The *blockdraw_mp* package

MetaPost macros for the creation of block diagrams and bond graphs

Henrik Tidefelt
tidefelt@isy.liu.se

January 21, 2007

Abstract

This is not the right place to motivate why MetaPost should be used for the production of technical illustrations, in particular those to be used in material produced using LaTeX. Block diagrams are no exception since consistent layout of blocks, arrangement of connections, formatting of labels, placement of labels, et c are such demanding tasks. This package provides macros that facilitate the drawing of block diagrams using MetaPost. To extend the capabilities to bond graphs is a small step, and such macros are also included in this package.

This document describes the package, but is not the main manual. However, the 2-sided appendix starting on an odd page may be used on its own as a reference sheet.

Contents

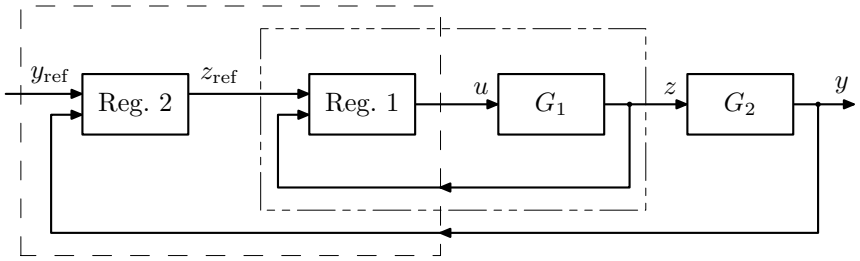

Figure 1. Example figure: block diagram.

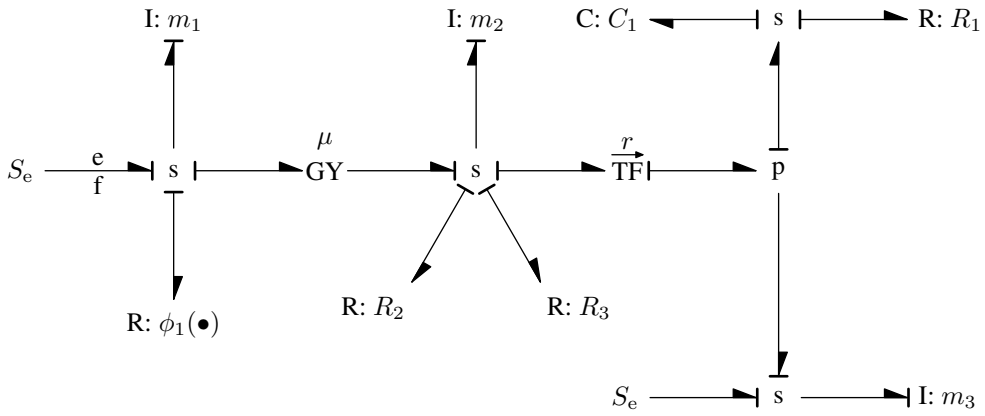

Figure 2. Example figure: bond graph.

1 Motivation

The pdf format has several advantages to PostScript, perhaps most importantly that previewers for pdf are more generally available and of better quality. This is obviously the reason why there is pdf(La)TEX. However, one problem for some potential pdf(La)TEX users is that the psfrag package is not compatible with pdf(La)TEX. On the other hand, since psfrag is not the most convenient way of putting good looking labels in a figure anyway, alternatives should be welcome. The drawing-language MetaPost is a powerful option for those who do not fear the expressiveness of text-based computer languages. This packages provide macros that facilitate the drawing of block diagrams and bond graphs in MetaPost. Two examples are given in figures 1 and 1.

Also the user who have no experience of MetaPost but needs a way to draw nice block diagrams without too much effort should consider using this package, because it might well be worthwhile to learn the few necessary bits of MetaPost. This typically amounts to looking at a few examples, learning how to invoke the MetaPost interpreter, and get acquainted with the *MetaPost output* format.

2 The Full Name

Although the main implementation file of this package is called *blockdraw.mp*, the package has not been given the name *blockdraw*. The full name of the package is precisely *blockdraw_mp*, and the reason for this is that the idea of drawing block diagrams with abstractions like those of this package can easily be, have been, and will be implemented in other languages than the presently targeted language MetaPost. For example, the implementation[1] I have been using mostly myself was for the language Drool (written by the author, not yet published). Nevertheless, the present implementation have been used successfully by several others, and support have been provided whenever asked for. Hence, although I am not a user of the package myself, I am convinced that it is truly useful.

3 This Document and the WWW-Based Documentation

The documentation provided in the appendix is meant to be only a brief summary of the variables and macros defined by the MetaPost-hosted package for drawing block diagrams and bond graphs. Details, illustrations, and examples can be found in the WWW-based documentation, currently available at http://www.control.isy.liu.se/~tidefelt/projects/blockdraw/. Still, the appendix might be useful as a reference sheet.

4 Files

The implementation consists of three files, which are located in the *implementation* directory:

- *shiftoff.mp* contains general-purpose macros for placing objects relative to a point. This file is included from *blockdraw.mp*.
- *blockdraw.mp* contains the things that are specific to drawing block diagrams.
- *bondgraph.mp* contains additional macros that are useful when drawing bond graphs.

The documentation and the sources needed to compile it are located in the *documentation* directory:

- *blockdraw_mp.tex* is the source of this document.
- *blockdraw_mp.pdf*
- *tiddetext.sty* contains various macros used.

[1] Note that it is not a well defined language that is implemented, but merely an idea of how to organize the drawing of block diagrams.

- *tighttoc.sty* is my ugly solution to make the table of contents consume less vertical space. Perhaps someone would like to have a look at it and make it a proper package...
- *cascadedemo.mp* is the source of the first demo figure.
- *cascadedemo.pdf*
- *bonddemo.mp* is the source of the second demo figure.
- *bonddemo.pdf*
- *docblockprefs.mp* is a file with settings to be used in block diagrams produces for this document. (Currently, there is only one such figure. A similar file could be creaded for bond diagrams to further emphasize how settings should be factored out in separate files, but I didn't just for brevity.)

A Miscellaneous macros

Name	Argument list	Description
`mspoint`	(*con*, *mediation*, *slide*)	Alternative to `point...of....`
`to_dir`	(*pt*)	(Low-level) Converts a direction vector to a direction constant.
`dir_to`	(*dir*)	(Low-level) Converts a direction constant to a direction unit vector.
`shiftoff`	(*pic*, *corner*)	Places *pic* relative to the origin, according to *corner* which shall be a direction constant (see below).
`shiftoffwlm`	(*pic*, *corner*)	Like `shiftoff`, but with "label margin".

B Block macros

Name	Argument list	Description
`sizedblock`	(*pict*, *center*, *rx*, *ry*)	For subsystems of explicitly specified size.
`longblock`	(*pict*, *center*)	For subsystems.
`squareblock`	(*pict*, *center*)	For simple subsystems.
`roundblock`	(*pict*, *center*)	For, for example, summations.
`splitdot`	(*center*)	A split point, drawn as a dot.
`termcircle`	(*center*)	A connection/terminal point, drawn as a small circle.
`pointpicture`	(*center*)	A <typename>picture</typename> that only serves to define a coordinate pair.
`conlabel`	(*shiftdir*, *pict*, *z*)	Similar to `label`, but honors the *textscale* setting.

C Point macros

Name	Argument list	Description
`leftpoint`	(*pict*, *total*, *index*)	For connections on the left side.
`rightpoint`	(*pict*, *total*, *index*)	For connections on the right side.
`bottompoint`	(*pict*, *total*, *index*)	For connections on the bottom side.
`toppoint`	(*pict*, *total*, *index*)	For connections on the top side.

D Connection macros

Name	Argument list	Description
`hhconnect`	(*point1*, *point2*, *mediation*, *slide*)	For horizontal to horizontal connections.
`vvconnect`	(*point1*, *point2*, *mediation*, *slide*)	For vertical to vertical connections.
`hvconnect`	(*point1*, *point2*)	For horizontal to vertical connections.
`vhconnect`	(*point1*, *point2*)	For vertical to horizontal connections.
`connect`	(*pict1*, *pict2*)	Automatic side to automatic side.
`lrconnect`	(*pict1*, *pict2*)	Left side to right side.
`rlconnect`	(*pict1*, *pict2*)	Right side to left side.
`btconnect`	(*pict1*, *pict2*)	Bottom side to top side.
`tbconnect`	(*pict1*, *pict2*)	Top side to bottom side.
`ltconnect`	(*pict1*, *pict2*)	Left side to top side.
`lbconnect`	(*pict1*, *pict2*)	Left side to bottom side.
`rtconnect`	(*pict1*, *pict2*)	Right side to top side.
`rbconnect`	(*pict1*, *pict2*)	Right side to bottom side.
`tlconnect`	(*pict1*, *pict2*)	Top side to left side.
`trconnect`	(*pict1*, *pict2*)	Top side to right side.
`blconnect`	(*pict1*, *pict2*)	Bottom side to left side.
`brconnect`	(*pict1*, *pict2*)	Bottom side to right side.
`llconnect`	(*pict1*, *pict2*, *slide*)	Left side to left side.
`rrconnect`	(*pict1*, *pict2*, *slide*)	Right side to right side.
`bbconnect`	(*pict1*, *pict2*, *slide*)	Bottom side to bottom side.
`ttconnect`	(*pict1*, *pict2*, *slide*)	Top side to top side.

E Bond graph macros

The macros `junction` and `junctionlbl` should only be used in the definitions of specialized macros such as `sjunction` and `tfjunction`.

Name	Argument list	Description
`junction`	(*symbol*, *center*)	Generic junction without parameter.
`junctionlbl`	(*symbol*, *arrowdir*, *lbl*, *center*, *drawarrow*)	Generic junction with parameter.
`terminal`	(*shiftdir*, *pict*, *point*)	Terminal elements.
`bgconnect`	(*pica*, *picb*)	Returns straight path from *pica* to *picb*, assuming that both pictures are circular with radii *smallblockr*. This is similar to the macro `connect` (documented here), which return paths with carpented segments.
`bond`	(*pth*)	Draws the half-arrow.
`causalmark`	(*pth*, *where*)	Draws the causality mark. *where* is a path time, and shall be 0 for the tail and infinity for the head.
`tbond`	(*pth*)	Draws the half-arrow and a causality mark at the tail.
`hbond`	(*pth*)	Draws the half-arrow and a causality mark at the head.
`terminalto`	(*jct*, *pict*, *point*)	Connection points to the junction.
`terminalfr`	(*jct*, *pict*, *point*)	Connection points to the junction.
`effortlabel`	(*pth*, *pict*)	Puts the label (*pict*) on the effort side of the path *pth*.
`flowlabel`	(*pth*, *pict*)	Puts the label (*pict*) on the flow side of the path *pth*.

F Standard user junction definitions

A bond graph application source file should define junction macros compatible with the following table.

Name	Argument list	Description
`sjunction`	(*center*)	For serial (type 1) junctions.
`pjunction`	(*center*)	For parallell (type 0) junctions.
`tfjunction`	(*center*, *arrowdir*, *lbl*)	For transformer junctions.
`gyjunction`	(*center*, *arrowdir*, *lbl*)	For gyrator junctions.

G Setting variables

Name	Type	Description
shiftofflabelmargin	Length	Similar to MetaPost's *labelmargin*.
longblockrx	Length	Horizontal size of standard rectangular blocks.
longblockry	Length	Vertical size of standard rectangular blocks.
smallblockr	Length	Size of square blocks.
connectionlw	Length	Width of connection lines.
blocklw	Length	Width of block frames.
textscale	Scalar	Scaling applied to all labels.
implicitdraw	Boolean	If true, generated objects are drawn before they are returned.
junctionimplicitdraw	Boolean	Like *implicitdraw*, but used in bond graphs.
useopenbonds	Boolean	If true, the asymmetric arrowheads used in bond graphs are not filled.

H Direction constants

Name	Description
to_center	Tells `shiftoff` and `shiftoffwlm` to center an object at the origin.
to_lft	Tells `shiftoff` and `shiftoffwlm` to place an object to the left of the origin, in a fashion analogue to the *lft* suffix on MetaPost's `label` macro.
to_ulft	Analogue to the corresponding `label` suffix.
to_top	Analogue to the corresponding `label` suffix.
to_urt	Analogue to the corresponding `label` suffix.
to_rt	Analogue to the corresponding `label` suffix.
to_lrt	Analogue to the corresponding `label` suffix.
to_bot	Analogue to the corresponding `label` suffix.
to_llft	Analogue to the corresponding `label` suffix.