DiaGraph for Drawing Block Diagrams with LaTeX

Tomomichi Hagiwara

1 What's DiaGraph?

This short writing introduces to you a small program called DiaGraph (DiaGraph), which might be useful to you if you are a LaTeX user and are feeling some difficulty or inconvenience in drawing a block diagram with it. The program is a kind of preprocessor, which generates the LaTeX code for the block diagram that you have in your mind, given its description in terms of its own language. I hope that the language is simple and friendly enough so that everyone can get accustomed to it within a few hours or less. As one of the authors of the program, I would be happy if you could enjoy it.

DiaGraph generates only a picture environment of LaTeX; you must either insert the resulting output file into your manuscript manually, or `\input` or `\include` it, usually inside a figure environment. To use the program, no knowledge about the picture environment is prerequisite, but having some knowledge about it would be helpful. The only new LaTeX command that you might have to learn is `\setlength`; see Section 4 for more details.

2 It's simple, isn't it?

Let me give you an idea about the language with the following simple example corresponding to Fig. 1. The basic idea is just to describe the way how pieces, such as lines, arrows and boxes, are connected to one another to form a block diagram.

```
R(1)
c("$ r $")
p
a(label-1)
R(1)
b(2,1,"$ G $")
c("stable")
r(1)
k(label-2)
R(1)
c("$ y $")
g(label-2)
d(1.5)
l(label-1)
U(label-1)
m
```

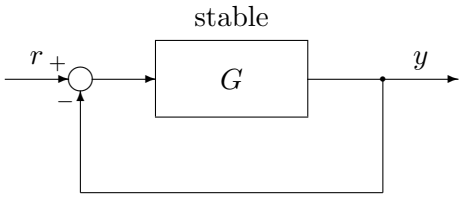

Figure 1: A simple block diagram

Each piece has a starting point and an ending point; for example, the starting point and ending point of an arrow are its tail and head, respectively. Unless otherwise specified, the starting point of a piece will automatically be placed at the ending point of the previous piece. In other words, when a piece is drawn, the *current position* will be moved to its ending point, and the next piece will be drawn from the current position. On the other hand, a few commands act on another position that also moves automatically with the placement of each piece, which we call the *subcurrent position*.

As you could guess from the example,

- Only one command should be described in a line. A command should be written in one line. A blank line is allowed. The last command in a source file should be followed by a new line.

- `R(1)` means a right arrow of length 1. `r(1)` means a line of length 1 "to the right" (by which its starting point and ending point would be clear). Similarly, `L` and `l` mean left, `U` and `u` up, `D` and `d` down. When one of these commands is issued, the *current direction* will be set to its direction. At the same time, the subcurrent position will be set at the middle of the arrow/line with the shift by the value of the parameter `\c` "to the left with respect to the current direction." (See Section 4 for the parameter.)

- `b(2,1,"*")` draws a box of width 2 and height 1 with the string `*` centered in it, toward the current direction. The current position will be moved to the middle of the "forward edge" of the box. Also, the subcurrent position will be set at the middle of the "left-side edge with respect to the current direction" plus the same shift as the above. (If the value of `\c` is negative, its "mirror image.")

- `a` means a summing junction, which must be always given a label. It has its starting and ending points at its center; the current position remains unchanged. The subcurrent position will be moved to the left of the center with respect to the current direction, with the distance of `\c`.

- `k` draws a forking junction (a disk with a tiny radius), which must be also labeled.

- `g` means jump to a label; it moves the current position to the point of the label. Also, the subcurrent position will be moved to the point with the same shift as described for `a`.

- `p` and `m` are respectively for plus and minus signs beside summing junctions. The current/subcurrent positions remain unchanged.

- `c("*")` draws the string `*` "at the subcurrent position"; the actual action depends on the current direction. If it is 'right(left),' the bottom(top)-center of the string will be placed at the subcurrent position. If it is 'up(down),' the right(left)-center will be placed at the subcurrent position. (If the value of `\c` is negative, exchange top and bottom, and up and down in the above.) These rules might look very complicated at a glance, but you will find them quite reasonable and simple actually.

Line/arrow drawing commands such as `l` and `U` can take a label (rather than a number indicating the length) as their argument, in which case the line/arrow will be drawn up to the nearest point to the label (i.e., the point with the same x or y coordinate as the point of the label) along the direction.

Other features are listed below.

- `R!` and `r!` are the same as `R` and `r`, respectively, except that a dashed arrow/line is drawn. Similarly for other directions.
- `B` is similar to `b`, but draws a dashed box.
- `s(1)` draws a sampler, which looks like a switch, of "length" 1 towards the current direction. `s()` draws a sampler of a default length.
- `n` is similar to `a` and `k`, but draws nothing; it just gives the current position a label for later use.
- `t(r)` changes the current direction into 'right.' `l`, `u` and `d` are the other possible arguments.
- `x(1)` moves the current position to the right by 1. `x(`*label*`)` moves it horizontally so that it has the same x coordinate as the point of the label. `y(1)` moves the current position upwards by 1. `y(`*label*`)` moves it vertically so that it has the same y coordinate as the point of the label. `X` and `Y` are for the subcurrent position. These commands draw nothing.
- A command line that begins with `%%` is a comment, which will be copied to the output LaTeX file after deleting the first `%`. (Using this feature, you can embed a LaTeX command to the output file, by placing only a single `%` before it in the source file. This enables you to draw a block diagram that cannot be drawn only with the built-in functions provided by DiaGraph, e.g. that with an oblique line.)

  On the other hand, a line that begins with `\%` will be regarded as a comment that should not be copied to the output file.

  `%%` after a command means an in-line comment. The text following it will be regarded as a comment, which will *not* be copied to the output LaTeX file.
- After having declared `$`*lengthlabel*`=2`, you can write for example `R($`*lengthlabel*`)` instead of `R(2)`. You can also write something like `x(-$`*lengthlabel*`)`; it is valid even if the value of `$`*lengthlabel* is negative. This might be useful when you are to adjust the sizes of similar pieces at one time. This kind of labels are called *length labels* or *numerical labels*, while the labels indicating positions are called *coordinate labels*.

3 How to use it?

The usage is simple. If `example.grp` is the source file of DiaGraph, just type in

```
diagraph example
```

Then, you will get the LaTeX code in the file `example.tex` on the working directory which you are on. If the file name of a source file does not end with `.grp`, you must type in the full name. (If the file name is specified together with the path, the delimiters for directories should be slashes, rather than backslashes, even on MS-DOS.)

You can specify some options before the file name:

- `-b`, `-B`

  boxes are drawn with thick lines; `-B` is for both solid line boxes and dashed line boxes, while `-b` is for only solid line boxes

- `-h`

  try specifying this option and previewing the result, when you are stuck; you would get some helpful information (I guess that you can understand the meaning of the information provided, so please allow me to omit the explanation about it)

- `-n`

  somewhat suppresses the output to the standard output

- `-s`

  each sampler will be drawn with dots, rather than a line (you might need this option when you draw a small sampler)

Multiple options should be specified like `-hn` rather than `-h -n`.

4 Parameters

DiaGraph has parameters on which its operation depends slightly. All the parameters have their default values, which can be changed by the user by providing the configuration file named `.diagraph` (or `diagraph.cnf` on MS-DOS). Local changes can be also made within a source file by writing a line like `\a=0.3`, which will be in effect only in the following lines.

- `\a`, `\k`

  the *radius* of a summing junction and the *diameter* of a forking junction, respectively

- `\b`, `\l`, `\r`, `\t`

  extra bottom, left, right and top margins, respectively, added to the smallest fictitious rectangle that covers all the pieces of the block diagram (that are drawn with the built-in functions other than `c` command); LaTeX will regard the block diagram as if it were a huge character with the size of the rectangle with the margins added

- `\c`

  determines the subcurrent position at which the string specified by the `c` command is put; note that the subcurrent position is determined as soon as those commands referring to the value of `\c` are issued, and thus the value of `\c` should be changed *before such commands are issued* (it is too late to change it just before the `c` command is issued, even though changing the sign of `\c` even after the subcurrent position has been determined will lead to the corresponding effect on the action of the `c` command)

- `\d`

  length of the solid line segment for a dashed box

- `\e`

  extra length added to the solid line segment just at the head of a dashed arrow

- \i, \o

  lengths of the solid line part and blank part for a dashed arrow/line, respectively

- \p, \v

  position of plus/minus signs beside a summing junction; \p to the back and \v to the left, with respect to the current direction, from its center

- \s

  default length for a sampler

The built-in default values of these parameters are as described in the sample configuration file `.diagraph` (the numbers in the brackets). If you like to change the default values, put a modified configuration file with, e.g., \a=0.3, on the directory where you are. Or, put it on the directory where your environment variable `DIAGRAPH` points. (Or, on the same directory as `diagraph.exe`, if MS-DOS.) The configuration file will be searched for in the above-mentioned order. If not found, then the built-in default values are assumed.

When changing the value of a parameter within a source file, you may write -\c, which means to change the sign of the value of \c. The default value (built-in or that given by the configuration file) can be recovered by simply writing \c. Similarly for other parameters, but some are not allowed to have negative values.

The built-in default values are designed so that they are proper when the unit length is 1cm. So, you must always specify \setlength{\unitlength}{1cm} (or another appropriate specification of the unit length that you like) in the preamble of the LaTeX file that includes the resulting output file. (Note that the default unit length of LaTeX is 1pt.)

5 More on DiaGraph

For your reference, some items in more details are listed below.

- R (and all its relatives), b, B and s may be preceded by -, in which case, not the starting point but the ending point of the piece will be placed at the current position. Except for that, the result will be the same as what you would get without -. This - is called a *minus command-prefix*. You may write like -R(*label*) (imagine what would happen by this).

  p and m can also be preceded by -, which might be necessary if you are to place a plus/minus sign beside an arrow that was drawn by a command with a minus command-prefix.

- When you embed a LaTeX command (see the item for %% in Section 2), you can use the internal variables of DiaGraph. "x and "y respectively denote the x and y coordinates of the current position. "& is the shorthand notation for "x,"y. When you use a length label, enclose it with double quotations as "$*lengthlabel*". Similarly, "**coordinatelabel*" and "/*coordinatelabel*" respectively denote the x and y coordinates of the point of the label. "@*coordinatelabel*" is the shorthand notation for "**coordinatelabel*","/*coordinatelabel*". DiaGraph always tries to interpret a double quotation if it appears in a command line that begins with a single %. So, it is better not to use a "real" double quotation within a LaTeX command to be embedded; if necessary at all, define e.g. \dq to denote a double quotation and use it instead.

- There is a rule for the names of labels, but the details are omitted. Roughly speaking, "ordinary" strings that do not begin with a numeral are OK. In particular, alpha-numeric characters always form the "ordinary" string. A name that violates the entire rule would be known to you, since it will cause an error message (syntax error).
- There is an upper limit for the number of labels, but in a normal situation, it won't be exceeded. Similarly for the length of a label. Should they be exceeded, an error message would let you know that.
- There is an upper limit for the length of a command line, but it would be long enough. An error message will be issued if it is exceeded.
- As in the preceding three items, the ability for detecting errors in the source file is high. Rule of thumb: you may do whatever does not cause an error message, probably.
- I admit that there are many things yet to be described, but please allow me to ask you to regard the way how it works as its function. Some of its operations might look somewhat strange to some of the users, but those are the specifications. (I never mean that there is no bug, but I believe that the probability is very small.)

6 Caution

DiaGraph should be used at your own risk; no one is accountable for any disadvantage caused by using it. However, I swear that I cannot imagine such possibility at all, except that *its output file could overwrite an existing* LaTeX *file*. Namely, when DiaGraph acts on the source file `example.grp`, it always overwrites the file `example.tex`, if it exists, assuming that it is just the output file generated by the previous run of DiaGraph on (the previous version of) the same source file, which you won't have to keep. Just in case where it is actually an important file, however, its back-up file will be created as `example.org`; the latter file will always be overwritten without making a further back-up file or issuing an error message—you may fail only once.

Simple solution: Make it a rule not to use the same name for a LaTeX document file and a DiaGraph source file.

7 Conclusion

Why don't you join us and enjoy the world!

A Sample LaTeX file

```
\documentstyle[11pt]{article}
\setlength{\unitlength}{1cm}
\begin{document}
\begin{figure}
    \centering
    \input{example}
    \caption{A simple block diagram\label{diagram}}
\end{figure}
\end{document}
```

B List of Commands

Lines and Arrows

`r l u d R L U D`
`r! l! u! d! R! L! U! D!`

Rectangles

`b B`

Summing and Forking Junctions, + and − Symbols, and Samplers

`a k p m s`

Writing Strings

`c`

Moving Current and Subcurrent Coordinates

`g x y X Y`

Saving Current Coordinates and Changing Current Directions

`n t`

DiaGraph Parameters

`\a \b \c \d \e \i \k \l \o \p \r \s \t \v`

Embedding LaTeX Commands and Comments

`% %% \%`

Other Features

Numerical Labels, Minus Command-Prefix