A CSL Model Checker for Continuous-Time Markov Chains

Yang Gao1, Ernst Moritz Hahn1,2*, Naijun Zhan1, and Lijun Zhang3

1 State Key Lab. of Comp. Sci., Institute of Software, Chinese Academy of Sciences, China
2 University of Oxford, United Kingdom
3 Technical University of Denmark, DTU Informatics, Denmark

Abstract. We present CCMC (Conditional CSL Model Checker), a model checker for continuous-time Markov chains (CTMCs) with respect to formulas specified in continuous-time stochastic logic (CSL). Existing CTMC model checkers such as PRISM or MRMC handle only binary CSL until path formulas. CCMC is the first tool that supports algorithms for analyzing multiple until path formulas. Moreover, CCMC supports a recent extension of CSL – conditional CSL formulas – which makes it possible to verify a larger class of properties on CTMC models. Our tool is based on our recent algorithmic advances for CSL, that construct a stratified CTMC before performing transient probability analyses. The stratified CTMC is a product obtained from original CTMC and an automaton extracted from a given formula, aiming to filter out the irrelevant paths and make the computation more efficient.

1 Introduction

Continuous-time Markov chains (CTMCs) play an important role in performance evaluation of networked, distributed and biological systems. The concept of formal verification for CTMCs was introduced by Aziz et al. [1]. In their seminal work continuous-time stochastic logic (CSL) was defined to specify properties of CTMCs. They showed that the model checking problem of CSL over CTMCs, i.e., whether a CTMC satisfies a given CSL property, is decidable. The approach has not yet been implemented, due to the high theoretical complexity. Later, efficient approximation algorithms have been studied by Baier et al. [2]. Based on this, several tools have been developed to support CSL model checking, such as PRISM [3] and MRMC [4]. Both of them can only deal with CSL properties with binary until path formulas.

Recently, we have extended the approximation algorithm in [2] to deal with multiple until path formulas [5]. The main idea is to exploit the notion of stratified CTMCs, which is a subclass of CTMCs that has the nice feature of allowing one to obtain the desired probability using a sequence of transient probability analyses. First, a deterministic finite automaton (DFA) is constructed for the formula being considered. Then, the product of the CTMC and the DFA is constructed, which is stratified by construction. This product CTMC can then be analyzed efficiently, using standard numerical methods for CTMCs.

Moreover, we have proposed an extension of CSL with probabilistic conditional operators [6], and in addition, we allow disjunction and conjunction of path formulas. With conditional CSL, one can for instance formulate the following conditional property:

The probability is at least 0.1 that in the interval [10, 20) the number of proteins becomes more than 5 and the gene becomes inactive, under the condition that the proteins increasingly accumulated from 0 to k within the same time interval,

\[ P_{\geq 0.1}(true_{[10,20)} \land g \mid f_1 U_{[10,20)} f_2 U_{[10,20)} \cdots f_k), \]

* Ernst Moritz Hahn is supported by ERC Advanced Grant VERIWARE.
where $f, g, f_1, \ldots, f_k$ are appropriate atomic propositions.

In this paper, we present the probabilistic model checker CCMC, which is based on the recent work in [5][6][7], and supports the multiple until and probabilistic conditional formulas. These formulas allow one to express a richer class of properties for CTMCs, thus we consider our tool an important complementation of PRISM [3] and MRMC [4].

2 Logic and Tool Architecture

The syntax of Conditional CSL (CCSL) is given by the following grammar:

$$\Phi : = f | \neg \Phi | \Phi \land \Phi | P_{\text{op}}(\phi) | P_{\text{op}}(\phi | \phi), \quad \varphi : = \varphi \lor \varphi | \varphi | \Phi_1 U \Phi_2 U \cdots U \Phi_K$$

where $f$ is an atomic proposition, $I_i$ are non-empty left-closed and right-open intervals on $\mathbb{R}_{\geq 0}$, $\bowtie \in \{<, \leq, \geq, >\}$, $0 \leq p \leq 1$, $K > 1$. $\Phi$ is called a state formula, while $\varphi$ is called a path formula. In particular, CCMC supports the multiple until path formula, in contrast to existing model checkers (e.g. [3,4]) which are restricted to binary path until, i.e., $K = 2$.

For the sake of efficiency, CCMC was implemented in C/C++ and consists of approximately 5000 lines of code. It has been applied on a number of relevant case studies from diverse areas (performance evaluation, biological models, etc.). CCMC is available for Linux platform with libc6 and GNU Scientific Library 1.15, and distributed under the GNU General Public License (GPL) Version 3. The binary code, source code and case studies can be downloaded from: http://lcs.ios.ac.cn/~gaoy/CCMC/homepage.xhtml.

The architecture and components of CCMC are depicted in Fig. 1. The inputs of CCMC include the model description files and the property file. The model description files can be written manually or generated by PRISM, including a state file and a transition matrix file. They will be loaded, where we use explicit sparse matrix representations. The property file keeps the CSL properties of interest.

The preprocessing component constructs the stratified CTMC, which is a product obtained from the original CTMC and an automaton extracted from the given formula [5]. For CCSL conjunctive path formulas, we need an extended product construction which also takes sub path formulas into account [6].

By the preprocessing procedure, the paths which are irrelevant to the given properties are filtered out and the model analysis component will carry out a forward transient probability computation. The verification results and other information can be visualized or exported into a file.

3 Experiments

In this section, we conduct experiments on some CTMC benchmarks from the PRISM webpage and other publications [8]. All experiments were performed on a Linux (Ubuntu 12.10) machine with an Intel(R) Core(TM) i7-2600 processor at 3.40GHz equipped with 3 GB of RAM. Due to space constraints, detailed information about the cases is provided on the CCMC webpage.

**PRISM Benchmark Suite.** Firstly, we compare our model checker CCMC with PRISM (sparse matrix engine) on verifying benchmark CTMC models. Here we use three models which
can be found on PRISM homepage\(^1\). The first one is a cyclic server polling system [9], the second one is a workstation cluster [10], and the third one is an embedded control system [11].

We consider binary CSL until formulas, which can also be handled by PRISM. Results and executing time (in seconds) are listed in Table 1. The meaning of parameter \( N \) is as on the PRISM homepage. Execution times of CCMC and PRISM for the analyses considered are almost equal.

<table>
<thead>
<tr>
<th>Polling Cluster</th>
<th>Embedded</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRISM</td>
<td>CCMC</td>
</tr>
<tr>
<td>8</td>
<td>3.077</td>
</tr>
<tr>
<td>9</td>
<td>6.915</td>
</tr>
<tr>
<td>10</td>
<td>15.361</td>
</tr>
<tr>
<td>11</td>
<td>33.793</td>
</tr>
<tr>
<td>12</td>
<td>73.729</td>
</tr>
<tr>
<td>13</td>
<td>159.745</td>
</tr>
<tr>
<td>14</td>
<td>344.065</td>
</tr>
</tbody>
</table>

**Table 1.** Comparison with PRISM. \( N \) and MAX are model parameters influencing the number of states.

**Random Robot.** We use this case (revised from [8]) which describes a robot on a grid with \( N \times N \) cells of different land types to show the ability of CCMC to verify conditional formula\(^2\). Now we focus on computing the probability that the robot goes across the flatlands, cementlands and grasslands under the condition that it will get stuck within time \( t \). This property can easily be expressed by a CCSL formula, that is, \( P=\gamma(\varphi | \psi) \) where \( \varphi = \text{flat } U_{(a_1, b_1)} \text{ cement } U_{(a_2, b_2)} \text{ grass } \) and \( \psi = \text{true } U_{(0, 1)} \) trap. We generate the grid randomly and the experiments result are listed in the left part of Table 2 where \( a_1 = 1, a_2 = 1.5, b_1 = 2, b_2 = 3 \) and \( t = 10 \).

**Cyclic Server Polling.** We reconsider the cycling server polling [9] and the CSL property: what is the probability of finding all the queues full in the whole round when the server serves them within \( T \) seconds, which can be expressed using a multiple until formula:

\[
P=\gamma(s = 1 \land s_1 U_{(0, T)} s = 2 \land s_2 U_{(0, T)} \cdots U_{(0, T)} s = k \land s_k), \text{ where } s = i \text{ (} i = 1, \cdots, k \text{) means that the server is at the } i\text{-th station, and } s_k = 1 \text{ shows that the } k\text{-th station is full. The right part of Table 2 shows the results by fixing } T = 10.
\]

**Table 2.** Experimental results.

2. Detailed information can be found on: [http://lcs.ios.ac.cn/~gaoy/CCMC/casestudies/casestudies.htm](http://lcs.ios.ac.cn/~gaoy/CCMC/casestudies/casestudies.htm).
CCSL formula, however, the product construction makes the original CTMC stratified and we need just perform the transient probability analyses at each time endpoint of the intervals which occur in the CCSL formula. Thus, the execution time only depends on the size of product CTMC and the number of time endpoints.

Remark 1. More recently, Donatelli et al. [12] have extended CSL such that path properties can be expressed via a deterministic timed automaton (DTA) with a single clock. Chen et al. [8,13] take this approach further and consider DTA specifications with multiple clocks as well. In the CPS case study, we compare our approach with the DTA based approach. In [8], a DTA is used to specify the property: What is the probability that after consulting all queues for one round, the server serves each queue one after the other within \( T \) time units? This property can be separated into two phases and formulated by multiple until formulas. At each phase, we construct the corresponding product CTMC which reduces the computation work a lot. As a result, we can handle larger models, and the running time is considerably improved. See the CCMC homepage for details.

4 Concluding Remarks

In this work, we have introduced CCMC, a probabilistic model checker for CTMC models. Its effectiveness and efficiency have been demonstrated through the successful analysis of several case studies. As future work, we will extend this work to Continuous-Time Markov Decision Procedure (CTMDP) models which can model and analyze the systems with both probabilistic and nondeterministic behaviors. We also want to explore the possibility to use symbolic data structures, such as MTBDDs or the one of the PRISM hybrid engine.

References