Lot-time diagram source generator

A.T. Hofkamp

April 17, 2007, 517/1440

Abstract

The LaTeX lottimediagram package by Van Eekelen ([1]) is very useful to make beautiful lot-time diagrams in publications. However, writing LaTeX source code for such a diagram from simulation output can be a time-consuming task, especially when the diagram is large or a lot of parallellism is present in the model.

The `lotdiagram.py` program described here aims at easing the latter. It takes simulation output in a pre-defined format, and automatically generates LaTeX code suitable for use by the lottimediagram LaTeX package.

1 The `lotdiagram.py` program

A single entity in the LaTeX lottimediagram package [1] (version `2004/05/28` was used here) is specified by the start time of processing, the end time of processing, the name of the machine, and the lot number (or numbers in case of a batch) being processed. In a simulation, this information is not readily available. However, the moment when processing starts, and the moment when processing ends at a machine is quite easy to obtain. From these events, the entities needed for the lottimediagram package can be constructed.

Also, normally, several machines operate in parallel in a simulation, which means that the events from different machines are merged together into one output stream. The `lotdiagram.py` program reads the output stream from the simulation, combines the events, and outputs LaTeX code suitable for processing by the lottimediagram package.

1.1 Input format

The `lotdiagram.py` program needs the events to be in a specific format. Each event must be written on a seperate line, and each line has the following form:

```
time machine lots
```

where `time` is a real number indicating the time of the event, `machine` is an identifier (presumably indicating the machine where the event occurred), and `lots` are the lot numbers affected by the event. If a single lot is affected by the event, `lots` should be a single unsigned integer number (for example '38'). If multiple lots are affected (for example, a batch machine starts processing 3 lots), a literal list should be used (for example '[2 3 4]', which means that lots 2, 3, and 4 are affected).

Note that start events and end events use the same format. The `lotdiagram.py` program understands that the first occurrence of a unique combination of machine and lot numbers means that a start event happened, and the second occurrence means that an end event has happened.

The program allows for different seperators in a line. Between `time` and `machine`, and between `machine` and `lots` you can use a space, a tab, or a comma character. In the case of multiple lots being affected, you can use either a space or a comma character between the lot numbers.

The order of the lot numbers is not relevant, '[2 3 4]' and '[4 2 3]' is equivalent (the program orders the lot numbers internally). Due to limiations in the LaTeX package, you are required to give a complete list when multiple lots are affected. In other words, there may be no gaps in the lot numbers. For example, the list [2 1 4] is incorrect, because lot number 3 is missing.

1.2 Output

To make the program output flexible in use, the `lotdiagram.py` program outputs as little as possible. By default, it only outputs a sequence of `\lot{}` and `\batch{}` lines. By using the `-d` (`--diagram`) option, additional `\begin{lottimediagram}` and `\end{lottimediagram}` lines are generated.

2 Example

Suppose we have the following χ 0.8 program:

```
// GMBMbE.chi

// generator, generates 14 lots
proc G(c:!nat) = |[ n:nat | n:=0 ; *[ n<14 -> c!n ; n:=n+1 ] ]|

// machine M processes single lots
proc M(a:?nat, b:!nat) =
|[ x:nat
 | *[ true
      -> a?x
       ; !time," M ",x,"\n"
       ; delta 2.2
       ; !time," M ",x,"\n"
       ; b!x
    ]
]|

// buffer, collects upto 2 lots in a batch before passing on
proc B(a:?nat, b:!nat*) =
|[ x,y:nat
 | *[ true
      -> a?x
       ; [ true; a?y   -> b![x,y]
         | true; b![x] -> skip
         ]
    ]
]|

// machine MB processes batches
proc MB(a:?nat*, b:!nat*) =
```

```
|[ x:nat*
 | *[ true -> a?x
       ; ! time, " MB ", x, "\n"
       ; delta 4
       ; ! time, " MB ", x, "\n"
       ; b!x
    ]
]|

// exit process, accepts batches
proc E(c:?nat*) = |[ x:nat* | *[ true -> c?x ] ]|

clus S() =
|[ gm,mb:-nat, bmb,mbe:-nat*
 | G(gm) || M(gm,mb) || B(mb,bmb) || MB(bmb,mbe) || E(mbe)
]|

xper = |[ S() ]|
```

In this example, a lot is represented here by a single natural number, which makes it easy to output batches (lists of lots). In many cases however, lots contain more information than just its identification number, for example its type. Batches of such lots can easily be printed by using a select expression like $[y.0|y : \text{lot} \leftarrow xs]$, where the first field of the lot-tuple contains the identification number of the lot, and xs is a list of lots.

Executing this program (`./GMBMbE > simoutput`) generates

```
0 M 0
2.2 M 0
2.2 M 1
2.2 MB [ 0 ]
4.4 M 1
4.4 M 2
6.2 MB [ 0 ]
6.2 MB [ 1 ]
6.6 M 2
6.6 M 3
8.8 M 3
8.8 M 4
10.2 MB [ 1 ]
10.2 MB [ 2 3 ]
11 M 4
11 M 5
13.2 M 5
13.2 M 6
14.2 MB [ 2 3 ]
14.2 MB [ 4 5 ]
15.4 M 6
15.4 M 7
17.6 M 7
17.6 M 8
```

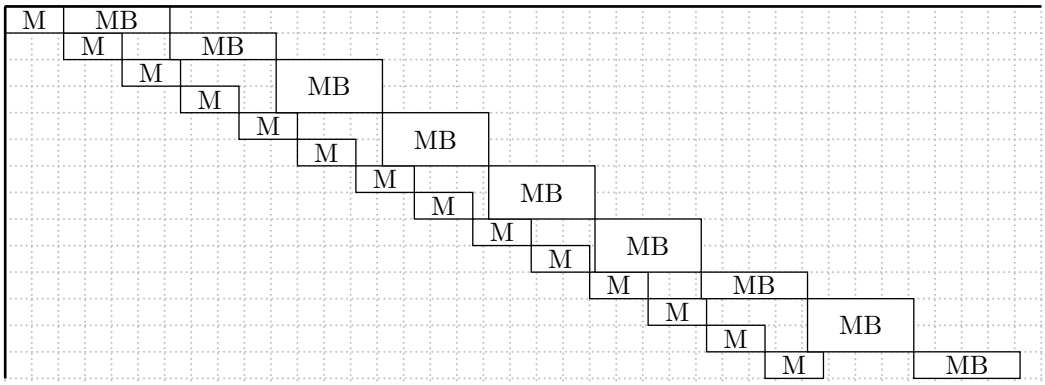
```
18.2 MB [ 4 5 ]
18.2 MB [ 6 7 ]
19.8 M 8
19.8 M 9
22 M 9
22 M 10
22.2 MB [ 6 7 ]
22.2 MB [ 8 9 ]
24.2 M 10
24.2 M 11
26.2 MB [ 8 9 ]
26.2 MB [ 10 ]
26.4 M 11
26.4 M 12
28.6 M 12
28.6 M 13
30.2 MB [ 10 ]
30.2 MB [ 11 12 ]
30.8 M 13
34.2 MB [ 11 12 ]
34.2 MB [ 13 ]
38.2 MB [ 13 ]
```

Applying the `lotdiagram.py` program (`python lotdiagram.py --diagram --input simoutput --output diagram.tex`) gives

```
\begin{lottimediagram}{14}{39}
  \lot(0)(0,2.2){M}
  \lot(1)(2.2,4.4){M}
  \lot(0)(2.2,6.2){MB}
  \lot(2)(4.4,6.6){M}
  \lot(1)(6.2,10.2){MB}
  \lot(3)(6.6,8.8){M}
  \lot(4)(8.8,11){M}
  \batch(2,3)(10.2,14.2){MB}
  \lot(5)(11,13.2){M}
  \lot(6)(13.2,15.4){M}
  \batch(4,5)(14.2,18.2){MB}
  \lot(7)(15.4,17.6){M}
  \lot(8)(17.6,19.8){M}
  \batch(6,7)(18.2,22.2){MB}
  \lot(9)(19.8,22){M}
  \lot(10)(22,24.2){M}
  \batch(8,9)(22.2,26.2){MB}
  \lot(11)(24.2,26.4){M}
  \lot(10)(26.2,30.2){MB}
  \lot(12)(26.4,28.6){M}
  \lot(13)(28.6,30.8){M}
  \batch(11,12)(30.2,34.2){MB}
  \lot(13)(34.2,38.2){MB}
```

```
\end{lottimediagram}
```

which (after processing with the lottimediagram package) delivers the following picture:



which is exactly what you would expect from the specification, in particular the gaps in the diagram in the processing of a lot are caused by the fact that the buffer process `B` did not output events.

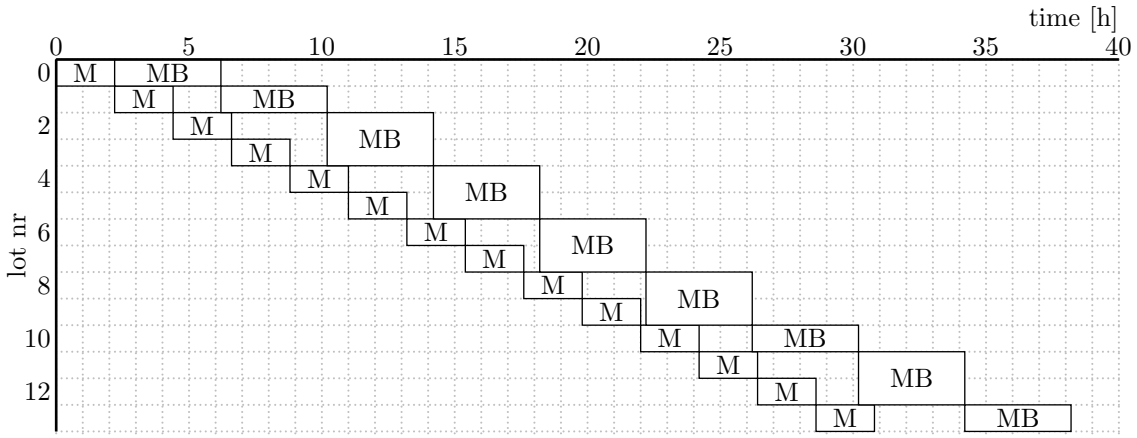
The missing legenda along the axis can easily be added manually once you are satisfied with the contents of the diagram. If you don't like to make manual changes to generated diagrams, you can drop the `--diagram` option (that is, run `python lotdiagram.py --input simoutput --output lotlines.tex`) and include the generated output in the following way

```
\begin{lottimediagram}{14}{40}
\timeaxislabel{time [h]}
\timetickinterval{5}
\lotaxislabel{lot nr}
\lottickinterval{2}

\input{lotlines}  % load the generated lotlines.tex file

\end{lottimediagram}
```

which reads the lot-time data from the `lotlines.tex` file, and generates the following (prettier) picture:



Acknowledgements

Thanks to Joost and Ad for writing the lottimediagram package, Erjen for creating an excuse to write this Python program, and both Joost en Erjen for reviewing an earlier version of this document.

References

[1] Joost van Eekelen. *Lot-time diagrams in LaTeX using package* `lottimediagram`, August 2004. Available at `http://se.wtb.tue.nl/documentation/`.