SMT-based Counterexample Generation for Discrete-Time Markov Chains

Bettina Braitling Ralf Wimmer Bernd Becker
Albert-Ludwigs-University Freiburg, Germany
{braitlin,wimmer,becker}@informatik.uni-freiburg.de
Nils Jansen Erika Ábrahám
RWTH Aachen, Germany
{jansen,abraham}@informatik.rwth-aachen.de

Counterexamples are a highly important feature of the model checking process. A counterexample in a digital circuit typically consists of a single path leading to a set of critical states of the system. However, in the probabilistic setting counterexamples may consist of a large number of paths whose common probability measure exceeds a given bound. In order to be able to handle large systems and to use the capabilities of modern SAT-solvers, bounded model checking (BMC) for discrete-time Markov chains (DTMCs) was established.

This Stochastic BMC (SBMC) approach reaches its limit for systems with a lot of paths with low probability. Since SBMC ignores the actual path probabilities and therefore finds the paths of the same length in a random order, counterexamples may consist of more paths than necessary. To solve this problem, we introduced the usage of SMT-solving over the reals for the BMC procedure.

The SMT-based SBMC (SSBMC) is able to prefer paths with higher probability by doing a binary search on the probability space: We include the transitions probabilities p_i into the BMC-formula such that the transition predicate $T_{\mathrm{SMT}}(s_i, s_{i+1}, \hat{p}_i)$ is satisfied iff there is a transition from s_i to s_{i+1} and $\hat{p}_i$ is assigned the logarithm of its probability. The logarithm is used to turn the multiplication of probabilities along a path into a summation. We can then compare the probability of the whole path to a threshold value p_t:

$$\mathrm{BMC}_k := \mathrm{Init}(s_0) \wedge \bigwedge_{i=0}^{k-1} \mathrm{Trans}_{\mathrm{SMT}}(s_i, s_{i+1}, \hat{p}_i) \wedge \mathrm{Goal}(s_k) \wedge \left(\sum_{i=0}^{k-1} \hat{p}_i \geq \log p_t \right).$$

By adjusting p_t we can determine the most probable path (up to a gap ε) using binary search.

Besides preferring paths with higher probability, SSBMC allows us to handle Markov Reward Models (MRMs) as well. An MRM is a DTMC with a reward function which may be used, e. g., to model costs of certain operations to count steps or to measure given gratifications. Such reward functions restrict the set of valid paths to those whose accumulated reward value lies within a given interval I.

In order to include rewards into our formula we extend it in a similar way as we have done for the probabilities: In the transition predicate $T_{\text{R-SMT}}$ an additional variable $\hat{r}_i$ is introduced, which stores the reward of the transition. The sum of these variables is compared to the bounds of the reward-interval I:

$$\mathrm{BMC}_k := \mathrm{Init}(s_0) \wedge \bigwedge_{i=0}^{k-1} \mathrm{Trans}_{\text{R-SMT}}(s_i, s_{i+1}, \hat{p}_i, r_i) \wedge \mathrm{Goal}(s_k) \wedge \left(\sum_{i=0}^{k-1} \hat{p}_i \geq \log p_t \right) \wedge \left(\sum_{i=0}^{k-1} r_i \in I \right).$$

Additionally to SMT we may also compute the bisimulation quotient of a system in order to minimize it. The quotient satisfies the same properties as the original system, but has fewer paths. Thus we get a smaller, more abstract representation of the counterexample: Each path in the bisimulation quotient stands for a bundle of equivalent paths in the original system. This yields further speed-up to counterexample generation. In case we want to work with the original model again, we are also able to efficiently translate the abstract counterexample back to the original system.

First experimental results with this SMT-based approach show clear advantages over the SAT-based formulation if the analyzed system contains relevant paths with different probabilities.