Reachability Analysis for Incomplete Networks of Markov Decision Processes*

Ralf Wimmer[§], Ernst Moritz Hahn[‡], Holger Hermanns[‡], and Bernd Becker[§]

[§]*Albert-Ludwigs-University Freiburg, Germany*
{wimmer,becker}@informatik.uni-freiburg.de
[‡]*Saarland University, Saarbrücken, Germany*
{emh,hermanns}@cs.uni-saarland.de

***Abstract*—Assume we have a network of discrete-time Markov decision processes (MDPs) which synchronize via common actions. We investigate how to compute probability measures in case the structure of some of the component MDPs (so-called blackbox MDPs) is not known. We then extend this computation to work on networks of MDPs that share integer data variables of finite domain. We use a protocol which spreads information within a network as a case study to show the feasibility and effectiveness of our approach.**

I. Introduction

Markov decision processes (MDPs) have been applied successfully to reason about quantitative properties in a large number of areas, in particular network protocols. In addition to probabilistic choices as in Markov chains, MDPs also contain nondeterministic choices. To obtain a stochastic process, the nondeterminism has to be resolved. Given a property, we are usually interested in minimal and maximal probabilities that the property holds where the minimum (maximum) is taken over all resolutions of the nondeterminism. This paper is about networks of MDPs in which for some components we only know the interface to the environment, but not their behavior. We denote such processes as *blackboxes*. A network of MDPs without blackboxes is again an MDP, and can be analyzed as such. In our setting however, because of this incomplete specification, we can only consider upper and lower bounds for both minimal and maximal probabilities, when considering all possible implementations of the blackboxes.

The problem studied here is motivated by the intention to formally analyze the behavior of peer-to-peer distributed hash table architectures, such as Pastry [1] or Chord [2]. These architectures are designed to be scalable, fault tolerant, and self-organizing. This is achieved by distributing and replicating the hash entries over a network of nodes, together with effective lookup and maintenance protocols. The individual nodes participating in, say, Chord can faithfully be modelled as MDPs, properly connected to reflect the link structure. Blackbox MDPs can then be used as very natural means to reflect a Byzantine faulty node, and also as an abstraction for entire subnetworks. Indeed we experiment with simple scenarios of Chord-like communication networks in the experimental section.

There exist a number of related areas in which blackboxes have been used successfully before. Nopper, Scholl, and Becker [3], [4] considered circuits in which some components are blackboxes of which only input and output interfaces are known. They consider two questions: (1) whether there exists an implementation such that a given CTL specification is fulfilled (*realizability*), and (2) whether the specification is fulfilled for all possible implementations (*validity*). Because of the potentially infinite state space of the blackboxes, these problems are undecidable. In turn, the authors consider approximate methods in a symbolic implementation. Their work is motivated by the advantages of using formal methods in early design phases, as well as the possibility to speed up the model checking process when exact implementations of system parts are not relevant for certain properties.

In [5], [6] a SAT-based bounded model checking (BMC) method was developed for such incomplete circuits. The task of this method is to prove that a given invariant property is violated for all possible blackbox implementations (*non-realizability*). Recently these BMC methods were also extended to incomplete networks of timed automata [7]. The assumption there is that in a network of timed automata which communicate via channels, common clocks, and integer variables, only the communication interface is available for a subset of the automata, but not their internal structure. The authors also strive for the falsification of invariant properties.

In [8], the compositional verification of (complete) networks of MDPs is considered. The idea is to avoid constructing the very large state space of the complete network to verify a given property. Instead, properties of individual processes are proven, and the correctness of the overall network is shown by assume-guarantee reasoning. This way, the behavior of network components is substituted by partial formulae, such that some information about the behavior

*This work was partly supported by the German Research Council (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS) (see www.avacs.org for more information).

of some processes is lost. This can be considered as a greybox variation of the setting considered by us, and, similar to our setting, the method may only derive lower and upper bounds for extremal probabilities. Because a priori all network components are known, by choosing the partial formulae accordingly, tighter bounds can be obtained than in our method. However, the analysis of [8] involves a substantial degree of manual intervention, needed to construct the appropriate partial formulae. In addition, it does not consider communication via shared integer variables as we do.

Organization of the paper. In the following section we will provide the definitions of MDPs and networks of MDPs. In Section III we will consider incomplete networks and show how bounds on the probabilities of reachability properties can be derived. In Section IV and Section V we do the same for integer-decorated MDPs. In Section VI we will present experimental results for a case study from the computer network domain. Section VII concludes the paper and presents ideas for future work.

II. Networks of Markov Decision Processes

In this section we introduce discrete-time Markov decision processes and networks thereof.

Let $\mathrm{Distr}(S)$ denote the set of probability distributions on a finite set S and $\delta_s \in \mathrm{Distr}(S)$ the probability distribution which assigns 1 to $s \in S$, and 0 to all other elements.

Definition 1 *A **discrete-time Markov decision process** (MDP) is a tuple $M = (S, s_I, A, P)$ such that S is a finite, non-empty set of states of which $s_I \in S$ is the initial state, A is a finite set of actions and $P \subseteq S \times A \times \mathrm{Distr}(S)$ is a set of transitions, which assign a probability distribution over the successor states to state/action-pairs. We assume that for each $s \in S$ and $a \in A$ there is at most one tuple (s, a, d) in P, i. e., $\forall s \in S\ \forall a \in A: \ \big|\{d \in \mathrm{Distr}(S) \,|\, (s, a, d) \in P\}\big| \leq 1$.*

We denote the set of actions among which can be chosen in the state $s \in S$ by $A_s = \{a \in A \,|\, \exists d \in \mathrm{Distr}(S) : (s, a, d) \in P\}$. We assume in the following that $A_s \neq \emptyset$ for all $s \in S$.

Markov decision processes combine nondeterministic and probabilistic behavior. Being in state $s \in S$, first one of the actions in A_s is chosen nondeterministically. According to the probability distribution of the chosen action, the successor state is determined probabilistically.

An infinite path in M is an infinite sequence $s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \cdots$ such that there is $(s_i, a_i, d_i) \in P$ with $d_i(s_{i+1}) > 0$ for all $i \geq 0$. A finite path is a finite sequence $s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} s_n$ with $d_i(s_{i+1}) > 0$ for all $0 \leq i < n$. The $(i+1)^{\text{th}}$ prefix of π, i. e., $s_0 \xrightarrow{a_0} \cdots \xrightarrow{a_{i-1}} s_i$, is denoted by $\pi\!\uparrow^i$. For a finite path π, the last state of π is denoted by $\mathrm{last}(\pi)$. The length $|\pi|$ of a finite path $\pi = s_0 \xrightarrow{a_0} \cdots \xrightarrow{a_{n-1}} s_n$ is n. We write $\mathrm{Paths}^{fin}_M(s)$ for all finite paths that start in state $s \in S$.

It is only possible to assign probabilities to paths if the nondeterminism has been resolved. This is done by an entity called *scheduler*. A scheduler σ for an MDP $M = (S, s_I, A, P)$ is a function $\sigma : (S \times A)^* \times S \to \mathrm{Distr}(A)$ such that, for all finite paths π, $\sigma(\pi)(a) > 0$ implies $a \in A_{\mathrm{last}(\pi)}$. The set of all schedulers for M is denoted by Sched_M. Given a scheduler σ, the probability $\Pr^\sigma_M(\pi)$ of a finite path $\pi = s_0 \xrightarrow{a_0} \cdots \xrightarrow{a_{n-1}} s_n$ is given by $\Pr^\sigma_M(\pi) = \prod_{i=0}^{n-1} \sigma(\pi\!\uparrow^i)(a_i) \cdot d_i(s_{i+1})$, provided that $(s_i, a_i, d_i) \in P$ for all $0 \leq i < n$, and 0 otherwise. For sets of infinite paths, a probability measure can be defined as follows: Let π be a finite path. The *cone* $C(\pi)$ consists of all infinite paths that have π as a prefix. We define its probability by $\Pr^\sigma_M\big(C(\pi)\big) = \Pr^\sigma_M(\pi)$. The probability space of M is then the smallest σ-algebra that contains the cones of all finite paths starting in the initial state.

An important class of properties are *reachability properties*. Given a set of target states $T \subseteq S$ in an MDP $M = (S, s_I, A, P)$, the question is: starting in state $s \in S$, what is the probability to walk along a path which contains a state of T?

Let $\mathrm{reach}_M(s, T) := \{s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots \xrightarrow{a_{n-1}} s_n \in \mathrm{Paths}^{fin}_M(s) \mid s_n \in T \wedge \neg\exists i : 0 \leq i < n \wedge s_i \in T\}$ be the set of finite paths that start in s and end in T without visiting T in between. For a fixed scheduler σ we have

$$\Pr^\sigma_M(s, T) = \sum_{\pi \in \mathrm{reach}_M(s,T)} \Pr^\sigma_M(\pi).$$

In order to answer the above question independently from the scheduler, one is normally interested in the maximal and minimal probability over all schedulers, which we denote by

$$\Pr^{\min}_M(s, T) = \min_{\sigma \in \mathrm{Sched}_M} \Pr^\sigma_M(s, T) \qquad \text{and}$$
$$\Pr^{\max}_M(s, T) = \max_{\sigma \in \mathrm{Sched}_M} \Pr^\sigma_M(s, T),$$

respectively. This allows to decide, e. g., whether the probability to reach a safety-critical state is below a given threshold.

A broader class of properties is described by the probabilistic computation tree logic PCTL [9]. To be able to perform PCTL model checking, it is sufficient to compute reachability properties for all states of an MDP. Since this requirement is fulfilled by the algorithms we are going to present in the following, we restrict our presentation to computing probabilities for reachability properties.

Typically large systems are not modeled as monolithic MDPs. Instead, MDP models for the individual components are created, which communicate via common actions. This is called a *network of MDPs*. Formally, a network of MDPs is a set $\mathcal{M} = \{M_1, \ldots, M_n\}$ of MDPs $M_i = (S_i, s^i_I, A_i, P_i)$ whose sets of actions do not need to be disjoint. We use

the typical interleaving semantics with synchronization: If an action a is executed, all MDPs with a in their action set must simultaneously execute a transition labeled with a. The other MDPs with $a \notin A_i$ remain idle. This is captured in the following definition:

Definition 2 *Let $M_i = (S_i, s_I^i, A_i, P_i)$ for $i = 1, 2$ be two MDPs. The **composition** $M_1 \| M_2 = (S, s_I, A, P)$ is an MDP with*

- $S = S_1 \times S_2$,
- $s_I = (s_I^1, s_I^2)$,
- $A = A_1 \cup A_2$, *and*
- $\big((s_1, s_2), a, d\big) \in P$ *iff*
  - *either $a \in A_1 \cap A_2$ and there are $(s_1, a, d_1) \in P_1$ and $(s_2, a, d_2) \in P_2$ such that $d\big((t_1, t_2)\big) = d_1(t_1) \cdot d_2(t_2)$ for all $(t_1, t_2) \in S$,*
  - *or $a \in A_1 \setminus A_2$ and there is $(s_1, a, d_1) \in P_1$ such that for all $t_1 \in S_1$ holds: $d\big((t_1, s_2)\big) = d_1(t_1)$ and for all $t_2 \in S_2 \setminus \{s_2\}$ and all $t_1' \in S_1$: $d\big((t_1', t_2)\big) = 0$,*
  - *or $a \in A_2 \setminus A_1$ and there is $(s_2, a, d_2) \in P_2$ such that for all $t_2 \in S_2$ holds: $d((s_1, t_2)) = d_2(t_2)$ and for all $t_1 \in S_1 \setminus \{s_1\}$ and all $t_2' \in S_2$: $d\big((t_1, t_2')\big) = 0$.*

This composition is commutative and associative. It can be used to construct an MDP for the whole system from the MDPs of the individual components.

III. Incomplete Networks of MDPs

In this paper, we consider the problem to derive information about a network of MDPs if only a subset of the components is available. We call this an *incomplete network of MDPs*. In an incomplete network, the internal structure of some of the MDPs may not be available. The only knowledge we have about the unavailable MDPs is the interface via which they interact with their environment. In our communication model, this interface is given by the set of actions that are used for synchronization with other (available) MDPs of the network.

Now assume that $\{M_1, \ldots, M_n\}$ is the set of available components of an incomplete network $\mathcal{M}$ of MDPs. Since the composition defined above is commutative and associative, we may compose the available MDPs first. In the following, we therefore assume that the available part of the network is given as a single MDP $M = M_1 \| M_2 \| \cdots \| M_n = (S, s_I, A, P)$ together with a set $A^C \subseteq A$ of actions that are used by the unavailable part of the network for synchronization with M. Please note that a transition $(s, a, d) \in P$ with $a \in A^C$ may only be taken if all blackbox MDPs with a in their action set simultaneously take a transition that is labeled with a. The blackbox blocks the execution of a common action $a \in A^C$ if the current state of the blackbox-MDP does not have an out-going transition labeled with a.

We use the notation $M \| BB$ to denote the network for a fixed implementation BB of the blackboxes.

Our goal is to compute bounds on the minimal and maximal probability of a given reachability property. For an incomplete network M, the target set T is given only for the available parts. For a corresponding complete network $M \| BB$ with S_{BB} being the states of the blackbox implementation, the according target set is $T \times S_{BB}$.

The extension of these reachability properties to PCTL requires a three-valued variant thereof, which was defined, e. g., in [10]. This is due to the fact that we are only able to derive bounds on probabilities. Hence, there are cases where we can neither verify nor refute that the probability of a certain property is within a given bound. The extension of our algorithms to three-valued PCTL model checking is however straightforward.

When computing bounds on the minimal reachability probability we encounter the following problem: If we allow arbitrary schedulers, there are some which only choose actions that are local to the blackboxes. Therefore, the available part of the network never executes a step, and the bounds on the minimal probability of a given reachability property are always 0 (except if we already start in the target set). It is highly unrealistic for practical systems that parts of the system never execute a step. In order to avoid this problem, we introduce a very weak notion of fairness, which is a special case of existing notions [11]. For the computation of bounds on the maximal values this problem does not occur, since it is desirable for optimal schedulers to make progress in the available components.

Definition 3 *Let σ be a scheduler for a network $M \| BB$ with A being the action set of M. We call σ **BB-fair** iff the probability to walk along a path on which infinitely often an action from A is taken equals 1. The set of BB-fair schedulers is denoted by $\mathrm{Sched}^{BB}_{M \| BB}$.*

We define

$$\Pr^{\min}_{M \| BB}(s, T \times S_{BB}) = \min_{\sigma \in \mathrm{Sched}^{BB}_{M \| BB}} \Pr^{\sigma}_{M \| BB}(s, T \times S_{BB})$$

as the minimal probability with which a state of $T \times S_{BB}$ is reached in the network $M \| BB$ when starting in state (s, s_{BB}) where s_{BB} is an arbitrary state of the blackbox. Analogously, with

$$\Pr^{\max}_{M \| BB}(s, T \times S_{BB}) = \max_{\sigma \in \mathrm{Sched}^{BB}_{M \| BB}} \Pr^{\sigma}_{M \| BB}(s, T \times S_{BB})$$

we denote the maximal probability thereof. We can formulate our goal as follows: For an incomplete network (M, A^C) of MDPs, compute values $\Pr^{\min,+}_M(s, T)$, $\Pr^{\min,-}_M(s, T)$, $\Pr^{\max,+}_M(s, T)$, and $\Pr^{\max,-}_M(s, T)$ such that for all implementations BB of the blackboxes the following inequalities

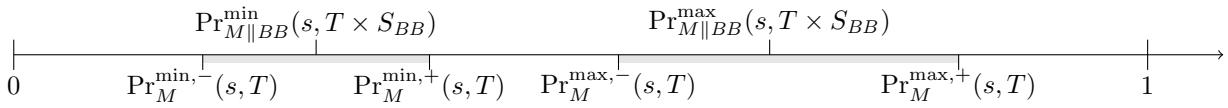

Figure 1. Over- and under-approximating probabilities for incomplete networks of MDPs

hold (cf. Fig. 1):

$$\Pr^{\min,-}_{M}(s,T) \leq \Pr^{\min}_{M\|BB}(s, T \times S_{BB}) \leq \Pr^{\min,+}_{M}(s,T),$$
$$\Pr^{\max,-}_{M}(s,T) \leq \Pr^{\max}_{M\|BB}(s, T \times S_{BB}) \leq \Pr^{\max,+}_{M}(s,T). \quad (1)$$

Definition 4 *Let $M = (S, s_I, A, P)$ be an incomplete network of MDPs with A^C being the set of actions for communication with the blackboxes. Let $\lfloor M \rfloor$ be the network that is obtained from M by removing all transitions whose action label is contained in A^C, i. e., $\lfloor M \rfloor = \big(S, s_I, A \setminus A^C, \lfloor P \rfloor\big)$ with $\lfloor P \rfloor = \big\{(s,a,d) \in P \,|\, a \in A \setminus A^C\big\}$.*

Accordingly, we denote by $\lceil M \rceil$ the network which results from M by ignoring the restrictions the blackbox implementation may impose on the executability of transitions. Hence $\lceil M \rceil = M$, but we neglect that synchronization with the blackbox can happen.

The construction of Definition 4 is only intended to be used for the computation of probability bounds over all possible blackbox implementations. It should not be involved in further composition operations.

Then, on the one hand, $\lfloor M \rfloor$ is an under-approximation of the behavior of $M\|BB$ for any possible implementation of the blackboxes. On the other hand, $\lceil M \rceil$ is an over-approximation thereof.

Theorem 1 *The probabilities computed from $\lfloor M \rfloor$ and $\lceil M \rceil$ have the following relation:*

$$\Pr^{\min}_{\lceil M \rceil}(s,T) \leq \Pr^{\min}_{M\|BB}(s, T \times S_{BB}) \leq \Pr^{\min}_{\lfloor M \rfloor}(s,T),$$
$$\Pr^{\max}_{\lfloor M \rfloor}(s,T) \leq \Pr^{\max}_{M\|BB}(s, T \times S_{BB}) \leq \Pr^{\max}_{\lceil M \rceil}(s,T). \quad (2)$$

Proof: Given a finite path $\pi = (s_0^1, s_0^2) \xrightarrow{a_0} (s_1^1, s_1^2) \xrightarrow{a_1} (s_2^1, s_2^2) \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} (s_n^1, s_n^2)$ of the composition of two models $M_1\|M_2$, we define $\mathrm{proj}_{M_i}(\pi) := s_0^i \xrightarrow{a_0} s_1^i \xrightarrow{a_1} s_2^i \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} s_n^i$ for $i = 1, 2$. For a scheduler σ on $M_1\|M_2$, we define $\mathrm{proj}_{M_i}(\sigma)$ such that

$$\mathrm{proj}_{M_i}(\sigma)(\pi)(a) := \sum_{\{\pi' \,|\, \mathrm{proj}_{M_i}(\pi') = \pi\}} \Pr^{\sigma}_{M_1\|M_2}(\pi') \cdot \sigma(\pi')(a).$$

Assume the blackbox implementation is $BB = (S_{BB}, s^I_{BB}, A_{BB}, P_{BB})$ and the non-blackbox part is $M = (S, s_I, A, P)$. The set of common actions A^C is given by $A^C = A \cap A_{BB}$. We define $M' := (S, s_I, A \cup A_{BB}, P')$ with $P'(s,a)(s') := P(s,a)(s')$ if $a \in A$, $P'(s,a)(s) := 1$ if $a \in A_{BB} \setminus A$ and 0 else. This way, M' extends $\lceil M \rceil$ by introducing self-loops labeled with actions which are actions of the blackbox only.

Consider a fair scheduler σ in $M\|BB$ which yields the minimal probability of reaching T, i. e., for which $\Pr^{\sigma}_{M\|BB}(s, T \times S_{BB}) = \Pr^{\min}_{M\|BB}(s, T \times S_{BB})$. The paths that reach $T \times S_{BB}$ in $M\|BB$ can be partitioned according to their projections on M'. Each path that reaches a target state in $M\|BB$ uniquely belongs to one path that reaches a target states in M'. Hence, due to the definition of M', we have

$$\mathrm{reach}_{M\|BB}(s, T \times S_{BB}) = \biguplus_{\pi \in \mathrm{reach}_{M'}(s,T)} \{\pi' \in \mathrm{reach}_{M\|BB}(s, T \times S_{BB}) \mid \mathrm{proj}_{M'}(\pi') = \pi\}.$$

Let $\sigma' := \mathrm{proj}_{M}(\sigma)$. Then, for $\pi \in \mathrm{reach}_{M'}(s,T)$

$$\Pr^{\sigma'}_{M'}(\pi) = \sum_{\{\pi' \in \mathrm{reach}_{M\|BB}(s, T \times S_{BB}) \,|\, \mathrm{proj}_{M'}(\pi') = \pi\}} \Pr^{\sigma}_{M\|BB}(\pi').$$

From this, it follows that $\Pr^{\sigma'}_{M'}(s,T) = \Pr^{\sigma}_{M\|BB}(S, T \times S_{BB})$. There exists a non-randomized fair scheduler σ'' with $\Pr^{\sigma''}_{M'}(s,T) \leq \Pr^{\sigma'}_{M'}(s,T)$ [11]. In M', actions of $A_{BB} \setminus A$ can only lead to self-loops with probability 1, which may in turn only be taken a finite number of times. Because of this, taking or not taking such self-loops does not influence the reachability probability. Thus, without loss of generality, we assume σ'' to only choose actions of A. Then, σ'' is also a scheduler in $\lceil M \rceil$, and has the same reachability probability in this model. So, $\Pr^{\min}_{\lceil M \rceil}(s,T) \leq \Pr^{\min}_{M\|BB}(s, T \times S_{BB})$.

Now consider a scheduler σ of $\lfloor M \rfloor$ with $\Pr^{\sigma}_{\lfloor M \rfloor}(s,T) = \Pr^{\min}_{\lfloor M \rfloor}(s,T)$. We define σ' such that for $s_1 \in S$ and $s_2 \in S_{BB}$ we have $\sigma'((s_1, s_2)) := \sigma(s_1)$. We have $\Pr^{\sigma'}_{M\|BB}(s, T \times S_{BB}) = \Pr^{\sigma}_{\lfloor M \rfloor}(s,T)$ and thus $\Pr^{\min}_{M\|BB}(s, T \times S_{BB}) \leq \Pr^{\min}_{\lfloor M \rfloor}(s,T)$.

Thus, we have shown both parts of the inequation for minimal reachability. The proof for maximal reachability works analogously. ∎

This means we can compute the bounds that we are looking for from $\lceil M \rceil$ and $\lfloor M \rfloor$ using the standard methods for MDPs. These are the tightest bounds we can obtain for an incomplete network of MDPs, because there are implementations of the blackboxes that actually yield these probabilities: Let $BB_l = \big(\{t_I\}, t_I, A^C \dot\cup \{a_{BB}\}, \{(t_I, a_{BB}, \delta_{t_I})\}\big)$ and $BB_u = \big(\{t_I\}, t_I, A^C, \{(t_I, a, \delta_{t_I}) \,|\, a \in A^C\}\big)$ be two implementations of the blackbox, both consisting of a single state with self-loops. Then the probability of reaching a target

state in $\lfloor M \rfloor$ coincides with the probability in $M \| BB_l$, and the probability in $\lceil M \rceil$ coincides with that in $M \| BB_u$.

IV. Extending MDPs with Bounded Integer Variables

In order to have more flexibility for modelling, systems are often decorated with variables with finite domains. This does not increase the principal modeling power of MDPs, but leads to more compact and more natural models. In this section we consider networks of MDPs with common bounded integer variables. These integer variables control the executability of actions. They may be assigned new values during a transition.

Let $I = \{i_1, \ldots, i_m\}$ be a set of bounded integer variables. For each $i_j \in I$, $\mathrm{range}(i_j) \subseteq \mathbb{Z}$ denotes the (finite) domain of i_j. The initial value of each integer variable is given by $\mathrm{init}(i_j) \in \mathrm{range}(i_j)$.

The set of *variable constraints* is given by the following context-free grammar with initial symbol Φ:

$$\Phi ::= \mathit{true} \mid f(i_1, \ldots, i_m) \sim c \mid (\Phi \wedge \Phi) \mid \neg\Phi$$

with a constant $c \in \mathbb{Z}$, variables $i_1, \ldots, i_m \in I$, a function $f : \mathbb{Z}^m \to \mathbb{Z}$, and a comparison operator $\sim \in \{<, \leq, =, \neq, \geq, >\}$. The set of variable constraints over I is denoted by $\Phi(I)$.

A *variable assignment* for I is a function $\nu : I \to \mathbb{Z}$ such that for all $i \in I$ the condition $\nu(i) \in \mathrm{range}(i)$ holds. The set of all variable assignments for I is denoted by $\mathfrak{A}(I)$. We extend ν to functions $f : \mathbb{Z}^m \to \mathbb{Z}$ by $\nu\big(f(i_1, \ldots, i_m)\big) = f\big(\nu(i_1), \ldots, \nu(i_m)\big)$. That a variable assignment ν satisfies a variable constraint φ (written $\nu \vDash \varphi$) can be defined as follows:

$$\begin{aligned} \nu &\vDash \mathit{true}, \\ \nu \vDash f(i_1, \ldots, i_m) \sim c \quad &\Leftrightarrow \quad \nu\big(f(i_1, \ldots, i_m)\big) \sim c, \\ \nu \vDash (\varphi_1 \wedge \varphi_2) \quad &\Leftrightarrow \quad \nu \vDash \varphi_1 \text{ and } \nu \vDash \varphi_2, \\ \nu \vDash \neg\phi \quad &\Leftrightarrow \quad \nu \nvDash \varphi. \end{aligned}$$

For two sets A and B, we denote the set of total functions from A to B by $\mathcal{F}(A, B)$, and the set of partial functions by $\mathcal{F}_p(A, B)$. Furthermore, we identify a tuple $(v_1, \ldots, v_m) \in \mathbb{Z}^m$ with the variable assignment $\nu \in \mathfrak{A}(I)$ that satisfies $\nu(i_j) = v_j$ for all $j = 1, \ldots, m$. If $\tilde{f} : I \to \mathcal{F}(\mathbb{Z}^m, \mathbb{Z})$ is a partial function, $\nu[\tilde{f}]$ denotes the variable assignment with

$$\nu[\tilde{f}](i) = \begin{cases} \tilde{f}(i)\big(\nu(i_1), \ldots, \nu(i_m)\big), & \text{if } \tilde{f} \text{ is defined for } i, \\ \nu(i), & \text{otherwise.} \end{cases}$$

Let $\tilde{f}, \tilde{g} \in \mathcal{F}_p(A, B)$ be partial functions such that $\tilde{f}$ is defined on a set $A_{\tilde{f}} \subseteq A$ and $\tilde{g}$ on $A_{\tilde{g}} \subseteq A$ with $A_{\tilde{f}} \cap A_{\tilde{g}} = \emptyset$. Then we write $\tilde{f} \uplus \tilde{g}$ for the function with

$$(\tilde{f} \uplus \tilde{g})(a) = \begin{cases} \tilde{f}(a), & \text{if } a \in A_f, \\ \tilde{g}(a), & \text{if } a \in A_g, \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

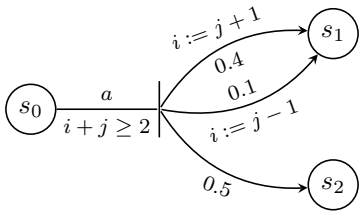

Figure 2. Illustration of a transition in an IMDP

Definition 5 *An **integer-decorated MDP** (IMDP) is a tuple $M = (S, s_I, A, I, P)$ such that S is a finite, non-empty set of states, $s_I \in S$ the initial state, A a finite set of actions, I a finite set of bounded integer variables and $P \subseteq S \times A \times \Phi(I) \times \mathrm{Distr}\big(S \times \mathcal{F}_p(I, \mathcal{F}(\mathbb{Z}^m, \mathbb{Z}))\big)$ the transition probabilities. For the elements $(s, a, g, d) \in P$, we require that, if $d(t, \tilde{f}) > 0$ and $\tilde{f}$ is defined for $i \in I$, then $\tilde{f}(i) \in \mathcal{F}\big(\mathrm{range}(i_1) \times \cdots \times \mathrm{range}(i_m), \mathrm{range}(i)\big)$.*

The last condition ensures that only values from $\mathrm{range}(i)$ are assigned to each integer variable i. This implies that the probability distributions assign a positive value to only finitely many alternatives.

A transition of an IMDP is illustrated in Fig. 2. The action label of this transition is a and its guard $i + j \geq 2$. This means the transition is only enabled if the sum of the values of i and j is at least 2. With probability 0.5 the successor state is s_2. The values of the variables do not change in this case. With probability 0.4 the next state is s_1 and the value of i is updated to the current value of j plus one. With probability 0.1, however, the successor state is also s_1, but the new value of i is the current value of j minus one. Variables that are not explicitly updated on a transition keep their value.

Every IMDP can be transformed into an ordinary MDP by putting the integer variables into the state space.

Definition 6 (Integer elimination) *Let $M = (S, s_I, A, I, P)$ be an extended MDP with $I = \{i_1, \ldots, i_m\}$. $\tilde{M} = (\tilde{S}, \tilde{s}_I, \tilde{A}, \tilde{P})$ is an equivalent ordinary MDP with*

- $\tilde{S} = S \times \mathrm{range}(i_1) \times \cdots \times \mathrm{range}(i_m)$.
- $\tilde{s}_I = \big(s_I, \mathrm{init}(i_1), \ldots, \mathrm{init}(i_m)\big)$
- $\tilde{A} = A$.
- $\big((s, v_1, \ldots, v_m), a, \tilde{d}\big) \in \tilde{P}$ *iff there is a transition* $(s, a, g, d) \in P$ *such that* $(v_1, \ldots, v_m) \vDash g$ *and*

$$\tilde{d}(s', v'_1, \ldots, v'_m) = \sum_{\substack{\tilde{f} \in \mathcal{F}_p\left(I, \mathcal{F}(\mathbb{Z}^m, \mathbb{Z})\right) \\ (v_1, \ldots, v_m)[\tilde{f}] = (v'_1, \ldots, v'_m)}} d(s', \tilde{f}).$$

The sum in the definition of $\tilde{d}$ is needed because different update functions can lead to the same values of the integer variables. Their probabilities are accumulated for the new transition.

A *(complete) network of IMDPs* is a set $\mathcal{M} = \{M_1, \ldots, M_n\}$ of IMDPs $M_i = (S_i, s_I^i, A_i, I_i, P_i)$. We assume that the guards of the transitions depend on variables from a set I with $I_i \subseteq I$ for all $1 \leq i \leq n$. In order to avoid write conflicts on the integer variables, we assume that I_i contains those variables which may be assigned new values in M_i and that all I_i are pairwise disjoint. That means each variable can be read by all processes but written by only one of them. The concepts for incomplete networks that we develop in the sequel can, however, also be applied to arbitrary strategies for resolving write conflicts.

Definition 7 *Let for $i = 1, 2$ $M_i = (S_i, s_I^i, A_i, I_i, P_i)$ denote an IMDP. The* ***composition*** *$M_1 \| M_2$ is an IMDP $M = (S, s_I, A, I, P)$ such that*

- $S = S_1 \times S_2$,
- $s_I = (s_I^1, s_I^2)$,
- $A = A_1 \cup A_2$,
- $I = I_1 \cup I_2$,
- $e = \big((s_1, s_2), a, g, d\big) \in P$ *iff either*
  - $a \in A_1 \cap A_2$ *and there are transitions* $e_i = (s_i, a, g_i, d_i) \in E_i$ *for* $i = 1, 2$ *such that* $g = (g_1 \wedge g_2)$ *and* $d\big(\tilde{f}, (t_1, t_2)\big) = d_1(\tilde{f}_1, t_1) \cdot d_2(\tilde{f}_2, t_2)$ *for partial functions* $\tilde{f}_i \in \mathcal{F}_p\big(I_i, \mathcal{F}(\mathbb{Z}^m, \mathbb{Z})\big)$ *with* $\tilde{f} = \tilde{f}_1 \uplus \tilde{f}_2$, *or*
  - $a \in A_1 \setminus A_2$ *and there is a transition* $e_1 = (s_1, g, a, d_1) \in P_1$ *such that for all* $\tilde{f}_1 \in \mathcal{F}_p\big(I_1, \mathcal{F}(\mathbb{Z}^m, \mathbb{Z})\big)$ *it is* $d\big(\tilde{f}_1, (t_1, s_2)\big) = d_1(\tilde{f}_1, t_1)$ *and* $d\big(\tilde{f}, (t_1, t_2)\big) = 0$ *otherwise, or*
  - $a \in A_2 \setminus A_1$ *and there is a transition* $e_2 = (s_2, g, a, d_2) \in P_2$ *such that for all* $\tilde{f}_2 \in \mathcal{F}_p\big(I_2, \mathcal{F}(\mathbb{Z}^m, \mathbb{Z})\big)$ *it is* $d\big(\tilde{f}_2, (s_1, t_2)\big) = d_2(\tilde{f}_2, t_2)$ *and* $d\big(\tilde{f}, (t_1, t_2)\big) = 0$ *otherwise.*

For each network of IMDPs, one can construct a single ordinary MDP by first computing the composition of all components and then eliminating all integer variables.

V. Incomplete Networks of IMDPs

With these preparations we can now define incomplete networks of IMDPs.

Definition 8 *An* ***incomplete network of integer-decorated MDPs****, communicating via common actions and common integer variables, is given by a set $\{M_1, \ldots, M_n\}$ of available IMDPs, a set A^C of actions used by the blackbox MDPs for synchronization, and a set I^C of integer variables which may be written by the blackboxes.*

In networks of IMDPs, blackboxes not only have the power to block the execution of common actions, but also to change the values of variables on which they have write access in an arbitrary way. In incomplete networks we can therefore neglect all updates to the blackbox integer variables from I^C, since between any two steps of the visible part a transition in a blackbox can be scheduled that updates the integer variables in I^C.

Without limitation of generality, we assume that the available part of the network is given as a single IMDP $M = (S, s_I, A, I^C, P)$. If more than one component is available, we can construct their composition, resulting in a single IMDP. After composition all integer variables which are not blackbox variables, i. e., the variables in $I \setminus I^C$, can be eliminated. We again restrict our presentation to reachability properties. Hence we assume that a set $T \subseteq S$ of target states is given and that our task is to compute the probability of eventually reaching a state in T. However, the techniques we are going to present in the following, can be applied to other properties as well.

In principle we can proceed in the same way as for incomplete networks of ordinary MDPs without integer variables. We declare all transitions with a blackbox action and all transitions whose guard depends on a variable from I^C as unsafe. To obtain the outer bounds in Fig. 1, we replace the guards of all unsafe transitions by $true$ and treat their actions as non-blackbox actions. For the inner bounds, we remove all unsafe transitions. This method allows to use the standard techniques for computing the desired probabilities on MDPs.

Tighter approximations, however, can be obtained by our second approach. It is based on the idea to maximize and minimize the probability over all values that the blackbox integer variables can take to obtain upper and lower bounds respectively. For complete networks of MDPs, the probability of reachability properties can be computed by solving an equation system that involves either maximum or minimum operators, ranging over all actions that are selectable in a state. For the solution of these equation systems, a technique called *value iteration* is typically applied.

We will first give equation systems that characterize the four probability bounds for incomplete networks and show afterwards how value iteration can be adapted for these equation systems.

The lower bound on the maximal probability (i. e., $\Pr_M^{\max,-}(s, T)$) is the solution of the following equation system:

$$\begin{aligned} x_s^{\max,-} &= 1, && \text{if } s \in T, \\ x_s^{\max,-} &= \min_{\nu \in \mathfrak{A}(I^C)} \Big(\max_{\substack{(s,a,g,d) \in P: \\ \nu \vDash g \wedge a \notin A^C}} \big(\sum_{t \in S} d(t) \cdot x_t^{\max,-} \big) \Big), \\ & && \text{if } s \notin T. \end{aligned} \tag{3}$$

The upper bound on the maximal probability (i. e., $\Pr_M^{\max,+}(s, T)$) is characterized by replacing $\min$ by $\max$

in Eq. (3):

$$x_s^{\max,+} = 1, \qquad \text{if } s \in T,$$
$$x_s^{\max,+} = \max_{\nu \in \mathfrak{A}(I^C)} \Big(\max_{\substack{(s,a,g,d) \in P: \\ \nu \vDash g}} \big(\sum_{t \in S} d(t) \cdot x_t^{\max,+} \big) \Big), \qquad \text{if } s \notin T. \tag{4}$$

In an analogous way, the bounds $\Pr_M^{\min,-}(s,T)$ and $\Pr_M^{\min,+}(s,T)$ on the minimal probability can be characterized. The correctness of these bounds could be shown along the lines of the proof for Theorem 1, that is by mapping schedulers from one side of the inequation to the other, while maintaining probability bounds.

In order to arrive at an efficient algorithm for solving these equation systems, we first make the observation that it is not necessary to take into account all possible variable assignments.

Let $\nu \in \mathfrak{A}(I)$ be a variable assignment and $s \in S$ a state. By

$$\mathcal{E}(s,\nu) = \big\{ e \in E \,|\, e = (s,a,g,d) \wedge \nu \vDash g \big\}$$

we denote the set of out-going transitions of s whose guard is satisfied by ν. We partition the assignments according to the sets of enabled transitions:

$$\Pi(s) = \big\{ \{ \nu \in \mathfrak{A}(I) \,|\, \mathcal{E}(s,\nu) = \mathcal{E}(s,\nu') \} \,\big|\, \nu' \in \mathfrak{A}(I) \big\}.$$

We define the corresponding sets of transitions that are enabled by all assignments in $C \in \Pi(s)$ as

$$\mathcal{E}(s,C) = \{ e \in E \,|\, e = (s,g,a,d) \wedge \forall \nu \in C : \nu \vDash g \}.$$

For computing the probability bounds it is sufficient to take into account one assignment per block of $\Pi(s)$. When computing the inner bounds in Fig. 1 we can restrict ourselves to those blocks of $\Pi(s)$ such that the set of enabled transitions is minimal. Contrarily, for the outer bounds only those blocks have to be considered whose corresponding set of enabled transitions is maximal. Therefore we set

$$\Pi^{\min}(s) = \{ C \in \Pi(s) \,|\, \nexists C' \in \Pi(s) : \mathcal{E}(s,C') \subsetneq \mathcal{E}(s,C) \},$$
$$\Pi^{\max}(s) = \{ C \in \Pi(s) \,|\, \nexists C' \in \Pi(s) : \mathcal{E}(s,C') \supsetneq \mathcal{E}(s,C) \}.$$

Then we can write Equations (3) and (4) as follows:

$$x_s^{\max,-} = 1, \qquad \text{if } s \in T,$$
$$x_s^{\max,-} = \min_{C \in \Pi^{\min}(s)} \Big(\max_{\substack{(s,g,a,d) \in \mathcal{E}(s,C) \\ \wedge a \notin A^C}} \big(\sum_{t \in S} d(t) \cdot x_t^{\max,-} \big) \Big), \qquad \text{if } s \notin T. \tag{5}$$

$$x_s^{\max,+} = 1, \qquad \text{if } s \in T,$$
$$x_s^{\max,+} = \max_{C \in \Pi^{\max}(s)} \Big(\max_{(s,g,a,d) \in \mathcal{E}(s,C)} \big(\sum_{t \in S} d(t) \cdot x_t^{\max,+} \big) \Big), \qquad \text{if } s \notin T. \tag{6}$$

Algorithm 1 Value Iteration for $\Pr_M^{\max,-}(\cdot,T)$

```
1: procedure ValueIteration(M = (S, s_I, A, I, P), A^C, I^C, T)
2:     for all s ∈ S do
3:         if s ∈ T then x_s^0 ← 1.0
4:         else x_s^0 ← 0.0
5:     end for
6:     n ← 0
7:     repeat
8:         for all s ∈ S do
9:             if s ∈ T then x_s^{n+1} ← 1.0
10:            else
11:                x_s^{n+1} ← min_{C ∈ Π^min(s)} ( max_{(s,g,a,d) ∈ E(C), ∧ a ∉ A^C} ( Σ_{t ∈ S} d(t) · x_t^n ) )
12:            end if
13:        end for
14:        n ← n + 1
15:    until ||x_s^n − x_s^{n−1}|| < ε for all s ∈ S
16:    return (x_s^n)_{s ∈ S}
17: end procedure
```

The corresponding equations for the bounds on the minimal probability are:

$$x_s^{\min,+} = 1, \qquad \text{if } s \in T,$$
$$x_s^{\min,+} = \max_{C \in \Pi^{\min}(s)} \Big(\min_{(s,g,a,d) \in \mathcal{E}(s,C)} \big(\sum_{t \in S} d(t) \cdot x_t^{\min,+} \big) \Big), \qquad \text{if } s \notin T. \tag{7}$$

$$x_s^{\min,-} = 1, \qquad \text{if } s \in T,$$
$$x_s^{\min,-} = \min_{C \in \Pi^{\max}(s)} \Big(\min_{\substack{(s,g,a,d) \in \mathcal{E}(s,C) \\ \wedge a \notin A^C}} \big(\sum_{t \in S} d(t) \cdot x_t^{\min,-} \big) \Big), \qquad \text{if } s \notin T. \tag{8}$$

Reachability probabilities for MDPs can be determined efficiently using a technique called *value iteration* [12]. An adaption of the value iteration technique for our incomplete networks of IMDPs is shown in Algorithm 1. The given variant computes $\Pr_M^{\max,-}(\cdot,T)$. For the other three bounds, only the update step in line 11 has to be adapted.

Theorem 2 *Algorithm 1 terminates for all $\varepsilon > 0$. For all states $s \in S$ the sequence $(x_s^n)_{n \in \mathbb{N}}$ converges to $\Pr_M^{\max,-}(s,T)$ for $n \to \infty$.*

For a proof of this theorem consider [13]. This paper states the correctness of value iteration for various variants of stochastic games. Computing the probabilities for an incomplete network of IMDPs corresponds to solving alternating stochastic games in which one player chooses the variable

Table I
CHORDS OF DIFFERENT SIZES (BUFFER SIZE $K = 1$)

n	$\lvert S\rvert$	$\lvert T\rvert$	$\Pr^{\max}$	Time [s]
11	2031	19014	0.7987	0.33
12	4079	42058	0.7987	0.68
13	8175	91950	0.7987	1.40
14	16367	199474	0.7987	3.06
15	32751	429872	0.7987	6.39
16	65519	921394	0.7987	13.96
17	65537	1005572	0.8000	0.80
18	262111	4316646	0.7997	63.90
19	524255	9141686	0.7997	135.31
20	1048543	19299802	0.7997	283.18

assignment for the integer variables and the other one an enabled transition of the current state. Therefore Theorem 2 directly follows from the results for alternating stochastic games.

VI. CASE STUDY

To evaluate our approach we implemented the value iteration technique for incomplete networks of IMDPs, which we have presented in the previous section. Our tool is able to read models which are specified using the guarded command language that is used by the probabilistic model checker PRISM [14]. We applied it to a case study which models information spread in a computer network.

Assume we have a computer network consisting of n nodes each of which has a message buffer with a capacity of K messages. This results in a potential state space of $(K+1)^n$ states, but some of them may be unreachable. One node of the network serves as the sender node which initially has a single message in its message buffer. The buffers of all other nodes are empty. A second node is the receiver node that should eventually obtain the message. If the buffer of a node contains at least one message and at least one of the neighbors' buffers has not reached its capacity limit yet, the following step is executed with probability $p > 0$: One of the messages is placed in the buffers of all neighbor nodes that are not full yet. After sending a message, the node decreases the number of messages in its buffer by one. With probability $1 - p$ a message is discarded. An example of such a network with three nodes together with the input file for PRISM is shown in Fig. 3. We want to compute the probability that finally a message is placed in the buffer of the receiver node, i. e., $\mathcal{P}_{=?}(\mathit{true}\,\mathcal{U}\, x_n > 0)$.

We modelled *Chord networks* [2] of different sizes with this formalism. A Chord network is a ring structure with additional shortcut links. If $N_0, \ldots, N_{n-1}$ are the nodes of such a Chord network of size n, then the set of successor nodes of N_i is $\{N_j \mid j = i + 2^p \bmod n \wedge 0 \leq p < \lceil \log_2 n \rceil\}$. In our experiments, the sender node is N_0, the receiver node N_{n-1}.

We put different nodes into blackboxes and computed $\Pr^{\max,-}$ to answer the question: "Is there a possibility that the message is received with a probability of at least p – independently of the behavior of the blackboxes?" The answer is yes if $\Pr^{\max,-}(s_I, T) \geq p$ holds for the initial state s_I where T is the set of all states in which the target node has received a message.

The experiments were run on an Intel Core2Duo processor with 2 GHz clock frequency and 4 GB of main memory under Kubuntu 10.04 Linux. We want to be able to handle chord networks with a non-trivial number of nodes. Therefore the buffer capacity of all nodes is set to the smallest reasonable value $K = 1$ because the size of the state space grows rapidly with increasing buffer capacity. The forwarding probability was set to $p = 0.8$. The results for the complete Chord networks are contained in Table I. One can observe that the size of the state space and the runtimes for the computations grow exponentially in the size of the network. The probability $\Pr^{\max}$, which we focus on, is approximately 0.8 for all instances. For the value iteration we set $\varepsilon = 10^{-8}$.

For the evaluation of our algorithm, we put one and two nodes of the Chord networks, respectively, into blackboxes, allowing them to behave in an arbitrary way. Due to restrictions in our prototypic implementation we were only able to add two blackboxes for $n \leq 16$. We restricted our selection of blackbox nodes to those which are not directly connected to the receiver node because this would lead to $\Pr^{\max,-} = 0$. The reason for this behavior is that the blackbox node would gain direct write access to the buffer of the target node and would be able to remove all messages from there. Furthermore we left out all combinations which would cause all paths from the sender to the receiver node to pass a blackbox node. Also in this case the computed probability would be 0.

Table II contains the results for a number of different blackbox configurations, including the configurations for which the smallest and the largest bound were obtained. We can observe that the required time for computing the probabilities is considerably smaller than for the complete network because of the smaller state space. It further decreases if a second node is put into a blackbox. The probabilities we were able to derive using our method are close to the probabilities of the complete networks. For example consider the Chord network with 16 nodes. The probability for the complete network is 0.7987. If we treat node N_1 as a blackbox module, we can derive a lower bound of 0.7654, and with N_5 as an additional blackbox of 0.7595. We can observe that the gap between the derived bound and the actual probability increases with the number blackboxes growing.

VII. CONCLUSION

In this paper we have shown how bounds on reachability probabilities can be derived from networks of Markov decision processes which are only partially available. The processes communicate via messages or via shared integer

```
mdp
const int K = 3;
const double p = 0.8;
global x0 : [0..K] init 1;
global x1 : [0..K] init 0;
global x2 : [0..K] init 0;

module node0
  [] (x0 > 0) & (x1 < K) ->
           p: (x0' = x0 - 1) & (x1' = min(x1 + 1, K))
    + (1 - p): (x0' = x0 - 1);
endmodule

module node1
  [] (x1 > 0) & ((x0 < K) | (x2 < K)) ->
           p: (x1' = x1 - 1) & (x2' = min(x2 + 1, K)) & (x0' = min(x0 + 1, K))
    + (1 - p): (x1' = x1 - 1);
endmodule
```

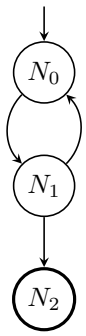

Figure 3. An example network consisting of three nodes, node_0 being the sender node and node_2 the receiver node, which is only implicitly given by its buffer x_2.

variables. Value iteration is used to derive the probability bounds. We have applied our technique to Chord networks where we allowed arbitrary behavior for a subset of the nodes, and studied lower bounds on the maximal probability of message delivery. Our experiments show that we are able to arrive at bounds close to the actual probability of message delivery in the complete network. This is both interesting and encouraging. As future work we are planning to investigate bounded model checking based on stochastic satisfiability problems (SSMT) [15]. This will have the advantage that we do not have to construct the composition and perform integer elimination prior to computing bounds. We hope that this enables us to handle significantly larger models.

REFERENCES

[1] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Int'l Conf. on Distributed Systems Platforms (Middleware)*, ser. Lecture Notes in Computer Science, R. Guerraoui, Ed., vol. 2218. Springer-Verlag, 2001, pp. 329–350.

[2] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for Internet applications," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 17–32, 2003.

[3] T. Nopper and C. Scholl, "Approximate symbolic model checking for incomplete designs," in *5th Int'l Conf. on Formal Methods in Computer-Aided Design (FMCAD)*, ser. Lecture Notes in Computer Science, A. J. Hu and A. K. Martin, Eds., vol. 3312. Springer-Verlag, 2004, pp. 290–305.

[4] T. Nopper, C. Scholl, and B. Becker, "Computation of minimal counterexamples by using black box techniques and symbolic methods," in *Int'l Conf. on Computer-Aided Design (ICCAD)*, G. G. E. Gielen, Ed. San Jose, CA, USA: IEEE Computer Society, Nov. 2007, pp. 273–280.

[5] M. Herbstritt, B. Becker, and C. Scholl, "Advanced SAT-techniques for bounded model checking of blackbox designs," in *7th Int'l Workshop on Microprocessor Test and Verification (MTV)*, M. S. Abadir, L.-C. Wang, and J. Bhadra, Eds. IEEE Computer Society, 2006, pp. 37–44.

[6] C. Miller, S. Kupferschmid, M. D. T. Lewis, and B. Becker, "Encoding techniques, Craig interpolants and bounded model checking for incomplete designs," in *13th Int'l Conf. on Theory and Applications of Satisfiability Testing (SAT)*, ser. Lecture Notes in Computer Science, O. Strichman and S. Szeider, Eds., vol. 6175. Springer-Verlag, 2010, pp. 194–208.

[7] C. Miller, K. Gitina, C. Scholl, and B. Becker, "Bounded model checking of incomplete networks of timed automata," in *11th Int'l Workshop on Microprocessor Test and Verification (MTV)*. IEEE Computer Society, 2010.

[8] M. Z. Kwiatkowska, G. Norman, D. Parker, and H. Qu, "Assume-guarantee verification for probabilistic systems," in *16th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, ser. Lecture Notes in Computer Science, J. Esparza and R. Majumdar, Eds., vol. 6015. Springer-Verlag, 2010, pp. 23–37.

[9] H. Hansson and B. Jonsson, "A logic for reasoning about time and reliability," *Formal Aspects of Computing*, vol. 6, no. 5, pp. 512–535, 1994.

[10] H. Fecher, M. Leucker, and V. Wolf, "Don't know in probabilistic systems," in *13th Int'l SPIN Workshop on Model*

Table II
RESULTS FOR CHORDS WITH BLACKBOXES (BUFFER SIZE $K = 1$)

n	BBs	$\Pr^{\max,-}$	Time [s]	BBs	$\Pr^{\max,-}$	Time [s]	BBs	$\Pr^{\max,-}$	Time [s]
11	N_1	0.7629	0.10	N_4	0.6400	0.10	N_5	0.7833	0.08
	N_1, N_5	0.7424	0.07						
12	N_1	0.7591	0.17	N_2	0.7554	0.18	N_5	0.7902	0.18
	N_6	0.7594	0.18	N_1, N_5	0.7459	0.11	N_2, N_6	0.6095	0.11
13	N_1	0.7680	0.34	N_2	0.7616	0.35	N_3	0.7909	0.34
	N_6	0.7627	0.34	N_7	0.7604	0.36	N_1, N_2	0.6400	0.20
	N_1, N_3	0.6400	0.20	N_1, N_6	0.6400	0.19	N_2, N_3	0.7130	0.24
14	N_1	0.7614	0.72	N_2	0.7569	0.71	N_3	0.7908	0.74
	N_4	0.7591	0.73	N_7	0.7664	0.72	N_8	0.7555	0.56
	N_1, N_2	0.5120	0.44	N_1, N_8	0.4017	0.41	N_2, N_3	0.7332	0.44
	N_2, N_4	0.5120	0.44	N_2, N_8	0.5120	0.33	N_3, N_4	0.6659	0.50
15	N_1	0.7615	1.53	N_2	0.7601	1.54	N_3	0.7911	1.52
	N_4	0.7601	1.59	N_5	0.7978	1.54	N_8	0.7613	1.52
	N_2, N_3	0.7109	0.98	N_2, N_4	0.4830	1.01	N_3, N_4	0.7249	1.01
	N_3, N_5	0.7697	0.98	N_4, N_8	0.4649	1.06	N_5, N_9	0.7713	1.02
16	N_1	0.7654	3.37	N_2	0.7610	3.31	N_3	0.7910	3.36
	N_4	0.7601	3.34	N_5	0.7980	3.42	N_6	0.7907	3.36
	N_1, N_3	0.6282	2.24	N_1, N_5	0.7595	2.27	N_3, N_4	0.7109	2.25
	N_3, N_5	0.7690	2.25	N_3, N_6	0.4697	2.22	N_5, N_9	0.7856	2.34
17	N_2	0.8000	6.36	N_3	0.8000	6.40	N_4	0.8000	6.39
	N_5	0.8000	6.45	N_6	0.8000	6.53	N_7	0.8000	6.49
18	N_3	0.7934	14.14	N_4	0.7680	13.84	N_5	0.7986	14.13
	N_6	0.7925	10.30	N_7	0.7986	14.14	N_8	0.7912	14.11
19	N_1	0.7932	30.57	N_4	0.7680	30.43	N_5	0.7984	30.58
	N_6	0.7987	30.84	N_7	0.7935	30.36	N_9	0.7997	30.73
20	N_1	0.7933	66.97	N_2	0.7929	66.44	N_5	0.7986	67.12
	N_6	0.7933	66.78	N_7	0.7987	67.63	N_9	0.7997	67.48

Checking Software, ser. Lecture Notes in Computer Science, A. Valmari, Ed., vol. 3925. Springer-Verlag, 2006, pp. 71–88.

[11] C. Baier and M. Z. Kwiatkowska, "Model checking for a probabilistic branching time logic with fairness," *Distributed Computing*, vol. 11, no. 3, pp. 125–155, 1998.

[12] R. E. Bellman, "A Markovian decision process," *Journal of Mathematics and Mechanics*, vol. 6, pp. 679–684, 1957.

[13] K. Chatterjee and T. A. Henzinger, "Value iteration," in *25 Years of Model Checking – History, Achievements, Perspectives*, ser. Lecture Notes in Computer Science, O. Grumberg and H. Veith, Eds., vol. 5000. Springer-Verlag, 2008, pp. 107–138.

[14] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker, "PRISM: A tool for automatic verification of probabilistic systems," in *12th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, ser. Lecture Notes in Computer Science, H. Hermanns and J. Palsberg, Eds., vol. 3920. Springer-Verlag, 2006, pp. 441–444.

[15] M. Fränzle, H. Hermanns, and T. Teige, "Stochastic satisfiability modulo theory: A novel technique for the analysis of probabilistic hybrid systems," in *11th Int'l Conf. on Hybrid Systems: Computation and Control (HSCC)*, ser. Lecture Notes in Computer Science, M. Egerstedt and B. Mishra, Eds., vol. 4981. Springer-Verlag, 2008, pp. 172–186.