PASS: Abstraction Refinement for Infinite Probabilistic Models*

Ernst Moritz Hahn[1], Holger Hermanns[1], Björn Wachter[1], and Lijun Zhang[2]

[1] Saarland University, Saarbrücken, Germany
[2] Oxford University Computing Laboratory, UK

Abstract. We present PASS, a tool that analyzes concurrent probabilistic programs, which map to potentially infinite Markov decision processes. PASS is based on predicate abstraction and abstraction refinement and scales to programs far beyond the reach of numerical methods which operate on the full state space of the model. The computational engines we use are SMT solvers to compute finite abstractions, numerical methods to compute probabilities and interpolation as part of abstraction refinement. PASS has been successfully applied to network protocols and serves as a test platform for different refinement methods.

1 Introducing PASS

Network protocols are subject to random phenomena like unreliable communication and employ randomization as a strategy for collision avoidance. Further, they are often distributed and thus inherently concurrent. To account for both randomness and concurrency, Markov decision processes (MDPs) are used as a semantic foundation as they feature both non-deterministic and probabilistic choice. Typically one is interested in computing (*maximal* or *minimal*) *reachability probabilities*, e.g., of delivering three messages after ten transmission attempts (under *best-case* and *worst-case* assumptions concerning the environment).

Probabilistic reachability is expressible in terms of least fixed points of a system of recursive equations [1] where the unknowns correspond to the probability of an individual state. For finite MDPs, probabilistic reachability can be reduced to linear programming [2] or solved approximately by value iteration. Current implementations, e.g., in the popular PRISM model checker [3], use numerical methods like value iteration. However, the infamous state explosion problem is even more severe than in the qualitative setting. Explicit-state methods do not scale well in presence of expensive numerical computations. Symbolic techniques are often not effective because the probabilities arising as intermediate results of computations exhibit little structure or regularity to exploit.

* This work is supported by the NWO-DFG bilateral project ROCKS, by the DFG as part of the Transregional Collaborative Research Center SFB/TR 14 AVACS and the Graduiertenkolleg "Leistungsgarantien für Rechnersysteme", and has received funding from the European Community's Seventh Framework Programme under grant agreement n^o 214755.

PASS[1] uses the same principal machinery, but aims at supporting infinite or very large models by resorting to counterexample-guided abstraction refinement (CEGAR): instead of exploring the state space of the model, PASS uses predicate abstraction to maintain a finite abstract model. Analysis of the abstract model is typically very efficient since it has few states. It yields probability intervals that are guaranteed to contain the probabilities in the original model. The difference between interval bounds quantifies the approximation error caused by abstraction. The abstraction is refined until the approximation error is small enough. Otherwise the abstract model provides diagnostic information to refine the abstraction. The process is described in [4,5]. A major difference to conventional CEGAR for predicate abstraction lies in the notion and interpretation of counterexamples: counterexamples are Markov chains rather than single paths.

Predicate abstraction for probabilistic models [6] and suitable refinement techniques [4] premiered in PASS. Preceding abstraction-refinement methods in the probabilistic setting like magnifying-lens abstraction [7] or RAPTURE [8] are restricted to finite models, since they locally unfold the state space of the original model. Our previous version PASS 1.0 has only been able to compute effective upper bounds on probabilities rather than probability intervals.

Kwiatkowska et al. pioneered game-based abstractions [9] which have the benefit of providing safe upper and lower bounds. The idea is that the abstraction distinguishes two kinds of non-determinism: non-determinism present in the original model and non-determinism that results from abstraction. This has been applied to a sequential C-like language with probabilistic choice but without concurrency [10]. In [11], concurrent probabilistic programs have been considered, but without refinement and only for finite models.

To be able to compute probability intervals, we have recently enhanced the PASS machinery with notions of game-based abstraction [5]. To this end, we have introduced a coarser game-based abstraction, called *parallel abstraction*. It can be efficiently computed for concurrent probabilistic programs and yields tight probability bounds, as shown by our experimental results [5]. Beside this feature, PASS has been improved in terms of robustness, efficiency and usability.

2 Architecture

The architecture of PASS, depicted in Figure 1, revolves around an abstraction refinement loop.

PASS reads programs in a concurrent, guarded-command language extending the one of PRISM. The semantics of a program is an MDP in which each state is a valuation of program variables. Initial states are specified by an expressions over program variables. The rest of the description consists of commands. Each command comprises a guard and a set of probabilistic alternatives. Each alternative is associated with a probability and an update formula. If a state fulfills the guard of a command, this state has a probabilistic choice to go to each state obtained by the respective update formula. Unlike PRISM, we allow variables

[1] The acronym stands for Predicate Abstraction for Stochastic Systems.

with infinite range. Probabilistic reachability properties are specified by giving an expression that defines the set of goal states. PASS then computes probability bounds to reach them.

We use the *predicate abstraction* method of [5] where probabilistic programs are abstracted to stochastic games [12]. The abstraction is implemented using SMT-based enumeration. The abstractions of the commands are stored in BDDs. Prior to the quantitative analysis, we perform a preprocessing step where we prune the abstract state space to the states that are both reachable from an initial abstract state and can reach a goal state. To this end, we employ a BDD-based forward and backward analysis respectively.

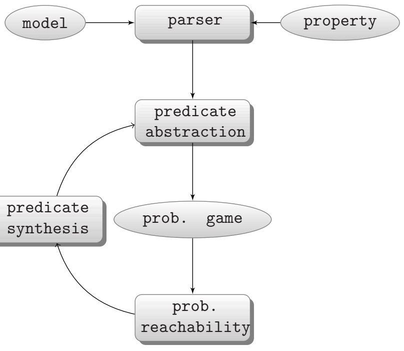

Fig. 1. Architecture of PASS

The stochastic game is first converted from a symbolic BDD-based representation to a sparse-matrix representation. Then the lower and upper bound probabilities are computed by *value iteration*. Value iteration also generates game strategies, a resolution of non-determinism in the game that witnesses the obtained probabilities. These strategies form the foundation for the notion of an abstract probabilistic counterexample [4], which can be used to either refute properties or provide diagnostic information to refine the abstraction.

PASS supports two different refinement methods: Probabilistic CEGAR [4], which analyzes probabilistic counterexamples based on the idea of strongest evidence [13], and the method in [5], which splits abstract states where certain strategies in the abstract game and the obtained bounds indicate a loss of precision. Both methods have their benefits. A comparison is given in [5].

3 Selected Features

Several new features have not been covered in previous publications [6,4,5].

Improved value iteration scheme. It is important to use an efficient value iteration scheme since this step has to be repeated after each refinement step. The order in which value iteration updates the probabilities at a state has a significant impact on the number of iterations. The value of a state depends on its successors. Following the dependencies in the evaluation order can significantly speed up value iteration [14]. PASS now performs value iteration according to a reversed depth-first order starting with the goal states.

Interpolation. PASS uses interpolation to analyze paths of the abstract model. We have written a wrapper to include different interpolation tools with implemented bindings for MathSAT [15], CSIsat [16] and FOCI [17].

On-the-fly Abstraction. To only compute transitions of abstract states that are actually reachable, PASS computes the abstraction layer-wise starting with the initial states interleaving state exploration with SMT-based abstraction. In order to benefit from learned clauses and avoid a repetitive build up of the SMT problem, on-the-fly abstraction employs incremental SMT solving across layers.

4 Concluding Remarks

PASS consists of approximately 18.000 lines of C++ code, and has been tested on a large number of case studies. It is available for Linux with libc6. A PASS executable and case studies can be downloaded from:

`http://depend.cs.uni-sb.de/pass`

References

1. Baier, C.: On Algorithmic Verification Methods for Probabilistic Systems, Habilitationsschrift, Universität Mannheim (1998)
2. Bianco, A., de Alfaro, L.: Model Checking of Probabilistic and Nondeterministic Systems. In: Thiagarajan, P.S. (ed.) FSTTCS 1995. LNCS, vol. 1026, pp. 499–513. Springer, Heidelberg (1995)
3. Hinton, A., Kwiatkowska, M., Norman, G., Parker, D.: PRISM: A Tool for Automatic Verification of Probabilistic Systems. In: Hermanns, H., Palsberg, J. (eds.) TACAS 2006. LNCS, vol. 3920, pp. 441–444. Springer, Heidelberg (2006)
4. Hermanns, H., Wachter, B., Zhang, L.: Probabilistic CEGAR. In: Gupta, A., Malik, S. (eds.) CAV 2008. LNCS, vol. 5123, pp. 162–175. Springer, Heidelberg (2008)
5. Wachter, B., Zhang, L.: Best Probabilistic Transformers. In: Barthe, G., Hermenegildo, M. (eds.) VMCAI 2010. LNCS, vol. 5944, pp. 362–379. Springer, Heidelberg (2010)
6. Wachter, B., Zhang, L., Hermanns, H.: Probabilistic Model Checking Modulo Theories. In: QEST (2007)
7. de Alfaro, L., Roy, P.: Magnifying-Lens Abstraction for Markov Decision Processes. In: Damm, W., Hermanns, H. (eds.) CAV 2007. LNCS, vol. 4590, pp. 325–338. Springer, Heidelberg (2007)
8. D'Argenio, P.R., Jeannet, B., Jensen, H.E., Larsen, K.G.: Reachability Analysis of Probabilistic Systems by Successive Refinements. In: de Luca, L., Gilmore, S. (eds.) PROBMIV 2001, PAPM-PROBMIV 2001, and PAPM 2001. LNCS, vol. 2165, pp. 39–56. Springer, Heidelberg (2001)
9. Kwiatkowska, M., Norman, G., Parker, D.: Game-based Abstraction for Markov Decision Processes. In: QEST, pp. 157–166 (2006)
10. Kattenbelt, M., Kwiatkowska, M., Norman, G., Parker, D.: Abstraction Refinement for Probabilistic Software. In: Jones, N.D., Müller-Olm, M. (eds.) VMCAI 2009. LNCS, vol. 5403, pp. 182–197. Springer, Heidelberg (2009)
11. Kattenbelt, M., Kwiatkowska, M., Norman, G., Parker, D.: Game-Based Probabilistic Predicate Abstraction in PRISM. In: QAPL (2008)
12. Condon, A.: The Complexity of Stochastic Games. Inf. Comput. 96, 203–224 (1992)
13. Han, T., Katoen, J.P.: Counterexamples in probabilistic model checking. In: Grumberg, O., Huth, M. (eds.) TACAS 2007. LNCS, vol. 4424, pp. 72–86. Springer, Heidelberg (2007)

14. Dai, P., Goldsmith, J.: Topological Value Iteration Algorithm for Markov Decision Processes. In: IJCAI, pp. 1860–1865 (2007)
15. Bruttomesso, R., Cimatti, A., Franzén, A., Griggio, A., Sebastiani, R.: The MathSAT 4 SMT Solver. In: Gupta, A., Malik, S. (eds.) CAV 2008. LNCS, vol. 5123, pp. 299–303. Springer, Heidelberg (2008)
16. Beyer, D., Zufferey, D., Majumdar, R.: CSIsat: Interpolation for LA+EUF. In: Gupta, A., Malik, S. (eds.) CAV 2008. LNCS, vol. 5123, pp. 304–308. Springer, Heidelberg (2008)
17. McMillan, K.L.: An Interpolating Theorem Prover. Theor. Comput. Sci. 345, 101–121 (2005)