`Genet`
A tool for the synthesis and mining of Petri nets

Josep Carmona
`jcarmonalsi.upc.edu`
Software Department
Universitat Politcnica de Catalunya

Contents

1.1 Overview of the tool

`Genet` is a tool for the synthesis and mining of k-bounded Petri nets. The input of the tool is a *transition system* [2] specifying a behavior. The tool derives a Petri net (PN) [6] that has some relation with the input transition system. Depending on the usage of the tool, two possible outcomes are possible:

1. Synthesis [4]: the PN derived has a reachability graph bisimilar to the input transition system.

2. Mining [3]: the set of possible traces in the PN derived is a superset of the ones possible in the input transition system.

Let us use a simple example to illustrate the usage of the tool for these two different options. Let $a \dots e$ be the set of events of a concurrent system. Imagine that we have the following behavior on these events:

Initially a and b are enabled concurrently, and any of these two events can trigger c. Then, when a, b and c have been observed, d and e are observed in sequence.

The informal behavior described above can be formally specified as a transition system:

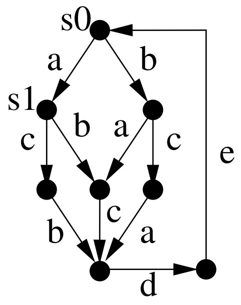


Figure 1.1: Transition system representing a given behavior.

Both a transition system and a Petri net can be easily specified in a text file. See Section 1.6 for a short description of each format.

Synthesis

The behavior of this example cannot be represented in a 1-bounded PN (unless the labels of transitions a and b are duplicated): the reason is due to the fact that there is an *or-causality* relation between the set of events a, b and the caused event c. However, a 2-bounded PN can represent the log, as shown in Figure 1.2.

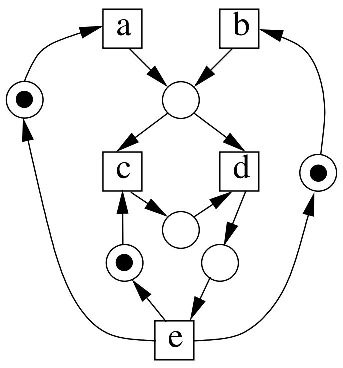


Figure 1.2: PN with behavior bisimilar to the TS of Figure 1.1.

The command line used for synthesizing the PN is:

```
# genet -k 2 or.sg
```

Mining

It might be the case that the user is interested in a 1-bounded PN that covers the behavior of the TS. In the mining option of `Genet`, the PN derived contains all the traces that are possible in the TS, and maybe more. Usually, the PN generated with the mining option is less complex (in terms of places, arcs and transitions) than the one derived for synthesis. The PN mined for the TS of Figure 1.1 is shown in Figure 1.3.

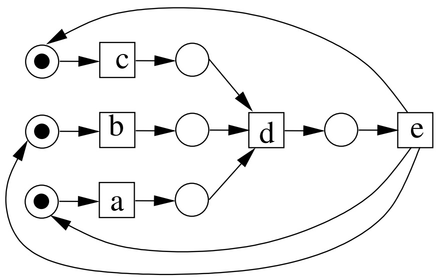


Figure 1.3: PN mined from the TS of Figure 1.1.

And the command line used for mining the PN is:

```
# genet -pm or.sg
```

1.2 User commands

USAGE

```
genet [options]  <name_of_transition_system_file> > <resulting_pn>
```

Tables 1.1 and 1.2 describe the options of the tool.

GENERAL OPTIONS (SYNTHESIS & MINING)

-*k n*	to look for an n-bounded Petri net
-*min s*	start the search for regions with bound $n \geq k \geq s$, where n has been assigned by the -k option . This may speed up the search.
-*enc* [0 | 1]	encoding of the states. 0: logarithmic encoding (default), 1:event-based encoding.
-*prj ev_list*	projects the initial transition system into the events in the list. ev_list is a list of events separated by commas.
-*h*	help

Table 1.1: General options (synthesis and mining).

OPTIONS FOR PETRI NET MINING

-*pm*	to apply Petri net mining (approximation of the excitation closure).
-*cov n*	n is the number of minimal regions to cover an event. This might be useful when the set of minimal regions is large and therefore one can weaken the condition to cover an event with few regions.
-*mg*	mining of a marked graph. This is experimental, and not very debugged option (self-loops still not treated properly).

Table 1.2: Options for mining.

1.3 External tool for visualization

For graphical visualization of a transition system or Petri net, the public domain tool `draw_astg` can be used. Given a Petri net described in the file *pn.g*, the command:

```
draw_astg pn.g -o pn.ps
```

generates in the new file *pn.ps* a graphical representation of the Petri net, in `Ghostscript` format. Then genet can be pipelined with `draw_astg`:

```
 # genet -k 2 or.sg | draw_astg -noinfo -nonames -o or.ps
```

where "-nonames" indicate to `draw_astg` to not provide names in the places, and "-noinfo" avoids reporting the role of the events.

It is also possible to visualize a transition system. The option "-sg" allows this:

```
 # draw_astg -sg or.sg -o or_ts.ps
```

`draw_astg` is a tool within the `petrify` tool suite [5], and can be downloaded from the following url:

```
http://www.lsi.upc.es/~jordicf/petrify/
```

1.4 Examples

1.4.1 Synthesis

Imagine five processes $P_1 \dots P_5$ that have access to three shared resources. Process P_1 access two resources, whereas every other process P_i only access one resource. The transition system expresssing this behavior is described in Figure 1.4.
where each process P_i has the events e_i for adquiring the resource and t_i for releasing it. Clearly, looking at the transition system it is difficult to realize what is going on in this system. If a 1 or 2-bounded Petri net is derived from this transition system, the visualization will also be very poor because several transitions have to be split in order to satisfy the synthesis conditions.

Now let us synthesize a 3-bounded Petri net from this transition system, and draw it with `draw_astg`:

```
 # genet -k 3 shared532.sg
```

The result, depicted in Figure 1.5, clearly summarizes the behavior of the system.

If, for instance, only part of the behavior must be observed, the projection option can be used. Supose we want to obtain a Petri net representing the behavior of the processes $P_2 \dots P_5$. The following command will obtain the Petri net of Figure 1.6.

```
 # genet -k 3 -prj e2,t2,e3,t3,e4,t4,e5,t5 shared532.sg
```

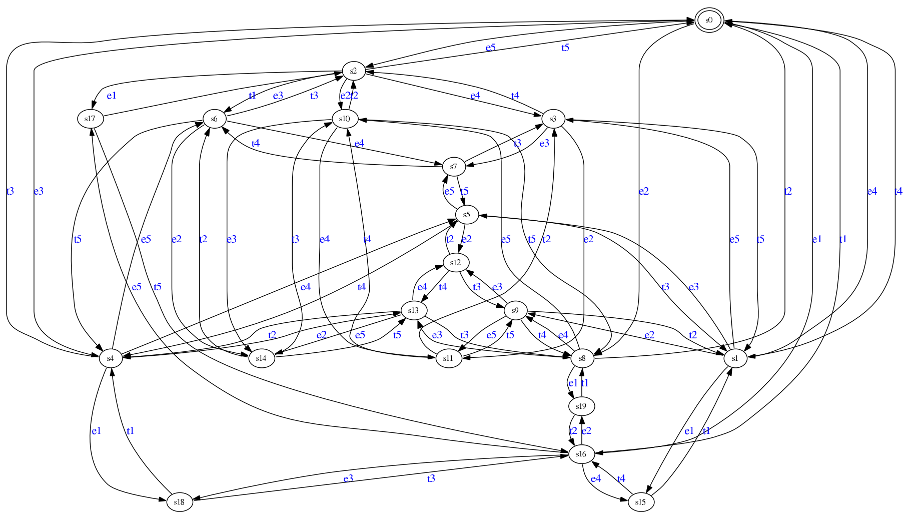


Figure 1.4: Shared resources system: transition system.

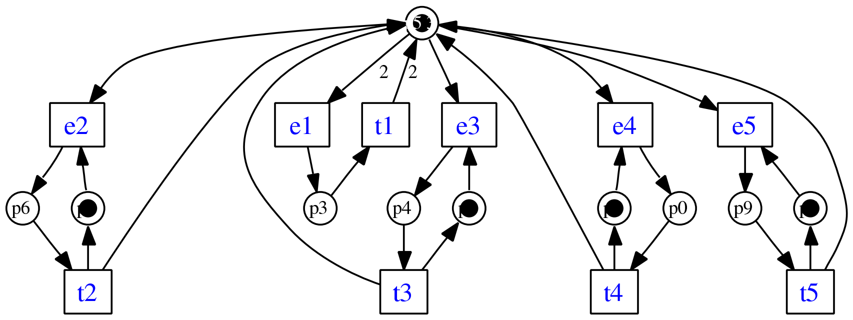


Figure 1.5: Shared resources system: Petri net.

1.4.2 Mining

In Process Mining, the transition system represents an event log of a system. Let us use the `hrbstFig6p21` example obtained from [1]. The initial transition system is depicted in Figure 1.7(a).

Now imagine that a 1-bounded Petri net should be obtained. If the net should reproduce exactly this transition system, the Petri net of Figure 1.7(b) will be derived. Clearly, the Petri net is complex and not easy to understand due to the neccessary splittings to exactly reproduce the log.

If the mining option of genet is used, the Petri net of Figure 1.7(c) results. It covers more traces than the ones in the log, but on the other side the visualization of the corresponding Petri net is much more understandable!

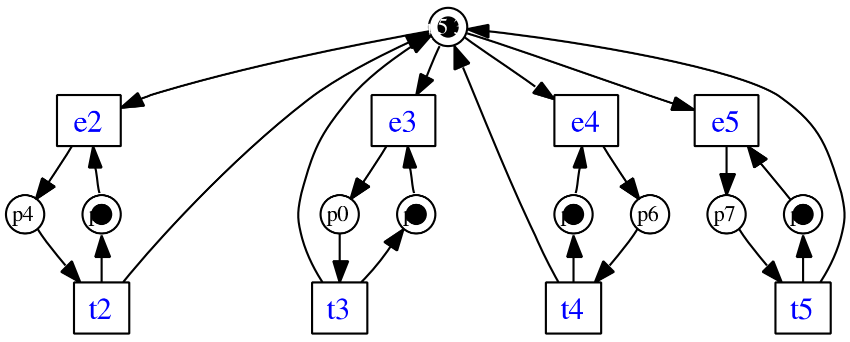


Figure 1.6: Synthesized Petri net on behavior regarding processes $P_2 \ldots P_5$.

1.5 Obtaining the tool

The tool can be obtained from the following url:

`http://www.lsi.upc.es/~jcarmona/genet.html`

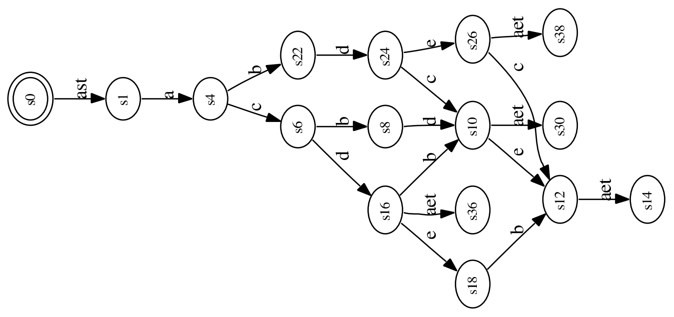
1.6 Transition system and Petri net file formats

The best is to use some examples of this tutorial. The transition system shown in Figure 1.1 is specified with the following file:

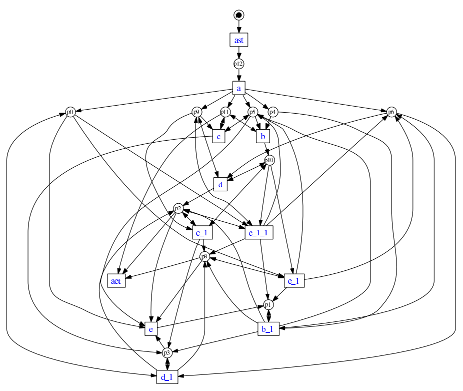
```
.outputs  a b c d e
.state graph
s0 a s1
s0 b s2
s1 b s5
s2 a s5
s1 c s3
s2 c s4
s5 c s6
s3 b s6
s4 a s6
s6 d s7
s7 e s0
.marking {s0}
.end
```

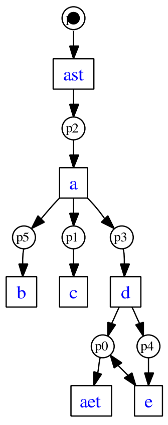
where:

- the first line enumerates the events of the system
- the second line specifies the type of the model (transition system is specified as *state graph*)



(a)





(b) (c)

Figure 1.7: (a) Transition system specifying the initial log, (b) Synthesized Petri net from the transition system of Figure 1.7(a), (c) Mined Petri net from the transition system of Figure 1.7(a).

- the line ".marking s0" determines the initial state of the transition system
- the last line should be ".end"
- each other line contains the definition of an arc (source_state, event, target_state)

The file format for a Petri net is somehow similar to the transition system file format. The file describing the Petri net of Figure 1.5 is the following:

```
.outputs e1 e2 e3 e4 e5 t1 t2 t3 t4 t5
.graph
p5 e1(2)
p5 e2
p8 e2
```

```
p1 e3
p5 e3
p2 e4
p5 e4
p5 e5
p7 e5
p3 t1
p6 t2
p4 t3
p0 t4
p9 t5
e1 p3
e2 p6
e3 p4
e4 p0
e5 p9
t1 p5(2)
t2 p5
t2 p8
t3 p1
t3 p5
t4 p2
t4 p5
t5 p5
t5 p7
.capacity  p0=3 p1=3 p2=3 p3=3 p4=3 p5=3 p6=3 p7=3 p8=3 p9=3
.marking { p1 p2 p5=3 p7 p8 }
.end
```

where:

- the first line enumerates the events of the system
- the second line specifies the type of the model (Petri net is specified as *graph*)
- the line ".marking " determines the initial token assignment of the places
- the line ".capacity " determines the maximal capacity for the places of the Petri net
- the last line should be ".end"
- each other line contains the definition of a (possibly weighted) arc. An arc can be from a transition to a place or viceversa. The arc can

be weighted, as it is shown in the line "p5 e1(2)", representing that transition $e1$ removes two tokens from place $p5$.

Bibliography

[1] Process mining. www.processmining.org.

[2] A. Arnold. *Finite Transition Systems.* Prentice Hall, 1994.

[3] Josep Carmona, Jordi Cortadella, and Michael Kishinevsky. A region-based algorithm for discovering Petri nets from event logs. In M. Dumas, M. Reichert, and M. C. Shan, editors, *BPM*, volume 5240 of *Lecture Notes in Computer Science*, pages 358–373. Springer, 2008.

[4] Josep Carmona, Jordi Cortadella, Michael Kishinevsky, Alex Kondratyev, Luciano Lavagno, and Alexandre Yakovlev. A symbolic algorithm for the synthesis of bounded Petri nets. In Kees M. van Hee and Rüdiger Valk, editors, *Petri Nets*, volume 5062 of *Lecture Notes in Computer Science*, pages 92–111. Springer, 2008.

[5] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev. Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers. *IEICE Transactions on Information and Systems*, E80-D(3):315–325, March 1997.

[6] T. Murata. Petri Nets: Properties, analysis and applications. *Proceedings of the IEEE*, pages 541–580, April 1989.