Symbolic Approximation of the Bounded Reachability Probability in Large Markov Chains

Markus N. Rabe[1], Christoph M. Wintersteiger[2], Hillel Kugler[2], Boyan Yordanov[2], and Youssef Hamadi[2]

[1] Saarland University, Germany
[2] Microsoft Research

Abstract. We present a novel technique to analyse the bounded reachability probability problem for *large* Markov chains. The essential idea is to incrementally search for sets of paths that lead to the goal region and to choose the sets in a way to easily determine the probability mass they represent. To effectively dispatch the resulting formulas using an SMT solver, we employ a finite-precision abstraction on the Markov chain and a custom quantifier elimination strategy. Through experimental evaluation on PRISM benchmark models we demonstrate the feasibility of the approach on models that are out of reach for previous methods.

1 Introduction

Probabilistic model checking is used in a wide array of applications, e.g., in reliability analysis, analysis of randomized algorithms, but also for analysis of system models that arise from the natural sciences like in computational biology [1]. Especially in the sciences, there has always been a large interest in the analysis of probabilistic models, as testified by countless applications of Markov chains, Markov decision process, and their associated analysis procedures. Versatile logics, such as PCTL [2], offer a flexible framework for specifying questions in probabilistic systems. We consider one of the main building blocks for the analysis of PCTL specifications: the bounded reachability probability problem. It asks for the probability that a given event, characterized by a set of states (the *goal region*), occurs within a given number of steps of the model.

The general area of probabilistic model checking has received increased interest recently, which is to a large degree due to advances made both in theory and in practical analysis tools, e.g., in model checkers like PRISM [3] and MRMC [4]. Like all model checkers, these tools face the state-space explosion problem, though the challenge of dealing with probabilities makes the analysis of even moderate size systems very difficult. Various strategies have been developed to manage the size of the state-space, including techniques like abstraction refinement (e.g., [5]), Stochastic SAT (SSAT) and Stochastic SMT (SSMT) [6, 7], generalized Craig interpolation [8], symmetry reduction [9], as well as bisimulation minimization [10, 11]. Other techniques use SAT-based path search to enumerate paths (possibly with cycles) that lead to the goal region [12, 13] and

```
1  module p1
2    x : [0..99] init 0;
3    [] x<50 -> 0.5:(x'=x+1) + 0.5:(x'=2*x);
4    [] x=50 -> (x'=x);
5  endmodule
```

Fig. 1: A symbolic DTMC in simplified PRISM syntax with one integer variable x with domain $[0, 99]$ and initial value 0. The first action is enabled if $x < 50$ and it offers 2 probabilistic choices: With probability 0.5, x is incremented or it is doubled. The second action is enabled when $x = 50$ and idles.

add up their probabilities; this approach was recently enhanced to enumerating path fragments in a BDD-representation [14]. Even though these approaches scale to models that cannot be solved by explicit state methods (i.e. numerical approaches), the number of states in these models is still fairly small compared to other symbolic techniques in automated (program) verification.

In this paper we present a new approach to approximate the bounded reachability probability in large Markov chains using solvers for satisfiability modulo theories (SMT-solvers). The approach starts with a new problem representation: Instead of focusing on probability distributions over states, we consider the whole probability space of *sequences of random decisions* up to the given step bound. We then iteratively approximate the bounded reachability probability through the SMT-based search for *sets of paths* that lead to the goal region.

Example. Consider the example system of Fig. 1. What is the probability that x is smaller than 20 after 8 steps? For example, the set of paths executing (x'=2*x) in the first 3 steps and anything in the 5 steps thereafter ensures that x is smaller than 20 until step 8 and it has a probability of 2^{-3}. A second set of paths starting with one execution of (x'=x+1), then four unrestricted steps, followed by three executions of (x'=x+1) is disjoint to the first set and also ensures to stay below 20. Hence its probability of 2^{-4} can be counted separately from the first set.

Organization. In Section 2 we characterize the probabilistic transition relation of a Markov chain *given in a symbolic representation* via an integral over a propositional formula, which enables a conceptually simple characterization of the bounded reachability probability. Next, we present the iterative approach to the approximation of the bounded reachability probability by searching sets of paths that lead to the goal region. We choose the shape of sets in a way that allows us to easily determine their probability mass (Section 3). To effectively solve the resulting formulas, we discuss a finite-precision abstraction (Section 4) to obtain a purely discrete problem that we can effectively dispatch to SMT solvers. To enhance the efficiency, we present a specialized quantifier elimination strategy that makes use of the convexity of the sets we search for (Section 5). In Section 6 we report on an experimental evaluation of our approach on a set of common benchmark models and discuss the findings. Section 7 discusses related work.

2 Preliminaries

We assume familiarity with the basic concepts of probability spaces and distributions and we start directly with the definition of the system model:

Definition 1 (Markov chain). *A (discrete-time)* Markov chain (S, s_{init}, P) *consists of a finite set of states S, an initial state $s_{init} \in S$, and a probabilistic transition function $P : S \to Dist(S)$ assigning each state $s \in S$ a probability distribution over successor states.*

An *execution* of a Markov chain is an infinite sequence of states. Although it is intuitively clear what the behavior of Markov chains is, we need to construct the probability space over executions carefully. We employ the theorem of Ionescu-Tulcea [15, Thm. 2.7.2] to build this probability space out of the infinite sequence of random experiments (random variables). Recursively, we define a random variable X_i over the sequences of states of length i as

$$X_i(s_0 \dots s_i) = \sum_{s_{i-1} \in S} P(s_{i-1})(s_i) \cdot X_{i-1}(s_0 \dots s_{i-1}) \ ,$$

and we define X_0 to assign probability 1 to s_{init}. The fact that each X_i, for $i \geq 0$, is a random variable, is easily verified.

This construction defines a σ-algebra over cylindrical sets of executions, i.e. sets of executions that are defined via a common prefix, and it yields a *unique* Borel-measurable probability space over the infinite executions of the Markov chain. For a given Markov chain M we denote this measure on executions (and on their prefixes) as $\Pr_M$, mapping Borel-measurable sets of finite and infinite executions to probabilities.

2.1 Bounded Reachability Probability

The analysis problem we consider in this paper is to determine the probability to reach a specified set of final states in a given number of steps. This problem is motivated by encodings of practical problems into Markov chains, where steps correspond to steps in time in the original system. To ask for the probability to reach the final states in a given number of steps is, therefore, often equivalent to asking what the probability is that a certain event happens at a certain time. The time bounded reachability probability problem is also a basic building block for model checking logics like PCTL [2].

Formally, for a given Markov chain $M = (S, s_{init}, P)$, a set of final states $F \subseteq S$, and a step number k we define the problem as computing

$$\Pr{}_M(F, k) = \Pr{}_M\big(\{s_0 s_1 \dots s_k \dots \in S^\omega \mid \ s_0 = s_{init} \ \wedge \ s_k \in F\}\big) \ .$$

Note that this formulation of the problem asks for the probability of reaching F after *exactly* k steps. The computation of the probability of reaching F in k or fewer steps is a variation of the problem requiring $\exists i.\ s_i \in F$ instead of $s_k \in F$.

2.2 Symbolic Markov Chains

We aim to analyze systems with large state spaces. Hence we begin with a symbolic encoding of the state space and the probabilistic transition function. Figure 1 shows an example system. The state space is described by variables $v_1, v_2, \ldots, v_n$ with given finite domains. The transition function is given by a list of actions, each of them describing a probability distribution over successor states. Actions have the following form:

$$(\mathit{guard}) \to p_1 : (\mathit{update}_1) + \cdots + p_m : (\mathit{update}_m)\ ;$$

The $\mathit{guard} : S \to \mathbb{B}$ is a predicate on states that indicates whether the action is *enabled*. If the system has multiple actions, their guards need to partition the (reachable) state space. Thus, whenever a guard holds in a state, it is executed.

Intuitively, when executing an action, one of its *probabilistic choices*, which are separated by the symbol $+$, is selected at random. The probability distribution over the probabilistic choices is defined by the expressions $p_1, \ldots, p_m : S \to [0,1] \cap \mathbb{Q}$. Each action a and probabilistic choice p entails a unique successor state given by an update function, $\mathit{update}_{a,p} : S \to S$.

The description of Markov chains in terms of actions is inspired by the PRISM input language [3], as it proved to be flexible enough for a wide range of application areas, such as distributed algorithms, communication protocols, security, dependability, and biology. The PRISM input language supports more features, like the parallel composition of multiple modules, but for the sake of simplicity, we restrict the discussion to the features described above. Our implementation presented in Section 6 does support modules, however.

2.3 The Markov Chain Entailed by a Symbolic Markov Chain

The state space of the Markov chain entailed by a symbolic Markov chain is simply the cross product of the domains of the variables $v_1, v_2, \ldots, v_n$. The initial state s_{init} is fixed by an expression in the symbolic model. To construct the transition relation, we first consider the execution of a particular action a in a state s. The probabilistic choices of a with their probabilities $p_1(s), \ldots, p_m(s)$, respectively, define a partitioning of the interval $[0,1]$ into the sub-intervals

$$I_{a,p_i}(s) = \left[\sum_{j=1}^{i-1} p_j(s), \sum_{j=1}^{i} p_j(s)\right)$$

for $i < m$ and $I_{a,p_m} = \left[\sum_{j<m} p_j,\ 1\right]$. To execute a step in the model, we draw a value r from the interval $[0,1]$ uniformly at random and then proceed according to the deterministic transition relation. For a pair of states s and s' and a given random value $r \in [0,1]$, the transition relation is defined as

$$T(s,s',r) = \bigwedge_{1 \le j \le n} \mathit{guard}_{a_j}(s) \implies \bigwedge_{1 \le i \le m} \left(r \in I_{a_j,p_i}(s) \implies \mathit{update}_{a_j,p_i}(s,s')\right).$$

That is, we determine which action a is enabled and then apply the update function of the probabilistic choice belonging to the sub-interval the random number r falls in.

The probabilistic transition function of the entailed Markov chain, is thus

$$P(s, s') = \int_0^1 T(s, s', r)\ dr\ , \tag{1}$$

where $T(s, s', r)$ is interpreted as 1 iff it holds true.

Other works in the area (e.g. [7, Definition 5.2]) define the probabilistic transition relation as a sum of the probabilities of all probabilistic choices that result in the specified state. Our definition untangles the possible system behaviors and the measure. This allows us to formulate a conceptually simple approximation algorithm (Section 3). Of course, both approaches to define the entailed Markov chain result in the same system behaviour.

3 Incremental Symbolic Approximation

In this section, we present a method to incrementally approximate the bounded reachability probability for a given (symbolic) Markov chain. It is based on a characterization of the bounded reachability probability as an integral over a propositional formula, similar to the formulation for the one-step probabilistic transition given in Subsection 2.3.

We begin by characterizing executions of length k, that is legal combinations of execution prefixes $\bar{s} = s_0 s_1 \dots s_k$ of sequences of random decisions $\bar{r} = r_1 r_2 \dots r_k$; we define

$$T^k(\bar{s}, \bar{r}) = \Big(s_0 = s_{init}\ \wedge\ \textstyle\bigwedge_{0 \le i \le k-1} T(s_i, s_{i+1}, r_{i+1}) \Big)\ . \tag{2}$$

Note that for all sequences $\bar{r}$ there is exactly one sequence of states that fulfills this condition. We are interested in all sequences of random decisions that lead to a given goal region F, i.e.,

$$T^k(\bar{r}, F) = \exists \bar{s} \in S^{k+1}.\ T^k(\bar{s}, \bar{r})\ \wedge\ s_k \in F\ .$$

Proposition 1. *Let M be a DTMC entailed by a symbolic Markov chain. For the bounded reachability probability for a given goal region F and step number k it holds that*

$$\Pr{}_M(F, k) = \int_0^1 \dots \int_0^1 T^k(r_0 \dots r_k, F)\ dr_0 \dots\ dr_k\ .$$

3.1 Identifying Cubes in the Probability Space

Proposition 1 leads to a new view on the problem. Instead of considering how the probability distributions over the state space evolve over time, we consider

the probability space over all sequences of random decisions. This '*state-less*' representation of the problem helps to attack the problem for models beyond the scale at which their state space can be represented explicitely or via (MT)BDDs.

We propose to exploit the additivity of the probability measure at the level of traces, i.e., to search for subsets $R_1, \ldots, R_n \subseteq [0,1]^k$ with $\forall i \forall \bar{r} \in R_i.\ T^k(\bar{r}, F)$ and then to count them separately:

$$\Pr{}_M(F,k) = \sum_i \left(\int_{R_i} 1 \; d\bar{r} \right) \quad + \quad \int_0^1 \ldots \int_0^1 T^k(\bar{r}, F) \ \wedge \ \bigwedge_i \ \bar{r} \notin R_i \ \ d\bar{r} \; .$$

It is important to pick sets R_i that are easy to integrate. By choosing them to be disjoint (closed and/or open) rectangles in $[0,1]^k$, we are able to obtain an arbitrarily close approximation and it is easy to determine the volume of each R_i. To see this, note that the space $T^k(\bar{r}, F)$ is the finite disjoint union of the sets $R(\bar{s}) = \{\bar{r} \in [0,1]^k \mid T^k(\bar{s}, \bar{r})\}$ with $s_k \in F$. The sets $R(\bar{s})$ are in general closed and open rectangles, as in each dimension they are defined by an upper bound and a lower bound given by the expressions $p_i(s)$ in the system description.

In practice it is of course desirable to find larger rectangles. Our proposal is essentially a greedy algorithm that searches for the next largest rectangle in a system: Check, for increasing rectangle sizes x, whether a rectangle of that size still exists; which translates to

$$\begin{aligned} \exists \bar{l}, \bar{u} \in [0,1]^k.\ \ & x \leq \textstyle\prod_{1 \leq i \leq k} u_i - l_i \ \wedge \\ & \forall \bar{r} \in [0,1]^k. \left(\textstyle\bigwedge_{1 \leq i \leq k} l_i \leq r_i \leq u_i \right) \implies T^k(\bar{r}, F) \ \wedge \ \textstyle\bigwedge_i \ \bar{r} \notin R_i \end{aligned} \tag{3}$$

Whenever we find a rectangle that satisfies the conditions above, we add it to the set of rectangles R_i and repeat the process. If no rectangle exists, we reduce the size of the rectangle to search for. It is clear that we can stop the process at any time and obtain an under-approximation, i.e., $\sum_i \left(\int_{R_i} 1 \; d\bar{r} \right) \leq \Pr_M(F,k)$.

Note that this method has an advantage over enumerating paths through the system, if there are multiple probabilistic choices that do not change the fact that the executions reach the goal region with the same probabilistic choices in other steps—the sequence of states visited may be different though.

4 A Finite-Precision Abstraction

Our problem formulation of Section 3 is not very amenable to efficient solving with automatic methods; to achieve this goal, we employ a layer of automatic abstraction refinement, where each abstraction is obtained by bounding the precision of the analysis. In practice, we encode each of the sub-problems in the SMT theory of uninterpreted functions and bit-vectors (SMT UFBV) as this theory offers an efficient quantifier elimination/instantiation strategy [16].

To encode the problem in this purely discrete theory, we discretize the random variables according to a precision parameter h. We propose a *symbolic* discretization technique on the level of the formula $T^k(\bar{s}, \bar{r})$ that maintains the

conciseness of the representation. That is, we do not need to consider every state or transition of the entailed DTMC, but the technique works directly on the symbolic description. This discretization preserves the probability measure up to an arbitrarily small error.

First, we replace the real valued variables $r \in [0,1]$ by discrete variables $r \in \{0, \ldots, 2^h - 1\}$, where each of the discretization levels now corresponds to a small portion ($\frac{1}{2^h}$) of the probability mass. Second, for every action of the symbolic Markov chain with m probabilistic choices, we discretize the intervals $I_{a,i}$ introduced in Section 2.3 according to a precision parameter h:

$$\left\lceil I_{a_j,p_i} \right\rceil_h(s) = \left[\left\lfloor 2^h \cdot \sum_{j=1}^{i-1} p_j(s) \right\rfloor, \left\lceil 2^h \cdot \sum_{j=1}^{i} p_j(s) \right\rceil \right) ,$$

for $i < m$ and $\left\lceil I_{a_j,p_m} \right\rceil_h = \left[\left\lfloor 2^h \cdot \sum_{j=1}^{m} p_j(s) \right\rfloor, 2^h - 1 \right]$.

This simplifies the encoding of the one-step transition relation $T(s, s', r)$ to

$$\bigwedge_{1 \le j \le n} \mathit{guard}_{a_j}(s) \implies \bigwedge_{1 \le i \le m} \left(r \in \left\lceil I_{a_j,p_i} \right\rceil_h(s) \implies \mathit{update}_{a_j,p_i}(s, s') \right) ,$$

which is a formula over a purely discrete space. The probability of a particular probabilistic choice now approximately corresponds to the number of values for r for which the transition relation holds true and shows this choice.

Due to the overlapping intervals $\left\lceil I_{a_j,p_i} \right\rceil_h$, some of the discretization levels are assigned to multiple intervals, but otherwise this transformation maintains a clear correspondence of the values of r. Thus, the approximated transition relation now represents an *over-approximation* of the original transition relation, or, in other words, the formula $T_h^k(\bar{s}, \bar{r}) = \left(s_0 = s_{init} \wedge \bigwedge_{0 \le i \le k-1} T(s_i, s_{i+1}, r_{i+1}) \right)$ is a relaxation of $T^k(\bar{s}, \bar{r})$ and we define $T_h^k(\bar{r}, F)$ to be $\{\bar{r} \in [0,1]^k \mid \exists \bar{s} \in S^{k+1}.\ T_h^k(\bar{s}, \lfloor \bar{r} \cdot 2^h \rfloor) \ \wedge \ s_k \in F\} \supseteq T^k(\bar{r}, F)$.

Replacing $T^k(\bar{r}, F)$ by $T_h^k(\bar{r}, F)$ in Equation 3 does not result in the desired approximation, as for the incremental symbolic search for *under-approximations of the probability*, we are interested in an under-approximation of the transition relation. We use the duality of the search for F and its complement $\bar{F}$ (that is $\Pr_M(F, k) = 1 - \Pr_M(\bar{F}, k)$) to derive that, we replace $T^k(\bar{r}, F)$ by $\neg T_h^k(\bar{r}, \bar{F})$, which yields an under-approximation, as desired. The reason for not starting with an under-approximation right away is to avoid the additional quantifier alternation that lures in the set $T^k(\bar{r}, F)$. In this way, the discretized version of Equation 3 has only one quantifier alternation from an existential quantifier to a universal quantifier.

4.1 Precision

The total probability mass affected by this approximation within one step of the transition relation, is the probability of the union of all ambiguous discretization

levels. For a given action with m probabilistic choices, there can be at most $m-1$ ambiguous discretization levels, hence the quality of the approximation for executing action a is $\frac{m-1}{2^h}$. The affected probability mass in one step of the system is easily obtained by considering the action with the maximal number of probabilistic choices.

When considering k steps, an obvious upper bound of the probability mass affected by the approximation is $\sum_{0<i\leq k} \frac{m-1}{2^h} \cdot (1-\frac{m-1}{2^h})^{i-1}$, which is smaller than $k \cdot \frac{m-1}{2^h}$. As we are free to choose the parameter h, it is feasible to keep the amount of affected probability mass arbitrarily small, because we have $\lim_{h\to\infty} \Pr_{M_h}(F,k) = \Pr_M(F,k)$.

Proposition 2. *For a given symbolic Markov chain M, its discretized variant M_h, a step number k, and a goal region F, we have*

$$\Pr_{M_h}(F,k) \leq \Pr_M(F,k) \text{ and}$$
$$\Pr_{M_h}(F,k) + k \cdot (\tfrac{m-1}{2^h}) \geq \Pr_{M_h}(F \cup \{s^*\}, k) \; .$$

As a consequence, the finite-precision abstraction of an under-approximation of the bounded reachability probability as discussed in Section 3 is still an under-approximation of the bounded reachability probability.

5 Implementation and Optimizations

We implemented our technique in a prototype model checker named *pZ3* to evaluate its practical efficacy. We use the Z3 theorem prover (specifically its theory for SMT UFBV [16]) as a back-end to dispatch the generated SMT instances. The input to the tool is a PRISM model file, a predicate F on the state space that represents the goal region, a step number $k \in \mathbb{N}$, and a target probability $p^* \in [0,1]$. The tool then determines whether the probability to reach a state satisfying F is larger than or equal to p^*. Answering similar questions, like whether the probability is below a certain bound, or within some upper and lower bounds is straightforward, but not currently implemented.

5.1 The Basic Encoding

By the *basic encoding*, we refer to the direct encoding of Eq. 3 in the theory of bit-vectors with quantifiers, using the finite-precision abstraction presented in Section 4. Thereby we completely rely on the SMT solver's ability to handle the quantifiers. We chose the theory of bit-vectors over a pure SAT encoding to make use of the word-level reasoning and optimizations of the SMT solver.

We omit the details of the translation of Eq. 3 into bit-vectors as this is as usual. However, it is interesting to note that the size of the generated SMT instance is (1) *logarithmic* in the domains of the variables, and (2) *linear* in the number of variables, the number of actions, the precision parameter h, number of updates, and the number of steps.

Note that the PRISM language supports modules that can perform actions jointly via a synchronization mechanism. We encode such synchronized actions of multiple modules compactly to avoid enumerating the exponential number of synchronizations actions.

This basic encoding, while correct, challenges the current state-of-the-art in SMT-solving as it produces large and complex quantified formulas that can not be quickly dispatched. In the following, we discuss optimizations, first and foremost a custom quantifier elimination strategy, that does enable checking Eq. 3 effectively for large Markov chains.

5.2 Custom Quantifier Elimination

To improve the performance of the SMT solver, we implemented a customized quantifier elimination procedure that relies on the notions of *example paths* and *close counter-examples*. The idea is to not search for a sufficiently large rectangle directly, but instead we pick a local environment by fixing an *example path* that leads to the goal region and only search for rectangles that contain this path, such that we find the largest rectangle containing at least this path. Abstractly, we pick candidate rectangles and check whether they are valid rectangles by searching for a path inside the rectangle that does *not* lead to the goal region, i.e., a counter-example. If we find such a counter-example, we remember it and generate a new candidate rectangle. This procedure is similar to what a strategy like model-based quantifier instantiation [17, 16] does with a problem like Eq. 3.

However, we may have to perform many queries to find a rectangle that does not contain a counter-example (i.e., a rectangle only containing paths that lead to the goal region). So, when searching for counter-examples, it is highly beneficial to rule out as many candidate rectangles as possible. Here, we exploit the fact that the rectangles we are looking for are convex. Therefore, counter-examples that are 'closer' to the example path rule out more candidate rectangles than those that are strictly further away. As a measure of distance between paths, we use the Hamming distance of the bit-strings that represent the random choices, as follows: Instead of encoding the random choices of each step by a bit-vector of length h, we consider these as h independent Boolean variables, such that we are able to compute the Hamming distance between two different instantiations of those variables. This entails a change of view from a k-dimensional space where each dimension has 2^h values, to a $k \cdot h$-dimensional space with Boolean values and it slightly changes the notion of rectangles: A rectangle in the bit-vector representation is not necessarily a rectangle in the Boolean representation, and vice versa. (Rectangles in the Boolean space are also called *cubes*.) Currently, we only support this restricted notion of shapes, but in general any type of convex polygonal shape can be used.

Using these definitions, we search for those counter-examples that are *closest* to the example path. We call these counter-examples *close counter-examples*. Finally, we provide a sketch of the process in Algorithm 1, which uses the following functions:

```
Data: Initial states I, k-step transition relation T^k, goal states F, target
      probability p*
Result: true if the probability to reach F from I via T^k is at least p*, false
        otherwise.
rectangles := ∅ ;
p := 0 ;
while p < p* do
    path := findPath (I, T^k, F, ¬rectangles) ;
    if path ≠ ∅ then
        closeCEs := ∅ ;
        rectangleFound := false;
        while not rectangleFound do
            rectangle := findCandidateRectangle(path, closeCEs);
            closeCE := findClosestCE(rectangle, path, I, T^k, ¬F);
            if closeCE ≠ ∅ then
                closeCEs := closeCEs ∪ { closeCE };
            else
                p := p + computeVolume(rectangle, rectangles);
                rectangles := rectangles ∪ { rectangle } ;
                rectangleFound := true;
            end
        end
    else
        return false;
    end
end
return true;
```

Algorithm 1: Quantifier elimination based on close counter-examples

findPath: Yields a path that starts with the initial state, follows the k-step transition relation and ends up in a state satisfying F. Paths that were already counted in previous runs of the outer loop are excluded by $\neg$*rectangles*. This is essentially an SMT-based bounded model checking query. We utilize both under- and over-approximations as described in Section 4. Searching paths in the over-approximation results in example paths that share their sequence of random decisions with a second path that does not lead to the goal region. As an optimization, we also search for example paths in the under-approximation of the transition relation that does not allow for overlapping intervals.

findCandidateRectangle: Finds a rectangle that contains the example path and avoids all counter-examples. Note that this check is completely agnostic to the transition relation.

findClosestCE: This function iteratively searches for counter-examples of increasing Hamming distance, starting with distance 0. Finding close counter-examples seems to be a hard task for SMT solvers—in many models this is

harder than finding a path that is not related by distance to the original path. Typically our tool spends >50% of its run time in this routine.

Note that, while we require example paths to not be covered by any of the identified rectangles, we do not require the rectangles to be intersection-free. This is an optimization that tries to avoid the fragmentation of the remaining parts of the probability space. The function *computeVolume(...)* finally computes the volume that the new rectangle adds to the union of all rectangles.

6 Experimental evaluation

We conducted a set of experiments to evaluate our technique and to determine its effectiveness in verifying the bounded reachability probability problem and its performance in relation to existing approaches.

Models. To obtain a benchmark set that is not biased toward our tool, we chose to consider all Markov chain models in the benchmark set that is delivered with the PRISM model checker. Out of those, we picked all models that come with a bounded reachability specification. This set comprises the *Bounded retransmission protocol* (BRP) [18], the Crowds protocol (CROWDS) [19], a contract signing protocol (EGL) [20], the self-stabilization protocol (HERMAN) [21], a model of von Neumann's NAND multiplexing [22], and a synchronous leader election protocol (LEADER) [23].[3] For each of these models, we considered multiple parameter settings, that, for example, control the number of participants in the protocol or the minimal probability that must be proven. Most of the instances considered satisfy the specified probability bounds. The full list of experiments can be found in Appendix A.

Experimental Setup. All experiments were performed on machines with two Intel Xeon L5420 quad core processors (2.5GHz, 16GB RAM). All tools were limited to 2GB of memory and the time limit was set to 2 hours (7200s).

Comparison to the PRISM model checker. For the comparison, we configured PRISM to use its symbolic MTBDD engine and we extended the available memory of PRISM's BDD library to 2GB. Figure 2 summarizes the comparison of pZ3 to PRISM. From this plot it is evident that pZ3 solves many of the large problem instances, for which PRISM runs out of time or memory. On Markov chains that are small or of moderate size, however, PRISM has a clear advantage. Especially for the models of the leader election protocol, the bounded retransmission protocol and the self-stabilization protocol, we observe that scaling the model parameters has little effect on the run time of pZ3, whereas PRISM exceeds the time or memory limits. For the models LEADER and HERMAN, all reachable states can be reached within the considered step bound ($> 10^{12}$ states

[3] These models and a detailed description for each of them can be found at http://www.prismmodelchecker.org/casestudies/

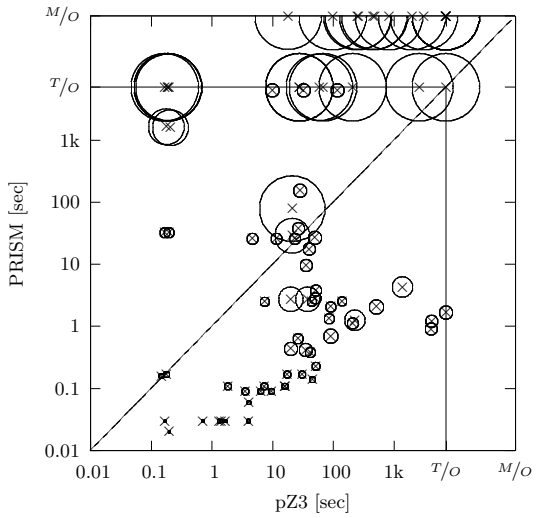
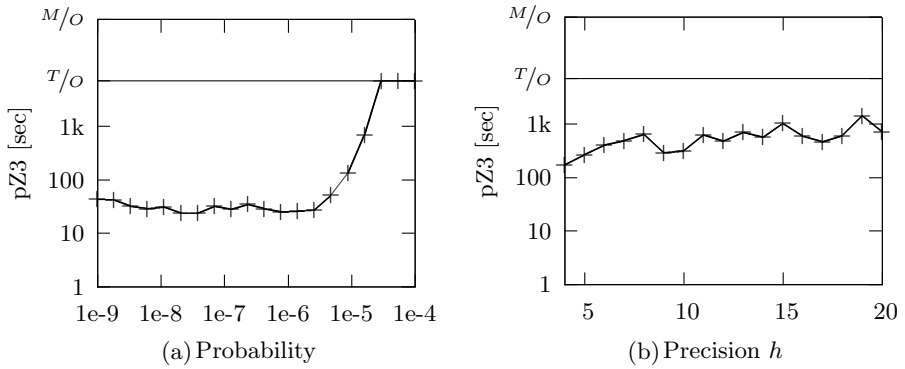


Fig. 2: The performance of PRISM vs the performance of pZ3 on all models and parameter settings. The size of the circles around datapoints indicate the logarithm of the size of the state space. For the full list of experiments, see Appendix A.

in case of the HERMAN model), suggesting that the advantage of pZ3 is due to its use of a symbolic reasoning engine rather than a variant of state space enumeration.

Scalability in the target probability. Figure 3a displays the scalability in the target probability. As expected for an incremental method, the run time increases when we specify a target probability close to the actual bound. The approximation quality that can be achieved varies greatly for the different models: While for the model of the leader election protocol 99% of the probability mass is found in acceptable time, in other models only a fraction of the actual probability mass is found by pZ3.

Scalability in the precision parameter. In theory, the precision parameter h plays an important role in the quality of the results as presented in Section 4. The probabilities computed by our approach are always sound under-approximations of the bounded reachability probability; regardless of the precision parameter. For the models in the PRISM benchmark suite, small values of the precision parameters, between $h = 1$ and $h = 8$, are often enough to verify many models. Nevertheless, for some models the precision might be an issue. Figure 3b shows the sensitivity or our approach with respect to h on the leader election protocol with 4 participants and 11 probabilistic alternatives each, where it has only a moderate effect on the run time. The choice of 11 probabilistic alternatives



(a) Probability (b) Precision h

Fig. 3: Run time of pZ3 for increasing (a) target probability on the BRP model with parameters $N = 16$ and $MAX = 2$ and (b) precision on the leader election model with parameters $N = 4$ and $K = 11$.

ensures that increasing the precision parameter actually increases the maximal probability that can be detected by our approach.

7 Related work

The first formulation of the bounded reachability problem for MDPs goes back to Bellman [24] and Shapley [25]. These are the foundation of the numerical methods included in MRMC [4], IscasMC[26], and Murphi [27] and enable precise model checking, but consider states and transition probabilities explicitly and so do not scale to problem sizes where the state space is not efficiently enumerable. Simulation based techniques and statistical model checking are well suited to explore the likelihood of relatively likely events in large executable systems. However, when the events are very unlikely, simulation based techniques struggle to produce results with small margins of error.

The first symbolic approach to analysis of MDPs is based on MTBDDs [28]. Experiments with the (MTBDD-based) tool PRISM show that the approach is limited to fairly small state spaces, compared to other symbolic techniques in general automated (program) verification.

Abstract interpretation [29] and general static analysis are widely employed techniques for approximate analysis of systems, but existing frameworks based thereupon are often limited to software-specific behavior (like numerical analysis) and their precisions strongly depends on the choice of abstract domains. Esparza and Gaiser [30], basing their work upon that of Hermanns et al. [5] and Kattenbelt et al. [31], as well as Monniaux [32] give a first taste of how abstract interpretation can be employed effectively in the probabilistic setting.

Fränzle et al. [33] proposed to encode the bounded reachability problem of Markov chains and Markov decision processes into Stochastic SAT (SSAT) Stochastic SMT (SSMT), hoping to replicate the tremendous progress SAT and SMT solvers brought to other fields. The proposed algorithms for SSAT and

SSMT branch over the probabilistic decisions and recursively add up the reachability probability of the individual branches. This requires exploration of an exponential number of probabilistic branches and hence the number of steps we can explore with this approach is limited, even for small models (cf. [7, Section 6.7]). SSAT and SSMT-based analysis of Markov chains is similar to our approach, but our method does not try do develop a specialized algorithm to solve SSAT instances; it instead builds upon general purpose SMT solvers. We make use of quantified theories to search for *sets of paths* that lead to the goal region.

Counterexample generation is a closely related branch of research [34, 35]. It is concerned with generating proof objects or counter-examples to the bounded reachability problem (or alternatively reward problems) that are—in principle—human readable. Also symbolic approaches have been explored for generating counter-examples for Markov chains [12–14]. Similar to the approach discussed here, these works present methods to iteratively find evidence for the probability of a given event in probabilistic systems. These methods mostly enumerate single paths (possibly with cycles [12] or fragments of paths [14]) such that they require a large number of calls to a SAT or SMT solver. In contrast, our method detects large sets of paths with few calls to a solver and builds on a fundamentally different representation: Instead of considering paths as sequences of states of the Markov chain, we consider paths to be sequences of random decisions.

Since the presented technique is not the first symbolic approach able to solve the bounded reachability probability problem in Markov chains, a comparison of all approaches would be in order. However, to the best of our knowledge there is no tool besides PRISM that (1) symbolically analyzes Markov chains (2) is publicly available and (3) is able to parse PRISM files. A first impression of the relative performance of recent counter-example generation techniques to the technique presented here can be obtained through the data presented in this work and in [14] (see also Appendix A). Both works consider the same models, though the model parameters in this work are often considerably higher. For example the leader election protocol seems to be not amendable to enumerating paths, whereas the search for sets of paths performs well. For the crowds protocol, however, enumerating paths (potentially with cycles and path fragments) by many SAT or SMT calls seems considerably faster than searching for sets of paths. This raises the question on combinations or generalizations of the methods to combine the best of both worlds.

8 Conclusion

We present a novel approach to iteratively approximate the bounded reachability probability in large Markov chains, which is based on a novel problem encoding in quantified SMT theories. We employ a finite-precision abstraction to obtain a discrete problem encoding. A specialized quantifier elimination strategy is given to effectively dispatch the encoded formulas. We demonstrate the feasibility of the approach on the set of benchmark models of the PRISM model checker. Especially for large models our tool is able to outperform PRISM, suggesting

that our method is a suitable, complementary approach in cases where existing methods do not scale.

Acknowledgments. This work was partially supported by the German Research Foundation (DFG) under the project SpAGAT within the Priority Program 1496 "Reliably Secure Software Systems"—RS^3.

References

1. Kwiatkowska, M.Z., Norman, G., Parker, D.: Using probabilistic model checking in systems biology. SIGMETRICS Performance Evaluation Review **35**(4) (2008)
2. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Aspects of Computing **6**(5) (1994) 512–535
3. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: Proc. of CAV. Volume 6806 of LNCS., Springer (2011)
4. Katoen, J.P., Zapreev, I.S., Hahn, E.M., Hermanns, H., Jansen, D.N.: The ins and outs of the probabilistic model checker MRMC. In: Proc. of QEST, IEEE Computer Society (2009) 167–176 www.mrmc-tool.org.
5. Hermanns, H., Wachter, B., Zhang, L.: Probabilistic cegar. In: Proc. of CAV. Volume 5123 of LNCS., Springer (2008)
6. Fränzle, M., Teige, T., Eggers, A.: Engineering constraint solvers for automatic analysis of probabilistic hybrid automata. Journal of Logic and Algebraic Programming **79**(7) (2010) 436–466
7. Teige, T.: Stochastic Satisfiability Modulo Theories: A Symbolic Technique for the Analysis of Probabilistic Hybrid Systems. Doctoral dissertation, Carl von Ossietzky Universität Oldenburg, Department of Computing Science, Germany (2012)
8. Teige, T., Fränzle, M.: Generalized Craig interpolation for stochastic Boolean satisfiability problems. In: Proc. of TACAS. Volume 6605 of LNCS., Springer (2011)
9. Kwiatkowska, M.Z., Norman, G., Parker, D.: Symmetry reduction for probabilistic model checking. In: Proc. of CAV. Volume 4144 of LNCS., Springer (2006)
10. Katoen, J.P., Kemna, T., Zapreev, I.S., Jansen, D.N.: Bisimulation minimisation mostly speeds up probabilistic model checking. In: Proc. of TACAS. Volume 4424 of LNCS., Springer (2007)
11. Dehnert, C., Katoen, J.P., Parker, D.: SMT-based bisimulation minimisation of Markov models. In: Proc. of VMCAI. Volume 7737 of LNCS., Springer (2013)
12. Wimmer, R., Braitling, B., Becker, B.: Counterexample generation for discrete-time Markov chains using bounded model checking. In Jones, N., Mller-Olm, M., eds.: Proc. of VMCAI. Volume 5403 of LNCS., Springer Berlin Heidelberg (2009) 366–380
13. Braitling, B., Wimmer, R., Becker, B., Jansen, N., Ábrahám, E.: Counterexample generation for Markov chains using SMT-based bounded model checking. In: FMOODS/FORTE, Springer (2011) 75–89
14. Jansen, N., Wimmer, R., Abraham, E., Zajzon, B., Katoen, J.P., Becker, B., Schuster, J.: Symbolic counterexample generation for large discrete-time markov chains. In: Proc. of FACS. Volume 7684 of LNCS., Springer (2014)
15. Ash, R., Doléans-Dade, C.: Probability and Measure Theory. Harcourt/Academic Press (2000)

16. Wintersteiger, C.M., Hamadi, Y., de Moura, L.M.: Efficiently solving quantified bit-vector formulas. Proc. of FMSD **42**(1) (2013)
17. Ge, Y., de Moura, L.: Complete instantiation for quantified formulas in satisfiabiliby modulo theories. In: Proc. of CAV. Volume 5643 of LNCS., Springer (2009)
18. Helmink, L., Sellink, M., Vaandrager, F.: Proof-checking a data link protocol. In Barendregt, H., Nipkow, T., eds.: Proc. of TYPES. Volume 806 of LNCS., Springer (1994) 127–165
19. Reiter, M., Rubin, A.: Crowds: Anonymity for web transactions. ACM Transactions on Information and System Security (TISSEC **1**(1) (1998) 66–92
20. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. Communications of the ACM **28**(6) (1985) 637–647
21. Herman, T.: Probabilistic self-stabilization. Information Processing Letters **35**(2) (1990) 63–67
22. von Neumann, J.: Probabilistic logics and synthesis of reliable organisms from unreliable components. In Shannon, C., McCarthy, J., eds.: Proc. of Automata Studies, Princeton University Press (1956) 43–98
23. Itai, A., Rodeh, M.: Symmetry breaking in distributed networks. Information and Computation **88**(1) (1990)
24. Bellman, R.: Dynamic programming. Princeton University Press (1957)
25. Shapley, L.S.: Stochastic games. Proceedings of the National Academy of Sciences **39**(10) (1953)
26. Hahn, E.M., Li, Y., Schewe, S., Turrini, A., Zhang, L.: IscasMC: A web-based probabilistic model checker. In: Proc. of FM. Volume 8442 of Lecture Notes in Computer Science., Springer (2014) 312–317
27. Della Penna, G., Intrigila, B., Melatti, I., Tronci, E., Venturini Zilli, M.: Bounded probabilistic model checking with the Murφ verifier. In Hu, A., Martin, A., eds.: Proc. of FMCAD. Volume 3312 of Lecture Notes in Computer Science., Springer (2004) 214–229
28. de Alfaro, L., Kwiatkowska, M., Norman, G., Parker, D., Segala, R.: Symbolic model checking of probabilistic processes using MTBDDs and the Kronecker representation. In: Proc. of TACAS. Volume 1785 of LNCS., Springer (2000)
29. Cousot, P., Cousot, R.: Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Proc. of POPL, ACM (1977)
30. Esparza, J., Gaiser, A.: Probabilistic abstractions with arbitrary domains. In: Proc. of SAS. Volume 6887 of LNCS., Springer (2011)
31. Kattenbelt, M., Kwiatkowska, M., Norman, G., Parker, D.: Abstraction refinement for probabilistic software. In: Proc. of VMCAI. Volume 5403 of LNCS., Springer (2009)
32. Monniaux, D.: Abstract interpretation of probabilistic semantics. In: Proc. of SAS. Volume 1824 of LNCS., Springer (2000)
33. Fränzle, M., Hermanns, H., Teige, T.: Stochastic satisfiability modulo theory: A novel technique for the analysis of probabilistic hybrid systems. In: Proc. of HSCC. Volume 4981 of LNCS., Springer (2008)
34. Andrs, M.E., Rossum, P.V.: Significant diagnostic counterexamples in probabilistic model checking. In: Proc. of HVC, Springer (2009) 129–148
35. Han, T., Katoen, J.P., Berteun, D.: Counterexample generation in probabilistic model checking. IEEE Transactions on Software Engineering **35**(2) (2009) 241–257

A Experimental data

Model	**Parameters**	**Prec.**	**Prob.**	**Steps**	$\log_{10}$(States)	**time pZ3**	**time PRISM**
BRP	N= 16 MAX= 2	$h = 8$	10^{-9}	14	2.83	6.49	0.09
BRP	N= 32 MAX= 4	$h = 8$	10^{-9}	14	3.34	31.25	0.17
BRP	N= 64 MAX= 4	$h = 8$	10^{-9}	14	3.72	52.79	0.23
BRP	N= 200 MAX= 4	$h = 8$	10^{-9}	14	4.13	26.41	0.64
BRP	N= 400 MAX= 4	$h = 8$	10^{-9}	14	4.43	86.54	1.33
BRP	N= 600 MAX= 4	$h = 8$	10^{-9}	14	4.61	92.40	2.09
BRP	N= 800 MAX= 4	$h = 8$	10^{-9}	14	4.74	51.23	2.89
BRP	N= 1000 MAX= 4	$h = 8$	10^{-9}	14	4.83	52.56	3.79
BRP	N= 2000 MAX= 4	$h = 8$	10^{-9}	14	5.13	36.00	9.72
BRP	N= 3000 MAX= 4	$h = 8$	10^{-9}	14	5.31	40.56	17.71
BRP	N= 4000 MAX= 4	$h = 8$	10^{-9}	14	5.43	50.32	26.85
BRP	N= 5000 MAX= 4	$h = 8$	10^{-9}	14	5.53	27.32	38.13
BRP	N= 10000 MAX= 4	$h = 8$	10^{-9}	14	5.83	28.94	153.73
BRP	N= 10^5 MAX= 4	$h = 8$	10^{-9}	14	n/a	27.90	t/o
BRP	N= 10^6 MAX= 4	$h = 8$	10^{-9}	14	n/a	28.05	t/o
BRP	N= 10^7 MAX= 4	$h = 8$	10^{-9}	14	n/a	68.35	t/o
BRP	N= 10^8 MAX= 4	$h = 8$	10^{-9}	14	n/a	60.56	t/o
NAND	N= 20 K= 2	$h = 8$	$5 \cdot 10^{-6}$	80	5.19	4108.10	0.91
NAND	N= 20 K= 3	$h = 8$	$5 \cdot 10^{-6}$	80	5.36	4234.30	1.23
NAND	N= 20 K= 4	$h = 8$	$5 \cdot 10^{-6}$	100	5.49	t/o	1.67
EGL	N= 5 L= 2	$h = 1$	0.5	31	4.53	42.38	0.38
EGL	N= 5 L= 4	$h = 1$	0.5	60	4.87	212.20	1.12
EGL	N= 7 L= 2	$h = 1$	0.5	100	5.87	92.28	0.70
EGL	N= 7 L= 4	$h = 1$	0.5	100	6.22	520.50	2.09
EGL	N= 10 L= 2	$h = 1$	0.5	100	7.82	227.78	1.26
EGL	N= 10 L= 4	$h = 1$	0.5	100	8.18	1396.62	4.38
CROWDS	Runs= 3 Size= 5	$h = 7$	0.005	15	3.08	17.68	0.17
CROWDS	Runs= 3 Size= 5	$h = 7$	0.007	15	3.08	45.99	0.14
CROWDS	Runs= 6 Size= 10	$h = 7$	0.005	15	5.55	20.14	0.44
CROWDS	Runs= 6 Size= 10	$h = 7$	0.007	15	5.55	35.51	0.42
CROWDS	Runs= 10 Size= 20	$h = 7$	0.005	15	9.35	20.34	2.76
CROWDS	Runs= 10 Size= 20	$h = 7$	0.007	15	9.35	38.76	2.76
CROWDS	Runs= 10 Size= 40	$h = 7$	0.005	15	11.90	21.29	29.00
CROWDS	Runs= 20 Size= 40	$h = 7$	0.005	15	17.80	21.37	79.98
CROWDS	Runs= 40 Size= 128	$h = 7$	0.005	15	n/a	264.53	m/o
CROWDS	Runs= 60 Size= 128	$h = 7$	0.005	15	n/a	251.15	m/o
LEADER	N= 3 K= 2	$h = 1$	0.1	4	1.41	0.72	0.03
LEADER	N= 3 K= 2	$h = 1$	0.5	4	1.41	1.45	0.03
LEADER	N= 3 K= 2	$h = 1$	0.9	4	1.41	1.70	0.03
LEADER	N= 3 K= 8	$h = 3$	0.1	4	3.02	1.86	0.11
LEADER	N= 3 K= 8	$h = 3$	0.5	4	3.02	3.57	0.09
LEADER	N= 3 K= 8	$h = 3$	0.9	4	3.02	7.38	0.11
LEADER	N= 3 K= 32	$h = 5$	0.1	4	4.82	4.76	26.13
LEADER	N= 3 K= 32	$h = 5$	0.5	4	4.82	11.70	25.91
LEADER	N= 3 K= 32	$h = 5$	0.9	4	4.82	23.86	25.99
LEADER	N= 3 K= 64	$h = 6$	0.1	4	5.72	10.11	6382.27
LEADER	N= 3 K= 64	$h = 6$	0.5	4	5.72	33.12	6375.88
LEADER	N= 3 K= 64	$h = 6$	0.9	4	5.72	118.54	6359.91

Model	**Parameters**	**Prec.**	**Prob.**	**Steps**	$\log_{10}$(States)	**time pZ3**	**time PRISM**
LEADER	N= 4 K= 2	$h = 1$	0.1	5	1.79	1.34	0.03
LEADER	N= 4 K= 2	$h = 1$	0.5	5	1.79	4.04	0.03
LEADER	N= 4 K= 2	$h = 1$	0.9	5	1.79	4.11	0.06
LEADER	N= 4 K= 8	$h = 3$	0.1	5	4.09	7.51	2.51
LEADER	N= 4 K= 8	$h = 3$	0.5	5	4.09	44.74	2.50
LEADER	N= 4 K= 8	$h = 3$	0.9	5	4.09	142.26	2.54
LEADER	N= 4 K= 32	$h = 5$	0.1	5	n/a	18.48	m/o
LEADER	N= 4 K= 32	$h = 5$	0.5	5	n/a	490.30	m/o
LEADER	N= 4 K= 32	$h = 5$	0.9	5	n/a	838.84	m/o
LEADER	N= 4 K= 64	$h = 6$	0.1	5	n/a	99.42	m/o
LEADER	N= 4 K= 64	$h = 6$	0.5	5	n/a	459.57	m/o
LEADER	N= 4 K= 64	$h = 6$	0.9	5	n/a	3130.91	m/o
LEADER	N= 6 K= 2	$h = 1$	0.1	7	2.53	9.66	0.09
LEADER	N= 6 K= 2	$h = 1$	0.5	7	2.53	15.88	0.11
LEADER	N= 6 K= 2	$h = 1$	0.9	7	2.53	16.28	0.11
LEADER	N= 6 K= 8	$h = 3$	0.1	7	n/a	210.56	t/o
LEADER	N= 6 K= 8	$h = 3$	0.5	7	n/a	2625.98	t/o
LEADER	N= 6 K= 8	$h = 3$	0.9	7	n/a	t/o	t/o
LEADER	N= 6 K= 32	$h = 5$	0.1	7	n/a	2000.57	m/o
LEADER	N= 6 K= 32	$h = 5$	0.5	7	n/a	t/o	m/o
LEADER	N= 6 K= 32	$h = 5$	0.9	7	n/a	t/o	m/o
LEADER	N= 6 K= 64	$h = 6$	0.1	7	n/a	t/o	m/o
LEADER	N= 6 K= 64	$h = 6$	0.5	7	n/a	t/o	m/o
LEADER	N= 6 K= 64	$h = 6$	0.9	7	n/a	t/o	m/o
HERMAN	N= 3	$h = 1$	0.1	20	0.90	0.17	0.03
HERMAN	N= 3	$h = 1$	0.99	20	0.90	0.20	0.02
HERMAN	N= 9	$h = 1$	0.01	20	2.71	0.15	0.16
HERMAN	N= 9	$h = 1$	0.05	20	2.71	0.18	0.17
HERMAN	N= 15	$h = 1$	10^{-4}	20	4.52	0.20	32.47
HERMAN	N= 15	$h = 1$	$5 \cdot 10^{-4}$	20	4.52	0.17	32.40
HERMAN	N= 21	$h = 1$	10^{-5}	20	n/a	0.19	t/o
HERMAN	N= 21	$h = 1$	$5 \cdot 10^{-5}$	20	n/a	0.19	t/o
HERMAN	N= 31	$h = 1$	10^{-8}	20	n/a	0.20	t/o
HERMAN	N= 31	$h = 1$	$5 \cdot 10^{-8}$	20	n/a	0.17	t/o
HERMAN	N= 41	$h = 1$	10^{-11}	20	12.34	0.21	1586.00
HERMAN	N= 41	$h = 1$	$5 \cdot 10^{-11}$	20	12.34	0.18	1661.00

Time presented in seconds. The number of states was determined by PRISM. For models on which PRISM reached a t/o or a m/o we assumed a large number (10^{18}) to generate the plot in Figure 2. "Prob." denotes the lower bound on the probability mass to find. We had to extend models LEADER, CROWDS, and HERMAN to support larger parameter values. The extended models were only used for those parameters not supported by the original model in the PRISM benchmark set.