Employing AI Techniques in Probabilistic Model Checking Position Paper

Robert P. Goldman and **Michael W. Boldt** and **David J. Musliner**
SIFT, LLC
319 First Avenue North, Suite 400
Minneapolis, MN 55401
rpgoldman@sift.net

Abstract

Probabilistic model-checking (PMC) is a new frontier in model checking verification, useful for checking randomized algorithms, communications protocols with random components, etc. AI techniques have been incorporated into conventional model-checking, with great success. We wish to see if corresponding successes can be found by applying AI techniques to PMC. In this position paper, we introduce PMC. We present the probabilistic automata and probabilistic bounded LTL used in modeling for PMC. We also introduce some existing tools for PMC. With this background in place, we discuss challenges and opportunities for incorporating AI techniques in PMC, and our current work in the area.

Introduction

Model-checking has been very successful in identifying and removing faults in conventional hardware and software. Heuristic search and AI planning are very closely related to model-checking verification, and such AI techniques have been successfully incorporated, under the rubric of directed model-checking (Edelkamp *et al.* 2008).

One new frontier in model-checking is model-checking stochastic systems, in the form of probabilistic automata. Probabilistic automata feature both conventional non-determinism and stochastic non-determinism. Consider, for example, a communications protocol involving a random element (for example, a random back-off used to resolve collisions). In order to assess the correctness of this algorithm we must consider both the random element, and the non-deterministic choice of, say, the bits in the message. Claims here are necessarily stochastic, e.g., "the probability that a message will fail to be delivered is less than 0.0001." Assessing these claims requires resolving non-stochastic non-determinism in a worst-case way, i.e., by means of an optimal adversary. The previous claim is actually, "no matter what the message content, the probability that a message will fail to be delivered is less than 0.0001." Finally, claims almost universally must be *time bounded*, since a system that runs forever, and has any non-transient chance of failure, will have probability 1 of reaching a failure state, by the law of large numbers. To summarize, probabilistic model-checking claims will be of the form "No matter how non-determinism is resolved, the probability of reaching a failure state, within x time units is less than p."

Note: We will use the term "probabilistic model checking (PMC)" to denote model-checking of systems modeled as probabilistic automata. We distinguish this from "statistical model checking (SMC)" which we will reserve for "statistical methods for model checking." SMC methods, based on statistical sampling, may be applied to both conventional and stochastic models.

Given the success in applying AI ideas in general, and planning ideas in particular, to conventional model-checking, we hope to achieve corresponding advances by incorporating AI planning ideas into probabilistic model checking. In this position paper we discuss challenges, opportunities, and our current work.

Probabilistic Automata

The models that interest us are *probabilistic automata* (PAs). PAs are a generalization of non-deterministic finite automata, in which some transitions, rather than leading to a single destination state, may have a discrete probability distribution over a set of possible successor states. More formally, we have the following:

Definition 1 *Probabilistic Automaton (PA):*[1] *A PA,* $\mathcal{A} = \langle S_\mathcal{A}, S_0, A_\mathcal{A}, \Delta_\mathcal{A} \rangle$ *where* $S_\mathcal{A}$ *is a set of states,* $S_0 \in S_\mathcal{A}$ *is the* initial state*,* $A_\mathcal{A} = E_\mathcal{A} \cup I_\mathcal{A}$ *is the* action set *of* $\mathcal{A}$*, partitioned into the set of* external *actions,* $E_\mathcal{A}$ *and* internal *actions,* $I_\mathcal{A}$*, where* $E_\mathcal{A} \cap I_\mathcal{A} = \emptyset$.[2] *The transition relation is* $\Delta_\mathcal{A} \subseteq S_\mathcal{A} \times A_\mathcal{A} \times \mathrm{Distr}(S_\mathcal{A})$*, where* $\mathrm{Distr}(S_\mathcal{A})$ *is the set of discrete distributions over* $S_\mathcal{A}$*.*

For the purposes of verification, we wish to assess claims in *probabilistic bounded time linear temporal logic*.

Definition 2 *Bounded Time LTL: Syntax:*

$$\phi := \lambda | \neg\phi | \phi \vee \phi | F^{\leq n}\phi | G^{\leq n}\phi | \phi U^{\leq n} \phi$$

where λ *is the set of propositions,* F *is the* eventually *modal operator,* G *is the* always *modal operator, and* U *is the* until *modal operator. The superscripts on the modal operators*

[1] Definition follows (Stoelinga 2002).

[2] External actions are used to define the composition of PAs, useful for compositional modeling.

indicate time bounds. For more details on LTL, see, e.g., (Emerson 1990).

Probabilistic Bounded LTL (PBLTL) adds a probability modality, P, permitting assertions of the form, $P(\phi) \leq \psi$, such as $P(F^{\leq 12}\text{fail} \vee \text{deadlock}) \leq 0.1$: "there is less than a 10% chance of reaching a failure or deadlock state within 12 time steps." In most cases, the verification challenge is to prove that the system will never reach an unsafe state – within a given time, and with no greater than a certain probability.

Since $\Delta_{\mathcal{A}}$ is a *relation*, we have pure nondeterminism, as well as stochasticity. Any probabilistic automaton, then, corresponds to a Markov Decision Process, in which making the action choices resolves the non-determinism. When the nondeterminism is *resolved*, we have a Markov chain.

It follows from the above that the problem of *model-checking* PBLTL assertions reduces to finding an optimal policy for the corresponding MDP, where the value of the policy is the probability of ϕ (recall that ϕ is a *time-bounded* assertion). While general assertions about PAs may require stochastic policies,[3] to model-check PBLTL assertions, deterministic policies suffice (De Alfaro 1998).[4] Note: however, that these policies typically require memory (i.e., are history-dependent).

The PRISM model-checking tool[5] provides analytic and statistical model-checking of probabilistic automata (as well as other forms of finite state machines) (Kwiatkowska, Norman, and Parker 2011). For analytic model-checking, PRISM uses either value iteration or, when it needs a stochastic policy,[6] a linear programming formulation of MDPs.[7]

Statistical Model Checking (SMC) is an alternative to analytic methods. SMC methods gather samples (i.e., sequences of steps through the model) to provide statistical evidence for (or against) the model satisfying a property. To the best of our knowledge, statistical model checking methods were first used by Younes and Musliner, in work on the CIRCA planning system (Younes and Musliner 2002; Younes, Musliner, and Simmons 2003). Their work had two particularly interesting features: because their models were so expressive— their models featured concurrency and arbitrary temporal distributions, so were Generalized Semi-Markov Processes, for which no analytic methods are available— they were forced to use sampling. They also adopted adaptive sampling techniques, because the precomputed sampling plans were so inefficient. Younes generalized the statistical safety-verification methods developed for CIRCA into more expressive statistical model checking in his prize-winning thesis (Younes 2005).

More recently, Henriques *et al.* (2012) have developed SMC methods for model-checking PAs. Since model checking PAs involves solving a Markov Decision Problem, they refer to their technique as SMC for MDPs or simply SMCMDP. Theirs is a two phase approach: first, they use reinforcement learning techniques, based on Q-learning, to choose the adversary's policy. Second, they determinize the policy, and use it to resolve the nondeterminism in the PA. Then they use sampling techniques (like those of Younes and Musliner), to determine whether the property will be violated, under the control of the adversary's policy. These techniques have been built into an extended version of the PRISM tool by researchers at SIFT, CMU, Oxford, and Newcastle (Uckun *et al.* 2011; Musliner, Woods, and Maraist 2012).

Unfortunately, as used to date, the technique only learns a deterministic, stationary policy for the adversary, which is not necessarily optimal (although they suggest that if time was inserted into the state space, their technique could provide a non-stationary policy). In general, a history-sensitive (non-stationary) policy may be needed for optimality. That means that the SMCMDP technique could produce a suboptimal policy, and mistakenly conclude that a system is safe when it is not. Additionally, while the learning technique is known to converge on the optimal policy, no bounds on its distance from optimality are given. Again, this means that a system could be mistakenly certified as safe (by stopping the process too soon).

Challenges

Given past successes in applying AI techniques (especially planning techniques) to conventional model-checking (Edelkamp *et al.* 2008; Edelkamp, Leue, and Lluch-Lafuente 2004; Albarghouthi, Baier, and McIlraith 2009), we wish to see if similar progress can be made in probabilistic model-checking. Unfortunately, achieving the same kind of carry-over is more difficult here, for at least two reasons. First, the specific MDP models used in the AI community do not align perfectly with those used in probabilistic model checking. Second, the techniques developed do not always provide the kind of performance guarantees and metrics needed for model-checking. We mention some particular areas where further work would be useful, and in the following section, discuss our work in the area.

Much existing work on MDPs assumes infinite horizon problems with time-discounted, expected reward used as the criterion for optimality (Puterman 1994). These models are mathematically convenient, and perform well with value and policy iteration, but are ill-suited for probabilistic model checking. Part of the convenience of this model is that there is always an optimal policy that is stationary and deterministic. As we discuss above, for time-bounded MDPs, on the other hand, there may be no optimal policy that is stationary. For PMC infinite-horizon solving and time-discounting is inappropriate.

Inspired by the probabilistic planning part of the International Planning Competition (IPC), some researchers are exploring alternative models. For example, Kolobov, *et al.* (2011), formalize MAXPROB MDPs, where the objective is not to minimize cost or maximize reward, but simply to maximize the probability of reaching the goal. This

[3] These may be necessary to achieve conjunctions of probability bounds.

[4] Cited in (Stoelinga 2002).

[5] Available at `prismmodelchecker.org`

[6] PRISM can model-check more expressive assertions than PBLTL.

[7] David Parker, personal communication.

Model	States	True Probability	Threshold Probability	Learning Samples	Correct w/o UCT	Correct w/ UCT
CSMA 2 2	1038	0.861	0.85	2000	12%	**77%**
CSMA 2 4	7958	0.768	0.76	2000	18%	**70%**
CSMA 2 6	66718	0.616	0.61	2000	25%	**54%**
CSMA 2 6	66718	0.616	0.61	4000	22%	**74%**

Figure 1: Initial results of UCT-guided SMCMDP on different-sized models of the probabilistic CSMA protocol. For each model, we asked SMCMDP and SMCMDP with UCT-guided sampling: using the given number of samples, is the threshold probability is less than the true probability. The percentage correct is out of 100 runs.

appears to be a better fit, but it assumes an indefinite horizon, where there is no time limit within which the goal must be reached. Since in many cases of probabilistic model-checking, the probability of reaching a property-violating state *eventually* is 1, this model is also unsuitable.

At the moment (see following section), we are investigating MDP solution techniques for PMC based on find-and-revise search (Bonet and Geffner 2003), and on Monte Carlo Tree Search (MCTS) (Browne *et al.* 2012). Challenges with MCTS have to do with the approximate nature of the search. While these algorithms are known to converge to the optimal policy, they don't give good bounds on their error *vis à vis* the optimal, and those bounds are typically formulated in terms of regret, from which it's not obvious that we can compute error bounds on the probability of properties. Recall that, in verification, we may be claiming that a system is safe based upon *failure* to find a policy that violates a property. Search algorithms, on the other hand, may be able to give such bounds. It remains to be seen if they will converge adequately quickly, whether we can find informative heuristics for PA verification, and if we can control the size of the resulting policies.

Current Work

Guided Sampling with Monte Carlo Tree Search

The Monte Carlo sampling process in SMCMDP can take a long time to converge. This problem can manifest itself either in the first phase when learning the adversary's policy (resolving non-determinism in the model), or in the second phase when, after the non-determinism has been resolved, we sample from resulting the Markov Chain to evaluate the PBLTL property's worst case probability.

To improve performance in the first phase of SMCMDP, we have been experimenting with Monte Carlo Tree Search (MCTS) methods (Browne *et al.* 2012). MCTS methods incrementally build a tree of the search space, and use data in the tree to resolve non-determinism when sampling. These methods have been very successful in difficult search applications, including Computer Go, and planning under uncertainty. We have developed a new policy search for SMCMDP, using the Upper Confidence bounds applied to Trees (UCT) MCTS method (Kocsis and Szepesvári 2006) to guide the sampling.

Our initial experiments with UCT-guided sampling look promising. As seen in Figure 1, UCT yields the correct answer much more often than Q-learning-based SMCMDP, learning from the same number of samples, for difficult problems where the threshold probability is very close to the true probability. This indicates that UCT helps SMCMDP learn a *better policy* with the *same number of samples*. We plan to extend these experiments to better analyze the costs and benefits of UCT-guided SMCMDP, including running them on different types of models.

Figure 2: A simple example where the optimal adversary requires a time-dependent policy.

Time-Dependent Policies

Current SMCMDP methods produce only stationary, memoryless policies (adversaries). But when performing *bounded time* model-checking, in general the optimal adversary policy is time-dependent. That means that a model-checker using a stationary policy may incorrectly label some systems as safe. The challenge of finding time-dependent policies is twofold: (1) keeping track of time in the model causes state space explosion, and (2) in general, a separate policy is required for every time unit, making the policy very large.

We have developed several test problems for experimenting with time-dependent vs. stationary adversaries. For the simple example in Figure 2, each transition takes one time tick, and the adversary wins by driving the model into `fail`. He starts at `s0` and must choose action `L` or `S`; after that the transitions are determined by the indicated probabilities. With 4 or more time ticks left, `L` has the highest probability of failure. However, with 3 time ticks left, `fail` is unreachable through `L`, so the adversary should choose action `S`.

We have begun exploring methods for efficiently developing time-dependent adversaries. With our adoption of UCT, we have developed a method which uses the greedy MCTS tree value to resolve non-determinism. Since the tree has no cycles, it is inherently time-dependent. It is likely that the MCTS tree does not include all reachable states, so we must include a way to resolve non-determinism outside of the tree. For this, we "warp" to a node in the tree with the same state and use that node to resolve the non-determinism. If no node in the tree shares the same state, we choose a random action.

Another possibility suggested by our adoption of UCT, is to compute the adversary's "moves" on-line, avoiding the need to store full policies. Other techniques we are considering include exploiting structured state models, and compressing large policies.

Conclusions

In this paper we have introduced the problem of Probabilistic Model Checking. We have discussed both analytic and statistical based approaches to PMC. We are interested in using more informed, AI-based approaches to PMC, hoping to replicate earlier successes in conventional, non-probabilistic

model checking. We present some challenges to transferring AI techniques for solving MDPs to PMC: these should provide interesting opportunities for researchers. We ourselves are exploring the use of MCTS and find-and-revise search techniques for PMC.

Acknowledgments

This material is based upon work supported by the AFRL under Contract No. PO 284227 under CMU prime contract FA9550-12-1-0146. Any opinions, findings and conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the AFRL.

References

Albarghouthi, A.; Baier, J. A.; and McIlraith, S. A. 2009. On the use of planning technology for verification. In *Proceedings of the workshop on verification and validation of planning systems (VVPS)*.

Bonet, B., and Geffner, H. 2003. Faster heuristic search algorithms for planning with uncertainty and full feedback. In Gottlob, G., and Walsh, T., eds., *Proceedings of the International Joint Conference on Artificial Intelligence*, 1233–1238. Morgan Kaufmann.

Browne, C. B.; Powley, E.; Whitehouse, D.; Lucas, S. M.; Cowling, P. I.; Rohlfshagen, P.; Tavener, S.; Perez, D.; Samothrakis, S.; and Colton, S. 2012. A survey of monte carlo tree search methods. *Computational Intelligence and AI in Games, IEEE Transactions on* 4(1):1–43.

De Alfaro, L. 1998. *Formal Verification of Probabilistic Systems*. Ph.D. Dissertation, Stanford University, Stanford, CA, USA.

Edelkamp, S.; Schuppan, V.; Bosnacki, D.; Wijs, A.; Fehnker, A.; and Aljazzar, H. 2008. Survey on directed model checking. In *MoChArt*.

Edelkamp, S.; Leue, S.; and Lluch-Lafuente, A. 2004. Directed explicit-state model checking in the validation of communication protocols. *STTT* 5(2-3):247–267.

Emerson, E. A. 1990. Temporal and modal logic. In van-Leeuwen, J., ed., *Handbook of Theoretical Computer Science*, volume B: Formal Models. MIT Press. chapter 16, 995–1072.

Henriques, D.; Martins, J. G.; Zuliani, P.; Platzer, A.; and Clarke, E. M. 2012. Statistical model checking for markov decision processes. In *Ninth Int'l Conf. on Quantitative Evaluation of Systems*, 84–93.

Kocsis, L., and Szepesvári, C. 2006. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*. Springer. 282–293.

Kolobov, A.; Mausam; Weld, D. S.; and Geffner, H. 2011. Heuristic search for generalized stochastic shortest path MDPs. In Bacchus, F.; Domshlak, C.; Edelkamp, S.; and Helmert, M., eds., *ICAPS*. AAAI.

Kwiatkowska, M.; Norman, G.; and Parker, D. 2011. PRISM 4.0: Verification of probabilistic real-time systems. In Gopalakrishnan, G., and Qadeer, S., eds., *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *LNCS*, 585–591. Springer.

Musliner, D. J.; Woods, T.; and Maraist, J. 2012. Identifying culprits when probabilistic verification fails. In *Proc. ASME Computers and Information in Engineering Conference*.

Puterman, M. L. 1994. *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. New York, NY: John Wiley & Sons, Inc.

Stoelinga, M. 2002. An introduction to probabilistic automata. *Bulletin of the European Association for Theoretical Computer Science* 78:176–198.

Uckun, S.; Kurtoglu, T.; Bunus, P.; Tumer, I.; Hoyle, C.; and Musliner, D. J. 2011. Model-based systems engineering for the design and development of complex aerospace systems. In *SAE Aerotech Congress and Exposition*.

Younes, H. L., and Musliner, D. J. 2002. Probabilistic plan verification through acceptance sampling. In *Proc. AIPS-02 Workshop on Planning via Model Checking*, 81–88.

Younes, H. L. S.; Musliner, D. J.; and Simmons, R. G. 2003. A framework for planning in continuous-time stochastic domains. In *International Conference on Automated Planning and Scheduling*, 195–204.

Younes, H. L. S. 2005. *Verification and Planning for Stochastic Processes with Asynchronous Events*. Ph.D. Dissertation, Carnegie Mellon University.