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

Jiři Barnat, Luboš Brim, Ivana Černa, Milan Češka, and Jana Tůmova
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 [7] – a model-checker that use 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 [5] 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 [6]. 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 when searching for an AEC. 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 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 differencies between ProbDiVinE and ProbDiVinE-MC. The ProbDiVinE tool [1] allow users to perform parallel verification of qualitative aspects of probabilistic systems using distributed-memory environmnet, 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 easier to be used by inexperienced users. Also the computational require-

*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.

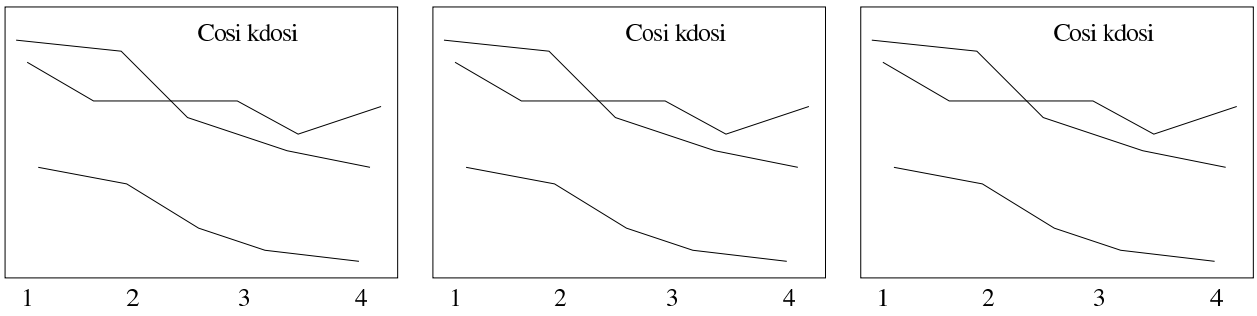

Figure 1. a) Overall runtimes. b) Reduction in the size of linear programm when redundant inequalities are removed. c) Time spent in LP solver for a single and partitioned linear pragram and various numbers of threads.

ments of the quantitative analysis simply render shared-memory parallelism more convenient. Both ProbDiVinE and ProbDiVinE-MC use the same ProbDVE input language [1].

- omit inequalities that cannot influence the result (detect them in parallel)
- decompose the dependency graph into SCC, solve individual SCC bottom-up and in parallel
- solve trivial SCCs without lpsolve

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 a systems with Intel Xeon 5130 and AMD Opteron(tm) 885 processors allowing us efficiently measure the performance of the tool when using 1 to 4 and 1 to 8 threads, respectively.

In this short paper we report on three different experiments related to quantitative verification only. Figure 1.a depicts the size of the linear program (the number of inequalitites) before and after the redundant inequalities are removed for 4 quite different systems. Figure 1 captures the time spent on solving the linear programm with a single thread if the linear program is not decomposed and the time spent on decomposing and solving all the parts of the decompose program using one and more threads. Finally, Figure 1.c reports on the overall runtime of the verification process using various number of CPU cores. The verification process involves the parallel detection of AECs in an implictly given graph, parallel detection of redundant inequalitites in the linear programm and their parallel removal, parallel decomposition of the linear program into its components, and all the concurrent calls to the LP solver. We claim that our approach is quite successful as overall runtimes tend to decrease as more CPU cores are used.

References

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

[2] B.Jeannet, P.dArgenio, and K.G. Lar. RAPTURE: A tool for verifying Markov Decision Processes. In *Proc. Tools Day / CONCUR02. 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. submitted for publication, 2006.

[4] E. M. Clarke, Jr., O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, Cambridge, Massachusetts, 1999.

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

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

[7] M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with PRISM: a hybrid approach. *STTT*, 6(2):128–142, 2004.

[8] M. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Proc. FOCS 1985*, pages 327–338. IEEE Computer Society Press, 1985.