Verification of Partial-Information Probabilistic Systems Using Counterexample-Guided Refinements

Sergio Giro[1,⋆] and Markus N. Rabe[2]

[1] Department of Computer Science, University of Oxford
[2] Department of Computer Science, Saarland University

Abstract. The verification of partial-information probabilistic systems has been shown to be undecidable in general. In this paper, we present a technique based on inspection of counterexamples that can be helpful to analyse such systems in particular cases. The starting point is the observation that the system under complete information provides safe bounds for the extremal probabilities of the system under partial information. Using classical (total information) model checkers, we can determine optimal schedulers that represent safe bounds but which may be spurious, in the sense that they use more information than is available under the partial information assumptions. The main contribution of this paper is a refinement technique that, given such a scheduler, transforms the model to exclude the scheduler and with it a whole class of schedulers that use the same unavailable information when making a decision. With this technique, we can use classical total information probabilistic model checkers to analyse a probabilistic partial information model with increasing precision. We show that, for the case of infimum reachability probabilities, the total information probabilities in the refined systems converge to the partial information probabilities in the original model.

1 Introduction

Verification algorithms for formalisms like Markov Decision Processes and their variants have been studied extensively in the last 20 years, given their wide range of applications. In these systems, there are two kinds of choices: non-deterministic and probabilistic (with probability values specified in the model). Non-deterministic choices are resolved using the so-called schedulers: by restricting a system to the choices of the scheduler, the restricted system collapses to a Markov chain, and probability values for properties can be calculated. Worst-case probability values are then defined by considering the maximum/minimum

⋆ This work was supported by DARPA and the Air Force Research Laboratory under contract FA8650-10-C-7077 (PRISMATIC). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of AFRL or the U.S. Government.

S. Chakraborty and M. Mukund (Eds.): ATVA 2012, LNCS 7561, pp. 333–348, 2012.

probability over all schedulers. In consequence, a counterexample is a scheduler that yields greater/less probability than allowed by the specification.

Background. In recent years, considerable attention has been paid to probabilistic systems in which the non-deterministic choices are resolved according to partial information (see [2,3,5,9] and references therein). The formalisms in these works can be seen as generalized versions of Decentralized Partial Observation MDPs. Using these formalisms we can model, for instance, a game in which players keep some information hidden.

The quantitative model checking problem was shown to be undecidable in general for partial information MDPs [10]. Some techniques are available for finite-horizon properties or to obtain over-approximations [11]. To the best of our knowledge, the technique for quantitative analysis in this paper is the first one in which the amount of information available is gradually refined. As a work in a similar direction, the first abstraction refinement technique for *non-probabilistic* partial-information games was proposed quite recently [6]. A recent work concerning *qualitative* properties in a setting similar to ours is [2].

Contribution. We present an iterative technique that allows to improve the accuracy of the values obtained using total information analysis on partial information systems. In order to do this, we present an algorithm to check whether a total information scheduler complies with the partial information assumptions. We also present a transformation (called *refinement*) that, given a scheduler that does not comply with the assumptions, modifies the model to exclude this scheduler and a whole class of schedulers that use the same unavailable information when making a decision. This transformation can be carried out using different criteria. For the case of infimum reachability probabilities, we show a criterion under which, by successively applying the refinements, the total information probabilities in the refined systems converge to the partial information probabilities in the original model.

Introductory Example. We illustrate the problem we address, and the usefulness of our technique, using the players A and B in Fig. 1 (the automaton $A \parallel B$ will be used later). To simplify, we assume that they play a turn-based game (the systems we consider in the rest of the paper allow for non-deterministic arbitrary interleaving). When the game starts, player A tosses a coin whose sides are labelled with 0 and 1. Then, B tosses a similar coin keeping the outcome hidden. In the next turn, A tries to guess if both outcomes agree: in the state $a0$, the guess of A is that an agreement happened, and his outcome has been 0 (the meaning of the other states is similar). After the guess, a synchronized transition (depicted with a dashed line) takes both A and B to the initial state, where another round starts. Player B wins if A fails to guess at least once. The problem under consideration is to calculate the minimum probability that B wins. Intuitively we can think that player A wants to prevent the system from reaching one of the states in which B wins. An example strategy for the first round for player A would be "if A's outcome is 0, then A guesses an agreement. Otherwise, it guesses a disagreement". In this scheduler/strategy, B wins iff its outcome is 1. Hence, for this scheduler/strategy,

the probability that B wins is the probability that B's outcome is 1, that is, $1/2$. It is easy to see that, for every scheduler, B wins in the first round with probability $1/2$. In subsequent rounds, player A might try different schedulers, but since all of them lead B to win the round with probability $1/2$, the probability that B has won after round N is $1 - (1/2)^N$. Hence, the probability that B wins the game in some round is 1. The minimum probability that B wins, quantifying over all schedulers, is then 1.

This result is not, however, the one yielded by standard tools for probabilistic model checking such as PRISM [12] or LiQuor [4]. Such tools verify this model by constructing the parallel composition $A \parallel B$ shown in Fig. 1 (double-framed boxes indicate the states in which B wins) and considering all schedulers for the composed model under total information. The problem with this approach (sometimes called the *compose-and-schedule* approach) is that there exist some unrealistic schedulers as the one shown in Fig. 2: in this scheduler, the probability of reaching a state in which B wins is 0. The scheduler simply guesses an agreement in case an agreement happened, and a disagreement otherwise. This is unrealistic, as in the original model A is unable to see the outcome of B.

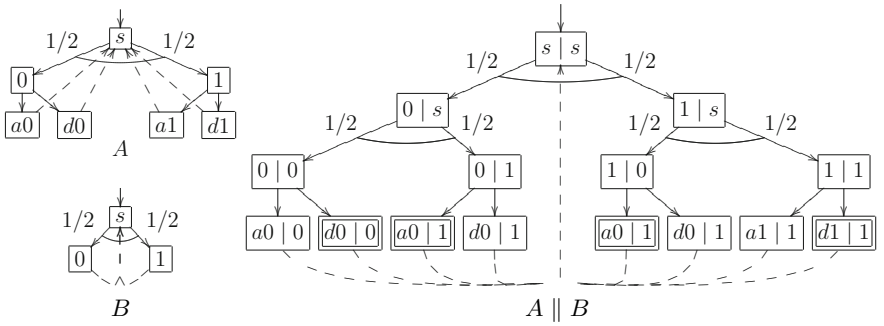


Fig. 1. Player A tries to guess if the choices agree

The problem illustrated by the example led to the definition of partial-information schedulers or distributed schedulers (in which a scheduler of the compound system is obtained by composing schedulers for each player). However, the verification of properties under partial-information schedulers was

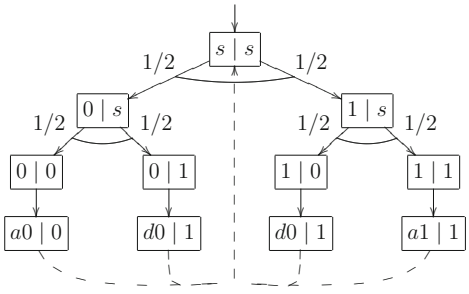
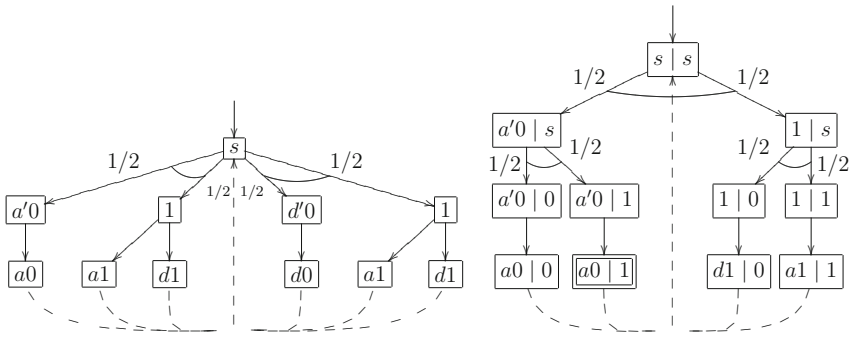


Fig. 2. An unrealistic scheduler for the parallel composition

proven to be difficult, and several hardness and undecidability results are known (see [10,2], just to name a few).

The result considering all schedulers can be seen as a safe (although overly pessimistic) bound on the minimum/maximum probability. In this paper, we present a technique to obtain tighter safe bounds. This technique works through a series of *refinements*: it starts by verifying the system as if total information were available, using standard algorithms for the total-information case. If the system is deemed correct, then it is also correct under partial information, as the set of schedulers under partial information is a subset of the ones under total information. If the system is deemed as incorrect, it can be checked whether the counterexample obtained is valid under partial information: that is, if all choices are resolved using only available information. If the scheduler is indeed valid, then we can conclude that the system under consideration is incorrect, and we can use the counterexample obtained as witness. For the case in which the counterexample is not valid under partial information (that is, the case in which there is a decision that is resolved according to information not available) we present a transformation that produces a system in which the spurious counterexample is less likely to occur in a new analysis under total information. We can analyse the resulting system by repeating this refinement each time we get a spurious counterexample, in the hope that eventually we find the system correct or we get a real counterexample. We show that, for infimum reachability probabilities, the refinements can be carried out in such a way that the results converge to the actual value for all systems.



(a) Refined model of A

(b) A scheduler after the first refinement

Fig. 3.

We can explain our counterexample-based transformation of the system by following our previous example: we first detect that, in the counterexample in Fig. 2, the player A performs a choice using unavailable information while in state 0, by noticing that its choices differ for the state $(0, 0)$ and the state $(0, 1)$ (the player also cheats in state 1, but can tackle one state at a time). The transformation forces (the refined model of) A to choose beforehand what the move will be in state 0, this choice being resolved during the coin toss. If the state reached is 0, player A must adhere to its previous decision. The refined model of A is

shown in Fig. 3(a). Roughly speaking, the non-determinism at state 0 has been "pulled backwards". If we apply the compose-and-schedule approach, the compound system still has some unrealistic counterexamples, as A can still cheat in state 1. One of such unrealistic counterexamples is shown in Fig. 3(b). However, the minimum probability that B wins is now 1 for all schedulers (as eventually A passes through state 0, in which A cannot cheat). Since our transformation ensures that 1 is a lower bound for the minimum probability, we know that the result is 1 and the verification finishes.

This verification, calculating the exact result after one refinement, can be contrasted with the naïve approach of computing the minimum probability p_N that B wins before round N, increasing N successively. These probabilities can be computed by considering each of the schedulers for A up to round N. The value of p_N is $1 - (1/2)^N$, and so this approximation never reaches the actual value 1. In addition, as general schedulers depend on the local history of A, computing p_N involves computations for 2^{2^N} different schedulers.

2 The Model

In this section we introduce Markov Decision Processes, together with a notion of parallel composition suited to define partial-information schedulers.

Markov Decision Processes. Let $\mathrm{Dist}(A)$ denote the set of discrete probability distributions over the set A. The support of A is denoted by $\mathrm{supp}(A)$.

A Markov Decision Process (MDP) $\mathcal{M}$ is a quadruple (S, Q, Σ, T) where S is a finite set of states, $Q \subseteq S$ is a set of initial states, Σ is the finite alphabet of the system, each element in Σ is called a label, and $T \subseteq S \times \Sigma \times \mathrm{Dist}(S)$ is a transition structure: if $\mu = (s, \alpha, d) \in T$ then there is a transition μ with label α enabled in s, and the probability of reaching t from s using this transition is $d(t)$. When no confusion arises, we write $\mu(t)$ instead of $d(t)$. We write $\mathrm{en}(s)$ for the set of transitions (t, α, d) with $t = s$. The label of μ is written $\mathrm{label}(\mu)$. We assume that subindices and superindices map naturally from MDPs to their constituents and so, for instance, S_p is the set of states of $\mathcal{M}_p$.

A path in an MDP is a (possibly infinite) sequence $\rho = s^0.\mu^1.s^1.\cdots..\mu^n.s^n$, where $\mu^i \in \mathrm{en}(s^{i-1})$ and $\mu^i(s^i) > 0$ for all i. If ρ is finite, the last state of ρ is denoted by $\mathrm{last}(\rho)$, and the length is denoted by $\mathrm{len}(\rho)$ (a path having a single state has length 0). Given two paths ρ, σ such that $\mathrm{last}(\rho)$ is equal to the first state of σ, we denote by $\rho \cdot \sigma$ the concatenation of the two paths. The set of finite paths of an $\mathcal{M}$ is denoted by $\mathrm{Paths}_{\mathcal{M}}$. We write $s \xrightarrow{\mu} t$ to denote $\mu \in \mathrm{en}(s) \wedge \mu(t) > 0$. Overloading the notation, we write $s \xrightarrow{\alpha} t$ iff $\exists \mu : \mathrm{label}(\mu) = \alpha \wedge s \xrightarrow{\mu} t$. Given a set of target states U, the set $\mathrm{reach}(U)$ comprises all infinite paths w such that at least one state in w is in U.

The standard semantics of MDPs is given by schedulers. A (total information) scheduler η for an MDP is given by a state $\mathrm{init}_\eta \in Q$ and a function $\eta\colon \mathrm{Paths}_{\mathcal{M}} \to T$ such that, if $\mathrm{en}(\mathrm{last}(\rho)) \neq \emptyset$, then $\eta(\rho) \in \mathrm{en}(\mathrm{last}(\rho))$. In words, the scheduler chooses an enabled transition based on the previous path.

For simplicity, in case $\mathrm{en}(\mathrm{last}(\rho)) = \emptyset$ we define $\eta(\rho) = \varsigma_{\mathrm{last}(\rho)}$, where $\varsigma_{\mathrm{last}(\rho)}$ is a fictitious transition representing the fact that, after ρ, the system remains in $\mathrm{last}(\rho)$ forever. Accordingly, we define $\varsigma_s(s) = 1$ for all s. The set of all schedulers for $\mathcal{M}$ is denoted by $\mathrm{Sched}(\mathcal{M})$.

The set $\mathrm{Paths}(\eta)$ contains all the paths $s^0.\mu^1.s^1.\cdots.\mu^n.s^n$ such that $s^0 = \mathrm{init}_\eta$, $\eta(s^0.\mu^1.s^1.\cdots..\mu^{i-1}.s^{i-1}) = \mu^i$ and $s^{i-1} \xrightarrow{\mu^i} s^i$ for all i. We say that two schedulers η, ζ are *equivalent* (denoted $\eta \equiv \zeta$) iff $\mathrm{Paths}(\eta) = \mathrm{Paths}(\zeta)$ (note that this implies $\eta(\rho) = \zeta(\rho)$ for all $\rho \in \mathrm{Paths}(\eta)$).

The probability $\Pr^\eta(\rho)$ of the path ρ under η is $\prod_{i=1}^n \mu^i(s^i)$ if $\rho \in \mathrm{Paths}(\eta)$. If $\rho \notin \mathrm{Paths}(\eta)$, then the probability is 0.

We are interested on the probability of (sets of) infinite paths. Given a finite path ρ, the probability of the set $\rho^\uparrow$ comprising all the infinite paths that have ρ as a prefix is defined by $\Pr^\eta(\rho^\uparrow) = \Pr^\eta(\rho)$. In the usual way (that is, by resorting to the Carathéodory extension theorem) it can be shown that the definition on the sets of the form $\rho^\uparrow$ can be extended to the σ-algebra generated by the sets of the form $\rho^\uparrow$. Since the measure of any set in the σ-algebra is determined by the measure in the sets $\rho^\uparrow$, it follows that for all measurable sets $\mathcal{H}$: $\eta \equiv \zeta \implies \Pr^\eta(\mathcal{H}) = \Pr^\zeta(\mathcal{H})$.

Composition of variable-based MDPs**.** The systems we compose exchange information through the use of shared variables. In consequence, we assume that the state of an MDP is given by a valuation on a set of variables V. The variables in the set $W \subseteq V$ are the *write* variables. The set of all valuations on a set V is denoted by $\mathcal{V}[V]$. The value of variable v in state s is denoted by $s(v)$. Given a state s and a set $V' \subseteq V$, we define the *restriction* $[s]_{V'}$ as the valuation on V' such that $[s]_{V'}(v) = s(v)$ for all $v \in V'$. Given an MDP $\mathcal{M}_p$ whose set of variables is V_p, we write $[s]_p$ for $[s]_{V_p}$ and $[s]_p^W$ for $[s]_{W_p}$.

The set of states of a variable-based MDP $\mathcal{M}_p$ is the set $\mathcal{V}[V_p]$. We assume, however, that the transitions of variable-based MDPs are of the form (s, α, d) where d is a distribution on $\mathcal{V}[W_p]$ (instead of of $\mathcal{V}[V_p]$). In order to comply with the definition of MDP given in the previous subsection, we can lift d to $\mathcal{V}[V_p]$ in the obvious way by defining $d'(t) = 0$, if $t(v) \neq s(v)$ for some $v \notin W_p$; and $d'(t) = d([t]_p^W)$, otherwise. In what follows, when writing $\mu(t)$, we mean $d(t)$ if $t \in \mathcal{V}[W_p]$, or $d'(t)$ if $t \in \mathcal{V}[V_p]$.

We say that the MDPs $M_1, \cdots, M_N$, are compatible if $\forall_{p \neq q}$: $W_p \cap W_q = \emptyset$. Given a set of compatible MDPs $\{M_1, \cdots M_N\}$, let $\mathcal{M}(\alpha)$ be the subset comprising the modules such that $\alpha \in \Sigma_p$, $W(\alpha)$ be $\cup_{M_p \in \mathcal{M}(\alpha)} W_p$ and $\neg W(\alpha)$ be $V \setminus W(\alpha)$. We define the parallel composition $\mathcal{M} = \|_{p=1}^N M_p$ as the MDP (S, Q, Σ, T) such that:

- S is the set of valuations on $\bigcup_p V_p$
- Q is the set of states s such that $[s]_p^W \in Q_p$ for all p
- $\Sigma = \bigcup_{p=1}^N \Sigma_p$
- $\mu = (s, \alpha, d) \in T$ iff for all $M_p \in \mathcal{M}(\alpha)$ there exists $\mu_p \in \mathrm{en}_p([s]_p)$ such that $\mathrm{label}(\mu_p) = \alpha$, $\mu(t) = \prod_{M_p \in \mathcal{M}(\alpha)} \mu_p([t]_p^W)$ if $[s]_{\neg W(\alpha)} = [t]_{\neg W(\alpha)}$ and $\mu(t) = 0$ whenever $[s]_{\neg W(\alpha)} \neq [t]_{\neg W(\alpha)}$.

Given a transition $\mu = (s, \alpha, d)$ in $\mathcal{M}$, we define $[\mu]_p = ([s]_p, \alpha, d')$, where $d'(t') = \sum_{\{t|[t]_p^W = t'\}} d(t)$ for all $t' \in \mathcal{V}[W_p]$. Note that $\mu \in \text{en}(s) \Longrightarrow [\mu]_p \in \text{en}([s]_p)$.

Control functions. The definitions introduced so far map easily to modelling languages with shared variables such as PRISM, but in order to consider partial information we also need to introduce a concept resembling input/output as in Probabilistic I/O Automata (PIOA [3]): the *control function*.

The technique we present considers a composition $\|_{p=1}^{N} M_p$ together with a function control: $\Sigma \to \{M_p\}_p \cup \{\bot\}$. If $\text{control}(\alpha) = M_p$, the intended meaning is that M_p decides to execute the transitions with this label, while the other modules M_q with $\alpha \in \Sigma_q$ react to this transition. We formalize this meaning when introducing partial-information schedulers. Labels with $\text{control}(\alpha) = \bot$ are not controlled by any module (it can be thought that they are controlled by an external entity that has full knowledge). For the previous definition to make sense, we require $\text{control}(\alpha) \neq \bot \implies \alpha \in \Sigma_{\text{control}(\alpha)}$. We impose the following condition, analogous to the input-enabledness condition for Input/Output Automata [3]:

$$\alpha \in \Sigma_p \wedge \text{control}(\alpha) \neq \bot \wedge \text{control}(\alpha) \neq M_p \implies \forall s_p \in S_p : \exists t_p : s_p \xrightarrow{\alpha} t_p \,. \quad (1)$$

In other words, whenever the module $\text{control}(\alpha)$ chooses to execute a transition, it will not be blocked because of other modules not having an α transition enabled. Given a transition μ we write $\text{control}(\mu)$ instead of $\text{control}(\text{label}(\mu))$.

The control/reaction mechanism is similar to the PIOA, but our definition for MDPs is simpler, as it does not need input and output schedulers as in Switched PIOA [3] nor tokens [3] (or interleaving schedulers [9]) for interleaving non-determinism.

If the model is specified in a language that does not allow control specifications (such as PRISM), we can take $\text{control}(\alpha) = M_p$ if M_p is the only module with $\alpha \in \Sigma_p$, and $\text{control}(\alpha) = \bot$ if $\alpha \in \Sigma_p \cap \Sigma_q$ for some $p \neq q$.

Partial-information schedulers for MDPs. Our partial-information schedulers require the choices of a module to be the same in all paths in which the information observed by such module is the same. Given a path ρ the information available to M_p is called the *projection* of ρ over M_p (denoted $[\rho]_p$). In order to define projections, we say that a transition μ *affects* module M_p iff $\text{label}(\mu) \in \Sigma_p$ or $s \xrightarrow{\mu} t$ for some s, t such that $[s]_p \neq [t]_p$. Projections are defined inductively as follows: the projection of the path s^0 is $[s^0]_p$. The projection $[\rho.\mu.s]_p$ is defined as $[\rho]_p.[s]_p$ if μ affects M_p or $[\rho.\mu.s]_p = [\rho]_p$, otherwise. Alternatively, projections might have been defined to include also information about the transitions. Our definition is simpler and, in addition, it is easy to emulate the (apparently) more general definition by keeping the last transition executed as part of the state of the module.

Definition 1 (Partial-information scheduler). *Given* $\mathcal{M} = \|_{p=1}^{N} M_p$, *the set* $\text{PISched}(\mathcal{M})$ *comprises all schedulers* η *such that for all modules* M_p, *paths* $\rho, \sigma \in \text{Paths}(\eta)$ *with* $[\rho]_p = [\sigma]_p$ *the two following conditions hold:*

$$\text{label}(\eta(\rho)) = \text{label}(\eta(\sigma)) \in \Sigma_p \implies [\eta(\rho)]_p = [\eta(\sigma)]_p \quad (2)$$

$$\text{control}(\eta(\rho)) = \text{control}(\eta(\sigma)) = M_p \implies \text{label}(\eta(\rho)) = \text{label}(\eta(\sigma)) \,. \quad (3)$$

To understand (2), recall that in a state s there might be different transitions enabled, say μ, λ, with $\text{label}(\mu) = \text{label}(\lambda) = \alpha$. Hence, (2) ensures that, if the module M_p synchronizes in a transition labelled with α after both ρ and σ, then the particular transition used in the synchronization must be the same after both ρ and σ. In Eq. (3) we require that, if M_p is the module that chooses the label in both ρ and σ, then the chosen label must be the same.

Given an MDP $\mathcal{M} = \|_{p=1}^{N} M_p$, a set $\mathcal{H}$ of infinite paths and $\mathcal{B} \in [0,1]$, we are interested in the problem of deciding if $\Pr^\eta(\mathcal{H}) \leq \mathcal{B}$ for all $\eta \in \text{PISched}(\mathcal{M})$. The inequality holds iff $\sup_{\eta \in \text{PISched}} \Pr^\eta(\mathcal{H}) \leq \mathcal{B}$. This problem has been proven undecidable, even in the particular case in which $\mathcal{H}$ is the set of paths that reach a particular state [10]. In contrast, it is well-known that the model-checking problem under total information is solvable in polynomial time on the size of the model for logics such as PCTL* and LTL [7].

3 Refinement

In this section we show how a system can be verified under partial information by using total-information techniques on successive refinements of the original system. Our technique starts by verifying the system $\mathcal{M}$ under total information, that is, by calculating $\mathcal{S}_{\text{Tot}} = \sup_{\eta \in \text{Sched}} \Pr^\eta(\mathcal{H})$. If $\mathcal{S}_{\text{Tot}}$ is less than or equal to the allowed probability $\mathcal{B}$ then the inclusion $\text{PISched} \subseteq \text{Sched}$ implies $\mathcal{S}_{\text{Par}} = \sup_{\eta \in \text{PISched}} \Pr^\eta(\mathcal{H}) \leq \mathcal{B}$, and hence the system is correct under partial information. In case that $\mathcal{S}_{\text{Tot}} > \mathcal{B}$, model-checking algorithms can provide a representation of a scheduler η such that $\Pr^\eta(\mathcal{H}) > \mathcal{B}$ (see [7]). If $\eta \in \text{PISched}$, then $\mathcal{M}$ is not correct under partial information, and η is a counterexample. If $\eta \notin \text{PISched}$, then we perform a transformation on the system that prevents the particular counterexample and with it a class of schedulers that violate the partial information assumption in a similar way.

We start the description of our technique by showing how to check if a scheduler η is in PISched.

3.1 Detection of Partial-Information Counterexamples

Under total information, for minimum/maximum reachability it suffices to consider only *globally Markovian* (GM) schedulers. A scheduler η is GM iff $\eta(\rho) = \eta(\sigma)$ for all $\rho, \sigma \in \text{Paths}(\eta)$ with $\text{last}(\rho) = \text{last}(\sigma)$. Moreover, the verification of general LTL and PCTL* formulae is carried out by reducing the original problem to problems for which GM schedulers are sufficient [7].

We address the problem of checking whether $\eta \in \text{PISched}$ for a given GM scheduler η. The method here resembles the well-known technique of *self-composition* [1]. We denote by $\eta^{>0}(s)$ the value of $\eta(\rho)$ for all $\rho \in \text{Paths}(\eta)$ with

$\text{last}(\rho) = s$. Note that a GM scheduler is completely determined by the value of $\eta^{>0}(s)$ in the states s reachable in η, in the sense that if two schedulers η, ζ reach the same states and $\eta^{>0}(s) = \zeta^{>0}(s)$ for all reachable states, then $\eta \equiv \zeta$.

We reduce the problem to that of checking whether η has *conflicting paths*. Given a GM η, we say that two states s, t with $[s]_p = [t]_p$ are η-*conflicting* for the module M_p iff either of the following holds

$$\text{label}(\eta^{>0}(s)) = \text{label}(\eta^{>0}(t)) \in \Sigma_p \wedge [\eta^{>0}(s)]_p \neq [\eta^{>0}(t)]_p \quad \text{or} \tag{4}$$

$$\text{control}(\eta^{>0}(s)) = \text{control}(\eta^{>0}(t)) = M_p \wedge \text{label}(\eta^{>0}(s)) \neq \text{label}(\eta^{>0}(t)) \,. \tag{5}$$

We say that two paths $\rho, \sigma \in \text{Paths}(\eta)$ are η-*conflicting* for M_p if $[\rho]_p = [\sigma]_p$, $\text{last}(\rho) = s$, $\text{last}(\sigma) = t$ and the states s, t are η-conflicting. From the definition of η-conflicting and the definition of PISched, we have the following equivalence for all GM η:

$$\eta \in \text{PISched} \iff \eta \text{ has no conflicting paths} \,. \tag{6}$$

Now we show how to check the (in)existence of conflicting paths for M_p. For all $s, t \in S$, $r_p \in S_p$, we define the relation $s \overset{r_p}{\rightsquigarrow} t$, that holds iff there exist paths ρ, σ such that $\rho \cdot \sigma \in \text{Paths}(\eta)$, $\text{last}(\rho) = s$, $\text{last}(\sigma) = t$ and $[\sigma]_p = r_p$. This relation can be extended naturally to the sequences $\pi_p = r_p^1 \cdots r_p^k$, so we can write $s \overset{\pi_p}{\rightsquigarrow} t$. By the definition of $\rightsquigarrow$, if $s \overset{\pi_p}{\rightsquigarrow} t$ then there exists a path π from s to t in η such that $[\pi]_p = \pi_p$. Consider the non-deterministic finite automaton $\text{Nfa}_p(\eta)$ that represents the relation $\overset{r_p}{\rightsquigarrow}$. In this automaton, each word starting in s and ending in t corresponds to a π_p such that $s \overset{\pi_p}{\rightsquigarrow} t$.

The problem of checking whether η has conflicting paths is now that of checking whether $\text{init}_\eta \overset{\pi_p}{\rightsquigarrow} s$ and $\text{init}_\eta \overset{\pi_p}{\rightsquigarrow} t$ for some η-conflicting s, t. This can be done by constructing the synchronous product automaton

$$\text{Nfa}_p^2(\eta) = \text{Nfa}_p(\eta) \times \text{Nfa}_p(\eta) \tag{7}$$

and checking whether it has a path from $(\text{init}_\eta, \text{init}_\eta)$ to some η-conflicting (s, t).

Superfluous conflicts. We show that some conflicts can be ignored. For instance, consider that we are calculating $\sup_\eta \Pr^\eta(\text{reach}(U))$, and we find that s, t are η-conflicting, and the probability $\Pr_s^\eta(\text{reach}(U))$ to reach U starting from s is 0. We say that this conflict is *superfluous*. Intuitively, one might think that η could be changed so as to not cheat in s, and the probabilities would not decrease, as the probability from s was already 0 in η: if all the conflicts are superfluous, then, we should be able to construct a partial-information scheduler yielding the same probabilities. This intuitive reasoning can be proven, and we state it formally in the theorem below. For minimum probabilities, we say that the conflict is superfluous if $\Pr_s^\eta(\text{reach}(U)) = 1$.

Theorem 1. *Let* η *be such that* $\Pr^\eta(\text{reach}(U)) = \sup_{\eta'} \Pr^{\eta'}(\text{reach}(U))$ *(or, resp.,* $\Pr^\eta(\text{reach}(U)) = \inf_{\eta'} \Pr^{\eta'}(\text{reach}(U))$*). If all conflicts in* η *are superfluous, then there exists* $\eta^* \in \text{PISched}$ *such that* $\Pr^{\eta^*}(\text{reach}(U)) = \Pr^\eta(\text{reach}(U))$.

3.2 Refining a System for a Conflict

According to Eq. (6), if a counterexample η does not comply with the partial-information constraints, there exist two η-conflicting states s and t for a module M_p. Since these states are conflicting, we have that $[s]_p = [t]_p$ and either (4) or (5) holds. In words, two different transitions μ, λ are chosen in M_p while, because of the partial-information constraints the choices must coincide. Next, we show how to refine the module M_p. The refinement is modular, in the sense that only M_p is affected.

As illustrated in the example in the introduction, the idea is to split $[s]_p$ in such a way that μ and λ are not enabled in the same state. We present a general way to split states, that will be useful to develop a splitting criterion that ensures convergence for the case of the infimum probabilities (see Thm. 3).

In order to ensure input-enabledness, the transitions enabled in each of the split states must comply with a certain condition: given a state s_p of M_p, a *minimal choice set* is a set F of transitions such that, for each label α not controlled by M_p there is transition labelled with α in F and, if there is a controlled transition enabled in s_p, then F has at least one controlled transition. Intuitively, when a module restricts its choices to a minimal choice set F, it fixes the reactions for all non controlled transitions, and fixes the controlled transition to execute (if any). In the following, let $\mathcal{T}_1, \cdots, \mathcal{T}_Z$ be sets of transitions such that for every minimal choice set F there exists i such that $F \subseteq \mathcal{T}_i$. These sets are called *choice sets*. As an example, the simplest splitting criterion is, given two states s, t being η-conflicting for M_p, take $\mathcal{T}_1 = \mathrm{en}([s]_p) \setminus \{[\eta(s)]_p\}$ and $\mathcal{T}_2 = \mathrm{en}([s]_p) \setminus \{[\eta(t)]_p\}$ as choice sets.

The overall idea of the transformation is then simple: the state s_p being split is replaced by Z states $s_p^1, \cdots, s_p^Z$ in such a way that the transitions enabled in s^i correspond to $\mathcal{T}_i$. We construct the refined module M_q in the following definition.

Definition 2. *Given a module M_p in $\mathcal{M}$, $u \in S_p$, and choice sets $\{\mathcal{T}_i\}_{i=1}^Z$, the refinement $M_q(\eta, u, \{\mathcal{T}_i\}_{i=1}^Z)$ is defined as:*

- *$V_q = V_p \cup \{x\}$ where $x \notin V_p$ is a fresh variable name. The domain of x is $\{1, \cdots, Z\}$. In addition, $W_q' = W_p \cup \{x\}$*
- *S_q is the set of valuations on V_q*
- *$Q_q = \{v \mid [v]_p = u \wedge [v]_p \in Q_p\} \cup \{v \mid [v]_p \neq u \wedge v(x) = 1 \wedge [v]_p \in Q_p\}$*
- *$\Sigma_q = \Sigma_p \cup \{\alpha \mid \exists u' : [u']_p^W \xrightarrow{\alpha} [u]_p^W\}$*
- *for all $v \in S_q$, we have $\mu_q \in \mathrm{en}_q(r)$ iff some of the following conditions hold:*
  - $\mathrm{label}(\mu_q) \in \Sigma_q \setminus \Sigma_p$ *and $\mu_q(v') = 1$ for some v' such that $[v']_p^W = [v]_p^W$.*
  - $\mathrm{label}(\mu_q) \in \Sigma_p$,

$$\forall v', v'' \in \mathrm{supp}(\mu_q) : v'(x) = v''(x) \tag{8}$$

and there exists $\mu_p \in \mathrm{en}_p([v]_p)$ such that $\mu_p([u]_p^W) > 0$ and

$$\forall v' : \mu_q(v') = \mu_p([v']_p) \qquad \text{and} \tag{9}$$

$$[v]_p \neq u \ \vee \ (v(x) = i \wedge \mu_p \in \mathcal{T}_i \ ) \ . \tag{10}$$

- $\text{label}(\mu_q) \in \Sigma_p$, *and*

$$\forall v' \in \text{supp}(\mu) : v'(x) = 1 \tag{11}$$

and there exists $\mu_p \in \text{en}_p([v]_p)$ *such that* $\mu_p([u]_p^W) = 0$ *and (9) and (10) hold.*

Next we explain this transformation informally. If $u \in Q_p$, then in Q_q we have Z states corresponding to u_p, the variable x having different values in each state (note that these two states constitute the set of all states v such that $[v]_p = u$). For all the other initial states, we just set the value of x arbitrarily to 1 (we could have chosen any value in $\{1, \cdots, Z\}$), since the transitions that lead to u will update the variable regardless of the initial value.

Notice that the alphabet of the module is extended in order to synchronize in all transitions that might change the variables of M_p, and lead it to u.

The transitions in M_q that synchronize on the newly added labels only change the value of x in a non-probabilistic fashion. The transitions in M_q having labels that were already in Σ_p correspond to transitions μ_p that were already in M_p. We have two cases: first we consider the case in which μ_p reaches u. Each of the corresponding transitions μ_q changes the value of x in a non-probabilistic fashion (condition (8)) so that there are Z transitions corresponding to μ_p: each one setting x to a value in $\{1, \cdots, Z\}$. Wrt. the other variables, the transitions μ_q behave in the same way as in M_p (condition (9)). In addition, condition (10) ensures that, if μ_q is enabled in a state corresponding to u, then μ_q corresponds to a transition in M_p that is consistent with the value of x. In case μ_p does reach not u, condition (11) specifies that the transition μ_q set the value of x to 1 (for the same reason that some initial states were set to such an arbitrary value). Note that the transitions μ_q must also respect the probabilities in μ_p, and be consistent with the value of x (as we require the conditions (9) and (10)).

If the module M_p in $\mathcal{M} = \|_r M_r$ is refined to $M_q(u, \{\mathcal{T}_i\}_{i=1}^Z)$, the refined system $\mathcal{M}(u, \{\mathcal{T}_i\}_{i=1}^Z)$ is $M_q(u, \{\mathcal{T}_i\}_{i=1}^Z) \ \|_{r \neq p} M_r$. control$(\alpha)$ remains unchanged.

Given a set of infinite paths $\mathcal{H}$ in an MDP $\mathcal{M}$ with variables V, we can obtain the corresponding set in $\mathcal{M}(u, \{\mathcal{T}_i\}_{i=1}^Z)$:

$$\mathcal{H}' = \{s'^0.\mu'^1.s'^1.\cdots \mid [s'^0]_V.[\mu'^1]_V.[s'^1]_V \cdots \in \mathcal{H}\} .$$

The following theorem ensures that we can apply refinements without changing the partial-information extremal probability values.

Theorem 2. $\sup_{\eta \in \text{PISched}(\mathcal{M})} \Pr^\eta(\mathcal{H}) = \sup_{\eta \in \text{PISched}(\mathcal{M}(u, \{\mathcal{T}_i\}_{i=1}^Z))} \Pr^\eta(\mathcal{H}')$

3.3 Convergence

In the beginning of this section we introduced the problem of computing $\mathcal{S}_{\text{Par}}$, in order to know whether $\mathcal{S}_{\text{Par}} \leq \mathcal{B}$. Depending on the particular system and property being checked, and on how the choice sets are chosen, the probabilities in

the refined systems might actually converge or not to $\mathcal{S}_{\text{Par}}$. The next subsection shows that, when calculating infimum values for reachability properties, there exists a sequence of choice sets that ensures that the probabilities converge.

Suppose the variables in the original system are $x_1, \cdots, x_N$, and that given a system with N variables, the new variable introduced in Definition 2 is x_{N+1}. Given $\mathcal{M}$, we write $N(\mathcal{M})$ for the number of variables in $\mathcal{M}$. Let $V \downarrow i$ be the set of variables $\{x_1, \cdots, x_i\}$. An i-state is an element of $\mathcal{V}[V \downarrow i]$. We say that two transitions are an i-choice in state s_p if they are enabled in s_p and $1 \leq i \leq N$ is the minimum value such that $[\mu]_{W_p \cap V \downarrow i} \neq [\lambda]_{W_p \cap V \downarrow i}$. Exceptionally, if $\text{label}(\mu) \neq \text{label}(\lambda)$ and $\text{control}(\text{label}(\mu)) = \text{control}(\text{label}(\lambda)) = M_p$, we say that μ, λ are a 0-choice. When refining a system with N variables, we eliminate (at least) one i-choice, where trivially $i \leq N$, and for each transition leading to the refined state we create new $(N+1)$-choices: the different transitions μ_q corresponding to a transition μ_p in Definition 2 can be distinguished only by the values they assign to x_{N+1}.

Consider a scheduler $\eta \notin \text{PISched}$. By Eq. (6), it has at least two conflicting states s, t. We say that μ, λ, is an i-conflict in η iff there exist s, s', s_p, such that $[s]_p = [s']_p = s_p$, $[\eta(s)]_p = \mu$, $[\eta(t)]_p = \lambda$, and μ, λ are an i-choice in s_p. We say that μ, λ is a minimum conflict if it is an i-conflict with minimum i, quantifying over all conflicts in η.

Informally, the following theorem states that, if every time we search for a minimum conflict, and we refine all the states having such a conflict, then the infimum probabilities under total information (in the refined systems) converge to the infimum probability under partial information (in the original system).

Theorem 3. *Given an* MDP $\mathcal{M}$ *and a set of states* U*, consider the sequence of* MDP*s defined inductively as follows:* $\mathcal{M}_0$ *is* $\|_p M_p$*. Given* $\mathcal{M}_k$ *and* $\eta_k \notin \text{PISched}(\mathcal{M}_k)$*, we define* $\mathcal{M}_{k+1}$ *by defining intermediate systems* $\mathcal{M}_{k,l}$*. Let* $S_{\min}$ *be the states with a minimum conflict* μ, λ *for* M_p *in* η_k*. The system* $\mathcal{M}_{k,0}$ *is* $\mathcal{M}_k$*; given* $\mathcal{M}_{k,l}$*, if there exists* s^{l+1} *such that* $[s^{l+1}]_{V \downarrow N(\mathcal{M}_k)} \in S_{\min}$ *then we let* $\mathcal{M}_{k,l+1} = \mathcal{M}_{k,l}([s^{l+1}]_p, \{\mathcal{T}_1, \mathcal{T}_2\})$ *, where* $\mathcal{T}_1 = \text{en}([s^{l+1}]_p) \setminus \{\mu' \mid [\mu']_{V_p \downarrow N(\mathcal{M}_k)} = \mu\}$ *and* $\mathcal{T}_2 = \text{en}([s^{l+1}]_p) \setminus \{\lambda' \mid [\lambda']_{V_p \downarrow N(\mathcal{M}_k)} = \lambda\}$*. If no such* s_p^{l+1} *exists, we let* $\mathcal{M}_{k+1} = \mathcal{M}_{k,l}$*. There exists* l *such that* $\mathcal{M}_{k+1} = \mathcal{M}_{k,l}$*. Moreover, we have*

$$\lim_{k \to \infty} \inf_{\eta \in \text{Sched}} \Pr^{\eta}_{\mathcal{M}_k}(\text{reach}(U_k)) = \inf_{\eta \in \text{PISched}} \Pr^{\eta}_{\mathcal{M}}(\text{reach}(U)) \,,$$

where U_k *is the set of all states* s *in* $\mathcal{M}_k$ *such that* $[s]_{V \downarrow N(\mathcal{M})} \in U$*.*

For simplicity, in the theorem above we assume that we are unlucky and we never find a partial information counterexample. In case we do, we can simply make $\mathcal{M}_{k+1} = \mathcal{M}_k$. We also preserve indices across refinements: if M_p is refined in $\mathcal{M}_{k,l}$, then M_p is the refined module in $\mathcal{M}_{k,l+1}$.

There is no similar convergence for upper bounds of supremum probabilities: together with the computable lower bounds $\lim_{n \to \infty} \Pr^{\eta}(\text{reach}(U_n))$ (where $\text{reach}(U_n)$ is the set of paths reaching U before n steps) such upper bounds would turn the approximation problem for the supremum decidable, and it is not [10].

4 Experimental Results

In spite of Thm. 3, we do not have an upper bound on the number of refinements needed to get a probability ϵ-close to the actual one. In consequence, we implemented a preliminary prototype of the refinement technique in PRISM, to check whether some improvements in worst-case probabilities can be found in practically acceptable times.

Variants implemented. We found that, in some cases, the criterion for selecting conflicts in Thm. 3 (which we refer to as "minimum variable conflict" or MVC) does not improve the probabilities quickly enough. An explanation for this behaviour can be seen in a small example: in a system with N variables, suppose that in a module we have $s \xrightarrow{\mu} t \xrightarrow{\lambda} r$, and in r there is a non-deterministic choice that the fictitious counterexamples resolve according to a hidden coin that is tossed by another module in a transition than synchronizes with μ. In the first refinement, we pull the non-determinism backwards to s'. For the scheduler not to cheat any longer, we need to pull the non-determinism to s but, in the first variant, this only occurs after all the N-conflicts have been refined, as the choices introduced in s' are $(N+1)$-choices.

We implemented a second variant (which we refer to as "shortest projection conflict" or SPC) that performs a breadth-first search on the automaton $\mathrm{Nfa}_p^2(\eta)$ in Eq. (7), in order to obtain the shorter projection ρ_p for which the partial information restrictions are violated. In the previous example, if $s.\mu.t.\lambda.r$ is the smallest conflicting path in the first counterexample, then $s.\mu.t$ is the smallest conflicting path in the second counterexample, and the cheating is eliminated in the second refinement. In addition, in this variant we split each state into several states: if the conflict concerns controlled transitions, then each choice set has a single controlled transition (and also has all non-controlled transitions); if the conflict concerns a reaction to a certain label α, then each choice set has a single transition labelled with α (and also all controlled transitions, and transitions with labels other than α). This is easy to implement and yields better results in practice. Examples can be constructed to show that SPC does not converge in all cases.

Table 1. Experimental results

					SPC				MVC		
	N	k	$\lvert S\rvert$ initial	Initial prob.	$\lvert S\rvert$ final	Final prob.	Time (s)	Superf. conflicts	$\lvert S\rvert$ final	Final prob.	Time (s)
Light bulb	3	4	712	0.44	2376731	0.30 *	2973	10	8506371	0.44	12418
	3	5	1042	0.62	5774983	0.50	16762	0	7592182	0.62	11317
	4	4	2069	0.09	893665	0.05 *	1561	80	2015056	0.09	19983
	4	5	3453	0.23	7708257	0.15	26363	156	8294102	0.23	14182
Crowds	3	-	109	1.00	6393	0.40 *	6	0	17705	0.40 *	54
	4	-	237	1.00	163634	0.30 *	530	0	308510	0.55	20050
AFS	20	7	6600	1.00	6978	0.23 *	197	0	6978	0.23 *	198
	25	8	10125	1.00	10719	0.22 *	710	0	10719	0.22 *	728

Experiments. We analysed three systems: (1) the crowds protocol [14]: a group of N entities tries to deliver a message to an evil party in an anonymous way. We ask for the maximum probability that the evil party guesses the sender correctly. (2) the protocol for anonymous fair service (AFS) in [11] (precisely, the variant called AFS1 in [11]), which involves two clients and a server. We ask for the worst-case probability that one of the clients gets k services more than the other one before the first N services happen; (3) the problem of the light bulb and the prisoners [15]: there is a group of N prisoners. In each round, a prisoner is selected at random and taken to a room where he can switch the light bulb. The only information available to each prisoner is the state of the light bulb when he enters the room. The prisoners win the game if one of them can, at some point, guess that all of them have been in the room (they lose in case of a wrong guess). We ask for the maximum probability that the prisoners lose the game in k steps. For (2), we reused in [10]; the other models were written specifically, as there are no existing PRISM models with partial information constraints.

The results are shown in Table 1. The experiments were ran on a six-core Intel Xeon at 2.80 GHz with 32Gb of memory, in order to be able to analyse the problem of the light bulb and the prisoners for as many refinements as possible. The table shows the number of states in the initial system, and in the final system after the last refinement, and similarly for the probability values. We set a time-out of 8hs. for experiments: the cases where the last scheduler had no conflicts (or all of them were superfluous) are marked with a $*$. In these cases we also indicate the number of superfluous conflicts in the last scheduler, as a measure of the usefulness of Thm. 3. For timed-out experiments, the time reported is the time spent to compute the last probability computed.

The two criteria have similar performance for AFS, but SPC outperforms MVC in the other systems. Even when SPC cannot find the realistic probability, in the light bulb case study, it is able to obtain improvements for the probabilities. It is also remarkable that, for AFS, we ran the experiments for the same parameters as in [11] ($N = 20$, $k = 1..20$, not shown in the Table 1) and the probabilities obtained using our technique coincide with the ones in [11]: these were known to be safe bounds, but thanks to our technique now we know that they were in fact exact values.

5 Concluding Remarks

Decidability. Although Thm. 3 ensures that the successive values converge to the infimum reachability probability, we found no way to check if a given approximation is within a certain error threshold: although the bounds get tighter, there is no way to know when we are close enough to the actual value. These approximations are still valuable, as it is not known whether the approximation problem is decidable for infimum probabilities. It is undecidable for the supremum [10], but the infimum/supremum problems are not trivially dual to each other: in fact, for probabilistic finite automata (which are a particular case of the models in [3,10] and here) the approximation problem is undecidable for the supremum [13] and decidable for the infimum [8].

Complexity. Our result of convergence (Thm. 3) concerns the computation of the infimum reachability probability. The problem of, given a system such that the infimum is 0 or 1, determining which of the cases holds, can be shown to be NP-hard using a similar argument as in [9, Thm. 5.5]. This NP-hardness result implies that, unless $P = NP$, for each error threshold ϵ there is no polynomial algorithm to approximate the value within relative error ϵ (these problems are often called *inapproximable*). We do not have any bounds on the amount of refinements needed to get an approximation: however. Given this hardness result, the best we can expect is that our technique is useful in practical cases, as shown in Sec. 4.

Further work. We are particularly interested in linking this approach with existing ones. For instance, we plan to study if it can be applied to the game-like setting in [6], or whether it can be adapted for qualitative properties as studied in [2]. In addition, we plan to consider conflict selection criteria other than "minimum-variable-first" and "shortest-projection-first", to find better results in practical cases. Given that our results are appealing to quantitative verification, a natural step forward would be to extend the results here to consider discounts and rewards.

References

1. Barthe, G., D'Argenio, P.R., Rezk, T.: Secure information flow by self-composition. In: CSFW, pp. 100–114. IEEE Computer Society (2004)
2. Chatterjee, K., Doyen, L., Henzinger, T.A.: Qualitative Analysis of Partially-Observable Markov Decision Processes. In: Hliněný, P., Kučera, A. (eds.) MFCS 2010. LNCS, vol. 6281, pp. 258–269. Springer, Heidelberg (2010)
3. Cheung, L., Lynch, N.A., Segala, R., Vaandrager, F.W.: Switched pioa: Parallel composition via distributed scheduling. TCS 365(1-2), 83–108 (2006)
4. Ciesinski, F., Baier, C.: Liquor: A tool for qualitative and quantitative linear time analysis of reactive systems. In: QEST, pp. 131–132. IEEE CS (2006)
5. de Alfaro, L., Henzinger, T.A., Jhala, R.: Compositional Methods for Probabilistic Systems. In: Larsen, K.G., Nielsen, M. (eds.) CONCUR 2001. LNCS, vol. 2154, pp. 351–365. Springer, Heidelberg (2001)
6. Dimitrova, R., Finkbeiner, B.: Abstraction refinement for games with incomplete information. In: FSTTCS. LIPIcs, vol. 2, pp. 175–186 (2008)
7. Forejt, V., Kwiatkowska, M., Norman, G., Parker, D.: Automated Verification Techniques for Probabilistic Systems. In: Bernardo, M., Issarny, V. (eds.) SFM 2011. LNCS, vol. 6659, pp. 53–113. Springer, Heidelberg (2011)
8. Giro, S.: An algorithmic approximation of the infimum reachability probability for probabilistic finite automata. CoRR, abs/1009.3822 (2010)
9. Giro, S.: On the automatic verification of distributed probabilistic automata with partial information. PhD thesis, FaMAF – Universidad Nacional de Córdoba (2010), `http://cs.famaf.unc.edu.ar/~sgiro/thesis.pdf`
10. Giro, S., D'Argenio, P.R.: Quantitative Model Checking Revisited: Neither Decidable Nor Approximable. In: Raskin, J.-F., Thiagarajan, P.S. (eds.) FORMATS 2007. LNCS, vol. 4763, pp. 179–194. Springer, Heidelberg (2007)

11. Giro, S., D'Argenio, P.R.: On the verification of probabilistic i/o automata with unspecified rates. In: SAC, pp. 582–586. ACM (2009)
12. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of Probabilistic Real-Time Systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer, Heidelberg (2011)
13. Madani, O., Hanks, S., Condon, A.: On the undecidability of probabilistic planning and related stochastic optimization problems. Artif. Intell. 147(1-2), 5–34 (2003)
14. Reiter, M.K., Rubin, A.D.: Anonymous web transactions with crowds. Commun. ACM 42(2), 32–38 (1999)
15. van Ditmarsch, H.P., van Eijck, J., Wu, W.: One hundred prisoners and a lightbulb - logic and computation. In: KR (2010)