

My Way

July 20, 2006

Extensible Arrows in ConT_EXt
Aditya Mahajan

This document describes how to use *extensible arrows* in ConT_EXt. Extensible arrows adapt to the width of the content placed on their top or bottom. They are implemented by having fixed *head* and *tail* (left and right part) with an extensible *shaft* (middle part). Most of the commonly used arrows are defined in ConT_EXt (with definitions borrowed from L^AT_EX). ConT_EXt also provides a clean syntax to define new arrows.

1 Introduction

Sometimes one needs to typeset arrows like $\xrightarrow[\text{below}]{\text{above the arrow}}$. ConTeXt provides extensible arrows to get this result. This *My Way* describes how to use extensible arrows and how to define new ones on your own.

2 Usage Syntax

The following extensible arrows are defined in ConTeXt.

<code>\xrightarrow</code>	\rightarrow
<code>\xleftarrow</code>	\leftarrow
<code>\xequal</code>	$=$
<code>\xrightarrow</code>	\Rightarrow
<code>\xleftarrow</code>	\Leftarrow
<code>\xleftrightarrow</code>	\Leftrightarrow
<code>\xleftrightarrow</code>	\leftrightarrow
<code>\xmapsto</code>	\mapsto
<code>\xtwoheadrightarrow</code>	\twoheadrightarrow
<code>\xtwoheadleftarrow</code>	\twoheadleftarrow
<code>\xrightharpoonup</code>	\rightharpoonup
<code>\xrightharpoonup</code>	\rightharpoonup
<code>\xleftharpoonup</code>	\leftharpoonup
<code>\xleftharpoonup</code>	\leftharpoonup
<code>\xhookrightarrow</code>	\hookrightarrow
<code>\xhookrightarrow</code>	\hookrightarrow
<code>\xleftrightarrow</code>	\Leftrightarrow
<code>\xleftrightarrow</code>	\Leftrightarrow

The syntax of using these commands is

```
\arrowname[options]{below}{above}
```

```
\xrightarrow [.1.] {..} {..}
                OPTIONAL  OPTIONAL  OPTIONAL
1  none  small  medium  big  normal  DIMENSION
2  TEXT
3  TEXT
```

where `<arrowname>` is one of the arrows shown above. The options effect the spacing around the above and below material. It can be one of the keywords or a number (which is converted to math units `mu`.) Compare:





Defining new arrows

<code>\xrightarrow</code>	
<code>\xrightarrow[none]</code>	
<code>\xrightarrow[small]</code>	
<code>\xrightarrow[medium]</code>	
<code>\xrightarrow[big]</code>	
<code>\xrightarrow[50]</code>	

All the arguments are optional, so you can also use

```
\arrowname[option]{above}
\arrowname{above}
\arrowname[option]
\arrowname
```

for example

<code>\xrightarrow{above}</code>	
<code>\xrightarrow[big]{above}</code>	
<code>\xrightarrow[big]</code>	
<code>\xrightarrow</code>	

3 Defining new arrows

There are two types of extensible arrows (i) Those with a single arrow (e.g. `\xrightarrow`, `\xleftarrow`, etc.) and, (ii) those with two arrows (e.g. `\xleftarrow-rightharpoons`, `\leftrightharpoons` etc.)

Here I will explain how to define both kinds of arrows. First you need to define an *extensible arrow fill*, which consists of three parts, which define the left, middle and right part of the arrow, where the middle part is stretched in a way that the arrow is at least as long as the material above and below it. This can be define using `\mtharrowfill`, e.g. for `\xrightarrow` one defines

```
\def\rightarrowfill{\mtharrowfill \relbar \relbar \rightarrow}
```

Defining new arrows

Here `\relbar` is `-` character, which is basically a horizontal bar at the middle of math line. Thus `\rightarrowfill` is an arrow fill in which the left part is `\relbar`, the middle stretchable part is `\relbar` and the right part is `\rightarrow`. Similarly, for `\xleftarrow` one defines

```
\def\leftarrowfill{\mtharrowfill \leftarrow \relbar \relbar}
```

and so on for different arrows.

An extensible arrow can be defined using the `\definearrow` command.

```
\definearrow [.1.] [...2,...] [...3,...]
                                OPTIONAL
1 IDENTIFIER
2 TEXT
3 TEXT
```

It can best be explained by means of an example. `\xrightarrow` is defined as

```
\definematharrow [xrightarrow] [0359] [\rightarrowfill]
```

The first argument is the name of the arrow (`\xrightarrow` in this case.) The second argument consists of a set of 4 numbers and specify the spacing correction in math units μ . These numbers define:

1st number: left arrow-tip correction

2nd number: right arrow-tip correction

3rd number: left space (multiplied by `\matharrfactor` and advanced by `\math-arrextra`)

4th number: right space (multiplied by `\matharrfactor` and advanced by `\math-arrextra`)

The third argument is the name of the extensible fill, that we defined earlier. This gives us an extensible arrow of the first kind. The third argument is optional when the arrow is redefined later (this is useful for font specific tweaking of the skips.) For example,

```
\math{\xrightarrow{\above}}
\definematharrow[xrightarrow][0000]
\math{\xrightarrow{\above}}
\definematharrow[xrightarrow][55{50}{50}]
\math{\xrightarrow{\above}}
```

Concluding Comments

gives $\overrightarrow{\text{above}}$ $\overrightarrow{\text{above}}$ $\overrightarrow{\text{above}}$

To define an extensible arrow the second kind, (e.g. `\xrightleftharpoons`) we need to define two extensible fills.

```
\def\rightharpoonupfill%
  {\mtharrowsfill \relbar \relbar \rightharpoonup}
\def\leftharpoondownfill%
  {\mtharrowsfill \leftharpoondown \relbar \relbar }
```

Next we need to tell `\definearrow` to stack them together.

```
\definematharrow [\xrightleftharpoons] [3095,0359]
  [\rightharpoonupfill,\leftharpoondownfill]
```

The second and the third set of arguments consist of comma separated values. The first element of the second argument (3095) corresponds to the spacing correction of top arrow fill (`\rightharpoonupfill`). Similarly, 0359 corresponds to bottom arrow fill `\leftharpoondownfill`). Stacking them on top of each other we get

$\overrightarrow{\overrightarrow{\text{rightharpoonupfill}}}$
 $\overleftarrow{\overleftarrow{\text{leftharpoondownfill}}}$

4 Concluding Comments

This is just meant as a usage guide for *extensible arrows* in ConTeXt, and is by no means exhaustive. If you really want to understand the nitty gritty of the implementation, you need to look at the code (in `math-ext.tex`).

