MC4CSL^{TA}: an efficient model checking tool for CSL^{TA}

Elvio Amparore Gilberto and Susanna Donatelli
Dipartimento di Informatica, Università di Torino, Italy
{amparore.elvio,susi}@di.unito.it

Abstract

This demo presents MC4CSL^{TA}, an efficient Markov chain Model Checker for (four) the stochastic logic CSL^{TA}. In CSL^{TA} formulas are either steady state queries or path formulas, where accepting paths are specified through a Deterministic Timed Automata (DTA). MC4CSL^{TA} takes as input a labeled CTMC, a query, one or more DTA involved in the query (possibly expressed in parametric form), and computes the needed probability to assess the truth value of the formula. MC4CSL^{TA} is written in C++, and is publicly available for the research community.

1. Major functionalities

CSL^{TA} is a stochastic logic for continuous Markov chains (CTMCs) that has been defined in [5]. CSL^{TA} is characterized by the possibility of specifying path formulas through a single-clock Deterministic Timed Automata (DTA). If $\lambda \in [0,1]$ is a probability, $p \in AP$ an atomic proposition and $\bowtie \in \{\leq, <, >, \geq\}$ a comparison operator, a CSL^{TA} state formula Φ is defined by:

$$\Phi ::= p \mid \neg\Phi \mid \Phi \wedge \Phi \mid \mathcal{S}_{\bowtie\lambda}(\Phi) \mid \mathcal{P}_{\bowtie\lambda}(\mathcal{A})$$

where $\mathcal{A}$ is a single-clock Deterministic Timed Automata.

Model checking of CSL^{TA} is the same as for CSL [4], but for the case of $\mathcal{P}_{\bowtie\lambda}(\mathcal{A})$. The model checking procedure for $\mathcal{P}_{\bowtie\lambda}(\mathcal{A})$, defined in [5], requires the construction of a non-ergodic *Markov Regenerative Process*[6] called the "synchronized product" $\mathcal{M}\times\mathcal{A}$ of the CTMC $\mathcal{M}$ and of the DTA $\mathcal{A}$, and its solution in steady state. In a recent paper [3] we have shown how the CSL^{TA} model checking problem for $\mathcal{P}_{\bowtie\lambda}(\mathcal{A})$ can be reduced to the steady state solution of a Deterministic Stochastic Petri Nets (DSPN). The work in [3] describes how the DSPN can be built starting from a CTMC $\mathcal{M}$ and a DTA $\mathcal{A}$. This demo tool presentation describes MC4CSL^{TA}, the tool that we have built upon the ideas described in [3]. The tool implements the translation into DSPN and its solution, as well as the chain of translation required for nested formulas.

The tool is a *single-line command* tool invoked by:

```
CslTA-Solver.exe  model.cslta
```

where `model.cslta` is a file that contains the CTMC $\mathcal{M}$, the DTA $\mathcal{A}$, and one or more evaluation commands. A user guide is included in the distribution, and therefore we give here only the most salient features of the tool.

The CTMC $\mathcal{M}$ is specified in a simple textual form, which includes the possibility of defining rate parameters and parametric action names for transitions, to allow a simple way to verify formulas for different set of CTMC parameters. Transition rates can be both rate parameters or simple arithmetic expressions involving rate parameters. The CTMC can be also produced automatically (again with a simple line command) from higher level models defined using either the tool PRISM or GreatSPN to produce a text file to be included in our `model.cslta` input file.

The DTA $\mathcal{A}$ is specified in textual form, with the peculiarity that DTA can be parametric in the set of time constants, in the set of propositions to be associated with locations, and in the set of action names to be associated with edges. At evaluation time a parametric DTA needs to be instantiated by providing an adequate set of parameters. This feature greatly enhances re-use (for example a simple DTA of a classical CSL Until formula can be re-used for any Until formula that we may want to check). It is assumed that time constants are given in a list that will be instantiated on an increasing set of values.

Evaluation commands take the form given below:

```
RESULT r = EVAL(Model, s0 |=
 PROB_TA < 0.1 (Until[ 10, 50 | | NOT waiting, waiting])
);
```

which instruct the tool to check whether the CTMC Model, for the initial state s0, satisfies the path formula $\mathcal{P}_{<0.1}(\mathcal{A})$, where $\mathcal{A}$ is the DTA named `Until`, that verifies $\Phi_1\ \mathcal{U}^{[\alpha,\beta]}\Phi_2$ instantiated with $\alpha = 10$, $\beta = 50$, $\Phi_1 = \neg waiting$ and $\Phi_2 = waiting$.

The execution of the CSL^{TA} solver command produces the truth value of the formula as well as the actual probability of the CTMC paths that satisfy the formula.

The tool allows for the specification of nested formulas, which in CSL^{TA} are realized by allowing a DTA location to be labeled with a boolean expression that includes CSL^{TA} formulas as well. In the tool this is realized by allowing a DTA to be instantiated with an actual parameter for location

propositions which is actually a CSL^{TA} formula which may include another DTA instantiation.

Another interesting feature of the tool is that it checks determinism of the DTA *before* building the MRP, looking only at the DTA. Since there are cases in which a DTA may lead to a deterministic behavior, depending on the structure of the CTMC, whenever a DTA is not surely deterministic the translation into DSPN includes a set of transitions that check the deterministic property of the DTA at "run-time", during the MRP generation.

2. Results and tool availability

The example we use is a CTMC derived from a stochastic Petri net model of a polling system with failure. The system has three service centers (queues), with a closed load of K clients each, organized in a cyclic order. A single server goes around the centers, stopping to provide service if a client is waiting in the queue. The server has three different states: working, degraded and failed (*work*, *deg*, *fail* for short), that determine the speed of the service provided. When the server fails, it is not repaired, and the system goes into a final absorbing state, with all clients waiting in the queues. The DTA that defines the path formula to be checked is depicted in Figure 1, and it has been instantiated with Φ_1 = *work*, Φ_2 = *deg*, and Φ_3 = *fail*, and various sets of α and β. The probability of the paths accepted by the DTA is plotted in Figure 2 for a varying set of parameters, where K is the load in each queue and λ is the arrival rate at queues, while the same value for α and β have been used, plotted on the x-axis.

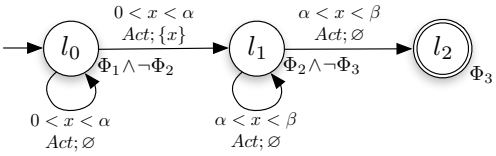

Figure 1. DTA for the example.

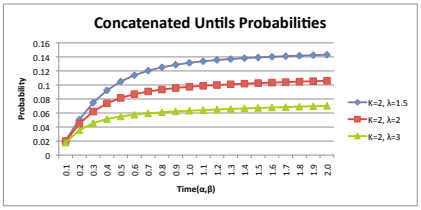

Figure 2. Probability of accepted paths.

MC4CSL^{TA} produces a DSPN and then uses our new DSPN tool DSPN-Tool [2], that we have recently implemented on top of GreatSPN. The DSPN tool implements a classical MRP solution, based on the Embedded DTMC (EMC) construction, as well as a recently proposed algorithm [1] that exploits the sequential structure of the transient behavior of the MRP, if any. Table 2 reports the size of the CTMC (states and transitions), the *Tangible Reachability Graph*(TRG) size of the DSPN produced by the translation, the number of entries of the EMC and the time to execute the classical solution or the method proposed in [1] for the query of figure 2. The table clearly shows the cost of building and storing the EMC. The last column builds the EMC only for a subset of the TRG, exploiting the sequentiality in the transient behavior of the MRP. An alternative solution algorithm could be the iterative technique proposed by German in [7], that avoids the EMC construction, which at the moment has not been defined for non-ergodic MRPs.

K	CTMC states	CTMC transits.	|TRG| size	EMC P entries	EMC (sec)	Sequent. (sec)
1	80	372	218	2953	0.32	0.23
2	297	1521	812	37801	1.88	0.89
3	736	3936	2018	229153	9.28	2.25
4	1475	8085	4052	918001	26.79	4.81
5	2592	14436	7130	2834569	84.51	8.83
6	4165	23457	11468	-	-	15.23
8	8991	51381	24788	-	-	40.15
10	16577	95601	45740	-	-	98.98
12	27547	159861	76052	-	-	228.26

Table 1. MC algorithms benchmark.

The tool can be downloaded from: http://www.di.unito.it/∼susi/PDS10/CslTA-Solver.zip. The available version implements the translation into DSPN for the generation of the MRP, which is then solved through our new DSPN-Tool tool [2]. We are currently finishing the complete implementation of the new MRP solver [1] that exploits the specific structure of MRPs generated by our DSPNs. A preliminary implementation has been used to produce the results in Table 2.

References

[1] E. Amparore and S. Donatelli. A Component-based Solution Method for Non-Ergodic Markov Regenerative Processes. Paper submitted for publication.

[2] E. Amparore and S. Donatelli. DSPN-Tool: a new DSPN and GSPN solver for GreatSPN. Tool demo presentation accepted at QEST 2010, September 15-18, Williamsburg, USA, IEEE-CS Press.

[3] E. G. Amparore and S. Donatelli. Model Checking CSL^{TA} with Deterministic and Stochastic Petri Nets. IEEE Computer Society Press, 2010. Accepted for pubblication at DSN-PDS 2010.

[4] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton. Model-checking continuous-time Markov chains. *ACM Trans. Comput. Logic*, 1(1):162–170, 2000.

[5] S. Donatelli, S. Haddad, and J. Sproston. Model checking timed and stochastic properties with CSL^{TA}. *IEEE Trans. Softw. Eng.*, 35(2):224–240, 2009.

[6] R. German. *Performance Analysis of Communication Systems with Non-Markovian Stochastic Petri Nets*. John Wiley & Sons, Inc., New York, NY, USA, 2000.

[7] R. German. Iterative analysis of Markov regenerative models. *Perform. Eval.*, 44:51–72, April 2001.