Incremental Bisimulation Abstraction Refinement

Lei Song*§, Lijun Zhang†, Holger Hermanns*, and Jens Chr. Godskesen‡
*Saarland University, Germany
†State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences
‡IT University of Copenhagen, Denmark
§Max-Planck-Institut für Informatik, Saarbrücken, Germany

***Abstract*—Abstraction refinement techniques in probabilistic model checking are prominent approaches to the 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 paper 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.**

I. Introduction

The model checking of large or infinite-state systems has been revolutionized by the invention of counterexample guided abstraction-refinement (CEGAR) [8]. 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 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 concurrent probabilistic systems [31]. Extending both labeled transition systems and Markov chains, they are convenient to use to model systems with nondeterminism and randomization. PAs are akin to Markov decision processes (MDP) and form the backbone model of successful model checkers such as PRISM [26] 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 [9], [10], [19], [7], [25], [34], [11], [28].

The first abstraction refinement techniques have been proposed [9] 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 in [10]. Probabilistic counterexample-guided abstraction-refinement [19] is a natural extension of CEGAR to 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 with two players games provide both upper and lower bounds for maximal/minimal probabilities [34], [25], [7].

The probabilistic CEGAR approach has been successfully applied to several examples [19], [34], [25] to study reachability properties, and an extension to general PCTL properties has been developed [7]. That approach however involves expensive simulation checking and counterexample generation. The algorithm for computing simulation runs in time $\mathcal{O}(m^2 n)$ [36] where n and m are the number of states and transitions of the MDP. Moreover in [7, Theorem 3.11] it was shown that the problem of finding the smallest counterexample is NP-hard, and it is also unlikely to be efficiently approximable.

This paper 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 [30] 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 [32], and combine this theoretical insight with an effective computational procedure. With some inspiration from modal transition systems [27], we use probabilistic may- and must-quotient automata to represent the behavior 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, so for each step indexed relation, we define a sequence of relations, some of which can be computed in polynomial time, and 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. In 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 to be not 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 [4], [7] 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.

Experimental results carried out with a prototypical implementation of the PIBAR approach demonstrate its effectiveness. We compare with PRISM [26] 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 paper are as follows:

1) We propose a novel framework of probabilistic bisimulation guided abstraction, avoiding the need to analyze counterexamples.
2) 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.
3) We report on a prototypical implementation of PIBAR, and demonstrate that the approach can accelerate probabilistic model checking in many cases.

Organization of the paper: Section II recalls some notations used in the body of the paper. In section III we recall the definition of strong i-depth bisimulation. We propose may quotient and must quotient of a probabilistic system w.r.t. an equivalence relation in Section IV. We describe in detail how PIBAR works in Section V. The experimental results are discussed in Section VI. Section VIII concludes the paper.

II. Preliminaries

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

For a 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 $\mathcal{R}$, 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 such that $[\mu]_\mathcal{R}(C) = \sum_{s\in C} \mu(s)$ for each $C \in S/\mathcal{R}$.

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

Definition 1 *For a relation $\mathcal{R}$ over S and $C \subseteq S$, define $\mathcal{R}^\downarrow(C) = \{s' \mid s' \ \mathcal{R} \ s \wedge s \in C\}$. We say C is $\mathcal{R}$* downward closed *iff $C = \mathcal{R}^\downarrow(C)$.*

We use $\mathcal{R}^\downarrow(s)$ as the shorthand of $\mathcal{R}^\downarrow(\{s\})$, and $\mathcal{R}^\downarrow = \{\mathcal{R}^\downarrow(C) \mid C \subseteq S\}$ to 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 [18] shows that each $\mathcal{R}$ downward closed set can be seen as a union of equivalence classes of $\equiv_\mathcal{R}$.

Lemma 1 (Lemma 5.1 [18]) *Let $\mathcal{R} \subseteq S \times S$ be a relation, and let $C \subseteq S$ be a $\mathcal{R}$ downward closed set, then C is a union of equivalence classes of $\equiv_\mathcal{R}$.*

A. Probabilistic Automata

We recall the notion of probabilistic automata, as coined by Segala [30].

Definition 2 *A* probabilistic automaton *is a tuple $\mathcal{P} = (S, \to, s_0, \mathsf{AP}, L)$ where S is a countable set of states, $\to \subseteq S \times Dist(S)$ is a transition relation, $s_0 \in S$ is a the initial state, AP is a set of atomic propositions, and $L : S \to 2^{\mathsf{AP}}$ is a labeling function.*

In this paper we assume S is finite, and the PA is image-finite, i.e. $\{\mu \mid (s,\mu) \in \to\}$ is finite for each $s \in S$. A transition $(s,\mu) \in \to$ is denoted by $s \to \mu$. A *path* is a finite or infinite sequence $\omega = s_0 s_1 s_2 \ldots$ of states. For each $i \geq 0$ there exists a distribution μ such that $s_i \to \mu$ and $\mu(s_{i+1}) > 0$. We use $lstate(\omega)$ to denote the last state of ω

if ω is finite. The set $Path$ contains all paths, and $Path(s_0)$ only contains paths starting from s_0. Similarly, $Path^*$ is the set of all finite paths, and $Path^*(s_0)$ contains those starting from s_0. Also we use $\omega[i]$ to denote the $(i+1)$-th state for $i \geq 0$, $\omega|^i$ to denote the fragment of ω ending at $\omega[i]$, and $\omega|_i$ to denote the fragment of ω starting from $\omega[i]$.

We introduce the definition of *schedulers* to resolve nondeterminism. A scheduler is a function $\sigma : Path^* \rightarrow Dist(\rightarrow)$ such that $\sigma(\omega)(s,\mu) > 0$ implies $s = lstate(\omega)$. A scheduler σ is *deterministic* if it returns only Dirac distributions, that is, the next step is chosen deterministically.

The *cone* of a finite path ω, denoted by C_ω, is the set of paths having ω as their prefix, i.e., $C_\omega = \{\omega' \mid \omega \leq \omega'\}$ where $\omega' \leq \omega$ iff ω' is a prefix of ω. Fixing a starting state s_0 and a scheduler σ, the measure $Prob_{\sigma,s_0}$ of a cone C_ω, where $\omega = s_0 s_1 \ldots s_k$, is defined inductively as follows: $Prob_{\sigma,s_0}(C_\omega)$ equals 1 if $k = 0$, and for $k > 0$,

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

Let $\mathcal{B}$ be the smallest algebra that contains all the cones and is closed under complement and countable unions. By standard measure theory [16], [29], this algebra is a *σ-algebra* and all its elements are the measurable sets of paths. Moreover, $Prob_{\sigma,s_0}$ can be extended to a unique measure on $\mathcal{B}$.

B. PCTL

We recall the syntax of PCTL [17] which is a probabilistic extension of CTL. Over the set AP of atomic propositions, PCTL is formed according to the following grammar:

$$\begin{aligned} \phi &::= true \mid a \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid P_{\bowtie q}(\psi) \\ \psi &::= \mathsf{X}\,\phi \mid \phi_1 \,\mathsf{U}\, \phi_2 \mid \phi_1 \,\mathsf{U}^{\leq n}\, \phi_2 \end{aligned}$$

where $a \in \mathsf{AP}$, $\bowtie \in \{<,>,\leq,\geq\}$, $q \in [0,1]$. We refer to ϕ and ψ 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

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

The satisfaction relation $\omega \models \psi$ for path formulae is defined as follows::

$$\begin{aligned} &\omega \models \mathsf{X}\,\phi && \text{iff } \omega[1] \models \phi \\ &\omega \models \phi_1 \,\mathsf{U}\, \phi_2 && \text{iff } \exists j \geq 0. \omega[j] \models \phi_2 \\ & && \wedge \forall 0 \leq k < j. \omega[k] \models \phi_1 \\ &\omega \models \phi_1 \,\mathsf{U}^{\leq n}\, \phi_2 && \text{iff } \exists 0 \leq j \leq n. \omega[j] \models \phi_2 \\ & && \wedge \forall 0 \leq k < j. \omega[k] \models \phi_1 \end{aligned}$$

In this paper we are especially interested in the safety fragment of PCTL i.e. safety PCTL, denoted as PCTL_{safe}, whose syntax is given as follows:

$$\begin{aligned} \phi &::= true \mid a \mid \neg a \mid \phi_1 \wedge \phi_2 \mid P_{\geq q}(\psi) \mid P_{>q}(\psi) \\ \psi &::= \mathsf{X}\,\phi \mid \phi_1 \,\mathsf{U}\, \phi_2 \mid \phi_1 \,\mathsf{U}^{\leq n}\, \phi_2 \end{aligned}$$

In PCTL_{safe}, only $\geq$ (or $>$) is allowed in the probabilistic operator, and negation only occurs at atomic propositions. In the sequel let PCTL^i denote the subset of PCTL in which the path formula is restricted to: $\psi ::= \mathsf{X}\,\phi \mid \phi_1 \,\mathsf{U}^{\leq j}\, \phi_2$ where $j \leq i$, similarly for PCTL^i_{safe}. Moreover for a given logic $\mathcal{L}$, we write $s \equiv_{\mathcal{L}} r$ iff $s \models \phi$ implies $r \models \phi$ and vice versa for any state formula ϕ of $\mathcal{L}$. Similarly, we write $s \preceq_{\mathcal{L}} r$ iff $r \models \phi$ implies $s \models \phi$ for any state formula ϕ of $\mathcal{L}$.

C. Bisimulation and Simulation

Weight functions [22] help us to define simulation and bisimulation on PA.

Definition 3 *Let $\mathcal{R} = S \times S$ 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:*

1) *$\Delta(s,r) > 0$ implies that $s \mathrel{\mathcal{R}} r$,*
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 $\mu \sqsubseteq_{\mathcal{R}} \nu$ iff there exists a weight function for μ and ν with respect to $\mathcal{R}$.

Strong simulation and bisimulation for PAs is defined as follows [30]. Since we are only interested in strong relations throughout this paper, we take the liberty to drop the prefix 'strong'. All relations considered throughout the entirety of this paper are strong relations.

Definition 4 *A relation $\mathcal{R} \subseteq S \times S$ is a simulation relation iff $s \mathrel{\mathcal{R}} r$ implies that $L(s) = L(r)$ and for each $s \rightarrow \mu$, there exists $r \rightarrow \nu$ such that $\mu \sqsubseteq_{\mathcal{R}} \nu$. If $\mathcal{R}$ 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 $s \mathrel{\mathcal{R}} r$.*

In [30] it was shown that $\precsim$ and $\sim$ are sound (but not complete) for PCTL and PCTL_{safe} respectively.

Lemma 2 ([30]) $\sim \; \subsetneq \; \equiv_{\mathsf{PCTL}}$, $\precsim \; \subsetneq \; \preceq_{\mathsf{PCTL}_{safe}}$.

According to Lemma 2, bisimulation preserves PCTL equivalence i.e. bisimilar states satisfy the same PCTL formulae.

D. Sound and Complete Bisimulation for PCTL

The inclusion in Lemma 2 is strict. Below we recall a variation of bisimulation which is sound and complete for PCTL equivalence [32].

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, $Prob_{\sigma,s}(C, C', n, \omega)$ equals 1 if $s \in C'$, and else if $n > 0 \wedge (s \in C \setminus C')$, then

$$\begin{aligned} &Prob_{\sigma,s}(C, C', n, \omega) \\ &= \sum_{r \in supp(\mu')} \mu'(r) \cdot Prob_{\sigma,r}(C, C', n-1, \omega r) \end{aligned} \tag{1}$$

where $\sigma(\omega)(s, \mu') = 1$, otherwise $Prob_{\sigma,s}(C, C', n, \omega) = 0$. We are now ready to introduce an indexed family of i-depth (bi)simulations. We let $s \sim_0 r$ and $s \precsim_0 r$ iff $L(s) = L(r)$.

Definition 5 *A relation $\mathcal{R} \subseteq S \times S$ is an i-depth simulation with $i \geq 1$ if $s \mathrel{\mathcal{R}} r$ 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*

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

If $\mathcal{R}$ 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 $s \mathrel{\mathcal{R}} r$.

In comparison with Definition 4, this definition does not require the matching of distributions out of s and r. The essential difference to the standard definition is: We only consider conditional reachability probabilities up to at most i steps.

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

Lemma 3 ([32]) 1) *$\sim_i$ is an equivalence relation, and $\precsim_i$ is a preorder 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* 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. More importantly, Clause 4 assures that there always exists $i \geq 0$ such that $\sim_i = \equiv_{\mathsf{PCTL}}$ and $\precsim_i = \preceq_{\mathsf{PCTL}_{safe}}$ for any PA. In other words, for increasing i, the i-depth bisimulations define a sequence of relation which converges to PCTL equivalence eventually.

III. A Two Dimensional Bisimulation Grid

Lemma 3 shows that by increasing i we can obtain a sequence of bisimulation relations converging to the PCTL-equivalence. In this section we refine the bisimulation $\sim_i$ further to a subsequence of equivalence relations. Our PIBAR framework is based on the induced two dimensional grid of relations. First, we define the size of a downward closed set as follows:

Definition 6 *Let $\mathcal{R} \subseteq S \times S$ be a relation, and $C \subseteq S$ be a $\mathcal{R}$ downward closed set, the size of C, denoted as $size(C)$, is defined as the number of equivalence classes of $\equiv_{\mathcal{R}}$ in C i.e. $size(C) = |\{C' \in S/\equiv_{\mathcal{R}} \mid C' \subseteq C\}|$.*

The concept of size will now be incorporated into a refined bisimulation definition. First, we set $s \precsim_{0,j} r$ and $s \sim_{0,j} r$ iff $L(s) = L(r)$ for any $j \geq 0$. Moreover:

Definition 7 *A relation $\mathcal{R} \subseteq S \times S$ is an (i, j)-depth simulation with $i, j \geq 1$ if $s \mathrel{\mathcal{R}} r$ implies $s \precsim_{i-1,j} r$ and for any $\mathcal{R}$ downward closed sets C, C' and scheduler σ 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 $\mathcal{R}$ 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 $s \mathrel{\mathcal{R}} r$.

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 4 1) *$\sim_{i,j}$ is an equivalence relation 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* PA, *there exists $j \geq 0$ such that $\sim_{i,j} = \sim_i$ and $\precsim_{i,j} = \precsim_i$.*

Proof:

1) Clause 1 is straightforward from Definition 5.
2) Let $\mathcal{R} = \{(s, r) \mid s \precsim_{j,i} r\}$, we show that $\mathcal{R}$ is a (k, i)-depth simulation provided that $j \geq k$. Suppose $s \mathrel{\mathcal{R}} r$ i.e. $s \precsim_{j,i} r$, according to Definition 4, we have $Prob_{\sigma',r}(C, C', l, r) \leq Prob_{\sigma,s}(C, C', l, s)$ for any $\mathcal{R}$ downward closed sets C and C' where $l \leq j$. This implies $Prob_{\sigma',r}(C, C', l, r) \leq Prob_{\sigma,s}(C, C', l, s)$ for any $l \leq k$, therefore $\mathcal{R}$ is a (k, i)-depth simulation.
3) Let $\mathcal{R} = \{(s, r) \mid s \precsim_{i,j} r\}$, we now show that $\mathcal{R}$ is a (i, k)-depth simulation provided that $j \geq k$. Obviously if for any $\mathcal{R}$ downward closed set C, it

$$
\begin{array}{ccccccccc}
\sim_{1,1} & \supseteq & \sim_{1,2} & \supseteq & \sim_{1,3} & \supseteq & \cdots & \supseteq & \sim_{1} \\
\rotatebox{90}{$\subseteq$} & & \rotatebox{90}{$\subseteq$} & & \rotatebox{90}{$\subseteq$} & & & & \rotatebox{90}{$\subseteq$} \\
\sim_{2,1} & \supseteq & \sim_{2,2} & \supseteq & \sim_{2,3} & \supseteq & \cdots & \supseteq & \sim_{2} \\
\rotatebox{90}{$\subseteq$} & & \rotatebox{90}{$\subseteq$} & & \rotatebox{90}{$\subseteq$} & & & & \rotatebox{90}{$\subseteq$} \\
\sim_{3,1} & \supseteq & \sim_{3,2} & \supseteq & \sim_{3,3} & \supseteq & \cdots & \supseteq & \sim_{3} \\
\rotatebox{90}{$\subseteq$} & & \rotatebox{90}{$\subseteq$} & & \rotatebox{90}{$\subseteq$} & & & & \rotatebox{90}{$\subseteq$} \\
\vdots & & \vdots & & \vdots & & \ddots & & \vdots \\
\rotatebox{90}{$\subseteq$} & & \rotatebox{90}{$\subseteq$} & & \rotatebox{90}{$\subseteq$} & & & & \rotatebox{90}{$\subseteq$} \\
\sim_{n,1} & \supseteq & \sim_{n,2} & \supseteq & \sim_{n,3} & \supseteq & \cdots & \supseteq & \sim_{n} = \equiv_{\mathsf{PCTL}}
\end{array}
$$

Figure 1. Inclusion among the bisimulations considered.

holds $\mathit{size}(C) \leq k$, then it also hold that $\mathit{size}(C) \leq j$, hence the following proof is straightforward.

4) 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 Fig. 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.

IV. Bisimulation Quotient

The quotients shall be the keys in the PIBAR approach. Inspired by modal transition systems, we define the may and must quotient systems as follows:

Definition 8 (May 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 *of* $\mathcal{P}$ *w.r.t.* $\mathcal{R}$, *is denoted as* $\mathcal{P}^{\diamond}_{\mathcal{R}} = (S/\mathcal{R}, \rightarrow_{\diamond}, [s_0]_{\mathcal{R}}, \mathsf{AP}, L)$, *where* $L([s]_{\mathcal{R}}) = L(s)$ *for each* s, *and* $[s]_{\mathcal{R}} \rightarrow_{\diamond} [\mu]_{\mathcal{R}}$ *iff there exists* $r \in [s]_{\mathcal{R}}$ *such that* $r \rightarrow \mu$.

Definition 9 (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* must quotient *of* $\mathcal{P}$ *w.r.t.* $\mathcal{R}$, *is denoted as* $\mathcal{P}^{\Box}_{\mathcal{R}} = (S/\mathcal{R}, \rightarrow_{\Box}, [s_0]_{\mathcal{R}}, \mathsf{AP}, L)$, *where* $L([s]_{\mathcal{R}}) = L(s)$ *for each* s, *and* $[s]_{\mathcal{R}} \rightarrow_{\Box} [\mu]_{\mathcal{R}}$ *iff for all* $r \in [s]_{\mathcal{R}}$ *there exists* $r \rightarrow \nu$ *such that* $[\mu]_{\mathcal{R}} = [\nu]_{\mathcal{R}}$.

The function L is overloaded for simplicity. Note the only difference between may and must quotients is the definition of transitions of the quotient systems. In Definition 8, we let the transition of $[s]_{\mathcal{R}}$ be the union of the transition for each $r \in [s]_{\mathcal{R}}$, on the other hand, in Definition 9 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: $\mathcal{P} \precsim \mathcal{P}'$ iff $s_0 \precsim s'_0$ 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 1 *Let* $\mathcal{P}$ *be a* PA *and* $\mathcal{R}$ *be an equivalence relation, we have*

$$\mathcal{P}^{\Box}_{\mathcal{R}} \precsim \mathcal{P} \precsim \mathcal{P}^{\diamond}_{\mathcal{R}}.$$

Proof: We only prove that $\mathcal{P} \precsim \mathcal{P}^{\diamond}_{\mathcal{R}}$ i.e. $s_0 \precsim [s_0]_{\mathcal{R}}$, since the other one can be proved in a similar way. Let $\mathcal{R} = \{(s, [r]_{\mathcal{R}}) \mid s \in [r]_{\mathcal{R}}\}$. Apparently $(s_0, [s_0]_{\mathcal{R}}) \in \mathcal{R}$, therefore it is enough to show that $\mathcal{R}$ is a simulation according to Definition 4. Suppose that $s \rightarrow \mu$, we need to show that there exists $[r]_{\mathcal{R}} \rightarrow \nu$ such that $\mu \sqsubseteq_{\mathcal{R}} \nu$.

We know that for each $s \in [r]_{\mathcal{R}}$, $s \rightarrow \mu$ implies $[r]_{\mathcal{R}} \rightarrow [\mu]_{\mathcal{R}}$ according to Definition 8. 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 $\mathcal{R}$, therefore condition 1) in Definition 3 is satisfied. Moreover $\mu(s') = \sum_{[t]_{\mathcal{R}} \in S/\mathcal{R}} \Delta(s', [t]_{\mathcal{R}}) = \Delta(s', [s']_{\mathcal{R}})$ and $[\mu]_{\mathcal{R}}([s']_{\mathcal{R}}) = \sum_{t \in S} \Delta(t, [s']_{\mathcal{R}}) = \sum_{t \in [s']_{\mathcal{R}}} \Delta(t, [s']_{\mathcal{R}}) = \sum_{t \in [s']_{\mathcal{R}}} \mu(t)$, hence condition 2) and 3) are also satisfied. Consequently, $\mu \sqsubseteq_{\mathcal{R}} [\mu]_{\mathcal{R}}$ which completes the proof. ■

V. Probabilistic Incremental Bisimulation Abstraction Refinement

For an equivalence relation $\mathcal{R}$, we can first construct the may and must quotients. 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 refinement terminates. 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 abstract system is too coarse and needs to be further refined. Lemma 3 and 4 indicate that $\sim_{i,j}$ defines a grid of bisimulations 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 be increased by 1, and j is reset to 0. In other words, we walk through the grid in Fig. 1 in a 'horizontally-first' manner. The thus resulting algorithm is shown in Algorithm 1, where the refinement process starts with the coarsest relation $\mathcal{R} = \sim_{0,0} = \{(s, r) \mid L(s) = L(r)\}$. The Algorithm 1 will for sure terminate since there exists 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 $\sim_{i,j-1}$ i.e. all the splitters (C_1, C_2) such that $\mathit{size}(C_1) < j$ and $\mathit{size}(C_2) < j$ have

Algorithm 1 The Algorithm of PIBAR

```
Input:
      A PA $\mathcal{P}$ and a property $\phi$ of $\mathsf{PCTL}_{safe}$
Output:
      $True$ if $\mathcal{P} \models \phi$, else $False$
 1.1: $i = 0$;
 1.2: while ($True$) do
 1.3:    $i++, j = 0$;
 1.4:    while ($\mathcal{R} \neq \sim_i$) do
 1.5:       // Compute $\sim_{i,j}$;
 1.6:       $\mathcal{R} = Refine(\mathcal{P}, \mathcal{R}, i, j++)$;
 1.7:       $\mathcal{P}^{\diamond}_{\mathcal{R}} = May(\mathcal{P}, \mathcal{R})$;
 1.8:       $\mathcal{P}^{\square}_{\mathcal{R}} = Must(\mathcal{P}, \mathcal{R})$;
 1.9:       if ($\mathcal{P}^{\diamond}_{\mathcal{R}} \models \phi$) then
1.10:          return $True$;
1.11:       else if ($\mathcal{P}^{\square}_{\mathcal{R}} \not\models \phi$) then
1.12:          return $False$;
1.13:       end if
1.14:    end while
1.15: end while
```

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 $\sim_{i,j}$;
 2.1: // Initialize the set of splitter.
 2.2: $splitters = \{(C_1, C_2) \mid C_1, C_2 \in \mathcal{R}^{\downarrow} \wedge (size(C_1) = j \vee size(C_2) = j)\}$;
 2.3: while ($splitters \neq \emptyset$) do
 2.4:    // Get a splitter and remove it from the splitter set;
 2.5:    $(C_1, C_2) = splitters.GetAndRemoveFirst()$;
 2.6:    for all $C \in S/\mathcal{R}$ do
 2.7:       Compute $Prob_{min}(s, C_1, C_2, i)$ for each $s \in C$;
 2.8:       Divide $C$ 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;
 2.9:    end for
2.10:    Add new generated splitters to $splitters$;
2.11: end while
```

been considered, therefore only splitters (C_1, C_2) such that either $size(C_1) = j$ or $size(C_2) = j$ 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_1 only via states in C_2 in at most i steps i.e. $Prob_{min}(s, C_1, C_2, i)$, we then divide C into several disjoint subsets such that $s, r \in C$ are in the same subset iff $Prob_{min}(s, C_1, C_2, i) = Prob_{min}(r, C_1, C_2, i)$. After refining $\mathcal{R}$, we will introduce some new blocks to the current partition thus we need to update the set of splitters by adding pairs like (C_1, C_2), where C_1 and C_2 are 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 i.e. when $\mathcal{R} = \equiv_{\mathsf{PCTL}}$.

Theorem 2 *Algorithm 1 is sound and complete.*

Proof: First note that when Algorithm 2 terminates, we can make sure that $\mathcal{R} = \sim_{i,j}$, since all the possible splitters have been considered. In the inner loop of Algorithm 1, we keep increasing j until $\sim_i$ is obtained. Lemma 4 shows that the inner loop will always terminate. When $\sim_i$ is obtained, we increase the parameter i by 1 and start the inner loop again. The termination of the outer loop is guaranteed by Lemma 3. Since in the worst case, $\mathcal{R}$ will be equal to $\equiv_{\mathsf{PCTL}}$, and we can make sure that either $\mathcal{P}^{\diamond}_{\mathcal{R}} \models \phi$ or $\mathcal{P}^{\square}_{\mathcal{R}} \not\models \phi$ holds. ∎

In theory, the time complexity of Algorithm 1 is high since we have to consider every possible splitter at each iteration. In the worst case the number of splitters is equal to $2^n * 2^n$ i.e. 2^{2n}, and similarly for the space complexity since we have to store all the splitters untouched. But we will show in the following, Algorithm 1 performs much better in practice.

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

Example 1 *Suppose we have a state t such that t have two transitions: $t \to \mathcal{D}_s$ and $t \to \mathcal{D}_r$, where s and r are shown in Fig. 2. We assume that s_2, s_4, and s_5 are absorbing i.e. can only evolve into themselves with probability 1, while the transitions of s_1 and s_3 are shown in the right side of Fig. 2. Moreover, assume all the states have distinct labels with state s_4 has label ●. It is easy to check that $s \sim_1 r$, thus in the quotient systems induced by $\sim_1$, states s and r will be grouped together, while the other states remain.*

Let $\mathcal{R} = \sim_1 = \{(s, r), (r, s)\} \cup ID$ be the equivalence relation induced by the current partition and $\phi = P_{\geq 0.6}(true\,\mathsf{U}\,\neg$●$)$ where ID is the identity relation. Since the minimal probability for the paths of t satisfying $true\,\mathsf{U}\,\neg$● is equal to 0.64 i.e. by choosing the transition to r and then choosing the red transitions in Fig. 2, therefore $t \models \phi$. Note that we also have $t^{\diamond}_{\mathcal{R}} \models \phi$, since according to Definition 8, block $[s]_{\mathcal{R}}$ (or equivalently $[r]_{\mathcal{R}}$) containing s and r will have the same transitions as r in the may quotient (up to $\equiv_{\mathcal{R}}$), while in the must quotient $[s]_{\mathcal{R}}$ will have the same transitions as s. Therefore $t^{\diamond}_{\mathcal{R}} \models \phi$, and we conclude that $\mathcal{R}$ is a fine enough equivalence relation to preserve ϕ.

On the other hand, if we let $\phi = P_{\geq 0.65}(true\,\mathsf{U}\,\neg$●$)$, we know from above discussion that $t \not\models \phi$. But if we use the same partition as before, we have $t^{\square}_{\mathcal{R}} \models \phi$, since in $t^{\square}_{\mathcal{R}}$ the middle transition of r will be abstracted away from block $[s]_{\mathcal{R}}$, as the transition is absent in s. Therefore the minimal

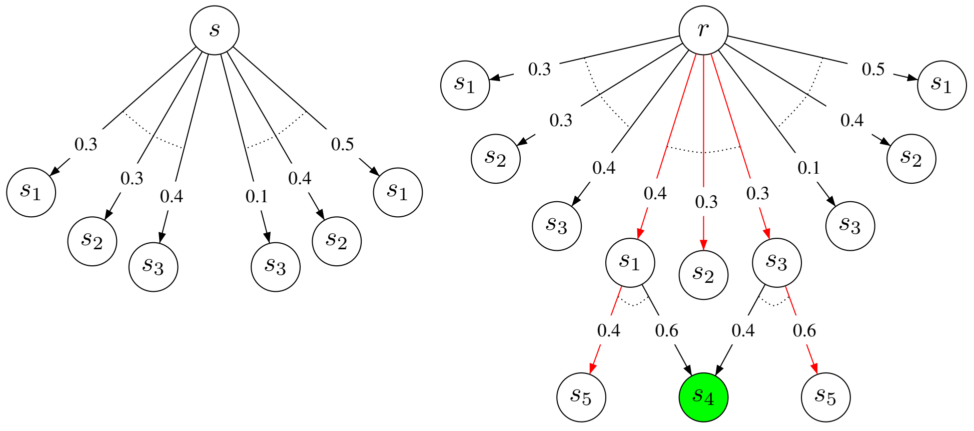

Figure 2. Counterexample of probabilistic bisimulation.

probability for the paths of $t^{\square}_{\mathcal{R}}$ satisfying $true\,\mathsf{U}\,\neg$● is the same as s i.e. 0.66. This means that $\mathcal{R}$ is too coarse to preserve ϕ, thus we need to refine it again. Moreover, note that $s \not\sim_2 r$, because for instance $r \not\models P_{\geq 0.65}(true\,\mathsf{U}^{\leq 2}\,\neg$●$)$, but $s \models P_{\geq 0.65}(true\,\mathsf{U}^{\leq 2}\,\neg$●$)$. Therefore in the next refinement step, s and r will be distinguished i.e. $\mathcal{R} = \sim_2 = ID$, and we will conclude $t^{\square}_{\mathcal{R}} \not\models \phi$.

However if we compute the classical bisimulation in [30] directly, s and r will never be grouped together since the middle transition of r has no way to be simulated by s even if we apply combined transitions, therefore $s \not\sim r$. Consequently, the abstract system induced by the classical bisimulation will be too fine for many properties like $P_{\geq 0.6}(true\,\mathsf{U}\,\neg a)$.

We have the following lemma concerning the complexity of computing the bisimulation relations.

Lemma 5 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 $\sim_1$ is NP-complete [14], similarly we can show for $i > 1$ that $\sim_i$ is NP-complete too. The proof is by reducing the following 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 $\sum_{d\in D} d = 0$. More details on the strategy are found in [14, Theorem 4].

For the second clause, we note that $\sim_{1,1}$ can be computed using a standard partition refinement algorithm, which has been applied for both DTMCs [13] and MDPs [3]. ∎

Remark 1 *Until now we have only dealt 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 3, this process will terminate. Then we are able to check arbitrary PCTL formulae on the result. This idea can be further enhanced to work with PCTL^* properties, by exploiting the sequence of bisimulation relations converging to PCTL^* equivalences [32].*

VI. Experimental Results

We have implemented our PIBAR approach in a prototype tool using C♯. All the results were obtained on a Laptop equipped with an Intel Core i5-2410M CPU 2.3GHz processor and 4GB RAM running Windows 7 Professional. All the examples are taken from the PRISM webpage http://www.prismmodelchecker.org, including the asynchronous leader election protocol [21], randomized consensus shared coin protocol [2], IEEE 802.3 CSMA/CD protocol, and randomized self-stabilizing algorithms (Beauquiepr et al. [5] and Israeli & Jalfon [20]). 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. The 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 get the quotients. For all the examples, Algorithm 1 terminated within one minute with an enough refined abstract system except for the last case which were almost 10 minutes. As we can see, the abstract systems are much smaller than the original ones. For example for "csma4_2", we reduce the state space 45 times and the transition space 74 times.

In Table II we compare the time to check properties on the original systems and the abstract systems. For checking on the abstract systems, the time should contain two parts: Time for abstracting and time for checking properties. For most of the examples, Algorithm 1 runs faster than PRISM if we consider the total time for checking all the properties except for the last case. For instance in "coin4", PRISM takes more than 163 seconds to check all the properties, while the time for using PIBAR to these properties is only about 23 seconds (the abstract time in Table I plus the checking time in Table II). In Table II, some properties are not well-formed PCTL_{safe} properties, for instance $P_{max}(true\,\mathsf{U}(\mathit{finished} \wedge \neg \mathit{agree}))$ [1]. We notice

[1]Properties like $P_{min}(true\,\mathsf{U}(\mathit{finished} \wedge \neg \mathit{agree}))$ are well-formed PCTL_{safe} formula, since $P_{min}(\psi) = q$ is equivalent to $P_{\geq q}(\psi)$ for any ψ

Table I
EXPERIMENT RESULTS (ABSTRACT)

Protocol	n	m	n'	m'	Abs.(s)
leader3	364	654	47	67	0.033
leader6	237656	760878	9012	24457	25.95
coin2	1296	2412	720	959	0.056
coin4	104576	351712	8827	16905	8.55
csma2_6	66718	93072	10948	10961	3.04
csma4_2	761962	1327068	16725	17808	55.8
beauquier5	1024	3840	37	89	0.01
beauquier9	262144	1769472	7031	31613	28.62
israeli-jalfon12	4095	43008	223	1056	0.3
israeli-jalfon18	262143	4128768	7684	65790	591.17

that in our experiments the abstract system returned by Algorithm 1 also preserves properties like $P_{max}(\psi)$ i.e. non-safety property $P_{\leq p}(\psi)$, which indicates that Algorithm 1 often terminates with the $\mathcal{R}$ quite close (if not equal) to $\equiv_{\mathsf{PCTL}}$.

The tool and its source code can be downloaded at http://www.itu.dk/people/leis.

VII. RELATED WORK

Probabilistic abstraction refinement techniques have first been studied in [9], [10]. While their approach focuses on the reachability probabilities, our approach deals in principle with all PCTL properties. Moreover, the major difference between PIBAR and [9], [10] is that different refinement strategies are adopted. Specifically, in [9] whenever it is necessary to refine an abstract system, first those blocks are identified in which the concrete states have different futures i.e. 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 [6], [1], [33]. In [10] this refinement method is further improved. In the present paper we instead adopt a novel refinement strategy which is directly based on the sequence of bisimulations [32]. 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 [34] refinement steps are guided by counterexamples expressed by a set of paths violating the property. As already mentioned further extensions include stochastic games [34], [25] for obtaining both upper and lower bounds, and also extensions carrying over to probabilistic software verification [24], [15] exists.

In [11], [28] 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 concrete states 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, i.e. it deal 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 above, that the approach in [7] has been introduced to handle *arbitrary* PCTL properties as well: but that approach involves repeated computations of simulation relations which is slow in practice [36]. To the best of our knowledge, the CEGAR approach of [7] has not yet been implemented.

The PIBAR approach has a flavor similar to the bisimulation based minimization approach for Markov chains [23]. 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 closely related to [12], [35], in which the *classical bisimulation* [30] 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 [12] 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 paper, 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.

Table II
EXPERIMENT RESULTS (PROPERTIES)

Protocol	Properties	Time(s)	$[Time]$(s)
leader3	$P_{\geq 1}(true\ \mathsf{U}^{\leq 1000}\ elected)$	0.033	0.03
leader6	$P_{\geq 1}(true\ \mathsf{U}^{\leq 1000}\ elected)$	38.8	1.29
coin2	$P_{\geq 1}(true\ \mathsf{U}\ finished)$	0.025	0.033
	$P_{min}(true\ \mathsf{U}(finished \wedge all\_coins\_equal\_1))$	0.382	0.274
	$P_{min}(true\ \mathsf{U}(finished \wedge all\_coins\_equal\_0))$	0.384	0.266
	$P_{max}(true\ \mathsf{U}(finished \wedge \neg agree))$	0.025	0.033
coin4	$P_{\geq 1}(true\ \mathsf{U}\ finished)$	0.089	1.196
	$P_{min}(true\ \mathsf{U}(finished \wedge all\_coins\_equal\_1))$	48.19	4.46
	$P_{min}(true\ \mathsf{U}(finished \wedge all\_coins\_equal\_0))$	54.27	4.28
	$P_{max}(true\ \mathsf{U}(finished \wedge \neg agree))$	60.83	4.77
csma2_6	$P_{min}(true\ \mathsf{U}\ min\_backoff\_after\_success \leq 5)$	0.477	0.438
	$P_{max}(true\ \mathsf{U}\ min\_backoff\_after\_success \leq 5)$	2.75	1.731
	$P_{min}(\neg collision\_max\_backoff\ \mathsf{U}\ all\_delivered)$	1.962	0.578
	$P_{max}(\neg collision\_max\_backoff\ \mathsf{U}\ all\_delivered)$	9.853	2.877
	$P_{min}(true\ \mathsf{U}\ max\_collisions \geq 5)$	0.489	0.203
	$P_{max}(true\ \mathsf{U}\ max\_collisions \geq 5)$	0.705	0.313
csma4_2	$P_{min}(true\ \mathsf{U}\ min\_backoff\_after\_success \leq 1)$	2.847	0.436
	$P_{max}(true\ \mathsf{U}\ min\_backoff\_after\_success \leq 1)$	6.28	2.124
	$P_{min}(\neg collision\_max\_backoff\ \mathsf{U}\ all\_delivered)$	27.092	1.826
	$P_{max}(\neg collision\_max\_backoff\ \mathsf{U}\ all\_delivered)$	49.437	5.53
	$P_{min}(true\ \mathsf{U}\ max\_collisions \geq 5)$	0.029	0.001
	$P_{max}(true\ \mathsf{U}\ max\_collisions \geq 5)$	0.002	0.001
beauquier5	$P_{min}(true\ \mathsf{U}^{\leq 1000}\ stable)$	0.085	0.018
beauquier9	$P_{min}(true\ \mathsf{U}^{\leq 1000}\ stable)$	34.108	5.668
israeli-jalfon12	$P_{min}(true\ \mathsf{U}^{\leq 1000}\ stable)$	0.693	0.057
israeli-jalfon18	$P_{min}(true\ \mathsf{U}^{\leq 1000}\ stable)$	75.534	6.799

VIII. CONCLUSION AND FUTURE WORK

In this paper we proposed 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. Many relations in Fig. 1 can be computed efficiently in polynomial time, thus another interesting direction is 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

[1] R. Alur, C. Courcoubetis, N. Halbwachs, D. L. Dill, and H. Wong-Toi. Minimization of timed transition systems. In *Proceedings of the Third International Conference on Concurrency Theory*, CONCUR '92, pages 340–354, London, UK, UK, 1992. Springer-Verlag.

[2] J. Aspnes and M. Herlihy. Fast randomized consensus using shared memory. *J. Algorithms*, 11(3):441–461, Sept. 1990.

[3] C. Baier, B. Engelen, and M. Majster-Cederbaum. Deciding bisimilarity and similarity for probabilistic processes. *J. Comput. Syst. Sci.*, 60(1):187–231, Feb. 2000.

[4] C. Baier, J.-P. Katoen, H. Hermanns, and V. Wolf. Comparative branching-time semantics for Markov chains. *Inf. Comput.*, 200(2):149–214, 2005.

[5] J. Beauquier, M. Gradinariu, and C. Johnen. Memory space requirements for self-stabilizing leader election protocols. In *Proceedings of the eighteenth annual ACM symposium on Principles of distributed computing*, PODC '99, pages 199–207, New York, NY, USA, 1999. ACM.

[6] A. Bouajjani, J.-C. Fernandez, N. Halbwachs, P. Raymond, and C. Ratel. Minimal state graph generation. *Sci. Comput. Program.*, 18(3):247–269, June 1992.

[7] R. Chadha and M. Viswanathan. A counterexample-guided abstraction-refinement framework for Markov Decision Processes. *ACM Trans. Comput. Logic*, 12(1):1:1–1:49, Nov. 2010.

[8] E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement for symbolic model checking. *J. ACM*, 50(5):752–794, Sept. 2003.

[9] P. R. D'Argenio, B. Jeannet, H. E. Jensen, and K. G. Larsen. 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*, PAPM-PROBMIV '01, pages 39–56, London, UK, 2001. Springer-Verlag.

[10] P. R. D'Argenio, B. Jeannet, H. E. Jensen, and K. G. Larsen. Reduction and refinement strategies for probabilistic analysis. In *Proceedings of the Second Joint International Workshop on Process Algebra and Probabilistic Methods, Performance Modeling and Verification*, PAPM-PROBMIV '02, pages 57–76, London, UK, 2002. Springer-Verlag.

[11] L. de Alfaro and P. Roy. Magnifying-lens abstraction for markov decision processes. In *Proceedings of the 19th international conference on Computer aided verification*, CAV'07, pages 325–338, Berlin, Heidelberg, 2007. Springer-Verlag.

[12] C. Dehnert. Symbolic Bisimulation Minimization of Markov Models. Master's thesis, RWTH Aachen University, 2011. Diplomarbeit.

[13] S. Derisavi, H. Hermanns, and W. Sanders. Optimal state-space lumping in markov chains. *Information Processing Letters*, 87(6):309–315, 2003.

[14] J. Desharnais, M. Tracol, and A. Zhioua. Computing distances between probabilistic automata. In *Ninth Workshop on Quantitative Aspects of Programming Languages*, QAPL'11, pages 148–162, 2011.

[15] J. Esparza and A. Gaiser. Probabilistic abstractions with arbitrary domains. In *Proceedings of the 18th international conference on Static analysis*, SAS'11, pages 334–350, Berlin, Heidelberg, 2011. Springer-Verlag.

[16] P. Halmos. *Measure theory*, volume 1950. Springer-Verlag New York;, 1974.

[17] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Asp. Comput.*, 6(5):512–535, 1994.

[18] H. Hermanns, A. Parma, R. Segala, B. Wachter, and L. Zhang. Probabilistic logical characterization. *Inf. Comput.*, 209(2):154–172, Feb. 2011.

[19] H. Hermanns, B. Wachter, and L. Zhang. Probabilistic CEGAR. In *Proceedings of the 20th international conference on Computer Aided Verification*, CAV '08, pages 162–175, Berlin, Heidelberg, 2008. Springer-Verlag.

[20] A. Israeli and M. Jalfon. Token management schemes and random walks yield self-stabilizing mutual exclusion. In *Proceedings of the ninth annual ACM symposium on Principles of distributed computing*, PODC '90, pages 119–131, New York, NY, USA, 1990. ACM.

[21] A. Itai and M. Rodeh. Symmetry breaking in distributed networks. *Inf. Comput.*, 88(1):60–87, July 1990.

[22] B. Jonsson and K. G. Larsen. Specification and refinement of probabilistic processes. In *Proceedings of the Sixth Annual Symposium on Logic in Computer Science*, LICS'91, pages 266–277. IEEE Computer Society, 1991.

[23] J.-P. Katoen, T. Kemna, I. Zapreev, and D. N. Jansen. 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*, TACAS'07, pages 87–101, Berlin, Heidelberg, 2007. Springer-Verlag.

[24] M. Kattenbelt, M. Kwiatkowska, G. Norman, and D. Parker. Abstraction refinement for probabilistic software. In *Proceedings of the 10th International Conference on Verification, Model Checking, and Abstract Interpretation*, VMCAI '09, pages 182–197, Berlin, Heidelberg, 2009. Springer-Verlag.

[25] M. Kattenbelt, M. Kwiatkowska, G. Norman, and D. Parker. A game-based abstraction-refinement framework for Markov decision processes. *Form. Methods Syst. Des.*, 36(3):246–280, Sept. 2010.

[26] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: verification of probabilistic real-time systems. In *Proceedings of the 23rd international conference on Computer aided verification*, CAV'11, pages 585–591, Berlin, Heidelberg, 2011. Springer-Verlag.

[27] K. G. Larsen. Modal specifications. In J. Sifakis, editor, *Automatic Verification Methods for Finite State Systems*, volume 407 of *Lecture Notes in Computer Science*, pages 232–246. Springer Berlin Heidelberg, 1990.

[28] P. Roy, D. Parker, G. Norman, and L. d. Alfaro. Symbolic magnifying lens abstraction in markov decision processes. In *Proceedings of the 2008 Fifth International Conference on Quantitative Evaluation of Systems*, QEST '08, pages 103–112, Washington, DC, USA, 2008. IEEE Computer Society.

[29] W. Rudin. *Real and complex analysis*. Tata McGraw-Hill Education, 2006.

[30] R. Segala. *Modeling and Verification of Randomized Distributed Realtime Systems*. PhD thesis, MIT, 1995.

[31] R. Segala and N. Lynch. Probabilistic simulations for probabilistic processes. *Nordic J. of Computing*, 2(2):250–273, June 1995.

[32] L. Song, L. Zhang, and J. C. Godskesen. Bisimulations meet PCTL equivalences for probabilistic automata. In *Proceedings of the 22nd international conference on Concurrency theory*, CONCUR'11, pages 108–123, Berlin, Heidelberg, 2011. Springer-Verlag.

[33] R. F. L. Spelberg, H. Toetenel, and M. Ammerlaan. Partition refinement in real-time model checking. In *Proceedings of the 5th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, FTRTFT '98, pages 143–157, London, UK, UK, 1998. Springer-Verlag.

[34] B. Wachter and L. Zhang. Best probabilistic transformers. In *Proceedings of the 11th international conference on Verification, Model Checking, and Abstract Interpretation*, VMCAI'10, pages 362–379, Berlin, Heidelberg, 2010. Springer-Verlag.

[35] R. Wimmer, M. Herbstritt, H. Hermanns, K. Strampp, and B. Becker. Sigref: a symbolic bisimulation tool box. In *Proceedings of the 4th international conference on Automated Technology for Verification and Analysis*, ATVA'06, pages 477–492, Berlin, Heidelberg, 2006. Springer-Verlag.

[36] L. Zhang and H. Hermanns. Deciding simulations on probabilistic automata. In *Proceedings of the 5th international conference on Automated technology for verification and analysis*, ATVA'07, pages 207–222, Berlin, Heidelberg, 2007. Springer-Verlag.