Parameter Synthesis for Probabilistic Systems*

Christian Dehnert  Sebastian Junges  Nils Jansen  Florian Corzilius
Matthias Volk  Harold Bruintjes  Joost-Pieter Katoen  Erika Ábrahám

RWTH Aachen University

Many systems that are subject to verification give rise to probabilities; examples include randomized distributed algorithms, security, systems biology, or embedded systems. State-of-the-art probabilistic model checkers like PRISM [7] mostly work under the assumption that all model probabilities are a priori known. However, at early development stages, certain system quantities require parametric probabilistic models to be specified, where transition probabilities are given by real-valued parameters. Here, we focus on so-called parametric Markov chains (pMC), see Figure 1(a). The model checking goal is to compute rational functions, i.e., a fraction of polynomials over the system’s parameters, which describe probabilities of reaching certain states. This can be done via state elimination [2], as incorporated by available tools such as PARAM [4], the parametric version of PRISM [7], and our tool PROPhESY [3], which employs improved variants [5]. For the common PRISM-benchmarks, PROPhESY performs best on nearly all instances. In general, systems with two parameters having up to 10 million states are handled within reasonable time. Consider Figure 1(b), where all intermediate states between $s_0$, $s_4$, and $s_5$ have been eliminated from the previous pMC. The functions $f_{s_0,s_4}$ and $f_{s_0,s_5}$ describe the probability of reaching $s_4$ and $s_5$ from $s_0$. For instance:

$$f_{s_0,s_4} = \frac{40p^2 + 20pq + 6p + 3q}{68p^2 + 34pq + 34q^2 + 34p + 17q}$$

With the exception of PROPhESY, the available tools just output the rational function, sometimes accompanied by constraints ensuring well defined probability distributions. The problem of parameter synthesis is therefore not addressed directly, posing the question of which parameter values lead to the satisfaction of certain properties of interest. We address this problem as follows: To give

---

*The results presented here have been published at CAV 2015 [3], where our tool also passed the CAV Artifact Evaluation.
the user a feasible and usable approach, an (approximate) partitioning of the parameter space into safe and unsafe regions is computed. Each parameter instantiation within a safe region satisfies the requirement, while inside unsafe regions, no instantiation meets the requirement.

This is approached in an incremental fashion: After the rational function is computed, the first step is to sample the rational function up to a user-adjustable degree. This yields a coarse abstraction of the true partitioning; a typical sampling result can be seen in Figure 1(c).

The goal then is to divide the parameter space into regions which are certified to be safe or unsafe. This is done in an iterative CEGAR-like fashion. First, a region candidate assumed to be safe or unsafe is automatically generated. An SMT solver like Z3 [6] or SMT-RAT [1] is then used to verify the assumption. In case it was wrong, a counterexample in the form of a contradicting sample point is provided with which the abstraction/sampling is refined, giving a finer abstraction of the solution space. Using this, new region candidates are generated. A very coarse partition into such regions is shown in Figure 1(d), a fine partition covering over 90% of the parameter space is shown in Figure 1(e). For the used benchmarks, a coverage of over 95% can be achieved within seconds.

Literatur
