The COMICS Tool – <u>C</u>omputing <u>M</u>inimal <u>C</u>ounterexample<u>s</u> for DTMCs

Nils Jansen[1], Erika Ábrahám[1], Matthias Volk[1], Ralf Wimmer[2],
Joost-Pieter Katoen[1], and Bernd Becker[2]

[1] RWTH Aachen University, Germany
[2] Albert-Ludwigs-University Freiburg, Germany

Abstract. This paper presents the tool COMICS 1.0, which performs model checking and generates counterexamples for DTMCs. For an input DTMC, COMICS computes an abstract system that carries the model checking information and uses this result to compute a *critical subsystem*, which induces a counterexample. This abstract subsystem can be refined and concretized *hierarchically*. The tool comes with a command line version as well as a graphical user interface that allows the user to interactively influence the refinement process of the counterexample.

1 Introduction

Discrete-time Markov chains (DTMCs) are widely used to model safety-critical systems with uncertainties. Model checking *probabilistic computation tree logic* (PCTL) properties can be performed by prominent tools like Prism [1] and Mrmc [2]. Unfortunately, the implemented numerical methods do not provide diagnostic information in form of *counterexamples*, which are very important for debugging and are also needed for CEGAR frameworks [3].

Although different approaches [4,5,6] were proposed for probabilistic counterexamples, there is still a lack of efficient and user-friendly *tools*. To fill this gap, we developed the tool COMICS, supporting SCC-based model checking [7] and, in case the property is violated, the *automatic* generation of *abstract counterexamples* [5], which can be subsequently refined either automatically or user-guided.

While most approaches represent probabilistic counterexamples as sets of paths, we use (hierarchically abstracted) subgraphs of the input DTMC, so-called *critical subsystems*. The user can refine abstract critical subsystems *hierarchically* by choosing system parts of interest which are to be concretized and further examined. All computation steps of the hierarchical counterexample refinement can be *guided and revised*. Though refinement can be done until a fully concrete counterexample is gained, it seems likely that the user can gain sufficient debugging information from abstract systems considering real-world examples with millions of states. The tool's graphical user interface (GUI) permits *visualization, reviewing and creation* of test cases.

The only other available tool we are aware of is DiPro [8], which supports both DTMCs and CTMCs but no abstract counterexamples, which is crucial for

the handling of large systems. It also does not allow the user to influence the search by using his or her expertise. Comparative experiments show that we can compute reasonably smaller counterexamples in shorter time with our tool.

In Section 2 we give a brief introduction to the methods implemented in our tool. We describe the features and architecture and report on benchmarks in Section 3. We conclude the paper in Section 4. The tool, a detailed manual, and a number of benchmarks are available at the `COMICS` website[3].

2 Foundations

In this section we briefly explain the algorithms implemented in `COMICS` (see [5] for more details). We use the standard definitions for DTMCs and PCTL.

Model checking time-unbounded PCTL properties for DTMCs can be reduced to the following problem: Given a DTMC M with one *initial state* s_I and a set of *target states* T, decide whether the probability to reach T from s_I is below an upper bound[4] $\lambda \in [0,1] \subset \mathbb{R}$. In case this bound is violated, a *counterexample* can be given as a set of finite paths of M leading from s_I to T with a cumulated probability mass greater than λ.

In [7] we proposed a model checking approach for DTMCs based on *hierarchical abstraction*. The result is an abstract DTMC, which represents the total probabilities of reaching target states from the initial state by single transition probabilities. The abstraction is hierarchically refinable, where the refinement of an abstract state might again contain abstract states. Based on this approach, in [5] we presented a method to compute and represent counterexamples as *critical subsystems*, consisting of subsets of the original DTMC's states and transitions such that the probability of reaching target states from the initial state within the subsystem still exceeds the probability bound λ. We compute these subsystems using path searches on the abstract DTMCs: either the *global search* (`GS`), which searches for most probable paths from s_I to T, or the *local search* (`LS`), which connects fragments of already found paths to extend the current subsystem. Abstract subsystems can be refined by *selecting* and *concretizing* abstract states and performing path search again to reduce the number of concretized states and transitions in the subsystem.

3 The COMICS Tool

`COMICS` can be used either as a command-line tool or with a GUI, the latter allowing the user to actively influence the process of finding a counterexample. The program consists of approximately 20 000 lines of code in five main compo-
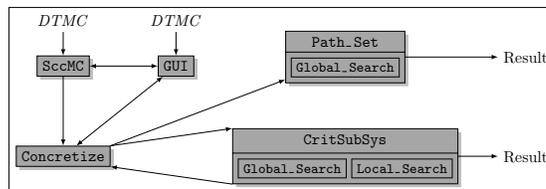

Fig. 1. Architecture of `COMICS`

nents (see Fig. 1). The GUI is implemented in `Java`, all other components in `C++`. The user may select *exact* or *floating point* arithmetics for the computations.

---

[3] http://www-i2.informatik.rwth-aachen.de/i2/comics/

[4] We only consider unbounded PCTL properties.

Fig. 2. Screenshot of `COMICS`'s GUI with an instance of the crowds protocol

`SccMC` performs *model checking* for an input DTMC and returns an abstract DTMC to `Concretize` or to `GUI`. `Concretize` *selects and concretizes* some states, either automatically or user-guided via the GUI. `CritSubSys` can be invoked on the modified system to compute a critical subsystem using `GS` or `LS`. The result is given back to `Concretize` for further refinement or returned as the result. *Heuristics* for the number of states to concretize in a single step as well as for the choice of states are offered. It is also possible to predefine the number of concretization steps. Counterexample representations as *sets of paths* and as *critical subsystems* are offered. The first case yields a *minimal counterexample* [4]. The `GUI` provides a *graph editor* for specifying and modifying DTMCs. A large number of *layout algorithms* increase the usability even for large graphs. Both concrete and abstract graphs can be *stored*, *loaded*, *abstracted*, and *concretized* by the user. As the most important feature, the user is able to *control the hierarchical concretization* of a counterexample. If an input graph seems too large to display, the tool offers to operate without the graphical representation. In this case the abstract graph can be computed and refined in order to reduce the size. Fig. 2 shows one abstracted instance of the *crowds protocol* benchmark [9], where the probability of reaching the unique target state is displayed in the information panel on the right as well as on the edge leading from the initial state to the target state. The initial state is abstract and can therefore be expanded.

Fig. 3 provides a comparison with DIPRO [8]. We applied our tool using `GS`, `LS` and the k-shortest path (kSP) approach [4] to the *crowds protocol* and the *probabilistic contract signing protocol* [10] for different probability thresholds all smaller than the model checking result (total prob.). We measured the size of the counterexample (states), the probability of reaching target states (prob.) and the computation time excluding the initial model checking. TO denotes timeout, MO out of memory and ERR wrong result. On the crowds protocol, `GS` performs best, while `LS` computes in general smaller counterexamples. kSP is the fastest method for contract signing, however, the representation of the result consists of a huge number of paths instead of a small subsystem of the input DTMC.

		crowds						contract signing			
states		3515		18817	198199	485941	1058353	33790	156670	737278	1654782
transitions		6035		32677	198199	857221	1872313	34813	157693	753663	1671165
total prob.		0.2346		0.4270	0.7173	0.809	0.8731	0.5156	0.5156	0.5039	0.5039
prob. threshold		0.15	0.23	0.25	0.35	0.4	0.4	0.5	0.5	0.5	0.5
GS	# states	629	1071	2036	5198	5248	5250	6827	37601	140034	369448
	prob.	0.1501	0.2301	0.25	0.3503	0.4002	0.4001	0.5	0.5	0.5	0.5
	time (s)	0.02	0.38	0.38	7.97	16.36	18.78	0.36	2.98	238.82	605.81
LS	# states	182	900	943	4180	6368	TO	6657	37377	MO	MO
	prob.	0.1501	0.2302	0.2501	0.3501	0.4		0.5	0.5		
	time (s)	0.14	1.11	6.1	619.06	2455.46		8	54.58		
kSP	# states	1071						6827	37601	140034	369444
	prob.	0.15	TO	TO	TO	TO	TO	0.5	0.5	0.5	0.5
	time (s)	6.58						1.93	0.13	0.69	1.49
DiPro	# states	938	2901	3227	9005	ERR	ERR	13311	74751	MO	MO
	prob.	0.1675	0.2334	0.254	0.3533			0.5	0.5		
	time (s)	2.02	7.06	7.87	44.34			1210	7114		

Fig. 3. Results for crowds and contract signing (TO $> 2h$)

4 Conclusion and Future Work

We presented version 1.0 of our tool COMICS which generates abstract, hierarchically refinable counterexamples for DTMCs. In the future, we will integrate the computation of *minimal* critical subsystems [6] and the adaption of our approaches to *symbolic data structures*. We are also working on an *incremental version of the Dijkstra algorithm* for path search and on *compositional counterexamples*.

References

1. Kwiatkowska, M.Z., Norman, G., Parker, D.: Prism 4.0: Verification of probabilistic real-time systems. In: Proc. of CAV. Volume 6806 of LNCS, Springer (2011) 585–591
2. Katoen, J.P., Zapreev, I.S., Hahn, E.M., Hermanns, H., Jansen, D.N.: The ins and outs of the probabilistic model checker MRMC. Perform. Eval. **68**(2) (2011) 90–104
3. Hermanns, H., Wachter, B., Zhang, L.: Probabilistic CEGAR. In: Proc. of CAV. Volume 5123 of LNCS, Springer (2008) 162–175
4. Han, T., Katoen, J.P., Damman, B.: Counterexample generation in probabilistic model checking. IEEE Trans. on Software Engineering **35**(2) (2009) 241–257
5. Jansen, N., Ábrahám, E., Katelaan, J., Wimmer, R., Katoen, J.P., Becker, B.: Hierarchical counterexamples for discrete-time Markov chains. In: Proc. of ATVA. Volume 6996 of LNCS, Springer (2011) 443–452
6. Wimmer, R., Jansen, N., Ábrahám, E., Becker, B., Katoen, J.P.: Minimal critical subsystems for discrete-time Markov models. In: Proc. of TACAS. LNCS, Springer (2012)
7. Ábrahám, E., Jansen, N., Wimmer, R., Katoen, J.P., Becker, B.: DTMC model checking by SCC reduction. In: Proc. of QEST, IEEE CS (2010) 37–46
8. Aljazzar, H., Leitner-Fischer, F., Leue, S., Simeonov, D.: DiPro – A tool for probabilistic counterexample generation. In: Proc. of SPIN. Volume 6823 of LNCS, Springer (2011) 183–187
9. Reiter, M.K., Rubin, A.D.: Crowds: Anonymity for web transactions. ACM Trans. on Information and System Security **1**(1) (1998) 66–92
10. Norman, G., Shmatikov, V.: Analysis of probabilistic contract signing. Journal of Computer Security **14**(6) (2006) 561–589