩ There is a closed circle on the left of this arrow, for example '$\bullet\!\!-$'.
- ⟨*leftarrowtriangle*⟩ There is an open triangle on the left of this arrow, for example '$\blacktriangleleft\!\!-$'.

A *rightarrow* has the same possibilities, visually mirrored around the vertical axis, and with '*left*' swapped with '*right*'. For example:

$\rightarrow$ is ⟨*leftarrownone*⟩⟨*rightarrowhead*⟩.

$\leftrightarrow$ is ⟨*leftarrowhead*⟩⟨*rightarrowhead*⟩.

$\bullet\!\!\rightharpoonup$ is ⟨*leftarrowbullet*⟩⟨*rightarrowturn*⟩.

These arrows can be extended using *extarrow* glyphs, which say how many extension pieces to put into the arrow. An *extarrow* is one of:

- ⟨*extarrowone*⟩ which extends the arrow.
- ⟨*extarrowneg*⟩ which negates the arrow.
- ⟨*extarrowoneneg*⟩ which extends the arrow and negates it.

If there is a negation, it should come in the middle of the list of extensions, for example:

$\not\Leftarrow$ is ⟨*leftarrowdblhead*⟩⟨*rightarrowdblhead*⟩⟨*extarrowneg*⟩.

$\Longleftarrow\!\!\!\!\!/$ is ⟨*leftarrowdblhead*⟩⟨*rightarrowdblhead*⟩⟨*extarrowoneneg*⟩.

$\not\Longleftrightarrow$ is ⟨*leftarrowdblhead*⟩⟨*rightarrowdblhead*⟩⟨*extarrowone*⟩⟨*extarrowneg*⟩⟨*extarrowone*⟩.

Note that not every font will be able to produce every combination of *leftarrow* and *rightarrow*, for example some pi fonts do not allow $\mapsto$ to be extended. In fact, some of the combinations make no sense, such as ⟨*leftarrowhook*⟩⟨*rightarrowdblhead*⟩.

Such nonexistent arrows should be set with an eye-catching error glyph such as '■' and if possible should put a 'Warning:' special into the `dvi` file.

We should not specify anything about the appearance of these control glyphs, which should give font implementors enough freedom to convert most pi fonts into this encoding. One possible implementation of these control glyphs is by ligaturing and kerning, for example by using the ligatures:

$$\langle leftarrowhead\rangle\langle rightarrowhead\rangle \rightarrow \langle leftrightarrow\rangle$$
$$\langle leftrightarrow\rangle\langle extarrowone\rangle \rightarrow \langle leftrightarrow\rangle\langle rightarrow\rangle$$
$$\langle leftrightarrow\rangle\langle rightarrow\rangle \rightarrow \langle leftarrow\rangle\langle rightarrow\rangle$$
$$\langle rightarrow\rangle\langle extarrowone\rangle \rightarrow \langle rightarrow\rangle\langle rightarrow\rangle$$
$$\langle rightarrow\rangle\langle rightarrow\rangle \rightarrow \langle arrowext\rangle\langle rightarrow\rangle$$

and appropriate kerns for:

$$\langle leftarrow\rangle\langle rightarrow\rangle$$
$$\langle arrowext\rangle\langle rightarrow\rangle$$

then the 'arrow-building kit' will build any length of '↔' that is required, for example:

```
\def\longrightarrow{\mathrel
   {\leftarrowheadchar
      \rightarrowheadchar
      \arrowextonechar}}
```

Note that as long as the user does not use the control glyphs of type mathord in their document, the ligtable will never be used (rule 14 of Appendix G) and so input such as:

```
{\rightarrow} \rightarrow
```

will not unexpectedly ligature to something else. There are a few features missing from these control glyphs:

- The simpler arrows should be in fixed slots, so that they can be accessed quickly through a `\mathchardef`. We should ensure that this `\mathchardef` is always of type mathrel, otherwise strange ligaturing might happen.
- There is currently no syntax for accessing the negation glyphs on their own, which are necessary for building negated arrow leaders similar to `\arrowfill`.

There are eight other 'arrowlike' glyphs that are defined as special glyphs, and do not fit into the 'arrow-building kit'. These are '↻', '↺', '↰', '↱', '→', '←', '$\vec{x}$' and '⇔'. These glyphs should be accessed directly.

All that these control glyphs are doing is simulating the extensible vertical glyphs such as the growing delimiters. If there were a horizontal equivalent of growing delimiters then no such trickery would be required!

4 Diagonal arrows

There is no easy way to build diagonal extended arrows. All the font can do is provide the building blocks, and leave TEX macros to build the arrows. The font standard should specify how each of the diagonal extended arrows is to be constructed, and any font dimensions that will be necessary for the task.

5 Conclusions

Setting extensible vertical arrows with TEX is simple, since the growing delimiter mechanism already supports it. Setting growing horizontal arrows is trickier, and requires clever ligtable programming, but it can still be done inside the font, and without use of TEX macros. Setting diagonal arrows can only be done by brute force and TEX programming.

◇ Alan Jeffrey
University of Sussex
alanje@cogs.susx.ac.uk