ProbDiVinE-MC: Multi-Core LTL Model Checker for Probabilistic Systems*

Jiří Barnat, Luboš Brim, Ivana Černá, Milan Češka, and Jana Tůmová
Faculty of Informatics, Masaryk University, Botanicka 68a, 60200 Brno, Czech Republic

Abstract

We present a new version of PROBDIVINE *– a parallel tool for verification of probabilistic systems against properties formulated in linear temporal logic. Unlike the previous release [1], the new version of the tool allows for both quantitative and qualitative model-checking. It is also strictly multi-threaded, therefore, protects users from unwanted burden of parallel computing in a distributed-memory environment.*

1. Introduction

Model-checking of probabilistic systems splits into qualitative and quantitative branch. While in qualitative verification the procedure decides whether the property holds with probability one or less, in the quantitative approach the procedure decides whether the probability of a certain property meets a given lower or upper bound.

There are several model-checking tools available for qualitative and quantitative verification of probabilistic systems. Similarly to the non-probabilistic case, the tools suffer from the well known state space explosion problem. Therefore, they apply various techniques to fight it and to extend the applicability of the tool. For branching time logic, we should mention PRISM [6] – a model-checker that uses symbolic state space representation, or MDP model-checker RAPTURE [2] that builds upon an automatic abstraction refinement and essential state reduction techniques. For linear time logic, the standard partial order reduction was implemented in LiQuor [3]. An alternative approach for coping with the state space explosion is also to investigate alternate computer architectures such as parallel or distributed systems. As an example of this approach we refer to the previous release of PROBDIVINE [1] that was capable of utilizing aggregate memory of computers in a network of interconnected workstations.

In this paper we consider automata-theoretic approach for enumerative LTL model-checking of probabilistic systems represented as Markov decision processes (MDP). The property to be verified is negated and the negation is expressed as a semi-deterministic Büchi automaton [4] which is then multiplied with an MDP of the system under consideration into a product Büchi MDP. Having the graph of the Büchi MDP, the problem of *qualitative* verification is reduced to the problem of detection of a reachable accepting ergodic component (AEC) in the graph [5]. The detection of AECs may be done in time almost linear with respect to the size of the product MDP. Therefore, the main factor that limits the applicability of a qualitative tool are the memory requirements for storing visited states of the MDP. However, this is not the case in the *quantitative* approach where the detection of AECs is further succeeded by a transformation of the product MDP into a set of linear inequalities (linear program) to be solved by some linear programming solver. Experience shows, that the solver is the bottleneck point as finding the optimal solution to the linear program is rather expensive in time as compared to the AEC detection. What our tool does to solve the quantitative problem efficiently is that it applies several subtle techniques to decompose the linear program into many smaller ones and uses parallel calls to the solver to partially remedy the limiting computational time factor.

2. ProbDiVinE-MC

First, we would like to state explicitly what are the differences between the previous release, referred to as PROBDIVINE, and the new version, reffered to as PROBDIVINE-MC. PROBDIVINE [1] allows users to perform parallel verification of qualitative aspects of probabilistic systems using distributed-memory environment, i.e. using aggregate power of computers in a cluster. On the other hand, PROBDIVINE-MC allows users to perform both *parallel qualitative* and *parallel quantitative* verification of probabilistic systems using shared-memory environment. Shared-memory parallelism became popular in recent years mainly due to the general availability of multi-cored CPUs and due to the fact that unlike the distributed-memory applications, shared-memory applications are eas-

*This work has been supported in part by the Czech Science Foundation grant No. GA201/06/1338 and by the Academy of Sciences of the Czech Republic grant No. 1ET408050503.

Model	# inequalities for LP solver			% of the whole graph	
	whole graph	reduced graph	largest subproblem	reduced graph	largest problem
Cons	55734	52776	30774	94.7	55.2
Phils	2817561	184604	246	6.6	Almost 0
Stabi	6897480	5983080	0	86.7	0
Leads	8800096	5678656	0	64.5	0
Crypts	8954217	132183	0	1.5	0

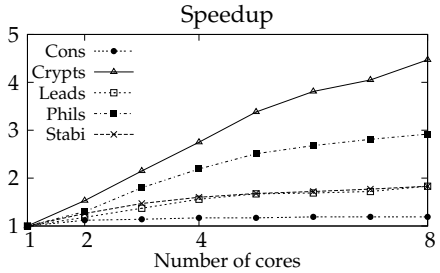

Figure 1. a) Size of LP problem with respect to used reduction techniques b) Overall speedup.

ier to be used by inexperienced users. Also the computational requirements of the quantitative analysis simply render shared-memory parallelism more convenient. Both PROBDIVINE and PROBDIVINE-MC use the ProbDVE input language [1].

For the construction of the tool, we have analyzed and reviewed the process of quantitative model checking phase by phase and pointed out several algorithmic modifications that have significant impact on a general performance of a quantitative verification tool. We also identified independent subtasks within individual phases of the process and apply parallel processing when possible. In particular, the tool employs our parallel technique to identify and remove states of MDP that are irrelevant with respect to the solution of the verification process. Then it applies a parallel algorithm for the detection of strongly connected components to partition the global LP problem into independent subproblems, solves the subproblems in parallel, and computes the global solution from the solutions to the subproblems. We have also identified a particular type of a subproblem for which the solution can be derived directly from the inequalities without using a call to an external LP solver.

3. Experimental evaluation

We have implemented the tool using the DiVinE Library and generally available LP solver `lpsolve`. We run a set of experiments on machines equipped with Intel Xeon 5130 and AMD Opteron(tm) 885 processors allowing us efficiently measure the performance of the tool when using 1 to 8 threads. In this paper we report on two different experiments related to quantitative verification only.

The table in Figure 1.a) captures the size of the linear programming problem (the number of inequalities to be solved by an LP solver) for five different models before and after the application of our reduction techniques. The first column (whole graph) gives the size before any reduction, the second column (reduced graph) gives the size when redundant inequalities were removed, and the third column (largest subproblem) gives the maximal size of a subproblem solved by an LP solver. Note that in some cases (largest subproblem = 0) we were able to find the global solution without calling LP solver at all. The size of the problem plays a crucial role in the performance of the tool. For example, the time consumed by the `lpsolve` to solve the model of Philosophers (Phils) was more than 2 days in the case of the whole graph, 45 minutes in the case of the reduced graph, and only 38 seconds, when all our reduction techniques were applied.

The verification process involves the parallel detection of AECs in an implicitly given graph, parallel detection and removal of redundant inequalities in the linear program, parallel decomposition of the linear program into subprograms, and all the concurrent calls to the LP solver. Figure 1.b) reports on the overall speed-up in the verification process we achieved using our tool on various number of CPU cores. We claim that our approach is quite successful as overall runtimes tend to decrease as more CPU cores are used. The tool is available at `http://anna.fi.muni.cz/probdivine`.

References

[1] J. Barnat, L. Brim, I. Černá, M. Češka, and J. Tůmová. ProbDiVinE: A Parallel Qualitative LTL Model Checker. In *Proc. of QEST'07*, pages 215–216. IEEE Computer Society, 2007.

[2] B.Jeannet, P. de Argenio, and K. Larsen. RAPTURE: A tool for verifying Markov Decision Processes. In *Proc. Tools Day / CONCUR'02. Tech.Rep. FIMU-RS-2002-05*, pages 84–98. MU Brno, 2002.

[3] F. Ciesinski and C. Baier. LiQuor: A tool for Qualitative and Quantitative Linear Time analysis of Reactive Systems. In *Proc. of QEST'06*, pages 131–132. IEEE Computer Society, 2006.

[4] C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907, 1995.

[5] L. de Alfaro. *Formal Verification of Stochastic Systems*. PhD thesis, Stanford University, Department of Computer Science, 1997.

[6] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In *Proc. of TACAS'06*, volume 3920 of *LNCS*, pages 441–444. Springer, 2006.