TP Cosmos

Benoît Barbot

ETR Aout 2017

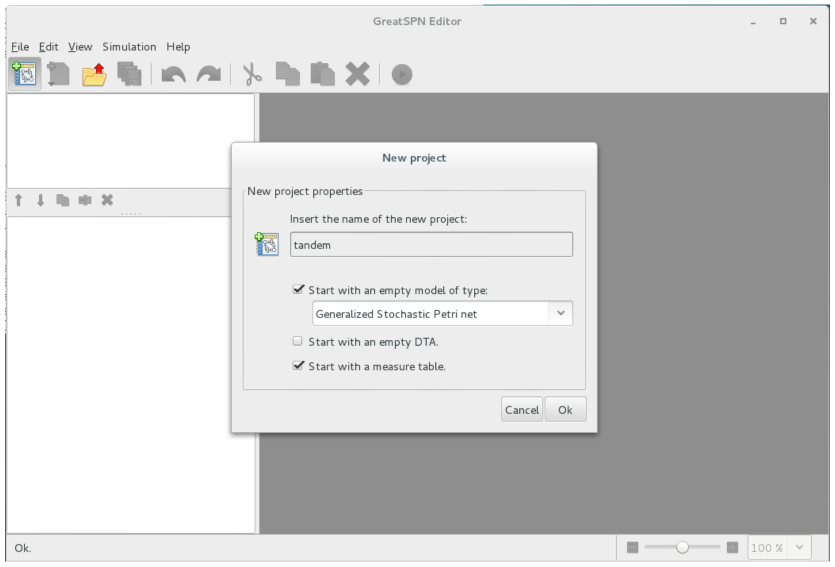
1 Building a model

We will use the GreatSPN Graphical Editor (`http://www.di.unito.it/~amparore/mc4cslta/editor.html`) which is graphical editor for stochastic Petri net.

▷ When using the ETR2017 virtual machine, start the editor by opening a terminal and writing:

```
> java -jar tools/cosmos-1.5/bin/Editor.java &
```

▷ Click on the upper left icon to create a new model. The following windows should open:



New project properties

Insert the name of the new project: tandem

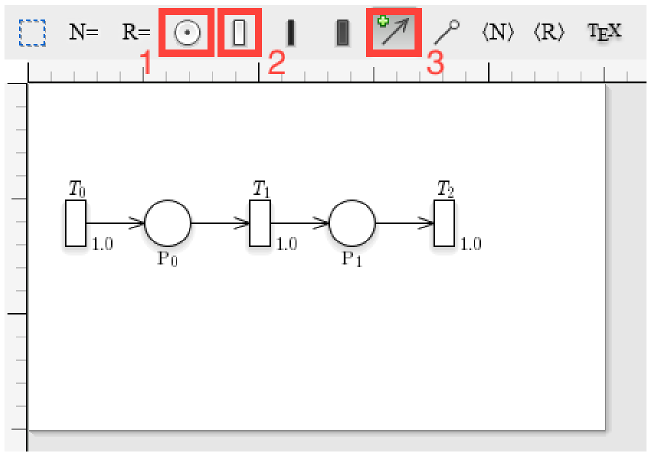
☑ Start with an empty model of type: Generalized Stochastic Petri net

☐ Start with an empty DTA.

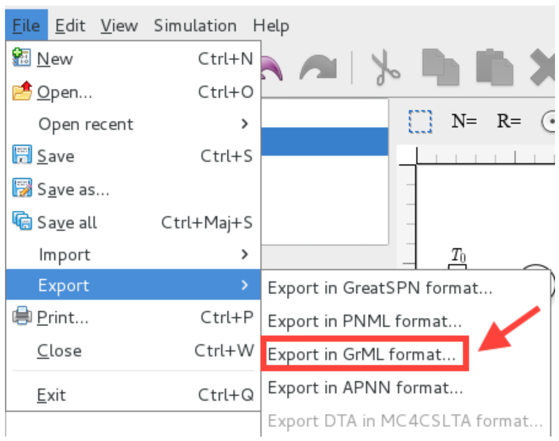
☑ Start with a measure table.

▷ Fill the windows as on the previous screen shoot.

▷ Draw a tandem queues system by adding two places (icon 1), three exponential distributions (icon 2) and adds arc between places and transitions (icon 3) as in the following screenshot:



▷ Export the model in the GrML file format using the file menu -> export -> export in GrML.



2 Simple use of the tool

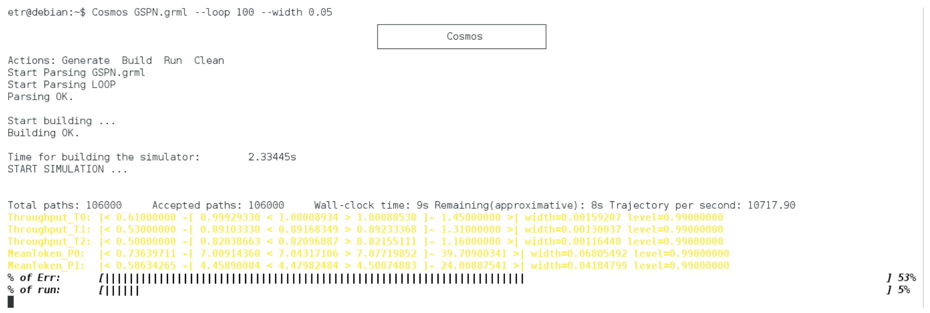
Cosmos can be used to compute simple performance evaluation indexes of a model by generating an HASL formula. This is done using the "–loop X" option which generates a simple LHA which loops for X time units and then accept the trajectory.

▷ Open a terminal and go to the directory where you have exported the model.

▷ write down

```
> Cosmos GSPN.grml --loop 100 --width 0.05
```

You should see something similar to the following screenshot:



```
etr@debian:~$ Cosmos GSPN.grml --loop 100 --width 0.05
                                                                  Cosmos

Actions: Generate  Build  Run  Clean
Start Parsing GSPN.grml
Start Parsing LOOP
Parsing OK.

Start building ...
Building OK.

Time for building the simulator:        2.33445s
START SIMULATION ...


Total paths: 106000     Accepted paths: 106000     Wall-clock time: 9s Remaining(approximative): 8s Trajectory per second: 10717.90
Throughput_T0: |< 0.61000000 -[ 0.99929330 < 1.00008934 > 1.00088538 ]- 1.45000000 >| width=0.00159207 level=0.99000000
Throughput_T1: |< 0.53000000 -[ 0.89103330 < 0.89168349 > 0.89233368 ]- 1.31000000 >| width=0.00130037 level=0.99000000
Throughput_T2: |< 0.50000000 -[ 0.82038663 < 0.82096887 > 0.82155111 ]- 1.16000000 >| width=0.00116448 level=0.99000000
MeanToken_P0:  |< 0.73639711 -[ 7.00914360 < 7.04317106 > 7.07719852 ]- 39.70900341 >| width=0.06805492 level=0.99000000
MeanToken_P1:  |< 0.58634265 -[ 4.45890084 < 4.47982484 > 4.50074883 ]- 24.00887541 >| width=0.04184799 level=0.99000000
% of Err:      [||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||                                                  ] 53%
% of run:      [||||||                                                                                                                  ] 5%
█
```

You can observe that Cosmos start by parsing the model and the LHA then the simulator is built and simulations start. Cosmos dynamically updates the state of simulation. The first line in black shows some data on the number of trajectories and simulation time. Each of the following lines (in yellow) is the estimation of an HASL expression. These lines read as follows:

```
Expr name: |< min value -[ confint low < mean value > confint up -] max value >|
```

Where `confint` stand for "confidence interval" and are followed by the width of the confidence interval and the confidence level used for the confidence interval computation.

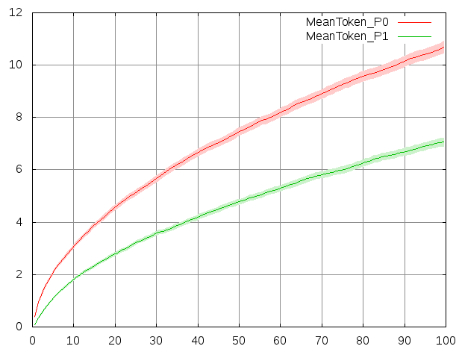
The last two lines indicate the progress of the simulation. The first one `% of Err:` represents the width of the largest confidence interval with respect to the objective (given on the command line with "–width 0.05"). The second line `% of run:` represents the number of simulated trajectories with respect to the maximum allowed number (by default 2000000). The simulation stop when one of the goals is reached. You can stop the simulation by hitting `Ctr-C`.

You can visualise the behaviours of the model by plotting the mean trajectory.

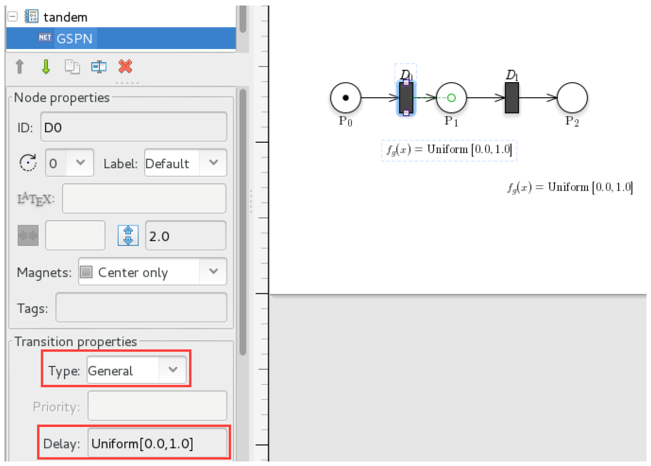
▷ Write down the following command line:

```
> Cosmos GSPN.grml --sampling 100 1 --max-run 100000
     --output-graph trace --gnuplot-driver png
```

The "–sampling X Y" option generate an LHA which samples the value each places each Y time unit and stop the simulation after X time unit. The "–max-run Z" option specify the maximal number of runs. The "–output-graph file" option output the result of the simulation in the file "file". The "–gnuplot-driver output" option start the gnuplot software and specify the type of output. At the end of the simulation you can open the file "trace.png" which should be similar to this one:



You can observe a well-known fact of queuing theory which is that if the rate of incoming client is not strictly smaller than the serving rates, then the system is unstable. The type and parameters of transition distribution can be modified in the left panel when the transition is selected. See below screenshot. In the interface, the parameter of exponential distribution is called "Rate", and it is called "Delay" for general distribution.



ID: D0
0 Label: Default
LaTeX:
2.0
Magnets: Center only
Type: General
Delay: Uniform[0.0,1.0]

▷ Change the rate of transition T_0 to 0.6 in GreatSPN, export the model and run Cosmos, observe the difference of behaviours.

▷ Change the type of transition T_1 to "General" and set the delay to "I[1.0]", which stand for Dirac distribution in 1. Change the sampling parameters to "–sampling 10 0.1" to plot a more precise graph.

If a model does not behave as it is intended to, it is useful to perform step by step simulation.

▷ write down the following command line

```
> Cosmos GSPN.grml --loop 100 --output-dot draw.pdf -i
```

Cosmos will stop before the first step of simulation and show you the available transitions. You can write down "s" to perform one step of simulation, or "fire

Ti" to fire transition Ti. If you write "d" and open the file "draw.pdf", it will contain a drawing of the Petri net updated with the current marking.

3 HASL formula

In this part we will design an LHA to evaluate a more complex property.

You can look at the LHA produced by Cosmos while using options "–loop" or "–sampling". By default these files are removed with all the temporary files when Cosmos exit. The option "–tmp-status keep" prevent Cosmos from cleaning up, all the temporary files are stored in a directory called "tmp" in the working directory.

▷ Look at the file "tmp/looplha.lha" and "tmp/samplinglha.lha".

▷ Build a new GSPN in GreatSPN with three places, and two transitions, set the initial marking of the first place to 1 and 0 for the others. Add arcs such that the only possible execution is for the token to go from the first place to the second by firing the first transition and then go to the third place by firing the second transition. Then set the distribution of both transitions to be a "General" distribution with delay "Uniform[0.0,1.0]".

▷ Open a new text file with an ".lha" extension (for example with gedit) and write down the following:

```
VariablesList = {x, y, r} ;
LocationsList = {lx, ly, lf, lp,lm};

%HASH Expression
p=4*PROB;

InitialLocations = {lx};
FinalLocations = {lp};

Locations={
%( location name, guard, flow )
(lx, TRUE , (x:1 , y:0) );
(ly, TRUE , (x:0 , y:1 ));
(lf, TRUE );
(lp, TRUE );
(lm, TRUE );
};

Edges={
%( (source, dest), label, guard, update )
((lx,ly),ALL, # , #);
((lx,lf),ALL, # , {r=x*x + y*y});
((lf,lp), #, r<=1 , #);
((lf,lm), #, r>=1 , #);
};
```

▷ Can you guess the value of p ?

▷ Run Cosmos by passing the model name and the lha name.

▷ Add an HASL expression to the LHA to compute the mean value of r.

▷ For the tandem queue model, write down an LHA computing the following LTL formula: “X ((Q1+Q2>0) U (Q1+Q2=10))”.