A CSL Model Checker for Continuous-Time Markov Chains

Yang Gao[1], Ernst Moritz Hahn[1,2⋆], Naijun Zhan[1], and Lijun Zhang[3]

[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*,
>
> $$\text{as } \mathcal{P}_{\geq 0.1}(true\ U_{[10,20)}\ f \wedge g \mid f_1\ U_{[10,20)}\ f_2\ U_{[10,20)} \cdots f_k),$$

⋆ 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:

$$\Phi := f \mid \neg\Phi \mid \Phi\wedge\Phi \mid \mathcal{P}_{\bowtie p}(\varphi) \mid \mathcal{P}_{\bowtie p}(\varphi \mid \varphi), \quad \varphi := \varphi\wedge\varphi \mid \varphi\vee\varphi \mid \Phi_1 U_{I_1} \Phi_2 U_{I_2} \cdots U_{I_{K-1}} \Phi_K$$

where f is an atomic proposition, I_i are non-empty left-closed and right-open intervals on $\mathbb{R}_{\geq 0}$, $\bowtie \in \{<, \leq, \geq, >\}$, $0 \leq p \leq 1$, $K > 1$. Φ is called a *state formula*, while φ 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.

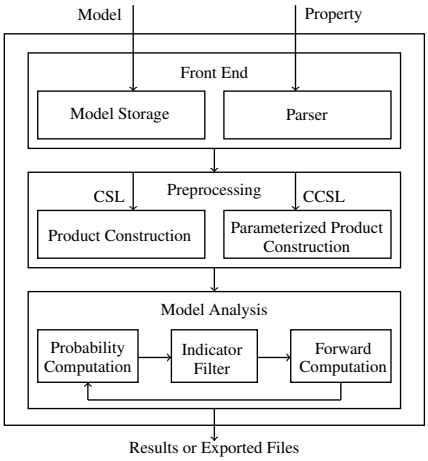

Fig. 1. CCMC Architecture.

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[4]. 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.

Polling				Cluster				Embedded			
N	states	PRISM	CCMC	N	states	PRISM	CCMC	MAX	states	PRISM	CCMC
8	3,073	0.016	0.01	2	276	0.006	<0.01	5	6,013	0.319	0.16
9	6,913	0.047	0.03	4	820	0.011	<0.01	8	8,548	0.437	0.22
10	15,361	0.077	0.06	8	2,772	0.014	0.01	10	10,238	0.53	0.29
11	33,793	0.161	0.14	16	10,132	0.051	0.03	20	18,688	0.925	0.50
12	73,729	0.341	0.41	32	38,676	0.166	0.11	50	44,038	2.715	1.14
13	159,745	0.804	1.04	64	151,060	0.639	0.45	100	86,288	4.285	2.73
14	344,065	1.853	2.43	128	597,012	2.871	3.09	200	170,788	8.535	6.00
15	737,281	4.069	5.73	256	2,373,652	12.345	19.53	500	424,288	21.649	20.46

Table 1. Comparism 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 [5]. 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, $\mathcal{P}_{=?}(\varphi \mid \psi)$ where $\varphi = \mathit{flat}\ U_{[a_1,b_1)}\ \mathit{cement}\ U_{[a_2,b_2)}\ \mathit{grass}$ and $\psi = \mathit{true}\ U_{[0,t)}\ \mathit{trap}$. 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: $\mathcal{P}_{=?}(s = 1 \wedge s_1\ U_{[0,T)}\ s = 2 \wedge s_2\ U_{[0,T)} \cdots U_{[0,T)}\ s = k \wedge s_k)$, where $s = i$ $(i = 1, \cdots, k)$ means that the server is at the i-th station, and $s_k = 1$ shows that the k-th station is full. The right part of Table 2 shows the results by fixing $T = 10$.

Random Robot							Cyclic Polling Server						
N	Before Product		After Product		time(s)	result	N	Before Product		After Product		time(s)	result
	states	transitions	states	transitions				states	transitions	states	transitions		
70	4900	14463	4869	14373	0.07	0.02602991	7	1345	6273	993	3456	<0.01	0.01779250
100	10000	29513	9941	29399	0.14	0.00213756	8	3073	15873	2305	8960	0.02	0.00908245
120	14400	43094	14288	42761	0.22	0.00210038	9	6913	39169	5249	22528	0.03	0.00462759
150	22500	67288	22301	66707	0.39	0.00106487	10	15361	94721	11777	55296	0.11	0.00235415
200	40000	119397	39748	118656	0.98	0.00107120	11	33793	225281	26113	133120	0.33	0.00119605
300	90000	268560	89330	266591	2.42	0.00077518	12	73729	528385	57345	315392	1.22	0.00060700
350	122500	366852	121695	364570	3.36	0.01079894	13	159745	1224750	124929	737280	5.06	0.00030786

Table 2. Experimental results.

Table 2 gives the number of states and transitions of original CTMC and product CTMC for each model. From this table, we can conclude that the product construction decreases the size of original CTMC since it filters out the irrelevant paths w.r.t. the properties we want to verify. For Random Robot, the size of product CTMC does not decrease so much, this depends on the

[4] PRISM is available at: `http://www.prismmodelchecker.org/`.

[5] Detailed information can be found on:
`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

1. Aziz, A., Sanwal, K., Singhal, V., Brayton, R.: Model-checking continous-time Markov chains. ACM TCL **1**(1) (2000) 162–170
2. Baier, C., Haverkort, B.R., Hermanns, H., Katoen, J.P.: Model-checking algorithms for continuous-time Markov chains. IEEE TSE **29**(6) (2003) 524–541
3. Hinton, A., Kwiatkowska, M., Norman, G., Parker, D.: PRISM: a tool for automatic verification of probabilistic systems. In: TACAS. Volume 3920 of LNCS., Springer (2006) 441–444
4. Katoen, J.P., Zapreev, I.S., Hahn, E.M., Hermanns, H., Jansen, D.N.: The ins and outs of the probabilistic model checker MRMC. PEVA **68**(2) (2011) 90–104
5. Zhang, L., Jansen, D.N., Nielson, F., Hermanns, H.: Efficient CSL model checking using stratification. LMCS **8**(2:17) (2012) 1–18
6. Gao, Y., Xu, M., Zhan, N., Zhang, L.: Model checking conditional CSL for continuous-time Markov chains. IPL (2012) 44–50
7. Zhang, L., Jansen, D.N., Nielson, F., Hermanns, H.: Automata-based CSL model checking. In: ICALP. LNCS 6756, Springer (2011) 271–282
8. Barbot, B., Chen, T., Han, T., Katoen, J., Mereacre, A.: Efficient CTMC model checking of linear real-time objectives. TACAS (2011) 128–142
9. Ibe, O., Trivedi, K.: Stochastic Petri net models of polling systems. IEEE JSAC **8**(9) (1990) 1649–1657
10. Haverkort, B., Hermanns, H., Katoen, J.P.: On the use of model checking techniques for dependability evaluation. In: SRDS. (October 2000) 228–237
11. Muppala, J., Ciardo, G., Trivedi, K.: Stochastic reward nets for reliability prediction. Communications in Reliability, Maintainability and Serviceability **1**(2) (July 1994) 9–20
12. Donatelli, S., Haddad, S., Sproston, J.: Model checking timed and stochastic properties with CSL^{TA}. IEEE TSE **35**(2) (2009) 224–240
13. Chen, T., Han, T., Katoen, J.P., Mereacre, A.: Quantitative model checking of continuous-time Markov chains against timed automata specifications. In: LICS, IEEE Comp. Soc. (2009) 309–318