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 $\alpha, \beta \in \{\leq, <, >, \geq\}$ a comparison operator, a CSL$^{TA}$ state formula $\Phi$ is defined by:

$$\Phi := p \lor \Phi \land \Phi \land S_{\geq \lambda}(\Phi) \land P_{\leq \lambda}(A)$$

where $A$ is a single-clock Deterministic Timed Automata.

Model checking of CSL$^{TA}$ is the same as for CSL [4], but for the case of $P_{\leq \lambda}(A)$. The model checking procedure for $P_{\leq \lambda}(A)$, defined in [5], requires the construction of a non-ergodic Markov Regenerative Process [6] called the “synchronized product” $M \times A$ of the CTMC $M$ and of the DTA $A$, and its solution in steady state. In a recent paper [3] we have shown how the CSL$^{TA}$ model checking problem for $P_{\leq \lambda}(A)$ 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 $M$ and a DTA $A$. 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 $M$, the DTA $A$, 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 $M$ 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 $A$ 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 $P_{<0.1}(A)$, where $A$ is the DTA named Until, that verifies $\Phi_1 \land U[10,50](\neg waiting \land waiting)$ 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\textsuperscript{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 $\Phi_1 = \text{work}$, $\Phi_2 = \text{deg}$, and $\Phi_3 = \text{fail}$, and various sets of $\alpha$ and $\beta$. 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 $\lambda$ is the arrival rate at queues, while the same value for $\alpha$ and $\beta$ have been used, plotted on the $x$-axis.

The tool can be downloaded from: http://www.di.unito.it/~susi/PDS10/CsITASolver.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.

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

Table 1. MC algorithms benchmark.

References