Incremental Bisimulation Abstraction Refinement

LEI SONG, Max-Planck-Institut für Informatik
LIJUN ZHANG, Chinese Academy of Sciences
HOLGER HERMANNS, Saarland University
JENS CHR. GODSKESEN, IT University of Copenhagen

Abstraction refinement techniques in probabilistic model checking are prominent approaches for verification of very large or infinite-state probabilistic concurrent systems. At the core of the refinement step lies the implicit or explicit analysis of a counterexample. This article proposes an abstraction refinement approach for the probabilistic computation tree logic (PCTL), which is based on incrementally computing a sequence of may- and must-quotient automata. These are induced by depth-bounded bisimulation equivalences of increasing depth. The approach is both sound and complete, since the equivalences converge to the genuine PCTL equivalence. Experimental results with a prototype implementation show the effectiveness of the approach.

Categories and Subject Descriptors: D2.4 [**Software Engineering**]: Software/Program Verification—*Model Checking*

General Terms: Algorithms, Experimentation, Verification

Additional Key Words and Phrases: Bisimulation, CEGAR, probabilistic automata

ACM Reference Format:
Lei Song, Lijun Zhang, Holger Hermanns, and Jens Chr. Godskesen. 2014. Incremental bisimulation abstraction refinement. ACM Trans. Embedd. Comput. Syst. 13, 4s, Article 142 (July 2014), 23 pages.
DOI: http://dx.doi.org/10.1145/2627352

1. INTRODUCTION

Model checking of large or infinite-state systems has been revolutionized by the invention of counterexample guided abstraction-refinement (CEGAR) [Clarke et al. 2003]. This approach, originally tailored to software model checking, automatically abstracts and refines a given system model, until either the property of interest is found to be satisfied, or a valid counterexample is found, demonstrating that the property is not satisfied (or the checker runs out of memory, or the user runs out of patience). During this process, which starts off from a very coarse abstract model, it often happens that the property is evaluated to false on the abstract system, while it is indeed true on the original system. In this case, the CEGAR machinery will provide an abstract counterexample, which is not valid in the original model, and therefore is used to refine the abstract model. The CEGAR approach has inspired a large body of work in

---

This article is an extension version of our conference paper [Song et al. 2013b].
The work has received support from the EU FP7 projects MEALS (295261), TREsPASS (318003), and SENSATION (318490), and from the DFG Sonderforschungsbereich AVACS, as well as from the National Natural Science Foundation of China (NSFC) under grant nos. 61361136002 and 91118007.
Author's addresses: L. Song (corresponding author) and H. Hermanns, Computer Science Department, Saarland University, Germany; L. Zhang, State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences; J. C. Godskesen, IT University of Copenhagen, Denmark; corresponding author's email: song@cs.uni-sb.de.org.

DOI: http://dx.doi.org/10.1145/2627352

related fields. It has also found its way into the area of model checking of probabilistic concurrent systems.

Probabilistic automata (PAs) are a very natural model of probabilistic concurrent systems [Segala and Lynch 1995]. Extending both labeled transition systems and Markov chains, they are convenient to represent systems with nondeterminism and randomization. PAs are akin to Markov decision processes (MDPs) and form the backbone model of successful model checkers, such as PRISM [Kwiatkowska et al. 2011], enabling the analysis of randomized concurrent systems. Despite the remarkable versatility of this approach, its power is limited by the state-space explosion problem, and several abstraction-refinement approaches have been proposed to alleviate that problem [D'Argenio et al. 2001, 2002; Hermanns et al. 2008; Chadha and Viswanathan 2010; Kattenbelt et al. 2010; Wachter and Zhang 2010; de Alfaro and Roy 2007; Roy et al. 2008].

The first abstraction-refinement techniques have been proposed [D'Argenio et al. 2001] for MDPs and implemented in the tool RAPTURE. Based on a chosen partition of the state space, an abstract MDP is constructed. Heuristics for getting a better refinement have been further studied [D'Argenio et al. 2002]. Probabilistic counterexample-guided abstraction-refinement [Hermanns et al. 2008] is a natural extension of CEGAR for the PA setting. It aims at identifying the maximal probability of reaching a set of goal states in the PA. For this, an abstract quotient PA is constructed based on an initially coarse partition of the state space. The reachability probability is computed then on the abstraction. This provides a safe upper bound on the probability in the original model. If the bound is not good enough, a counterexample can be derived, usually expressed as a set of paths, which are then used to refine the abstraction. Extensions of the abstraction yielding two-player games provide both upper and lower bounds for maximal/minimal probabilities [Wachter and Zhang 2010; Kattenbelt et al. 2010; Chadha and Viswanathan 2010].

The probabilistic CEGAR approach has been successfully applied to several examples [Hermanns et al. 2008; Wachter and Zhang 2010; Kattenbelt et al. 2010] for studying reachability properties, and an extension to general PCTL properties has been developed [Chadha and Viswanathan 2010]. That approach however involves expensive simulation checking and counterexample generation. The algorithm for computing simulation runs in $\mathcal{O}(m^2n)$ time [Zhang and Hermanns 2007], where n and m are the number of states and transitions of the MDP, respectively. Moreover, Chadha and Viswanathan [2010, Theorem 3.11] show that the problem of finding the smallest counterexample is NP-hard, and it is also unlikely to be efficiently approximable.

This article proposes a radically different approach to abstraction-refinement of probabilistic concurrent systems. The approach is compatible with and applicable to the full logic PCTL, and it works without counterexamples. Instead, the refinement is based on a sequence of incrementally computed bisimulations. We call the technique *Probabilistic Incremental Bisimulation Abstraction Refinement* (PIBAR).

The approach is rooted in the fact that probabilistic bisimulation equivalence [Segala 1995] is strictly finer than the equivalence induced by PCTL. Our approach turns this disturbing difference into an algorithmic idea. We harvest a recent characterization of PCTL equivalence as the limit of a sequence of step-indexed bisimulation relations [Song et al. 2013a] and combine this theoretical insight with an effective computational procedure. With some inspiration from modal transition systems [Larsen 1990], we use probabilistic may- and must-quotient automata to represent the behaviour of an abstract system. Intuitively, we work with a sequence of may- and must-quotient automata induced by the sequence of step-indexed bisimulation relations, which guarantees convergence to PCTL equivalence. But since the computation of the step-indexed relations is in general NP-complete, we define, for each step-indexed

relation, a sequence of relations, some of which can be computed in polynomial time. Moreover they converge to the step-indexed relation eventually. Still, the may- and must-abstractions we use are guaranteed to provide upper and lower bounds for maximal and minimal probabilities, respectively. In case that this interval is too large, the abstraction is refined by recomputing it for a finer bisimulation. In the recomputation, we reuse intermediate results from the previous iterations. This way, the PIBAR approach—just like the probabilistic CEGAR approaches—automatically abstracts and refines a given PA model until either the property of interest is found to be satisfied or the property is found not to be satisfied (or the checker runs out of memory, or the user runs out of patience). But it does so without resorting to any kind of counterexample analysis. The approach is complete in the sense that for finite-state systems, it terminates after a finite number of refinement steps. PIBAR is not restricted to reachability properties. As we will demonstrate, it instead works for the safety fragment [Baier et al. 2005; Chadha and Viswanathan 2010] of PCTL (where negation only appears at atomic propositions and probabilities appear lower-bounded only) and can be twisted to work with full PCTL by delaying the check until refinement has terminated, that is, PCTL-equivalence is obtained.

Experimental results carried out with a prototypical implementation of the PIBAR approach demonstrate its effectiveness. We compare with PRISM [Kwiatkowska et al. 2011] on a selection of case studies and report promising results. For several case studies, we obtain a sufficiently good bisimulation abstraction efficiently such that we can perform model checking on the may- and must-quotient automata.

Contributions. Our contributions in this article are as follows.

—We propose a novel framework of probabilistic bisimulation guided abstraction, avoiding the need to analyze counterexamples.
—Our algorithm works for the entirety of PCTL and is both sound and complete. This means it will always terminate and return the correct answer in an ideal setting with unlimited memory and time.
—We refine the abstract system by computing a sequence of step-depth indexed bisimulations.
—We propose a novel way to conduct property-driven abstraction and refinement.
—We report on a prototypical implementation of PIBAR and demonstrate that the approach can accelerate probabilistic model checking in many cases.

Organization of the Article. Section 2 recalls some notations, and in Section 3, we recall the definition of strong i-depth bisimulation. We propose may-quotient and must-quotient of a probabilistic system with respect to an equivalence relation in Section 4. We describe in detail how PIBAR works in Section 5. The experimental results are discussed in Section 6. Related work is discussed in Section 7, while Section 8 concludes.

2. PRELIMINARIES

We first introduce some notations which we will use throughout this article.

For a finite set S, a distribution is a function $\mu : S \to [0,1]$ satisfying $|\mu| := \sum_{s\in S} \mu(s) = 1$. We denote by $Dist(S)$ the set of distributions over S. We shall use $s, r, t, \ldots$ and $\mu, \nu \ldots$ to range over S and $Dist(S)$, respectively. The support of μ is defined by $supp(\mu) := \{s \in S \mid \mu(s) > 0\}$. A distribution μ is called *Dirac* if $|supp(\mu)| = 1$, and we let $\mathcal{D}_s$ denote the Dirac distribution with $\mathcal{D}_s(s) = 1$. For a distribution μ we also write it as $\{\mu(s) : s \mid s \in supp(\mu)\}$.

Given an equivalence relation $\mathcal{R}$ on S, let $[s]_\mathcal{R}$ denote the equivalence class $C \in S/\mathcal{R}$ such that $s \in C$, and $[\mu]_\mathcal{R}$ is the distribution over $S/\mathcal{R}$ such that $[\mu]_\mathcal{R}(C) = \sum_{s\in C} \mu(s)$ for each $C \in S/\mathcal{R}$.

Next we define the downward closure of a subset of states.

Definition 2.1 (*Downward Closure*). For a relation $\mathcal{R}$ over S and $C \subseteq S$, define

$$\mathcal{R}^{\downarrow}(C) = \{s' \mid \exists s \in C.s' \ \mathcal{R} \ s\}.$$

We say C is $\mathcal{R}$ *downward closed* it and only if $C = \mathcal{R}^{\downarrow}(C)$.

We shall use $\mathcal{R}^{\downarrow}(s)$ as the shorthand of $\mathcal{R}^{\downarrow}(\{s\})$, and let $\mathcal{R}^{\downarrow} = \{\mathcal{R}^{\downarrow}(C) \mid C \subseteq S\}$ denote the set of all $\mathcal{R}$ downward closed sets.

Given a relation $\mathcal{R}$, let $\equiv_{\mathcal{R}}$ be the largest equivalence relation contained in $\mathcal{R}$. The following lemma from Hermanns et al. [2011] shows that each $\mathcal{R}$ downward closed set can be seen as a union of equivalence classes of $\equiv_{\mathcal{R}}$.

LEMMA 2.2 (LEMMA 5.1 HERMANNS ET AL. [2011]). *Let $\mathcal{R} \subseteq S \times S$ and let $C \subseteq S$ be a $\mathcal{R}$ downward closed set, then C is a union of equivalence classes of $\equiv_{\mathcal{R}}$.*

2.1. Probabilistic Automata

We recall the notion of probabilistic automata, as coined by Segala [1995].

Definition 2.3 (*Probabilistic Automata*). A *probabilistic automaton* is a tuple $\mathcal{P} = (S, \rightarrow, s_0, \mathsf{AP}, L)$, where the following hold.

—S is a finite set of states.
—$\rightarrow \ \subseteq S \times \mathit{Dist}(S)$ is a finite set of transition relation.
—$s_0 \in S$ is the initial state.
—AP is a set of atomic propositions.
—$L : S \rightarrow 2^{\mathsf{AP}}$ is a labeling function.

A transition $(s, \mu) \in \rightarrow$ is denoted by $s \rightarrow \mu$. A *path* is a finite or infinite sequence $\omega = s_0 s_1 s_2 \ldots$ of states, such that for each $i \geq 0$ (except for the last state of a finite path), there exists a distribution μ with $s_i \rightarrow \mu$ and $\mu(s_{i+1}) > 0$. We introduce some notations as follows.

—$\mathit{lstate}(\omega)$, the last state of ω, provided ω is finite.
—$\omega[i] = s_i$ with $i \geq 0$, the $(i+1)$-th state on ω.
—$\omega|^i = s_0 s_1 \ldots s_i$, the fragment of ω ending at state $\omega[i]$.
—$\omega|_i = s_i s_{i+1} \ldots$, the fragment of ω starting from state $\omega[i]$.

Let $\mathit{Path}^{\omega}(\mathcal{P})$ and $\mathit{Path}^{*}(\mathcal{P})$ denote the sets containing all infinite and finite paths of $\mathcal{P}$, respectively. Let $\mathit{Path}(\mathcal{P}) = \mathit{Path}^{\omega}(\mathcal{P}) \cup \mathit{Path}^{*}(\mathcal{P})$. In case $\mathcal{P}$ is clear from the context, we simply omit it. We also let $\mathit{Path}(s_0)$ be the set containing all paths starting from s_0, similarly for $\mathit{Path}^{*}(s_0)$ and $\mathit{Path}^{\omega}(s_0)$.

Due to nondeterministic choices in PAs, a probability measure cannot be defined directly. As usual, we shall introduce the definition of *schedulers* to resolve the nondeterminism. Intuitively, a scheduler will decide which transition to choose at each step, based on the history execution. Formally, a scheduler is a function $\sigma : \mathit{Path}^{*} \rightarrow \mathit{Dist}(\rightarrow)$ such that $\sigma(\omega)(s, \mu) > 0$ implies $s = \mathit{lstate}(\omega)$ and $s \rightarrow \mu$. A scheduler σ is *deterministic* if it returns only Dirac distributions, that is, $\sigma(\omega)(s, \mu) = 1$ for some s and μ.

The *cone* of a finite path ω, denoted C_{ω}, is the set of infinite paths having ω as their prefix, that is, $C_{\omega} = \{\omega' \in \mathit{Path}^{\omega} \mid \omega \leq \omega'\}$, where $\omega \leq \omega'$ if and only if ω is a prefix of ω'. Fixing a starting state s and a scheduler σ, the measure $\mathit{Prob}_{\sigma,s}$ of a cone C_{ω}, where $\omega = s_0 s_1 \ldots s_k$, is defined inductively as $\mathit{Prob}_{\sigma,s}(C_{\omega}) = 0$ if $s \neq s_0$, $\mathit{Prob}_{\sigma,s}(C_{\omega}) = 1$ if $s = s_0$

and $k = 0$, otherwise,

$$Prob_{\sigma,s_0}(C_\omega) = Prob_{\sigma,s_0}(C_{\omega|^{k-1}}) \cdot \left(\sum_{(s_{k-1},\mu') \in \rightarrow} \sigma(\omega|^{k-1})(s_{k-1}, \mu') \cdot \mu'(s_k) \right).$$

Let $\mathcal{B}$ be the smallest algebra that contains all the cones and is closed under complement and countable unions. By standard measure theory [Halmos 1974; Rudin 2006], this algebra is a σ-*algebra*, and all its elements are measurable sets of paths. Moreover, $Prob_{\sigma,s_0}$ can be extended to a unique measure on $\mathcal{B}$.

2.2. PCTL

We recall the syntax of PCTL [Hansson and Jonsson 1994] which is a probabilistic extension of CTL [Clarke et al. 1986]. Over the set AP of atomic propositions, PCTL is formed according to the following grammar.

$$\Phi ::= a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid P_{\bowtie q}(\varphi)$$
$$\varphi ::= \mathsf{X}\,\Phi \mid \Phi_1\,\mathsf{U}\,\Phi_2 \mid \Phi_1\,\mathsf{U}^{\leq n}\,\Phi_2,$$

where $a \in \mathsf{AP}$, $\bowtie \in \{<, >, \leq, \geq\}$, $q \in [0, 1]$, and n is a positive integer. We write *true* as an abbreviation for $a \vee \neg a$ for some $a \in \mathsf{AP}$. We refer to Φ and φ as PCTL state and path formulae, respectively.

The satisfaction relation $s \models \Phi$ for state formulae is defined in a standard manner for boolean formulae. For the probabilistic operator, it is defined by

$$s \models P_{\bowtie q}(\varphi) \text{ iff } \forall\sigma. Prob_{\sigma,s}(\{\omega \in Path(s) \mid \omega \models \varphi\}) \bowtie q,$$

namely, $s \models P_{\bowtie q}(\varphi)$ if and only if whichever scheduler we choose, the probability of paths starting from s and satisfying φ always meets the given bound $\bowtie q$. The satisfaction relation $\omega \models \varphi$ for path formulae is defined in the same manner as in CTL.

$$\omega \models \mathsf{X}\,\Phi \text{ iff } \omega[1] \models \Phi,$$
$$\omega \models \Phi_1\,\mathsf{U}\,\Phi_2 \text{ iff } \exists j \geq 0.\ \omega[j] \models \Phi_2 \wedge \forall 0 \leq k < j.\ \omega[k] \models \Phi_1,$$
$$\omega \models \Phi_1\,\mathsf{U}^{\leq n}\,\Phi_2 \text{ iff } \exists 0 \leq j \leq n.\ \omega[j] \models \Phi_2 \wedge \forall 0 \leq k < j.\ \omega[k] \models \Phi_1.$$

In this article, we are especially interested in the safety fragment of PCTL, that is, safety PCTL [Baier et al. 2005], denoted PCTL_{safe}, whose syntax is given as follows.

$$\Phi ::= a \mid \neg a \mid \Phi_1 \wedge \Phi_2 \mid \Phi_1 \vee \Phi_2 \mid P_{\geq q}(\varphi) \mid P_{>q}(\varphi)$$
$$\varphi ::= \mathsf{X}\,\Phi \mid \Phi_1\,\mathsf{U}\,\Phi_2 \mid \Phi_1\,\mathsf{U}^{\leq n}\,\Phi_2$$

In PCTL_{safe}, only $\geq$ (or $>$) is allowed in the probabilistic operator, and negation only appears in atomic propositions. In the sequel, let PCTL^i denote the subset of PCTL in which the path formula is restricted to $\varphi ::= \mathsf{X}\,\Phi \mid \Phi_1\,\mathsf{U}^{\leq j}\,\Phi_2$, where $j \leq i$, similarly for PCTL^i_{safe}. In other words, PCTL^i only specifies (conditional) reachability probability up to i steps. Moreover for a given logic $\mathcal{L}$, we write $s \preceq_{\mathcal{L}} r$ iff $r \models \Phi$ implies $s \models \Phi$ for any state formula Φ of $\mathcal{L}$. Intuitively, all properties in $\mathcal{L}$ are considered safe, which cannot be violated; r represents a specification which defines all possible safe behaviors that a system is allowed to do. If s represents a concrete system implementing r, then its behaviors should be subsumed by the behaviors of r. Thus in case r satisfies a safety property Φ, s must also satisfy Φ. We write $s \equiv_{\mathcal{L}} r$ iff $s \preceq_{\mathcal{L}} r$ and $r \preceq_{\mathcal{L}} s$.

2.3. Bisimulation and Simulation

In this section, we introduce the definitions of bisimulation and simulation for PAs. In order to do so, we shall introduce the *weight functions* [Jonsson and Larsen 1991]

Definition 2.4 (*Weight Function*). Let $\mathcal{R}$ be a relation over S. A weight function for μ and ν with respect to $\mathcal{R}$ is a function $\Delta : S \times S \mapsto [0, 1]$ such that the following hold.

(1) $\Delta(s, r) > 0$ implies that $s \mathcal{R} r$.
(2) $\mu(s) = \sum_{r \in S} \Delta(s, r)$ for any $s \in S$.
(3) $\nu(r) = \sum_{s \in S} \Delta(s, r)$ for any $r \in S$.

We write $\mu \sqsubseteq_R \nu$ if and only if there exists a weight function for μ and ν with respect to $\mathcal{R}$.

Strong simulation and bisimulation for PAs are defined as follows [Segala 1995]. Since we are only interested in strong relations throughout this article, we take the liberty to drop the prefix 'strong'. All relations considered throughout this article are strong relations.

Definition 2.5 (*Strong Simulation and Bisimulation*). A relation $\mathcal{R} \subseteq S \times S$ is a *simulation* relation if and only if $s \mathcal{R} r$ implies that $L(s) = L(r)$ and for each $s \to \mu$, there exists $r \to \nu$ such that $\mu \sqsubseteq_R \nu$.

If a simulation relation $\mathcal{R}$ is symmetric, then $\mathcal{R}$ is a *bisimulation* relation. We write $s \precsim r$ ($s \sim r$) whenever there is a (bi)simulation relation $\mathcal{R}$ such that $s \mathcal{R} r$.

Segala [1995] showed that $\precsim$ and $\sim$ are sound (but not complete) for PCTL and PCTL_{safe}, respectively.

LEMMA 2.6 ([SEGALA 1995]). *$\sim \subsetneq \equiv_{\text{PCTL}}$ and $\precsim \subsetneq \preceq_{\text{PCTL}_{safe}}$.*

According to Lemma 2.6, bisimulation preserves PCTL equivalence, that is, bisimilar states satisfy the same PCTL formulae (soundness), but the other direction (completeness) does not hold, that is, states satisfying the same PCTL formulae are not necessarily bisimilar.

2.4. Sound and Complete Bisimulation for PCTL

The inclusion in Lemma 2.6 is strict. In this section, we will recall a variation of bisimulation which is both sound and complete for PCTL equivalence [Song et al. 2013a]. As in Song et al. [2013a], we are restricted to deterministic schedulers, which suffices for PCTL formulae [Baier and Katoen 2008, Remark 10.99].

Let $Prob_{\sigma,s}(C, C', n, \omega)$ denote the probability from s to states in C', via states in C possibly, in at most n steps under deterministic scheduler σ, where ω is used as a parameter of σ to keep track of the path. Formally,

$$Prob_{\sigma,s}(C, C', n, \omega) = \begin{cases} 1, & s \in C', \\ \sum\limits_{r \in supp(\mu')} \mu'(r) \cdot Prob_{\sigma,r}(C, C', n-1, \omega r), & n > 0 \wedge (s \in C \setminus C'), \\ 0, & \text{otherwise}, \end{cases}$$

where $\sigma(\omega)(s, \mu') = 1$ for some μ', that is, the transition $s \to \mu'$ is chosen with probability 1 by the scheduler σ given the history ω. In the case that $s \in C'$, the target states C' are already reached, so $Prob_{\sigma,s}(C, C', n, \omega) = 1$. In the case that neither $s \in C'$ nor $s \in C$, $Prob_{\sigma,s}(C, C', n, \omega) = 0$. Since it is not possible to reach states in C' only via states in C. Otherwise, we shall move one step further under the guidance of the scheduler σ, provided $n > 0$. Then $Prob_{\sigma,s}(C, C', n, \omega)$ is equal to the weighted sum of the probability

of successors of s to reach states in C' via only states in C in at most $n-1$ steps, that is, $Prob_{\sigma,r}(C, C', n-1, \omega r)$, where r is a successor of s under the given scheduler σ. Moreover, ω is updated to keep track of the visited path.

We are now ready to introduce an indexed family of i-depth (bi)simulations. We let $s \sim_0 r$ and $s \precsim_0 r$ if and only if $L(s) = L(r)$.

Definition 2.7 (*i-Depth Simulation and Bisimulation*). A relation $\mathcal{R} \subseteq S \times S$ is an i-depth simulation with $i \geq 1$ if $s \mathcal{R} r$ implies $s \precsim_{i-1} r$ and for any $\mathcal{R}$ downward closed sets C, C' and scheduler σ, there exists a scheduler σ' such that

$$Prob_{\sigma',r}(C, C', i, r) \leq Prob_{\sigma,s}(C, C', i, s). \tag{1}$$

If a simulation relation $\mathcal{R}$ is symmetric, then $\mathcal{R}$ is an i-depth bisimulation. We write $s \precsim_i r$ ($s \sim_i r$) whenever there is an i-depth (bi)simulation $\mathcal{R}$ such that $s \mathcal{R} r$.

In comparison with Definition 2.5, this definition does not require the matching of distributions out of s and r. The essential difference to the standard definition is that we only consider conditional reachability probabilities up to i steps. In the definition of PCTL_{safe}, we only have probability formulas in form of $P_{>q}(\varphi)$ or $P_{\geq q}(\varphi)$, that is, PCTL_{safe} is characterized by the minimal probabilities of satisfying some φ. A state s is considered "safer" than r (whenever r satisfies some safety properties, so is s), whenever the minimal probability of s satisfying φ is greater than r. This justifies the $\leq$ operator in Eq. (1).

The following lemma establishes some properties of i-depth (bi)simulation.

LEMMA 2.8 ([SONG ET AL. 2013A]).

(*1*) $\sim_i$ *is an equivalence relation, and* $\precsim_i$ *is a pre-order for each* $i \geq 0$.
(*2*) $\sim_i \subseteq \sim_j$ *provided* $i \geq j$.
(*3*) $\sim_i = \equiv_{\mathsf{PCTL}^i}$ *and* $\precsim_i = \preceq_{\mathsf{PCTL}^i_{safe}}$ *for each* $i \geq 0$.
(*4*) *For any finite* PA, *there exists* $i \geq 0$ *such that* $\sim_i = \equiv_{\mathsf{PCTL}}$ *and* $\precsim_i = \preceq_{\mathsf{PCTL}_{safe}}$.

Clause 2 says that we will obtain a finer bisimulation by increasing i. Clause 3 assures that i-depth bisimulation is both sound and complete for PCTL^i equivalence, similarly for i-depth simulation with respect to PCTL^i_{safe}. More importantly, Clause 4 assures that the i-depth bisimulations define a sequence of relations which equals PCTL equivalence eventually.

3. A TWO-DIMENSIONAL BISIMULATION GRID

Lemma 2.8 shows that by increasing i we can obtain a sequence of bisimulation relations converging to the PCTL-equivalence. In this section, we refine each bisimulation $\sim_i$ further to a subsequence of equivalence relations. Our PIBAR framework is based on the induced two-dimensional grid of relations. We first define the size of a downward closed set as follows.

Definition 3.1 (*Sizes of Downward Closed Sets*). Let $\mathcal{R} \subseteq S \times S$ be a relation, and $C \subseteq S$ a $\mathcal{R}$ downward closed set. The size of C, denoted $size(C)$, is defined as the number of equivalence classes of $\equiv_{\mathcal{R}}$ in C, that is, $size(C) = |\{C' \in S/\equiv_{\mathcal{R}} | \ C' \subseteq C\}|$.

The concept of size will now be incorporated into a refined bisimulation definition. Let $s \precsim_{0,j} r$ and $s \sim_{0,j} r$ if and only if $L(s) = L(r)$ for any $j \geq 0$. Moreover, we have the following definition.

Definition 3.2 ((i, j)-*Depth Simulation and Bisimulation*). A relation $\mathcal{R} \subseteq S \times S$ is an (i, j)-depth simulation with $i, j \geq 1$ if $s \mathcal{R} r$ implies $s \precsim_{i-1,j} r$ and for any scheduler

$$
\begin{array}{ccccccccccc}
\sim_{1,1} & \supseteq & \sim_{1,2} & \supseteq & \sim_{1,3} & \supseteq & \cdots & \supseteq & \sim_{1} & & \\
\cap\!\cup & & \cap\!\cup & & \cap\!\cup & & & & \cap\!\cup & & \\
\sim_{2,1} & \supseteq & \sim_{2,2} & \supseteq & \sim_{2,3} & \supseteq & \cdots & \supseteq & \sim_{2} & & \\
\cap\!\cup & & \cap\!\cup & & \cap\!\cup & & & & \cap\!\cup & & \\
\sim_{3,1} & \supseteq & \sim_{3,2} & \supseteq & \sim_{3,3} & \supseteq & \cdots & \supseteq & \sim_{3} & & \\
\cap\!\cup & & \cap\!\cup & & \cap\!\cup & & & & \cap\!\cup & & \\
\vdots & & \vdots & & \vdots & & \ddots & & \vdots & & \\
\cap\!\cup & & \cap\!\cup & & \cap\!\cup & & & & \cap\!\cup & & \\
\sim_{n,1} & \supseteq & \sim_{n,2} & \supseteq & \sim_{n,3} & \supseteq & \cdots & \supseteq & \sim_{n} & = & \equiv_{\mathsf{PCTL}}
\end{array}
$$

Fig. 1. Inclusion among the bisimulations considered.

σ and $\mathcal{R}$ downward closed sets C, C' with $size(C), size(C') \leq j$, there exists a scheduler σ' such that

$$Prob_{\sigma',r}(C, C', i, r) \leq Prob_{\sigma,s}(C, C', i, s).$$

If a simulation relation $\mathcal{R}$ is symmetric, then $\mathcal{R}$ is an (i, j)-depth bisimulation. We write $s \precsim_{i,j} r$ ($s \sim_{i,j} r$) whenever there is an (i, j)-depth (bi)simulation $\mathcal{R}$ such that $s \mathcal{R} r$.

Intuitively $\sim_{i,j}$ is almost the same as $\sim_i$ except that only downward closed sets with size not greater than j are considered. $\sim_{i,j}$ has the following properties.

LEMMA 3.3.

(1) $\sim_{i,j}$ *is an equivalence relation and* $\precsim_{i,j}$ *is a pre-order for each* $i, j \geq 0$.
(2) $\sim_{j,i} \subseteq \sim_{k,i}$ *and* $\precsim_{j,i} \subseteq \precsim_{k,i}$ *for any* $i, j, k \geq 0$ *provided* $j \geq k$.
(3) $\sim_{i,j} \subseteq \sim_{i,k}$ *and* $\precsim_{i,j} \subseteq \precsim_{i,k}$ *for any* $i, j, k \geq 0$ *provided* $j \geq k$.
(4) For any finite PA*, there exists* $j \geq 0$ *such that* $\sim_{i,j} = \sim_i$ *and* $\precsim_{i,j} = \precsim_i$.

PROOF.

(1) Clauses 1 and 2 are straightforward by Definition 2.7 and 3.2, respectively.
(2) Let $\mathcal{R} = \{(s, r) \mid s \precsim_{i,j} r\}$, we now show that $\mathcal{R}$ is an (i, k)-depth simulation provided that $j \geq k$. For any $\mathcal{R}$ downward closed set C, if $size(C) \leq k$, then we also have $size(C) \leq j$, hence the following proof is straightforward.
(3) We consider PAs with finite state spaces. In the worst case, each state belongs to a distinct equivalence class. Therefore there always exists $j \geq 0$ such that $\sim_{i,j} = \sim_i$ and $\precsim_{i,j} = \precsim_i$. □

We have defined a sequence of (i, j)-depth bisimulations which will converge to $\sim_i$ after a finite number of iterations. The inclusion hierarchy of all the bisimulations are summarized in Figure 1 where the lower-right corner indicates that there exists $n \geq 0$ such that $\sim_n = \equiv_{\mathsf{PCTL}}$. Obviously, all relations are coarser than it.

4. BISIMULATION QUOTIENT

Quotient constructions shall be the keys in the PIBAR approach. Inspired by modal transition systems, we define the may- and must-quotient systems as follows.

Definition 4.1 (*May- and Must-Quotient*). Given a PA $\mathcal{P} = (S, \rightarrow, s_0, \mathsf{AP}, L)$ and an equivalence relation $\mathcal{R}$ over S such that $\mathcal{R} \subseteq \{(s, r) \mid L(s) = L(r)\}$, the *may-quotient* and

must-quotient of $\mathcal{P}$ with respect to $\mathcal{R}$ are defined as

$$\mathcal{P}^{\diamond}_{\mathcal{R}} = (S/\mathcal{R}, \rightarrow_{\diamond}, [s_0]_{\mathcal{R}}, \mathsf{AP}, L),$$

$$\mathcal{P}^{\Box}_{\mathcal{R}} = (S/\mathcal{R}, \rightarrow_{\Box}, [s_0]_{\mathcal{R}}, \mathsf{AP}, L),$$

respectively, where the following hold.

—L is overloaded for simplicity such that $L([s]_{\mathcal{R}}) = L(s)$ for each s.
—$[s]_{\mathcal{R}} \rightarrow_{\diamond} [\mu]_{\mathcal{R}}$ if and only if there exists $r \in [s]_{\mathcal{R}}$ such that $r \rightarrow \mu$.
—$[s]_{\mathcal{R}} \rightarrow_{\Box} [\mu]_{\mathcal{R}}$ if and only if for all $r \in [s]_{\mathcal{R}}$ there exists $r \rightarrow \nu$ such that $[\mu]_{\mathcal{R}} = [\nu]_{\mathcal{R}}$.

The only difference between may- and must-quotients is the definition of transitions of the quotient systems. In the definition of may-quotient, we let the transitions of $[s]_{\mathcal{R}}$ be the union of transitions for each $r \in [s]_{\mathcal{R}}$. On the other hand, in the definition of must-quotient, we let the transitions of $[s]_{\mathcal{R}}$ be the joint transitions for all $r \in [s]_{\mathcal{R}}$.

Given two PAs $\mathcal{P}$ and $\mathcal{P}'$ with initial states of s_0 and s'_0, respectively, the simulation relations can be lifted to the automata level: $\mathcal{P} \precsim \mathcal{P}'$ if and only if $s_0 \precsim s'_0$ in the direct sum obtained from $\mathcal{P}$ and $\mathcal{P}'$. The following theorem shows the relation between the quotients and their original system.

THEOREM 4.2. *Let $\mathcal{P} = (S, \rightarrow, s_0, \mathsf{AP}, L)$ be a* PA *and $\mathcal{R}$ be an equivalence relation over S, we have $\mathcal{P}^{\Box}_{\mathcal{R}} \precsim \mathcal{P} \precsim \mathcal{P}^{\diamond}_{\mathcal{R}}$.*

PROOF. We only prove that $\mathcal{P} \precsim \mathcal{P}^{\diamond}_{\mathcal{R}}$, that is, $s_0 \precsim [s_0]_{\mathcal{R}}$; the other part can be proven in a similar way. Let $\mathcal{R} = \{(s, [r]_{\mathcal{R}}) \mid s \in [r]_{\mathcal{R}}\}$. Since $(s_0, [s_0]_{\mathcal{R}}) \in \mathcal{R}$, it suffices to show that $\mathcal{R}$ is a simulation according to Definition 2.5. Suppose that $s \rightarrow \mu$, we need to show that there exists $[r]_{\mathcal{R}} \rightarrow \nu$ such that $\mu \sqsubseteq_R \nu$.

We know that for each $s \in [r]_{\mathcal{R}}$, $s \rightarrow \mu$ implies $[r]_{\mathcal{R}} \rightarrow [\mu]_{\mathcal{R}}$ according to Definition 4.1. Let Δ be a function such that $\Delta(s', [s']_{\mathcal{R}}) = \mu(s')$ for each s', we need to show that Δ is a valid weight function between μ and $[\mu]_{\mathcal{R}}$. Since $(s', [s']_{\mathcal{R}}) \in \mathcal{R}$ according to the definition of $\mathcal{R}$, condition (1) in Definition 2.4 is satisfied. Moreover,

$$\mu(s') = \Delta(s', [s']_{\mathcal{R}}) = \sum_{[t]_{\mathcal{R}} \in S/\mathcal{R}} \Delta(s', [t]_{\mathcal{R}}), \textit{and}$$

$$[\mu]_{\mathcal{R}}([s']_{\mathcal{R}}) = \sum_{t \in [s']_{\mathcal{R}}} \mu(t) = \sum_{t \in [s']_{\mathcal{R}}} \Delta(t, [s']_{\mathcal{R}}) = \sum_{t \in S} \Delta(t, [s']_{\mathcal{R}}),$$

hence conditions (2) and (3) are also satisfied. Consequently, $\mu \sqsubseteq_R [\mu]_{\mathcal{R}}$, which completes the proof. □

5. PROBABILISTIC INCREMENTAL BISIMULATION ABSTRACTION REFINEMENT

In this section, we propose an abstraction refinement framework based on our notions of incremental bisimulation relations. Further, we also discuss a property driven extension of our framework.

5.1. Probabilistic Incremental Bisimulation Abstraction Refinement

Given a PA and an equivalence relation $\mathcal{R}$ over its state space, we can first construct the may- and must-quotients according to Definition 4.1. Then the verification of a PCTL_{safe} formula Φ can be done on the quotient systems. By Lemma 2.6 and Theorem 4.2, if $\mathcal{P}^{\diamond}_{\mathcal{R}} \models \Phi$, we know that $\mathcal{P} \models \Phi$, since $\mathcal{P} \precsim \mathcal{P}^{\diamond}_{\mathcal{R}}$, and the verification can be terminated. On the other hand, if $\mathcal{P}^{\Box}_{\mathcal{R}} \not\models \Phi$, we have $\mathcal{P} \not\models \Phi$ since, $\mathcal{P}^{\Box}_{\mathcal{R}} \precsim \mathcal{P}$. If neither $\mathcal{P}^{\diamond}_{\mathcal{R}} \models \Phi$ nor $\mathcal{P}^{\Box}_{\mathcal{R}} \not\models \Phi$ holds, this means that the current abstraction is too coarse and needs to be further refined. Lemmas 2.8 and 3.3 indicate that $\sim_{i,j}$ defines a grid of bisimulations

ALGORITHM 1: The Algorithm of PIBAR

Input: A PA $\mathcal{P}$ and a property Φ of PCTL_{safe}
Output: *True* if $\mathcal{P} \models \Phi$, else *False*
// Initialize $\mathcal{R}$ to be $\sim_0$, i.e., $(s,r) \in \mathcal{R}$ iff $L(s) = L(r)$
$i \leftarrow 0$;
$\mathcal{R} \leftarrow \{(s,r) \mid L(s) = L(r)\}$;
while *(True)* **do**
  // Once $\sim_i$ is computed, increase i by 1 and reset j to 0;
  $i \leftarrow i+1, j \leftarrow 0$;
  // Increase j until it is equal to the number
  // of equivalence classes in j, in which case $\mathcal{R} = \sim_i$;
  while *($j \leq |S/\mathcal{R}|$)* **do**
    // Compute $\sim_{i,j}$ by calling *Refine* in Algorithm 2;
    $\mathcal{R} = Refine(\mathcal{P}, \mathcal{R}, i, j)$;
    $j \leftarrow j+1$;
    $\mathcal{P}^{\diamond}_{\mathcal{R}} = May(\mathcal{P}, \mathcal{R})$;
    $\mathcal{P}^{\Box}_{\mathcal{R}} = Must(\mathcal{P}, \mathcal{R})$;
    if *($\mathcal{P}^{\diamond}_{\mathcal{R}} \models \Phi$)* **then**
      return *True*;
    end
    if *($\mathcal{P}^{\Box}_{\mathcal{R}} \not\models \Phi$)* **then**
      return *False*;
    end
  end
end

ALGORITHM 2: The Algorithm of Refine

Input: A PA $\mathcal{P}$, a relation $\mathcal{R}$, i, and j;
Output: A refined relation $\mathcal{R}$ equal to $\sim_{i,j}$;
// Initialize the set of splitter.
splitters $\leftarrow$ {all (C_1, C_2) satisfying Eq. (2)};
while (*splitters* $\neq \emptyset$) **do**
  // Get a splitter and remove it from the splitter set
  $(C_1, C_2) \leftarrow splitters.GetAndRemoveFirst()$;
  forall the $C \in S/\mathcal{R}$ **do**
    Compute $Prob_{min}(s, C_1, C_2, i)$ for each $s \in C$;
    Partition C according to the value of $Prob_{min}(s, C_1, C_2, i)$ such that only states with the same value are in the same subset;
  end
  Add new generated splitters to *splitters*;
end

which will converge to PCTL-equivalence after a finite number of steps. This gives us a straightforward way for refinement: The refinement is simply done by increasing j until $\sim_i$ is reached, then i will increased by 1, and j is reset to 0. In other words, we walk through the grid in Figure 1 in a horizontally-first manner. The thus resulting algorithm is shown in Algorithm 1, where the refinement process starts with the coarsest relation $\mathcal{R} = \sim_{0,0} = \{(s,r) \mid L(s) = L(r)\}$. Algorithm 1 will for sure terminate, because there exists an integer n such that $\sim_n \; = \; \equiv_{\mathsf{PCTL}}$ for any PA.

The algorithm for refining $\mathcal{R}$ is shown in Algorithm 2, where *splitters* are used to store all the splitters. A splitter is a pair of $\mathcal{R}$ downward closed sets (C_1, C_2). Before computing $\sim_{i,j}$, we have computed $\sim_{i,j-1}$, that is, all the splitters (C_1, C_2) such that

$size(C_1) < j$ and $size(C_2) < j$ have been considered. Therefore, only splitters (C_1, C_2) satisfying

$$(size(C_1) = j \wedge size(C_2) \leq j)) \vee (size(C_2) = j \wedge size(C_1) \leq j) \tag{2}$$

are taken into account. The refinement is done as follows: For each $C \in S/\mathcal{R}$, we first compute the minimal probability of each state $s \in C$ reaching states in C_2 only via states in C_1 in at most i steps, that is,

$$Prob_{min}(s, C_1, C_2, i) = \inf_{\sigma} Prob_{\sigma,s}(C_1, C_2, i, s),$$

with σ ranging over all schedulers of s. We then partition C into several disjoint subsets such that $s, r \in C$ are in the same subset if and only if $Prob_{min}(s, C_1, C_2, i) = Prob_{min}(r, C_1, C_2, i)$. After refining $\mathcal{R}$, we will incorporate some new blocks into the current partition. Therefore we need to update the set of splitters by adding pairs satisfying Eq. (2) and containing new equivalence classes.

The following theorem shows that Algorithm 1 is both sound and complete in the sense that it will always terminate and give the right answer. In the worst case, it will terminate when the PCTL-equivalent relation is obtained, that is, $\mathcal{R} = \equiv_{\mathsf{PCTL}}$.

THEOREM 5.1. *Algorithm 1 is sound and complete.*

PROOF. We first remark that once Algorithm 2 terminates, we are sure that $\mathcal{R} = \sim_{i,j}$, since all possible splitters have been processed. Termination is guaranteed because the inner loop of Algorithm 1 keeps increasing j until $\sim_i$ is obtained. Lemma 3.3 then implies that the inner loop always terminates. Upon obtaining $\sim_i$, parameter i is incremented in the outer loop, and the inner loop is starts over. The termination of the outer loop is implied by Lemma 2.8. Since in the worst case $\mathcal{R}$ will equal $\equiv_{\mathsf{PCTL}}$, it is assured that either $\mathcal{P}^{\diamond}_{\mathcal{R}} \models \Phi$ or $\mathcal{P}^{\Box}_{\mathcal{R}} \not\models \Phi$ holds. □

The theoretical-time complexity of Algorithm 1 is high because every potential splitter might need to be processed in each iteration. In the worst case, the size of $\mathcal{R}^{\downarrow}$ is 2^n. Therefore, a pessimistic estimate of the number of splitters is $2^n * 2^n$, thus 2^{2n}, and similarly for the space complexity, since all untouched splitters need to be stored. As we will see in Section 6, Algorithm 1 in practice stays far below the theoretical bounds.

To illustrate how PIBAR works, we discuss the following example.

Example 5.2. Suppose we face a state t as shown in Figure 2. We assume that s_2, s_4, and s_5 are absorbing, that is, can only evolve into themselves with probability 1, while the transitions of s_1 and s_3 are shown in the right side of Figure 2. Moreover, state s_4 is labeled $\triangle$, differently from all other states. We first observe that $s \sim_1 r$: The only non-trivial cases to consider concern the minimal probabilities from s and r to state sets $C \subseteq \{s_1, s_2, s_3\}$. Notably, states $s_1, s_2,$ and s_3 are pairwise distinguishable, because each can reach state s_4 with a different probability. For $C = \{s_1, s_2\}$, the minimal one-step probabilities from s and r to C equal 0.6, obtained taking the transition to distribution $\{0.3 : s_1, 0.3 : s_2, 0.4 : s_3\}$. We can reason similarly for other possible sets C. Therefore, $s \sim_1 r$, implying that in the quotient induced by $\sim_1$, states s and r will be grouped together, while all other states will be pairwise distinct, and thus form singleton blocks in the quotient. So, $\sim_1 = \{(s, r), (r, s)\} \cup ID$ is the equivalence relation $\mathcal{R}$ induced by the current partition, where ID is the identity relation.

We now assume that $\Phi = P_{\geq 0.6}(true\ \mathsf{U}\ \neg\triangle)$ is the property we are going to check. The minimal probability for the paths starting from t satisfying $true\ \mathsf{U}\ \neg\triangle$ is 0.64, which is induced by choosing the transition to r and then choosing the dashed transitions in Figure 2. Therefore, $t \models \Phi$ holds. According to Definition 4.1, block $[s]_{\mathcal{R}}$ (or equivalently $[r]_{\mathcal{R}}$) containing s and r has the same (up to $\equiv_{\sim_1}$) transitions as r in the may-quotient.

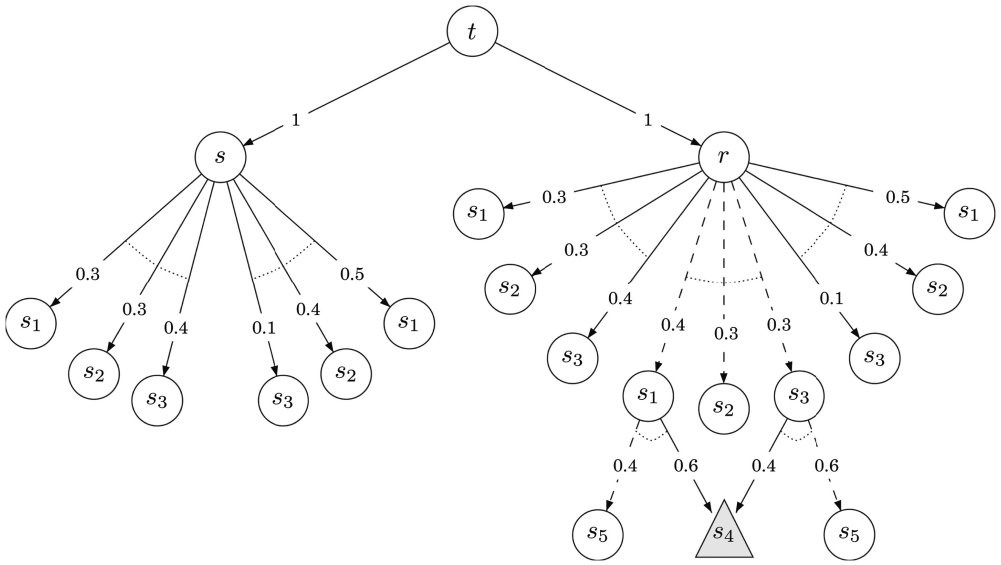

Fig. 2. A counterexample of the completeness of probabilistic bisimulation.

Therefore we can lift the reasoning showing that $t \models \Phi$ to the may-quotient to establish $t^{\diamond}_{\mathcal{R}} \models \Phi$. As a consequence, we can see that $\sim_1$ is fine enough to preserve Φ.

Now let us instead consider $\Psi = P_{\geq 0.65}(\mathit{true}\, \mathsf{U}\, \neg\triangle)$. We know from the preceding discussion that $t \not\models \Psi$, since there exists an execution that reaches $\neg\triangle$ from t with probability 0.64. If we now attempt to derive this fact by reasoning on the quotients induced by $\sim_1$ as before, we find that $t^{\diamond}_{\mathcal{R}} \not\models \Psi$, which implies neither $t \models \Psi$ nor $t \not\models \Psi$. So we now consider the must-quotient, where block $[s]_{\mathcal{R}}$ has the same (up to $\equiv_{\sim_1}$) transitions as s (opposed to r in the may-quotient). This implies that the middle transition of r (up to $\equiv_{\sim_1}$) existing from block $[s]_{\mathcal{R}}$ in the may-quotient is absent in the must-quotient, as the transition is absent in s. Therefore, the minimal probability for the paths of $t^{\square}_{\mathcal{R}}$ satisfying $\mathit{true}\, \mathsf{U}\, \neg\triangle$ is the same as s, namely, 0.66, and hence $t^{\square}_{\mathcal{R}} \models \Psi$. This means that $\sim_1$ is too coarse to preserve Ψ, and we need to refine it further. It turns out that $s \not\sim_2 r$. As for instance $r \not\models P_{\geq 0.65}(\mathit{true}\, \mathsf{U}^{\leq 2}\, \neg\triangle)$, but $s \models P_{\geq 0.65}(\mathit{true}\, \mathsf{U}^{\leq 2}\, \neg\triangle)$. Therefore, in the second refinement step, s and r will be distinguished, yielding an $\sim_2 = \mathcal{R}$ (namely, the identity relation), on which we will be able to conclude $t^{\square}_{\mathcal{R}} \not\models \Psi$. Obviously, $\sim_2$ is fine enough to preserve Ψ.

It is worthwhile to highlight that if we instead were using classical bisimulation [Segala 1995], states s and r would never be grouped together. This is rooted in the fact that the middle transition of r cannot be simulated by s even considering combined transitions. As a result, $s \not\sim r$, and the abstract system induced by the classical bisimulation would be too fine for properties such as $P_{\geq 0.6}(\mathit{true}\, \mathsf{U}\, \neg a)$. This is mainly caused by the fact that s and r are PCTL-equivalent [Song et al. 2013a], and $\sim$ is strictly finer than $\equiv_{\mathsf{PCTL}}$.

We establish the complexities of computing the relevant bisimulations as follows.

LEMMA 5.3.

(*1*) *It is NP-complete to check whether* $s \sim_i r$ *for any* $i \geq 1$ *and* s, r.
(*2*) $\sim_{1,1}$ *can be computed in polynomial time.*

PROOF (SKETCH). It has been shown that the problem of computing the simulation version of $\sim_1$ is NP-complete [Desharnais et al. 2011]. The proof idea proceeds by reducing the subset sum problem to our problem: Given n decimal numbers $d_1, d_2, \ldots, d_n$, can we find a set $D \subseteq \{d_i \mid 1 \leq i \leq n\}$ such that $\sum_{d \in D} d = 0$. In a similar way, one can show for $i \geq 1$ that $\sim_i$ is NP-complete, too.

For the second clause, we note that $\sim_{1,1}$ can be computed using a standard partition refinement algorithm, which has been applied to compute bisimulations for both DTMCs [Derisavi et al. 2003; Valmari and Franceschinis 2010] and MDPs [Baier et al. 2000]. This algorithm can be applied in our setting, since in order to compute $\sim_{1,1}$, we only need to consider individual equivalence classes. □

Likewise, $\sim_{2,1}$ can also be computed in polynomial time. We first compute $\sim_{1,1}$ and then consider all reachability probabilities up to two steps. The number of splitters we shall consider in Algorithm 2 is at most n^2, with n being the number of states. The other relations can be computed in a similar way, but with increased complexities.

Valmari and Franceschinis [2010] present an efficient algorithm running in time $O(n \log m)$ to compute bisimulation for DTMCs. Although the essence of $\sim_{1,1}$ is close to strong bisimulation on DTMCs, the approach cannot be extended to our setting directly, because the $O(n \log m)$ complexity crucially relies on the fact that in DTMCs $Prob_{min}(s, C, C_1, 1) = Prob_{min}(r, C, C_1, 1)$ and $Prob_{min}(s, C, C_2, 1) = Prob_{min}(r, C, C_2, 1)$ together imply $Prob_{min}(s, C, C_3, 1) = Prob_{min}(r, C, C_3, 1)$, for any C, C_1, C_2, and C_3 such that $C_2 \subset C_1$ and $C_3 = C_1 \setminus C_2$. This is in general not true in MDPs due to the existence of nondeterministic transitions.

5.2. Property-Driven Abstraction and Refinement

In general, $\sim_{i,j}$ is expensive to compute, since we have to consider exponentially many downward closed sets. In this section, we propose a method to avoid full exploration of all the downward closed sets.

Algorithm 1 works independently of any property. Once it terminates, we obtain an equivalence relation preserving all PCTL formulas. However, in case that we are only interested in whether a system satisfies a specific property, this relation may be too fine. This is because it suffices to preserve that very property, instead of all PCTL properties. In other words, we may utilize a technique called *property-driven abstraction and refinement* and perform the abstraction and refinement based on the given property. This technique has been applied for Markov chains [Katoen et al. 2007].

In our setting, this allows for drastic improvements: If we are facing the property $\Phi = P_{\geq 0.5}(\Phi_1 \,\mathsf{U}\, \Phi_2)$, it is enough to only consider a single splitter in Algorithm 2, namely, (C, C'), where $C = \{s \in S \mid s \models \Phi_1\}$ and $C' = \{s \in S \mid s \models \Phi_2\}$. The refinement can then proceed by simply increasing the depth, that is, the parameter i in Algorithm 1, as we will explain next.

Without loss of generality, we can assume a preprocessing step so that Φ_1 and Φ_2 are turned into atomic propositions, and all states satisfying neither Φ_1 nor Φ_2 are omitted. At the beginning, we let $\mathcal{R} = \{(s, r) \mid s, r \models \Phi_1 \wedge \neg\Phi_2\} \cup \{(s, r) \mid s, r \models \Phi_2\}$. By fixing the splitter to be (C, C'), we first compute the one-step conditional reachability probability $Prob_{min}(s, C, C', 1)$ for each state s, and then partition states such that s and r are in the same block if and only if $Prob_{min}(s, C, C', 1) = Prob_{min}(r, C, C', 1)$. In the case that the obtained abstract system is too coarse, we need to refine it. As before, the refinement proceeds by computing $Prob_{min}(s, C, C', i)$ with a greater i for each state. This process will continue until a fine enough abstraction is obtained, but throughout the entire process, only the splitter (C, C') is considered, which boils down to skipping the outer iteration of Algorithm 2, making it terminate in polynomial time. It also implies that

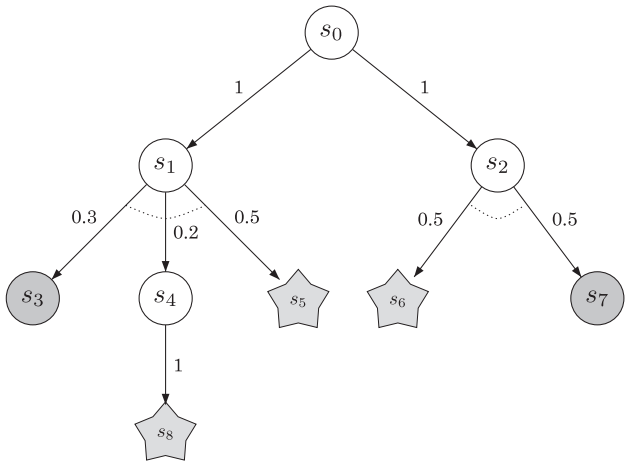

Fig. 3. An example of property-driven verification.

each inner loop of Algorithm 1 terminates with $j = 1$. As we will see in Section 6, this strategy often induces a smaller final quotient system.

THEOREM 5.4. *For a given property, the property driven refinement runs in $\mathcal{O}((m+n)\cdot n)$ time and in $\mathcal{O}(n)$ space, where n and m are the number of states and transitions, respectively.*

PROOF. It takes $\mathcal{O}(m)$ time to compute the minimal (maximal) i-bounded reachability probability for each state, provided that all the minimal (maximal) $i-1$ bounded reachability probabilities have been computed upfront, since we then have to consider each transition once. Using hashing, the partitioning process can be done in $\mathcal{O}(n)$ time in order to group states with the same minimal bounded reachability probability together. Therefore, one iteration of the refinement takes $\mathcal{O}(m+n)$ time. Moreover, it takes at most n refinements, since each state ends up in a distinct equivalence class in the worst case. Therefore, the property-driven abstraction and refinement will terminate in time $\mathcal{O}((m+n)\cdot n)$. The space complexity is easy to see: we only need to store the minimal (maximal) reachability probability for each state. □

The following example illustrates how the property driven abstraction and refinement works.

Example 5.5. Consider the PA in Figure 3, where state labels are indicated by their shapes and colors. For simplicity, we assume that states s_3, s_5, s_6, s_7, and s_8 are absorbing. Since states s_1 and s_2 can reach ●-states with different probabilities, they are neither bisimilar nor i-depth bisimilar for any i.

Let $\Phi = P_{\geq 0.5}$ (○ ∪ ⬠) be the property under consideration. We want to decide whether $s_0 \models \Phi$ holds or not. It is routine to verify that no pair of states in Figure 3 is 1-depth bisimilar. Thus without adopting the property-driven technique, we cannot do any abstraction, since already $\sim_1$ is the identity relation, and thus verification of Φ needs to be conducted on the original system.

However, we notice that in order to compute the conditional probability of reaching ⬠-states via ○-states, some transitions are irrelevant, for instance those from ○ to ●-states . By adopting the property driven technique, we shall fix the splitter to be (C, C') such that C and C' contain only ○-states and ⬠-states, respectively. We observe that s_1 and s_2 can reach the ⬠-states s_5, respectively, s_6 in one step, each with probability

0.5. They will thus be grouped together after the first round of refinement. It is not hard to see that by grouping states s_1 and s_2 into the same block, the resulting may-quotient system still satisfies Φ, which indicates that $s_0 \models \Phi$. This shows that by adopting property-driven abstraction and refinement, we may terminate with a coarser abstraction system than Algorithm 1.

5.3. Extensions

Until now we have only presented our framework to deal with PCTL_{safe} properties. However, Algorithm 1 can be used to deal with full PCTL with a slight change: We keep refining the systems by increasing the index i and j until PCTL equivalence is reached. By Theorem 2.8, this process will terminate. Then we are able to check arbitrary PCTL formulas on the result.

Further, we remark that the approach described in this article can be easily extended to work with PCTL* properties: As shown in Song et al. [2013a], there is a sequence of bisimulation relations converging to PCTL* equivalence. This sequence can be exploited to construct quotients to check PCTL* formulas. Detailed discussions are omitted as they are similar to the PCTL setting.

6. EXPERIMENTAL RESULTS

We have implemented PIBAR with and without property-driven abstraction and refinement in a prototype tool using JAVA. All results were obtained on a laptop with an Intel i7-3520 2.9GHz CPU and 4GB RAM running Ubuntu 12.04. The tool and its source code can be downloaded at `http://depend.cs.uni-sb.de/~song/pibar.html`. The benchmarks are taken from the PRISM webpage `http://www.prismmodelchecker.org`, including

—the asynchronous leader election protocol [Itai and Rodeh 1990],
—randomized consensus shared coin protocol [Aspnes and Herlihy 1990],
—IEEE 802.3 CSMA/CD protocol,
—randomized self-stabilizing algorithms [Beauquier et al. 1999; Israeli and Jalfon 1990].

In Table I, we compare the sizes of the original models and the abstract models, where the abstract models are as small as possible but large enough to preserve the properties we want to check. Columns n and m denote the number of states and transitions, respectively, in the original system, while n' and m' denote the number of states and transitions respectively in the abstract system. The last column "Abs.(s)" denotes the time in seconds used to construct the quotients. For all the examples, Algorithm 1 terminated within about one minute except for "ij18", which was more than ten minutes. As we can see, the abstract systems are much smaller than the original ones. For example, for "csma4_2", we reduce the number of states by a factor of 45 and the number of transitions by a factor of 74.

6.1. Reachability Probabilities

During the experiment, we consider the derivation of probabilistic reachability probabilities. Moreover, until further notice, we always compute precise values for all (maximal or minimal) reachability probabilities. This means that we do not exploit the fact that we can terminate refinement earlier if given a probability bound (as illustrated in Example 5.2.)

Let $\mathcal{P}$, $\mathcal{P}^{\Diamond}_{\mathcal{R}}$, and $\mathcal{P}^{\Box}_{\mathcal{R}}$ be as in Definition 4.1. Let $P_{max}(true\,\mathsf{U}\,stable)(\mathcal{P})$ and $P_{min}(true\,\mathsf{U}\,stable)(\mathcal{P})$ denote the maximal and minimal probabilities of reaching "stable" states in $\mathcal{P}$. We explain by means of example how precise values of extreme reachability probabilities can be obtained. Since $\mathcal{P}^{\Diamond}_{\mathcal{R}} \models P_{\geq q}(true\,\mathsf{U}\,stable)$ implies

Table I. Experiment Results (Abstract)

Protocol	n	m	n'	m'	$Abs.(s)$
leader3	364	654	47	81	0.038
leader5	27,299	74,365	1,527	4,386	2.506
leader6	237,656	760,878	9,012	30,103	42.658
coin2	1,296	2,412	720	1,197	0.011
coin4	104,576	351,712	8,827	21,141	10.198
csma2_6	66,718	93,072	7,504	14,706	3.075
csma4_2	761,962	1,327,608	9,901	17,526	61.203
ij16	65,535	1,048,576	2,249	28,736	20.498
ij18	262,143	4,128,768	7,684	114,814	688.095
beauquier5	1,024	3,840	512	1,920	0.093
beauquier7	16,384	86,016	523	2,817	0.862
beauquier9	262,144	1,769,472	7,031	47,869	61.68

$\mathcal{P} \models P_{\geq q}(true \,\mathsf{U}\, stable)$ for any $q \geq 0$ by Theorem 4.2, it holds $P_{min}(true \,\mathsf{U}\, stable)(\mathcal{P}^{\diamond}_{\mathcal{R}}) \leq P_{min}(true \,\mathsf{U}\, stable)(\mathcal{P})$ for any $\mathcal{R}$. Similarly, we can show that $P_{min}(true \,\mathsf{U}\, stable)(\mathcal{P}) \leq P_{min}(true \,\mathsf{U}\, stable)(\mathcal{P}^{\square}_{\mathcal{R}})$. Therefore the precise value of each minimal reachability probability can be obtained by keeping refining a system until the values obtained from the may- and must-quotient systems coincide. However, the maximal reachability probability cannot be obtained in a similar way, because Theorem 4.2 is restricted to properties in PCTL_{safe}. In order to obtain precise values of maximal reachability properties, we keep refining a system until the may- and must-quotients coincide, in which case the PCTL equivalence relation is reached.

For all properties considered in this section, we consider both unbounded and bounded versions. We choose 5,000 as the step bound of all bounded reachability properties. This is a somewhat arbitrary choice. It is "half-way" to the default maximum iteration number of the value iteration engine of PRISM, which is 10,000.

In Table II, we compare the time to check properties on the original systems and the abstract systems, where the last three columns denote the time spent to check the properties on the abstract systems, the time spent to checkthe properties on the original systems, and the acceleration ratio of PIBAR with respect to PRISM for checking all properties, respectively. For checking on the abstract systems, the time consists of three parts: (i) time for abstracting the system, (ii) time for generating the quotient system, and (iii) time for checking properties on the quotient system. For most of the examples, Algorithm 1 runs faster than PRISM if we consider the total time for checking all the properties except for "ij18". For instance in "csma4_2", PRISM takes more than one hour to check all the properties, while the time for using PIBAR to check these properties is less than two minutes (i.e., the abstract time in Table I plus the checking time in Table II), and we obtain an acceleration up to 64 times.

Generally, Algorithm 1 cannot deal with non-safety properties unless the PCTL-equivalence is eventually reached. However, we notice that in our experiments the abstract system returned by Algorithm 1 also preserves properties like $P_{max}(\varphi)$, that is, non-safety property $P_{\leq p}(\varphi)$, which indicates that Algorithm 1 often terminates with the $\mathcal{R}$ being very close (if not identical) to $\equiv_{\mathsf{PCTL}}$.

6.2. Property-Driven PIBAR

We now turn to the property-driven PIBAR approach, applying it to all cases considered in Table II. Experimental results are displayed in Table III, where column

Table II. Experiment Results (Properties)

Protocol	Properties	[*Time*](s)	Time(s)	Acc.
leader3	$P_{min}(true \cup^{\leq 5000} elected)$	0.015	0.062	1.758
	$P_{max}(true \cup^{\leq 5000} elected)$	0.007	0.041	
	$P_{min}(true \cup elected)$	0.001	0.002	
	$P_{max}(true \cup elected)$	0.001	0.004	
leader5	$P_{min}(true \cup^{\leq 5000} elected)$	0.491	6.554	4.041
	$P_{max}(true \cup^{\leq 5000} elected)$	0.596	7.487	
	$P_{min}(true \cup elected)$	0.03	0.333	
	$P_{max}(true \cup elected)$	0.053	0.479	
leader6	$P_{min}(true \cup^{\leq 5000} elected)$	6.607	240.906	7.916
	$P_{max}(true \cup^{\leq 5000} elected)$	6.699	235.695	
	$P_{min}(true \cup elected)$	1.75	3.952	
	$P_{max}(true \cup elected)$	3.743	5.932	
coin2	$P_{min}(true \cup^{\leq 5000} finished)$	0.096	0.126	1.000
	$P_{max}(true \cup^{\leq 5000} finished)$	0.087	0.117	
	$P_{min}(true \cup finished)$	0.032	0.008	
	$P_{max}(true \cup finished)$	0.035	0.01	
coin4	$P_{min}(true \cup^{\leq 5000} finished)$	6.948	31.712	2.718
	$P_{max}(true \cup^{\leq 5000} finished)$	5.735	31.038	
	$P_{min}(true \cup finished)$	2.984	12.337	
	$P_{max}(true \cup finished)$	6.234	12.173	
csma2_6	$P_{min}(\neg collision\_max\_backoff \cup^{\leq 5000} all\_delivered)$	3.055	10.204	1.260
	$P_{max}(\neg collision\_max\_backoff \cup^{\leq 5000} all\_delivered)$	3.043	9.471	
	$P_{min}(\neg collision\_max\_backoff \cup all\_delivered)$	1.8	2.883	
	$P_{max}(\neg collision\_max\_backoff \cup all\_delivered)$	12.786	7.367	
csma4_2	$P_{min}(\neg collision\_max\_backoff \cup^{\leq 5000} all\_delivered)$	3.879	1385.742	64.535
	$P_{max}(\neg collision\_max\_backoff \cup^{\leq 5000} all\_delivered)$	3.602	1385.743	
	$P_{min}(\neg collision\_max\_backoff \cup all\_delivered)$	3.406	1170.56	
	$P_{max}(\neg collision\_max\_backoff \cup all\_delivered)$	8.688	1270.965	
ij16	$P_{min}(true \cup^{\leq 5000} stable)$	2.813	40.54	3.078
	$P_{max}(true \cup^{\leq 5000} stable)$	2.322	40.822	
	$P_{min}(true \cup stable)$	0.757	0.78	
	$P_{max}(true \cup stable)$	0.547	0.758	
ij18	$P_{min}(true \cup^{\leq 5000} stable)$	17.419	219.509	0.620
	$P_{max}(true \cup^{\leq 5000} stable)$	13.388	220.905	
	$P_{min}(true \cup stable)$	4.318	4.996	
	$P_{max}(true \cup stable)$	1.912	3.909	
beauquier5	$P_{min}(true \cup^{\leq 5000} stable)$	0.109	0.274	2.158
	$P_{max}(true \cup^{\leq 5000} stable)$	0.091	0.241	
	$P_{min}(true \cup stable)$	0.005	0.068	
	$P_{max}(true \cup stable)$	0.005	0.071	
beauquier7	$P_{min}(true \cup^{\leq 5000} stable)$	0.202	4.312	6.795
	$P_{max}(true \cup^{\leq 5000} stable)$	0.163	4.014	
	$P_{min}(true \cup stable)$	0.007	0.069	
	$P_{max}(true \cup stable)$	0.006	0.031	
beauquier9	$P_{min}(true \cup^{\leq 5000} stable)$	8.589	95.185	2.353
	$P_{max}(true \cup^{\leq 5000} stable)$	6.394	92.363	
	$P_{min}(true \cup stable)$	2.722	0.97	
	$P_{max}(true \cup stable)$	0.822	0.22	

Table III. Experiment Results (Property Driven)

Protocol	Abs.(s)	Iterations	Acc.
leader3	0.066	24	1.225
leader5	2.025	51	4.843
leader6	18.702	65	13.406
coin2	0.037	54	0.723
coin4	19.18	125	2.256
csma2_6	7.451	194	0.628
csma4_2	95.662	187	45.527
ij16	5.011	19	7.307
ij18	107.319	22	3.112
beauquier5	0.011	11	3.085
beauquier7	0.403	14	10.915
beauquier9	10.104	14	6.587

"Iterations" denotes the number of refinements before termination, that is, the value of i after termination. From Table III, we can see the time taken to abstract "ij18" is only about 107 seconds by applying property-driven abstraction and refinement, while without applying property-driven technique it takes more than 688 seconds. However, the property-driven PIBAR does not always exceed PIBAR. For instance, for "coin4" and "csma4_2", property-driven PIBAR is slower than PIBAR. This difference is rooted in the structures of the models: Our experiments indicate that property-driven PIBAR is more efficient if the target states are in close reach from the initial states, otherwise PIBAR is more preferable. For instance in "coin4", the minimal expected number of steps needed from the initial states to states labeled with *finished* is more than 4,781, while for "ij18", the maximal expected number of steps before reaching target states labeled with *stable* is less than 153.

From Tables I, II, and III, we can see that PIBAR and property-driven PIBAR perform quite differently in practice. The reason being that they try to refine the system in different dimensions. With reference to Figure 1, plain PIBAR computes the relation in the first row, then the relations in the second row, and so forth, that is, $\equiv_{\mathsf{PCTL}}$ is approximated in a horizontally-first manner. Instead, property-driven PIBAR considers only a single splitter to refine, which means that the parameter j governing the inner loop of Algorithm 1 is 1 and that the outer loop in Algorithm 2 is basically omitted. Pictorially speaking, Figure 1 collapses into a single column, and the refinement simply proceeds by increasing the depth, that is, the parameter i in Algorithm 1. Thus we effectively converge to $\equiv_{\mathsf{PCTL}}$ from the top to bottom in a vertically-first manner.

6.3. Properties with Probabilistic Bounds

In the previous sections, we always computed precise values of reachability probabilities. In this section, we give an example showing that for PCTL properties with explicit probabilistic bounds, PIBAR may terminate after fewer iterations of refinement than computing precise values.

In Figure 4, we visualize the convergence for "csma4_2" with respect to the property $P_{max}(\neg \mathit{collision\_max\_backoff} \,\mathsf{U}\, \mathit{all\_delivered})$. By increasing the number of iterations, we obtain a finer quotient system, and the abstract result is closer to the real probability in the original model. However, in many cases it is not necessary to do so. For instance if we were to check whether "csma4_2" satisfies $P_{\geq 0.8}(\neg \mathit{collision\_max\_backoff} \,\mathsf{U}\, \mathit{all\_delivered})$, there is no need to know the precise

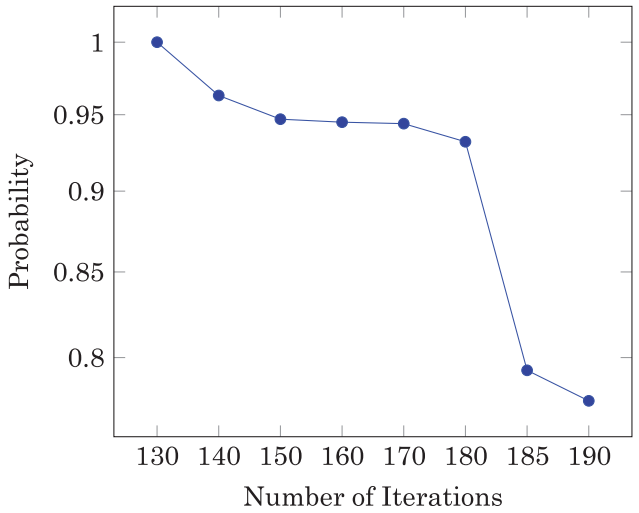

Fig. 4. csma4_2: The result gets more accurate as we increase the number of iterations.

probability of satisfying $\neg collision\_max\_backoff \cup all\_delivered$. In fact, we can stop refinement after 180 iterations, because at this point, we already know for sure the minimal probability of satisfying $\neg collision\_max\_backoff \cup all\_delivered$ is less than 0.8, and thus we can conclude that the property is not satisfied by "csma4_2". If the bound were 0.96, we could stop after 150 iterations. Notably, this is not an a posteriori analysis: The algorithm terminates after reaching the needed probability bound.

6.4. Comparison with Probabilistic CEGAR

We also compared our PIBAR implementation with PASS [Hahn et al. 2010] on finite probabilistic models. PASS implements probabilistic CEGAR [Hermanns et al. 2008] and game-based abstraction-refinement [Kattenbelt et al. 2010], where refinement steps are guided by counterexamples expressed by a set of paths violating the property. Our experimental evaluation shows that PIBAR usually performs better than PASS. This seems to be rooted in the fact that PASS relies on predicate abstraction and refinement, which may take many rounds of refinement before termination. For instance, for the "Bounded Retransmission Protocol" model with parameters 32 and 5 and property 1 considered in Hermanns et al. [2008], PASS took more than two minutes, while PIBAR took less than ten seconds. However, this purely runtime-oriented analysis has to be taken with a grain of salt, because there are two significant differences between PIBAR and PASS: PIBAR (i) works with explicit-state representations of models, while PASS works directly on the higher language level (such as the PRISM language). PASS in turn (ii) is able to deal with infinite-state models, which are entirely out of reach for the PIBAR approach. Therefore a direct comparison between PIBAR and PASS does not seem to be making much sense.

7. RELATED WORK

Probabilistic abstraction-refinement techniques have first been studied in D'Argenio et al. [2001, 2002]. While their approach focuses on the reachability probabilities, our approach deals in principle with all PCTL properties. Moreover, the major difference between PIBAR and D'Argenio et al. [2001, 2002], is that different refinement strategies are adopted. Specifically, in D'Argenio et al. [2001], whenever it is necessary to refine an abstract system, first those blocks are identified in which the concrete states have

different futures, that is, they can evolve into different distributions (up to the current equivalence relation). This refinement strategy is based on the bisimulation criteria and has been used in other models [Bouajjani et al.1992, Alur et al. 1992, Spelberg et al. 1998]. In D'Argenio et al. [2002] this refinement method is further improved. In the present article, we instead adopt a novel refinement strategy which is directly based on the sequence of bisimulations proposed in Song et al. [2013a]. The advantage of this refinement strategy is that it needs very few refinements before termination. For many practical examples, it even terminates after the first refinement.

In probabilistic CEGAR [Wachter and Zhang 2010], refinement steps are guided by counterexamples expressed by a set of paths violating the property. As already mentioned, further extensions include stochastic games [Wachter and Zhang 2010; Kattenbelt et al. 2010] for obtaining both upper and lower bounds, and also extensions carrying over to probabilistic software verification [Kattenbelt et al. 2009; Esparza and Gaiser 2011] exists.

In Roy et al. [2008] and de Alfaro and Roy [2007] an abstraction technique for MDPs called *magnifying lens abstraction* was introduced, which does not depend on counterexamples generation and analysis. It first partitions the states into regions (or blocks) and then computes upper and lower bounds on these regions. In order to do so, it considers only one region at a time and computes the bounds of the concretestates in it. The refinement of a region depends on the computed bounds of its concrete states. The magnifying lens abstraction technique is designed for reachability and safety properties, and moreover it is a property-driven abstraction, that is, it deals with one property each time.

Compared to previous approaches, PIBAR constructs abstractions based on the sequence of bisimulation relations converging to the PCTL equivalence. Thus it facilitates the verification of arbitrary PCTL properties. Notice, as said previously, that the approach in Chadha and Viswanathan [2010] has been introduced to handle arbitrary PCTL properties as well, but that approach involves repeated computations of simulation relations which is slow in practice [Zhang and Hermanns 2007]. To the best of our knowledge, the CEGAR approach of Chadha and Viswanathan [2010] has not yet been implemented.

The PIBAR approach has a flavor similar to the bisimulation-based minimization approach for Markov chains [Katoen et al. 2007]. In particular, $\sim_{1,1}$ can be computed in the same way as the bisimulation for Markov chains with minor changes, therefore it can be considered as an extension of this approach to the model of PAs. Our work is also related to Dehnert [2011] and Wimmer et al. [2006], in which the *classical bisimulation* [Segala 1995] has been computed symbolically: The polynomial algorithm turns out to be rather expensive in practice. The theoretical complexity of checking the PCTL equivalence relation is even worse: It is NP-complete. In Dehnert [2011], it has been shown that state space minimization based on the classical bisimulation usually does not speed up the verification, since the bisimulation is expensive to compute. In this article, we observe that for many cases, the classical bisimulation is too fine, and usually a quite coarser bisimulation for generating the quotient system is enough. Interestingly, even though with high theoretical complexity (NP-complete), our approach is efficient in almost all of the selected case studies.

8. CONCLUSION AND FUTURE WORK

In this article, we proposed the PIBAR framework, an abstraction-refinement framework based on a sequence of bisimulation equivalence relations. Our prototypical experiments show that PIBAR works well in practice, and very often it terminates after very few refinement steps.

As future work, we will extend PIBAR symbolically to be able to deal with larger systems. Inspired by Dehnert et al. [2013], an implementation of PIBAR based on SMT would also be interesting. Many relations in Figure 1 could be computed efficiently in polynomial time, thus another interesting direction would be to identify such relations and compute them first. Since continuous-time Markov decision process (CTMDP) is a continuous extension of PA, we plan to extend this framework further to deal with CTMDPs.

REFERENCES

Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, David L. Dill, and Howard Wong-Toi. 1992. Minimization of timed transition systems. In *Proceedings of the 3rd International Conference on Concurrency Theory*, Lecture Notes in Computer Science, vol. 630. Springer-Verlag, 340–354. http://dl.acm.org/citation.cfm?id=646727.703209

James Aspnes and Maurice herlihy. 1990. Fast randomized consensus using shared memory. *J. Algor.* 11, 3 (1990), 441–461. http://dx.doi.org/10.1016/0196-6774(90)90021-6

Christel Baier, Bettina Engelen, and Mila Majster-Cederbaum. 2000. Deciding bisimilarity and similarity for probabilistic processes. *J. Comput. Syst. Sci.* 60, 1 (2000), 187–231. http://dx.doi.org/10.1006/jcss.1999.1683

Christel Baier and Joost-Pieter Katoen. 2008. *Principles of Model Checking*. MIT Press.

Christel Baier, Joost-Pieter Katoen, Holger Hermanns, and Verena Wolf. 2005. Comparative branching-time semantics for Markov chains. *Inf. Comput.* 200, 2 (2005), 149–214.

Joffroy Beauquier, Maria Gradinariu, and Colette Johnen. 1999. Memory space requirements for self-stabilizing leader election protocols. In *Proceedings of the 18th Annual ACM Symposium on Principles of Distributed Computing (PODC'99)*. ACM, 199–207. http://doi.acm.org/10.1145/301308.301358

Ahmed Bouajjani, Jean-Claude Fernandez, Nicolas Halbwachs, and Pascal Raymond. 1992. Minimal state graph generation. *Sci. Comput. Program.* 18, 3 (1992), 247–269. http://dx.doi.org/10.1016/0167-6423(92)90018-7

Rohit Chadha and Mahesh Viswanathan. 2010. A counterexample-guided abstraction-refinement framework for Markov decision processes. *ACM Trans. Comput. Logic* 12, 1 (2010), 1:1–1:49. http://doi.acm.org/10.1145/1838552.1838553

Edmund Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. 2003. Counterexample-guided abstraction refinement for symbolic model checking. *J. ACM* 50, 5 (2003), 752–794. http://doi.acm.org/10.1145/876638.876643

Edmund M. Clarke, E. Allen Emerson, and A. Prasad Sistla. 1986. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.* 8, 2 (1986), 244–263. DOI: http://dx.doi.org/10.1145/5397.5399

Pedro R. D'Argenio, Bertrand Jeannet, Henrik Ejersbo Jensen, and Kim Guldstrand Larsen. 2001. Reachability analysis of probabilistic systems by successive refinements. In *Proceedings of the Joint International Workshop on Process Algebra and Probabilistic Methods, Performance Modeling and Verification.* Lecture Notes in Computer Science, vol. 2165, Springer-Verlag, 39–56. http://dl.acm.org/citation.cfm?id=645776.668429

Pedro R. D'Argenio, Bertrand Jeannet, Henrik Ejersbo Jensen, and Kim Guldstrand Larsen. 2002. Reduction and refinement strategies for probabilistic analysis. In *Proceedings of the 2nd Joint International Workshop on Process Algebra and Probabilistic Methods, Performance Modeling and Verification.* Lecture Notes in Computer Science, vol. 2399, Springer-Verlag, 57–76. http://dl.acm.org/citation.cfm?id=645777.668444

Luca de Alfaro and Pritam Roy. 2007. Magnifying-lens abstraction for Markov decision processes. In *Proceedings of the 19th International Conference on Computer Aided Verification*. Lecture Notes in Computer Science, vol. 4590, Springer-Verlag, 325–338. http://dl.acm.org/citation.cfm?id=1770351.1770402

Christian Dehnert. 2011. *Symbolic Bisimulation Minimization of Markov Models*. Master's thesis. RWTH Aachen University. Diplomarbeit.

Christian Dehnert, Joost-Pieter Katoen, and David Parker. 2013. SMT-based bisimulation minimisation of Markov models. In *Proceedings of the 14th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI).* Lecture Notes in Computer Science, vol. 7737, Springer, 28–47. http://dblp.uni-trier.de/db/conf/vmcai/vmcai2013.html#DehnertKP13

Salem Derisavi, Holger Hermanns, and William H. Sanders. 2003. Optimal state-space lumping in Markov chains. *Inf. Process. Lett.* 87, 6 (2003), 309–315. http://dx.doi.org/10.1016/S0020-0190(03)00343-0

Josée Desharnais, Mathieu Tracol, and Abir Zhioua. 2011. Computing distances between probabilistic automata. In *Proceedings of the 9th Workshop on Quantitative Aspects of Programming Languages (QAPL'11)*. 148–162.

Javier Esparza and Andreas Gaiser. 2011. Probabilistic abstractions with arbitrary domains. In *Proceedings of the 18th International Conference on Static Analysis*. Lecture Notes in Computer Science, vol. 6887, Springer-Verlag, 334–350. http://dl.acm.org/citation.cfm?id=2041552.2041577

Ernst Moritz Hahn, Holger Hermanns, Björn Wachter, and Lijun Zhang. 2010. PASS: Abstraction refinement for infinite probabilistic models. In *Proceedings of the 16th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*. Lecture Notes in Computer Science, Javier Esparza and Rupak Majumdar (Eds.), vol. 6015, Springer, 353–357.

Paul Richard Halmos. 1974. *Measure Theory*. Graduate Texts in Mathematics, Book 18. Springer-Verlag, New York, NY.

Hans Hansson and Bengt Jonsson. 1994. A logic for reasoning about time and reliability. *Formal Asp. Comput.* 6, 5 (1994), 512–535. http://dx.doi.org/10.1007/BF01211866

Holger Hermanns, Augusto Parma, Roberto Segala, Björn Wachter, and Lijun Zhang. 2011. Probabilistic logical characterization. *Inf. Comput.* 209, 2 (2011), 154–172. http://dx.doi.org/10.1016/j.ic.2010.11.024

Holger Hermanns, Björn Wachter, and Lijun Zhang. 2008. Probabilistic CEGAR. In *Proceedings of the 20th International Conference on Computer Aided Verification*. Lecture Notes in Computer Science, vol. 5123, Springer-Verlag, 162–175. http://dx.doi.org/10.1007/978-3-540-70545-1_16

Amos Israeli and Marc Jalfon. 1990. Token management schemes and random walks yield self-stabilizing mutual exclusion. In *Proceedings of the 9th Annual ACM Symposium on Principles of Distributed Computing (PODC'90)*. ACM, 119–131. http://doi.acm.org/10.1145/93385.93409

Alon Itai and Michael Rodeh. 1990. Symmetry breaking in distributed networks. *Inf. Comput.* 88, 1 (July 1990), 60–87. http://dx.doi.org/10.1016/0890-5401(90)90004-2

Bengt Jonsson and Kim Guldstrand Larsen. 1991. Specification and refinement of probabilistic processes. In *Proceedings of the 6th Annual Symposium on Logic in Computer Science (LICS'91)*. IEEE Computer Society, 266–277. http://dx.doi.org/10.1109/LICS.1991.151651

Joost-Pieter Katoen, Tim Kemna, Ivan Zapreev, and David N. Jansen. 2007. Bisimulation minimisation mostly speeds up probabilistic model checking. In *Proceedings of the 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Lecture Notes in Computer Science, vol. 4424, Springer-Verlag, 87–101. http://dl.acm.org/citation.cfm?id=1763507.1763519

Mark Kattenbelt, Marta Kwiatkowska, Gethin Norman, and David Parker. 2009. Abstraction refinement for probabilistic software. In *Proceedings of the 10th International Conference on Verification, Model Checking, and Abstract Interpretation*. Lecture Notes in Computer Science, vol. 5403, Springer-Verlag, 182–197. http://dx.doi.org/10.1007/978-3-540-93900-9_17

Mark Kattenbelt, Marta Kwiatkowska, Gethin Norman, and David Parker. 2010. A game-based abstraction-refinement framework for Markov decision processes. *Form. Methods Syst. Des.* 36, 3 (2010), 246–280. http://dx.doi.org/10.1007/s10703-010-0097-6

Marta Kwiatkowska, Gethin Norman, and David Parker. 2011. PRISM 4.0: Verification of probabilistic real-time systems. In *Proceedings of the 23rd International Conference on Computer Aided Verification*. Lecture Notes in Computer Science, vol. 6806, Springer-Verlag, 585–591. http://dl.acm.org/citation.cfm?id=2032305.2032352

Kim G. Larsen. 1990. In *Automatic Verification Methods for Finite State Systems*. Lecture Notes in Computer Science, vol. 407, Springer-Verlag, 232–246. DOI: http://dx.doi.org/10.1007/3-540-52148-8_19

Pritam Roy, David Parker, Gethin Norman, and Luca de Alfaro. 2008. Symbolic magnifying lens abstraction in markov decision processes. In *Proceedings of the 5th International Conference on Quantitative Evaluation of Systems (QEST'08)*. IEEE Computer Society, 103–112. DOI: http://dx.doi.org/10.1109/QEST.2008.41

Walter Rudin. 2006. *Real and Complex Analysis*. Tata McGraw-Hill Education.

Roberto Segala. 1995. Modeling and verification of randomized distributed realtime systems. Ph.D. Dissertation. MIT.

Roberto Segala and Nancy Lynch. 1995. Probabilistic simulations for probabilistic processes. *Nordic J. of Computing* 2, 2 (1995), 250–273. http://dl.acm.org/citation.cfm?id=642068.642075

Lei Song, Lijun Zhang, Jens Chr. Godskesen, and Flemming Nielson. 2013a. Bisimulations meet PCTL equivalences for probabilistic automata. *Logical Methods Comput. Sci.* 9, 2:7 (2013).

Lei Song, Lijun Zhang, Holger Hermanns, and Jens Chr Godskesen. 2013b. Incremental bisimulation abstraction refinement. In *Proceedings of the 13th International Conference on Application of Concurrency to System Design (ACSD'13)*. IEEE Computer Society, 11–20.

R. F. Lutje Spelberg, Hans Toetenel, and Marcel Ammerlaan. 1998. Partition refinement in real-time model checking. In *Proceedings of the 5th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*. Lecture Notes in Computer Science, vol. 1486, Springer-Verlag, 143–157. http://dl.acm.org/citation.cfm?id=646845.706939

Antti Valmari and Giuliana Franceschinis. 2010. Simple $O(m \log n)$ time Markov chain lumping. In *Proceedings of the 16th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Lecture Notes in Computer Science, vol. 6015, Springer-Verlag, 38–52. DOI: http://dx.doi.org/10.1007/978-3-642-12002-2_4

Björn Wachter and Lijun Zhang. 2010. Best probabilistic transformers. In *Proceedings of the 11th International Conference on Verification, Model Checking, and Abstract Interpretation*. Lecture Notes in Computer Science, vol. 5944, Springer-Verlag, 362–379. http://dx.doi.org/10.1007/978-3-642-11319-2_26

Ralf Wimmer, Marc Herbstritt, Holger Hermanns, Kelley Strampp, and Bernd Becker. 2006. Sigref: A symbolic bisimulation tool box. In *Proceedings of the 4th International Conference on Automated Technology for Verification and Analysis*. Lecture Notes in Computer Science, vol. 4218, Springer-Verlag, 477–492. http://dx.doi.org/10.1007/11901914_35

Lijun Zhang and Holger Hermanns. 2007. Deciding simulations on probabilistic automata. In *Proceedings of the 5th International Conference on Automated Technology for Verification and Analysis*. Lecture Notes in Computer Science, vol. 4762, Springer-Verlag, 207–222. http://dl.acm.org/citation.cfm?id=1779046.1779064

Received October 2013; accepted February 2014