Quantitative Model Checking Revisited: Neither Decidable Nor Approximable*

Sergio Giro and Pedro R. D'Argenio

FaMAF, Universidad Nacional de Córdoba - CONICET
Ciudad Universitaria - 5000 Córdoba - Argentina
{sgiro,dargenio}@famaf.unc.edu.ar

Abstract. Quantitative model checking computes the probability values of a given property quantifying over all possible schedulers. It turns out that maximum and minimum probabilities calculated in such a way are overestimations on models of distributed systems in which components are loosely coupled and share little information with each other (and hence arbitrary schedulers may result too powerful). Therefore, we focus on the quantitative model checking problem restricted to *distributed* schedulers that are obtained only as a combination of *local* schedulers (i.e. the schedulers of each component) and show that this problem is undecidable. In fact, we show that there is no algorithm that can compute an approximation to the maximum probability of reaching a state within a given bound when restricted to distributed schedulers.

1 Introduction

The model of Markov decision processes (MDP) [14] is a well-known formalism to study systems in which both probabilistic and nondeterministic choices interact. They are used in such diverse fields as operation research, ecology, economics, and computer science. In particular, MDPs (specially composition oriented versions like probabilistic automata [15] or probabilistic modules [7]) are useful to model and analyze concurrent systems such as distributed systems, and serve as the input model to succesful quantitative model checkers such as PRISM [9].

Analysis techniques for MDPs require to consider the resolution of all nondeterministic choices in order to obtain the desired result. For instance, one may like to use PRISM to find out which is the best probability value of reaching a goal under any possible resolution of the nondeterminism (a concrete instance being "the probability of reaching an error state is below the bound 0.01"). The resolution of such nondeterminism is given by the so called *schedulers* (called also adversaries or policies, see [14,1,15,5,17]). A scheduler is a function mapping traces to transitions or *moves* (or, in the more general case, traces to distributions on moves). Given the nondeterministic moves available at some state, the

* Supported by the CONICET/CNRS Cooperation project "Métodos para la Verificación de Programas Concurrentes con aspectos Aleatorios y Temporizados", ANPCyT project PICT 26135 and CONICET project PIP 6391. The first author was partially supported by the LSIS, UMR CNRS 6168, France.

J.-F. Raskin and P.S. Thiagarajan (Eds.): FORMATS 2007, LNCS 4763, pp. 179–194, 2007.

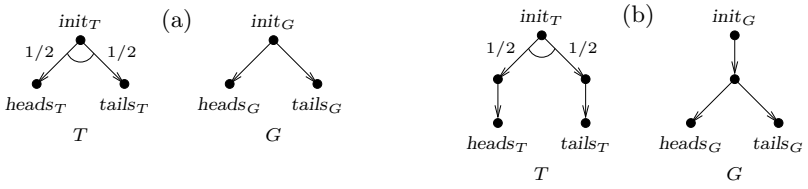

Fig. 1. T tosses a coin and G has to guess

scheduler chooses the move to perform based on the history until the actual state. Then, by quantifying over all the possible schedulers, we obtain maximal and minimal probabilities for (sets of) traces. Quantitative model checkers such as [9,10,4] are based on the technique introduced in [1], in which calculations are performed considering the set of *all* possible schedulers.

While this approach arises naturally, and it captures the intuitive notion of considering all the possible combinations of choices, this arbitrary type of schedulers may yield unexpected results. Consider the following example: a man tosses a coin and another one has to guess heads or tails. Fig. 1(a) depicts the models of these men in terms of MDPs. Man T, who tosses the coin, has only one move which represents the toss of the coin: with probability $1/2$ moves to state $heads_T$ and with probability $1/2$ moves to state $tails_T$. Instead, man G has two nondeterministic moves each one representing his choice: $heads_G$ or $tails_G$. An *almighty* scheduler for this system may let G guess the correct answer with probability 1 according to the following sequence: first, it lets T toss the coin, and then it chooses for G the left move if T tossed a head or the right move if T tossed a tail. Therefore, the maximum probability of guessing obtained by quantifying over these almighty schedulers is 1, even if T is a smart player that always hides the outcome until G reveals his choice. In this example, in which T and G do not share all information, we would like that the maximum probability of guessing (i.e., of reaching any of the states $(heads_T, heads_G)$ or $(tails_T, tails_G)$) is $1/2$. This observation is fundamental in distributed systems in which components are loosely coupled and share little information with each other. (We remark that a similar example is obtained under a synchronous point of view where G performs an idling move when T tosses the coin, and T idles when G performs his choice. See Fig. 1(b).)

This phenomenon has been first observed in [15] from the point of view of compositionality and studied in [6,7,3,2,19] in many different settings, but none of them aims for automatic analysis or verification with the exception of [6], in which temporal properties are quantified over a very limited set of schedulers (the so-called *partial-information policies*).

Therefore we focus on the question of model checking but, rather than considering all schedulers to calculate maximum and minimum probabilities, we decide to consider the model checking problem under the subset of *distributed* schedulers that are obtained by composing *local* schedulers (i.e. the schedulers of each component). Notice that the "almighty" scheduler of the example would not be a valid scheduler in this new setting since the choice of G depends only on

information which is external to (and not observable by) G. Distributed schedulers has been studied in [7] in a synchronous setting and in [3] in an asynchronous setting, and are related to the partial-information policies of [6].

The contribution of this paper is to show that the quantitative model checking problem, including pCTL and LTL model checking, restricted to distributed schedulers (called *schedulers for modules* in the rest of the paper) is undecidable. We prove that there is no algorithm to compute the maximum probability of reaching a set of states using only distributed schedulers. Moreover, we show that it is not even possible to compute an approximation of such a value within a given bound. We focus our proof on the synchronous setting of [7] but later show that it also applies to [3] and [6]. Since the model considered here can be encoded in the input language of model checkers such as PRISM or Rapture, our result equally applies to this setting. In other words, the probabilities usually returned by these model checkers overapproximate the ones possible in the more realistic interpretation of distributed schedulers, and there is no way to compute or approximate these values. The proof of undecidability is based on a result of [12] stating that it is undecidable to find an approximation to the maximum probability of accepting a word in a probabilistic finite-state automaton.

Organization of the paper. Next section recalls the model of [7] appropriately simplified to meet our needs. Section 3 presents the tools we use to prove undecidability, including the result of [12] and its relation to our setting. The main proof of undecidability is given in Sec. 4 together with some lemmas which are interesting on their own. In Sec. 5 we also show that our result extends to the models of [3] and [6], discuss other related works. Sec. 6 concludes the paper.

2 Modules and Schedulers

In this section we recall the model of [7] which gives the formal framework to prove our undecidability result. [7] introduces *(probabilistic) modules* to describe open probabilistic systems. Modules can be composed forming more complex modules (though we will not focus on this). On the other hand, modules are built out of *atoms*. Each atom groups the behavior that needs to be scheduled together. Hence, modules can be used to model a distributed system, and atoms can be used to model components in these distributed systems.

Each atom controls a set of variables in an exclusive manner and is allowed to read a set of variables that it may not control. That is, each variable can be modified at any time by only one atom but read by many. The change of values of variables is done randomly according to *moves* and they can take place whenever indicated by a *transition*. Thus an atom is a set of transitions that can only read the variables that the atom may read and can only change (according to some move) the variables that the atom controls. In what follows we give the formalization of these concepts.

Definition 1 (States and moves). *Let X be a set of variables. An X-*state *s is a function mapping each variable in X to a value. An X-*move *a is a probability distribution on X-states. Given scalars $\{\delta_i\}$ and X-moves $\{a_i\}$ such that*

$\sum_i \delta_i = 1$ *and* $\delta_i \geq 0$*, we write* $\sum_i \delta_i a_i$ *for the* X*-move resulting from the convex combination, i.e.,* $\left(\sum_i \delta_i a_i\right)(s) = \sum_i \delta_i a_i(s)$.

In the rest of the paper, we assume that the set of variables is finite as well as the set of their possible values.

Definition 2 (Transitions and convex closures). *Let* X *and* Y *be two sets of variables. A* probabilistic transition (s, a) *from* X *to* Y *consists of an* X*-state* s *and a* Y*-move* a*. Given a set* S *of transitions, the convex closure of* S *(denoted by* $\text{ConvexClosure}(S)$*) is the least set containing all the transitions* $(s, \sum_i \delta_i a_i)$ *for all* $(s, a_i) \in S$ *and* δ_i *as in Def. 1.*

Definition 3 (Atoms). *A* probabilistic X-atom A *consists of a set* $\text{read}_X(A) \subseteq X$ *of* read *variables, a set* $\text{ctr}_X(A) \subseteq X$ *of* controlled *variables and a finite set* $\text{Transitions}(A)$ *of transitions from* $\text{read}_X(A)$ *to* $\text{ctr}_X(A)$.

Atoms in [7] have two sets of transitions: one like ours and another one for initialization. In order to simplify the model, we dropped this second set and consider a unique initial state provided by the module.

Since our result does not require a framework as general as the original, we exclude external and private variables out of our definitions. It is easy to see that the definition of modules below agree with that of [7] when restricted to have only interface variables.

Definition 4 (Modules). *A* probabilistic X-module P *has an initial state and a finite set* $\text{Atoms}(P)$*. We write* $\text{Var}(P)$ *for* X*. The initial state* $\text{Init}(P)$ *is a* $\text{Var}(P)$*-state.* $\text{Atoms}(P)$ *is a finite set of* $\text{Var}(P)$*-atoms such that (1)* $\text{Var}(P) = \bigcup_{A \in \text{Atoms}(P)} \text{ctr}_X(A)$ *and (2)* $\forall A, A' \in \text{Atoms}(P) \bullet \text{ctr}_X(A) \cap \text{ctr}_X(A') = \emptyset$.

The semantics of a deterministic probabilistic system (i.e., Markov chains) is given by a probability distribution on traces (called *bundle* in this context). Nondeterministic probabilistic models –such as modules– exhibit different probabilistic behavior depending on how nondeterministic choices are resolved. Hence, the semantics of a module is given by a *set* of bundles. Each bundle of this set responds to a different way of resolving nondeterminism. The resolution of nondeterminism is done by a *scheduler*. A scheduler is a function that maps traces to distribution on possible moves. Such moves are convex combinations of the available moves at the end of the trace.

Definition 5 (Traces and bundles). *Let* n *be a positive integer. An* X*-trace* σ *of length* n *is a sequence of* X*-states with* n *elements. We write* $\sigma(i)$ *for the* i*-th element of* σ *and* $\text{last}(\sigma)$ *for the last element of* σ*. In addition, we write* $\text{len}(\sigma)$ *for the length of the sequence, and* $\sigma{\downarrow_n}$ *(if* $n \leq \text{len}(\sigma)$*) for the* n-th prefix*, i.e., the sequence of length* n *in which* $(\sigma{\downarrow_n})(i) = \sigma(i)$ *for all* $1 \leq i \leq n$*. We denote the* prefix order *by* $\sigma \sqsubseteq \sigma'$ *if* $\sigma = \sigma'{\downarrow_{\text{len}(\sigma)}}$.

An X*-bundle of length* n *is a probability distribution on* X*-traces of length* n*. The unique* X*-bundle of length* 1*, which assigns the probability* 1 *to the trace consisting only of the initial state is called the* initial bundle.

Definition 6 (Schedulers). *Let X and Y be two sets of variables, and s a starting Y-state. A* scheduler *η from X to Y is a function mapping every X-trace to a probability distribution on Y-states. If η is a scheduler from X to X, then the 1-outcome of η is the bundle $\mathbf{b}_1$ assigning 1 to the trace s. In addition, for all positive integers $i > 1$, the i-outcome of η is an inductively defined $X-$bundle $\mathbf{b}_i$ of length i: the bundle $\mathbf{b}_i$ is the extension of the bundle $\mathbf{b}_{i-1}$ such that $\mathbf{b}_i(\sigma) = \mathbf{b}_{i-1}(\sigma{\downarrow_{i-1}}) \cdot (\eta(\sigma{\downarrow_{i-1}}))(\sigma(i))$ for all $X-$traces of length i. We collect the set of i-outcomes of η ($i \geq 1$) in the set* $\mathrm{Outcome}(\eta)$ *of $X-$bundles. To simplify notation, we write* $\mathrm{Outcome}(\eta)(\sigma)$ *for* $\mathbf{b}_{\mathrm{len}(\sigma)}(\sigma)$.

Schedulers are the machinery to resolve nondeterminism. They do so by assigning probabilities to the different available moves at each point of an execution (i.e., a trace). In our framework, this is done by choosing a move which is a convex combination of the available moves. A scheduler can be *deterministic* (or *non-probabilistic*) in the sense that it assigns probability 1 to a single available move. In other words, a deterministic scheduler chooses a single move from the available ones at each point of the execution. In particular, we are interested on the scheduling within a single component, that is, within an atom.

Definition 7 (Schedulers for an atom). *Consider a probabilistic $X-$atom A. The set* $\mathrm{atom}\Sigma(A)$ *of* atom schedulers *for A contains all schedulers η from* $\mathrm{read}_X(A)$ *to* $\mathrm{ctr}_X(A)$ *such that* $(\sigma(n), \eta(\sigma)) \in \mathrm{ConvexClosure}(\mathrm{Transitions}(A))$ *for all* $\mathrm{read}_X(A)$*-traces σ of length $n \geq 1$. Let* $\mathrm{atom}\Sigma^d(A)$ *be set of deterministic schedulers for A, i.e., the subset of* $\mathrm{atom}\Sigma(A)$ *such that* $(\sigma(n), \eta(\sigma)) \in \mathrm{Transitions}(A)$.

Schedulers for atoms can observe all possible executions and all the (observed) state space. On the contrary, we are *not* interested in any arbitrary scheduler for a module. In a distributed setting each component schedules its own moves disregarding any behavior that does not affect its own state space. Hence, a global scheduler can only be obtained by the combination of the schedulers of each component. Similarly, we consider that a scheduler for a module only makes sense if it comes from the composition of schedulers of each of its atoms. Schedulers for a module are obtained by taking the product of schedulers for atoms as defined in the following.

Definition 8 (Projection and Product). *Let X and $X' \subseteq X$ be two sets of variables. The X'-projection of an X-state s is the X'-state $s[X']$ such that $(s[X'])(x) = s(x)$ for all variables $x \in X'$. The X'-projection of a trace σ is an X'-trace $\sigma[X']$ in which $\sigma[X'](i) = \sigma(i)[X']$ for $1 \leq i \leq \mathrm{len}(\sigma)$.*

Let X_1 and X_2 be two disjoint sets of variables. The product *of an X_1-state s_1 and an X_2-state s_2 is the $X_1 \cup X_2$-state $s_1 \times s_2$ such that $(s_1 \times s_2)(x_1) = s_1(x_1)$ for all $x_1 \in X_1$ and $(s_1 \times s_2)(x_2) = s_2(x_2)$ for all $x_2 \in X_2$. The* product *of an X_1-move a_1 and an X_2-move a_2 is the $X_1 \cup X_2$-move $a_1 \times a_2$ such that $(a_1 \times a_2)(s) = a_1(s[X_1]).a_2(s[X_2])$.*

Definition 9 (Product of schedulers). *If η_1 is a scheduler from X_1 to Y_1, and η_2 is a scheduler from X_2 to Y_2, such that $Y_1 \cap Y_2 = \emptyset$, then the* product

is the scheduler $\eta_1 \times \eta_2$ *from* $X_1 \cup X_2$ *to* $Y_1 \cup Y_2$ *such that* $(\eta_1 \times \eta_2)(\sigma) = \eta_1(\sigma[X_1]) \times \eta_2(\sigma[X_2])$ *for all* $X_1 \cup X_2$*-traces* σ*. If* Σ_1 *and* Σ_2 *are two sets of schedulers, then* $\Sigma_1 \times \Sigma_2 = \{\eta_1 \times \eta_2 | \eta_1 \in \Sigma_1 \text{ and } \eta_2 \in \Sigma_2\}$*.*

Since we restrict to modules without external variables, our definition of scheduler for a module is simpler than that of [7]. It is easy to see that the definition of schedulers below agree with that in [7] when restricted to our setting.

Definition 10 (Schedulers for a module). *Consider a probabilistic module* P*. The set* $\mathrm{mod}\Sigma(P)$ *of* module schedulers *for* P *contains all the schedulers from* $\mathrm{Var}(P)$ *to* $\mathrm{Var}(P)$ *having* $\mathrm{Init}(P)$ *as starting state which can be written as a product of schedulers for the atoms, i.e.:* $\mathrm{mod}\Sigma(P) = \{\prod_{A \in \mathrm{Atoms}(P)} \eta_A \mid \eta_A \in \mathrm{atom}\Sigma(A)\}$*. Let* $\mathrm{mod}\Sigma^d(P)$ *be the set of deterministic schedulers for the module, i.e.,* $\mathrm{mod}\Sigma^d(P) = \{\prod_{A \in \mathrm{Atoms}(P)} \eta_A \mid \eta_A \in \mathrm{atom}\Sigma^d(A)\}$*.*

Each scheduler defines a probability space on the set of infinite traces as stated below.

Definition 11 (Extensions and probability of traces). *For each* finite *trace* σ *of length* n*, we define the* set of extensions $[\sigma]$ *to be set of* infinite *traces such that for every* $\rho \in [\sigma]$*,* $\rho(n') = \sigma(n')$ *for all* $1 \leq n' \leq n$*.*

Let be P *a probabilistic module whose variables are* $\mathrm{Var}(P)$*, and let* $\eta \in \mathrm{mod}\Sigma(P)$ *be a scheduler for* P*. The probability* $\Pr^\eta([\sigma])$ *of a set of extensions is* $\mathrm{Outcome}(\eta)(\sigma)$*. This probability can be extended in the standard way to the least* σ*-algebra over the set of infinite traces containing the extensions (see [11]).*

Given the setting in the previous definition, one can talk of the probability $\Pr^\eta(\mathrm{reach}(U))$ of reaching the set of states U under η. Such probability is given by $\Pr^\eta(\{\rho \mid \exists n \bullet \rho(n) \in U\})$.

In the following, we define several shorthands and notations that will be convenient for the rest of the paper.

Let $\mathrm{en}_A(s)$ be the set of enabled moves in a Y-state s of an X-atom A with $X \subseteq Y$; that is, the set $\{a \mid (s[\mathrm{read}_X(A)], a) \in \mathrm{Transitions}(A)\}$.

A scheduler η for an X-atom A is a function from $\mathrm{read}_X(A)$-traces to distributions on $\mathrm{ctr}_X(A)$ such that $(\mathrm{last}(\sigma), \eta(\sigma)) \in \mathrm{ConvexClosure}(\mathrm{Transitions}(A))$. That is, $\eta(\sigma) = \sum_{a \in \mathrm{en}_A(s)} \delta_a a$, where $\sum_{a \in \mathrm{en}_A(s)} \delta_a = 1$. Hence, η can be alternatively seen as a function $\eta_f : \mathrm{ctr}_X(A)\text{-moves} \times \mathrm{read}_X(A)\text{-traces} \to [0,1]$, s.t. $\eta_f(a, \sigma) = \delta_a$ for all $a \in \mathrm{en}_A(s)$, and 0 otherwise. Conversely, if $\eta_f(a, \sigma) > 0 \Rightarrow a \in \mathrm{en}_A(\mathrm{last}(\sigma))$ and $\sum_{a \in \mathrm{en}_A(\mathrm{last}(\sigma))} \eta_f(a, \sigma) = 1$, for all trace σ, η_f defines a scheduler η s.t. $\eta(\sigma) = \sum_{a \in \mathrm{en}_A(s)} \eta_f(a, \sigma)\ a$. We will use η and η_f interchangeably according to our convenience.

If η is a scheduler for the module P, we will call η_A to the scheduler for the atom A of P such that $\eta = \eta_A \times \prod_{A' \in \mathrm{Atoms}(P) \setminus \{A\}} \eta_{A'}$. If η is deterministic and $\mathrm{Atoms}(P) = A_1, \ldots, A_n$ we ambiguously denote by $\eta(\sigma)$ the n-tuple of moves $(a_1, \ldots, a_n)$ such that $\eta(\sigma)(a_1, \ldots, a_n) = 1$. Notice that $\eta_{A_i}(\sigma) = a_i = \pi_i(\eta(\sigma))$ for all $1 \leq i \leq n$.

3 The Setting for the Proof of Undecidability

In this section we present the foundations for the proof of undecidability. We recall the result in [12] stating that it is undecidable to find an approximation to the maximum probability of accepting a word in a probabilistic finite-state automaton (PFA), and present a translation of PFAs into probabilistic modules. This setting is used in Sec. 4 to prove that it is not possible to determine the maximum probability of reaching a given set of states in a module. The maximum reachability problem is formally stated as follows:

Definition 12 (Maximum Reachability Problem). *Let P be a module, let U be a set of* $\mathrm{Var}(P)$*-states, and let* $\mathrm{reach}(U) = \{\rho \mid \exists n \bullet \rho(n) \in U\}$ *be the set of all infinite traces that pass through some state in U. The* maximum reachability problem *is to determine* $\sup_{\eta \in \mathrm{mod}_\Sigma(P)} \Pr^\eta(\mathrm{reach}(U))$.

In the following, we assume that any trace that reaches some state in U remains in U with probability 1. It is a standard assumption in reachability analysis of Markov decision processes in general to make target states absorbing (see e.g. [14,1,12]). The assumption in our setting is formally stated as follows.

Assumption 1 (States in U are absorbing). *Given an instance of the maximum reachability problem, we assume that the elements in U are absorbing, in the sense that* $\forall s \in U \bullet \Pr^\eta([\sigma\, s]) = \Pr^\eta(\biguplus_{s' \in U} [\sigma\, s\, s'])$ *for all η.*

The proof presented in the next section is based on the reduction of the probabilistic finite-state automata (PFA) maximum acceptance problem [12] to the maximum reachability problem on a module. Since this problem is undecidable, this reduction implies the undecidability of the maximum reachability problem.

A PFA is a quintuple (Q, Σ, l, q_i, q_f) where Q is a finite set of *states* with $q_i, q_f \in Q$ being the *initial* and *accepting state* respectively, Σ is the *input alphabet*, and $l : \Sigma \times Q \to (Q \to [0,1])$ is the *transition function* s.t. $l(\alpha, q)$ is a distribution for all $\alpha \in \Sigma$ and $q \in Q$. Notice that l is a total function. As in [12], we assume that q_f is absorbing, i.e. $l(\alpha, q_f)(q_f) = 1$ for all $\alpha \in \Sigma$.

In the following, we present the translation of PFA into modules and directly define the probability of accepting a word in the translated model. We do so to avoid introducing a probabilistic theory on traces for a slightly different setting.

A PFA is encoded in a module P_{PFA} with two variables and two atoms. Variable *st_pfa* takes values in Q recording the current state in the PFA. Variable *symbol* takes values in $\Sigma \cup \{\mathit{initial}\}$ and is used to indicate the next input issued to the PFA. In particular, *initial* is introduced for technical matters and only used in the first transition of the module to indicate that no selection has being issued yet. Atom A encodes the transition function l. Therefore it can read both variables, but can only control *st_pfa*. It is atom B the one that controls variable *symbol* and the one that introduces the nondeterminism in the selection of the input. Notice that A is completely deterministic (in the sense that, at every state, the value of *symbol* uniquely determines the next transition to execute). Since B takes the role of the environment selecting inputs, it cannot read (nor

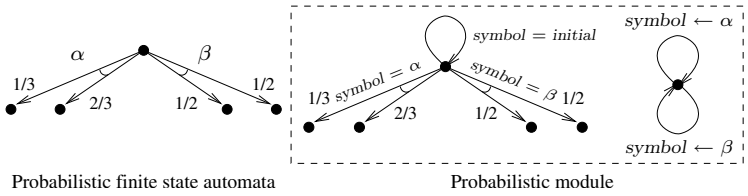

Fig. 2. From PFA to probabilistic modules

control) variable *st_pfa*. Hence, a word w over Σ is equivalent to the deterministic scheduler for B that chooses the symbols in the word.

The definition of P_{PFA} is formalized in the following.

Definition 13 (P_{PFA} and the probability of accepting a word). *Let $X = \{st\_pfa, symbol\}$. Let A be an X-atom with $\mathrm{read}_X(A) = X$ and $\mathrm{ctr}_X(A) = \{st\_pfa\}$, such that (1) $\mathrm{en}(s) = \{a_{l,s}\}$ for all s such that $s(symbol) \neq initial$, where $a_{l,s}(s') = l(s(symbol), s(st\_pfa))(s'(st\_pfa))$ for all $\{st\_pfa\}$-state s', and (2) $\mathrm{en}(s) = \{a_s\}$ for all s such that $s(symbol) = initial$, where $a_s(s) = 1$. Let B be an X-atom with $\mathrm{read}_X(B) = \mathrm{ctr}_X(B) = \{symbol\}$, and for all state s and symbol α it contains transition (s, a_α), where $a_\alpha(symbol = \alpha) = 1$. P_{PFA} is defined as the module containing atoms A and B above and having initial state s_i such that $s_i(st\_pfa) = q_i$ and $s_i(symbol) = initial$.*

Let $U = \{s \mid s(st\_pfa) = q_f\}$ be the set of accepting states. Then, the probability $\Pr(accepting\ w)$ of accepting an infinite word $w = w_1 w_2 \cdots$ of symbols from Σ is $\Pr^{\eta_A \times \eta_B}(\mathrm{reach}(U))$, where $\eta_B(\sigma) = a_{w_{\mathrm{len}(\sigma)-1}}$ (if $\mathrm{len}(\sigma) > 1$), $\eta_B(s) = a_s$ and η_A is the only possible deterministic scheduler for A.

Atom A is deterministic, since it has exactly one enabled move at every state. Hence there exists only one possible scheduler for A (the scheduler choosing the only possible move). In addition, it is also worth noting that, although we are dealing with infinite words, our criterion for acceptance is to *pass through* the accepting state using the word (i.e., a word is accepted iff a finite prefix reaches the accepting state). Figure 2 shows a simple PFA and its corresponding probabilistic module. In this figure, $symbol = \alpha$ indicates that the transition needs the value of the variable *symbol* to be α. In addition, $symbol \leftarrow \alpha$ indicates that the transition sets the value of the variable *symbol* to α.

Stated in terms of Def. 13, Corollary 3.4 in [12] states the following:

Lemma 1 (Corollary 3.4 in [12]). *For any fixed $0 < \epsilon < 1$, the following problem is undecidable: Given a module P_{PFA} as in Def. 13 such that either*

1. *P_{PFA} accepts some word with probability greater than $1 - \epsilon$, or*
2. *P_{PFA} accepts no word with probability greater then ϵ;*

decide whether case 1 holds.

[12] points out that, as a consequence of Lemma 1, the approximation of the maximum acceptance probability is also undecidable. This statement is formalized in the following corollary.

Corollary 1 (Approximation of the maximum acceptance probability is undecidable). *Given P_{PFA} as in Def. 13 and $\delta > 0$, the following problem is undecidable: find r such that $|r - \sup_w \Pr(\textit{accept } w)| < \delta$.*

4 Undecidability of the Reachability Problem for Modules

In this section, we prove the undecidability of the maximum reachability problem. Recall that the maximum reachability problem is to find the supremum over the set of all the schedulers for a given module. First of all, we show that infinite words can be seen as deterministic schedulers (Lemma 2). So, the problem of finding the supremum over the set of words is equivalent to the problem of finding the supremum over the set of *deterministic* schedulers. Next, we prove that the supremum quantifying over deterministic schedulers equals the supremum quantifying over all schedulers (Lemma 4) using the fact that, given a scheduler and a number N, a deterministic scheduler can be found which yields a larger probability until the N-th step (Lemma 3).

In the following, we prove not only the undecidability of the maximum reachability problem on modules, but also that the value of the maximum reachability probability cannot be approximated, i.e. given a certain threshold δ, there is no algorithm returning r such that $|r - \sup_{\eta \in \mathrm{mod}\Sigma(P)} \Pr^{\eta}(\mathrm{reach}(U))| < \delta$.

The following lemma states that each word in the PFA can be seen as a deterministic scheduler in P_{PFA} and vice versa.

Lemma 2 (Words and schedulers). *Given P_{PFA} as in Def. 13, each word w corresponds to a deterministic scheduler η and vice versa, in the sense that $\Pr(\textit{accepting } w) = \Pr^{\eta}(\mathrm{reach}(U))$.*

Proof. By definition, $\Pr(\text{accepting } w) = \Pr^{\eta_A \times \eta_B}(\mathrm{reach}(U))$, with η_A and η_B as in Def. 13.

Conversely, let η be a deterministic scheduler for P_{PFA}. Then $\eta = \eta_A \times \eta_B$ for some η_A (which is unique) and η_B. Note that, for any $n > 0$, there is exactly one $\{\textit{symbol}\}$-trace σ_n having probability greater than 0 and $\mathrm{len}(\sigma_n) = n$ which is defined by η_B. This is due to the fact that B has no probabilistic transitions and A cannot change the variable *symbol*. Then, take $w = w_1 \cdot w_2 \cdots$ to be the word defined by $w_n = \mathrm{last}(\sigma_{n+1})$. ($\sigma_1$ is ignored since variable *symbol* has the value *initial* in the first state.) □

As a consequence of Lemma 2 and Corollary 1 the computation of the maximum reachability probability restricted to deterministic schedulers –i.e. the computation of $\sup_{\eta \in \mathrm{mod}\Sigma^d(P)} \Pr^{\eta}(\mathrm{reach}(U))$– is an undecidable problem in general since it is undecidable for the particular case of modules obtained from PFA as in Def. 13. However, this fact does not guarantee that the problem is also undecidable when all module schedulers (not only deterministic ones) are considered (in fact, the problem *is* decidable for arbitrary global schedulers [1,5]). Our main contribution is to show that the problem is undecidable even if schedulers are not restricted to be deterministic.

The following lemma states that given a scheduler and a bound N, there is a deterministic scheduler that yields a larger probability of reaching U within the first N steps.

Lemma 3. *Given a scheduler η and $N \in \mathbb{N}$ there exists a deterministic scheduler η_d such that $\Pr^{\eta_d}(\text{reach}_N(U)) \geq \Pr^{\eta}(\text{reach}_N(U))$, where $\Pr^{\eta}(\text{reach}_N(U))$ denotes the probability of reaching U before the N-th step.*

Proof. Given an atom A^*, a $\text{read}_X(A^*)$-trace σ^* such that $\text{len}(\sigma^*) \leq N$ and a scheduler $\eta = \prod_{A \in \text{Atoms}(P)\setminus\{A^*\}} \eta_A \times \eta_{A^*}$, we find a scheduler $\det(\eta, \sigma^*)$ such that $\det(\eta, \sigma^*)$ coincides with η except for the choice corresponding to the trace σ^* in the atom A^*, in which $\det(\eta, \sigma^*)$ deterministically chooses a single action. Formally, $\det(\eta, \sigma^*)$ can be expressed in terms of η as follows: $\det(\eta, \sigma^*) = \prod_{A \in \text{Atoms}(P)\setminus\{A^*\}} \eta_A \times \eta'_{A^*}$, with $\eta'_{A^*}(\sigma^*, a^*) = 1$ for some a^* and $\eta'_{A^*}(\sigma, a) = \eta_{A^*}(\sigma, a)$ for all $\sigma \neq \sigma^*$. In addition, in the construction of $\det(\eta, \sigma^*)$ we choose a^* such that $\Pr^{\det(\eta,\sigma^*)}(\text{reach}_N(U)) \geq \Pr^{\eta}(\text{reach}_N(U))$. The core of the proof is to find such an a^*. Once obtained $\det(\eta, \sigma^*)$ for a trace σ^*, the final deterministic scheduler is calculated by repeating this process for all the local traces with length less than or equal to N.

In the following, let $k = \text{len}(\sigma^*)$ and define $r_N = \{\sigma | \, \text{len}(\sigma) = N \wedge \sigma(N) \in U\}$, $r_{N,\sigma^*} = r_N \cap \{\sigma | \sigma{\downarrow_k}[\text{read}_X(A^*)] = \sigma^*\}$ and $r_{N,\neg\sigma^*} = r_N \setminus r_{N,\sigma^*}$.

Note that, because of Assumption 1, $\Pr^{\eta}(\text{reach}_N(U)) = \Pr^{\eta}(\biguplus_{\sigma \in r_N} [\sigma])$, since $\Pr^{\eta}(\biguplus_{s'_i \in U} [\sigma s s'_1 \cdots s'_{N-(\text{len}(\sigma)+1)}]) = \Pr^{\eta}([\sigma s])$ for all $s \in U$.

Now, we start the calculations to find a^*.

$$\begin{aligned}
\Pr^{\eta}(\text{reach}_N(U)) &= \Pr^{\eta}(\biguplus_{\sigma \in r_N} [\sigma]) && \{\text{Explanation above}\}\\
&= \textstyle\sum_{\sigma \in r_N} \Pr^{\eta}([\sigma]) && \{\Pr \text{ is a measure}\}\\
&= \textstyle\sum_{\sigma \in r_{N,\sigma^*}} \Pr^{\eta}([\sigma]) + \sum_{\sigma \in r_{N,\neg\sigma^*}} \Pr^{\eta}([\sigma]) && \{\text{Commutativity}\}
\end{aligned}$$

Next, we examine the first summand. In the following calculation, let $P_{\sigma,i,A} = \sum_{a \in \text{en}_A(\sigma(i))} \eta_A(\sigma{\downarrow_i}[\text{read}_X(A)], a) \;\; a(\sigma(i+1))$.

$$\begin{aligned}
&\textstyle\sum_{\sigma \in r_{N,\sigma^*}} \Pr^{\eta}([\sigma])\\
=\;& \{\text{Definition 11}\}\\
&\textstyle\sum_{\sigma \in r_{N,\sigma^*}} \prod_{i=1}^{N-1} \prod_{A \in \text{Atoms}(P)} \sum_{a \in \text{en}_A(\sigma(i))} \eta_A(\sigma{\downarrow_i}[\text{read}_X(A)], a) \;\; a(\sigma(i+1))\\
=\;& \{\text{Arithmetics}\}\\
&\textstyle\sum_{\sigma \in r_{N,\sigma^*}} \Pr^{\eta}([\sigma{\downarrow_k}])\\
&\qquad \textstyle\big(\sum_{a \in \text{en}_{A^*}(\sigma(k))} \eta_{A^*}(\sigma{\downarrow_k}[\text{read}_X(A^*)], a) \;\; a(\sigma(k+1))\big)\\
&\qquad \textstyle\prod_{A \in \text{Atoms}(P)\setminus\{A^*\}} P_{\sigma,k,A}\\
&\qquad \textstyle\prod_{i=k+1}^{N-1} \prod_{A \in \text{Atoms}(P)} P_{\sigma,i,A}\\
=\;& \{\sigma{\downarrow_k}[\text{read}_X(A^*)] = \sigma^* \text{ (since } \sigma \in r_{N,\sigma^*}\text{), arithmetics}\}\\
&\textstyle\sum_{a \in \text{en}_{A^*}(\sigma^*(k))} \eta_{A^*}(\sigma^*, a)\\
&\qquad\qquad \textstyle\sum_{\sigma \in r_{N,\sigma^*}} \Pr^{\eta}([\sigma{\downarrow_k}]) \; a(\sigma(k+1))\\
&\qquad\qquad \textstyle\prod_{A \in \text{Atoms}(P)\setminus\{A^*\}} P_{\sigma,k,A}\\
&\qquad\qquad \textstyle\prod_{i=k+1}^{N-1} \prod_{A \in \text{Atoms}(P)} P_{\sigma,i,A}
\end{aligned}$$

$$\text{Let} \quad a^* = \arg\max_{a \in \mathrm{en}_{A^*}(\sigma(k))} \sum_{\sigma \in r_{N,\sigma^*}} \Pr^{\eta}([\sigma{\downarrow_k}])\; a(\sigma(k+1)) \prod_{A \in \mathrm{Atoms}(P) \setminus \{A^*\}} P_{\sigma,k,A} \prod_{i=k+1}^{N-1} \prod_{A \in \mathrm{Atoms}(P)} P_{\sigma,i,A}\,.$$

Then, since $\sum_{a \in \mathrm{en}_{A^*}(\sigma(k))} \eta_{A^*}(\sigma^*, a) = 1$, we have

$$\begin{aligned} \sum_{\sigma \in r_{N,\sigma^*}} \Pr^{\eta}([\sigma]) &\leq \sum_{\sigma \in r_{N,\sigma^*}} \Pr^{\eta}([\sigma{\downarrow_k}])\; a^*(\sigma(k+1)) \prod_{A \in \mathrm{Atoms}(P) \setminus \{A^*\}} P_{\sigma,i,A} \prod_{i=k+1}^{N-1} \prod_{A \in \mathrm{Atoms}(P)} P_{\sigma,i,A} \\ &= \sum_{\sigma \in r_{N,\sigma^*}} \Pr^{\eta'}([\sigma])\,, \end{aligned}$$

where η' is the scheduler that coincides with η except for trace σ^*, in which the scheduler for the atom A^* chooses a^*.

Since this change does not affect the extensions in $r_{N,\neg\sigma^*}$, we have that $\Pr^{\eta'}(r_N(U)) \geq \Pr^{\eta}(r_N(U))$. Thus, we define $\det(\eta, \sigma^*) = \eta'$.

Given a sequence of local traces $\sigma_1 \cdots \sigma_n$ (possibly belonging to different atoms), we extend the definition of det in order to handle finite sequences of traces as follows: $\det(\eta, \sigma_1 \cdots \sigma_n) = \det(\det(\eta, \sigma_n), \sigma_1 \cdots \sigma_{n-1})$.

Now, we can define a scheduler η^N being deterministic "until the N-th step" by considering the sequence $\sigma_1 \cdots \sigma_M$ comprising all local traces whose length is less or equal than N and computing $\det(\eta, \sigma_1 \cdots \sigma_M)$.

Since the choices after the N-th step do not affect the value of $\Pr(\mathrm{reach}_N(U))$, we construct the desired scheduler by taking η^N and modifying it to deterministically choose any move after the N-th step. □

Using the previous lemma, we prove that the maximum probability of reaching U is the same regardless whether it is quantified over all schedulers or only over deterministic schedulers.

Lemma 4. $\sup_{\eta \in \mathrm{mod}_{\Sigma^d}(P)} \Pr^{\eta}(\mathrm{reach}(U)) = \sup_{\eta \in \mathrm{mod}_{\Sigma}(P)} \Pr^{\eta}(\mathrm{reach}(U))$

Proof. Let $r = \sup_{\eta \in \mathrm{mod}_{\Sigma}(P)} \Pr^{\eta}(\mathrm{reach}(U))$. We prove that for every ϵ, there exists a deterministic scheduler η_ϵ such that $r - \Pr^{\eta_\epsilon}(\mathrm{reach}(U)) < \epsilon$, thus proving the lemma.

Given $\epsilon > 0$, there exists a scheduler η such that $r - \Pr^{\eta}(\mathrm{reach}(U)) < \epsilon/2$.

Note that we can write $\mathrm{reach}(U)$ as

$$\biguplus_{\{n \in \mathbb{N}_0\}} \biguplus_{\{\sigma' \mid \forall i \bullet \sigma(i) \notin U \wedge \mathrm{len}(\sigma') = n\}} \biguplus_{\{s \in U\}} [\sigma' \cdot s]\;.$$

Then, since Pr is a measure

$$\Pr^{\eta}(\mathrm{reach}(U)) = \sum_{\{n \in \mathbb{N}_0\}} \sum_{\{\sigma' \mid \forall i \bullet \sigma(i) \notin U \wedge \mathrm{len}(\sigma') = n\}} \sum_{\{s \in U\}} \Pr^{\eta}([\sigma' \cdot s])\,.$$

So, there exists N such that

$$\Pr^{\eta}(\mathrm{reach}(U)) - \sum_{n=0}^{N} \sum_{\{\sigma' \mid \forall i \bullet \sigma(i) \notin U \wedge \mathrm{len}(\sigma') = n\}} \sum_{\{s \in U\}} \Pr^{\eta}([\sigma' \cdot s]) < \epsilon/2\,.$$

By virtue of Lemma 3, we know that there exists a deterministic scheduler η_d such that $\Pr^{\eta_d}(\text{reach}_N(U)) \geq \Pr^{\eta}(\text{reach}_N(U))$. Then, $\Pr^{\eta_d}(\text{reach}(U)) \geq \Pr^{\eta}(\text{reach}(U)) - \epsilon/2$. This result yields,

$$\begin{aligned} r - \Pr^{\eta_d}(\text{reach}(U)) &= r - \Pr^{\eta}(\text{reach}(U)) + \Pr^{\eta}(\text{reach}(U)) - \Pr^{\eta_d}(\text{reach}(U)) \\ &< \epsilon/2 + \epsilon/2 = \epsilon\,. \end{aligned}$$

□

Though Lemma 4 is the basis for our undecidability result, it has a value of its own: it states that, for any probabilistic module P and reachability target U, it suffices to consider only deterministic schedulers to calculate the maximum probability of reaching some state in U. Lemma 4 yields to our main result:

Theorem 1 (Approximation of the maximum reachability problem is undecidable). *Given a probabilistic module P, a set U of states and $\delta > 0$, there is no algorithm that computes r such that $|r - \sup_{\eta \in \text{mod}\Sigma(P)} \Pr^{\eta}(\text{reach}(U))| < \delta$.*

Proof. Suppose, towards a contradiction, that the problem is decidable. Take an instance of P_{PFA} as in Def. 13. Then, using Lemmas 4 and 2, we can compute r such that

$$\begin{aligned} \delta &> |r - \sup_{\eta \in \text{mod}\Sigma(P)} \Pr^{\eta}(\text{reach}(U))| = |r - \sup_{\eta \in \text{mod}\Sigma^d(P)} \Pr^{\eta}(\text{reach}(U))| \\ &= |r - \sup_w \Pr(\text{accept } w)| \end{aligned}$$

thus contradicting Corollary 1. □

Often reachability properties are only of interest if they are compared to a probability value (e.g. the maximum probability of an error is smaller than 0.01). This kind of problems are also undecidable. If this were not the case, a procedure to calculate an approximation to the maximum reachability probability can be easily constructed using bisection (see, e.g., [13]). This result is formally stated in the following corollary.

Corollary 2. *Let $\star$ denote some operator in $\{\leq, \geq, <, >, =\}$. There is no algorithm that returns yes if $\sup_\eta \Pr^{\eta}(\text{reach}(U)) \star q$ or returns no, otherwise, for a given module P and number q.*

Moreover, there exists no algorithm that, given a module P, a number q and a threshold ϵ returns yes if $r \star q$ for some r such that $|r - \sup_\eta \Pr^{\eta}(\text{reach}(U))| < \epsilon$ or returns no, otherwise.

5 Impact and Related Work

Undecidability is frequent in problems involving control and partial information (e.g. [18]). Since a scheduler can be seen as a controller which enables appropriate moves, control is closely related to scheduling. This fact gave us a clue about the result presented in this paper. In [18], a finite state automaton can execute an action only if a set of infinite-state controllers allows it. The state of a controller (and, hence, the actions it allows to execute) is updated each time an action

observed by the controller happens. Although schedulers can be seen as infinite-state automata —since the language obtained by taking the (distributions on) moves prescribed by the scheduler is not restricted— we could not prove our result using the results in [18] by simply taking the controllers to be schedulers. Moreover, global schedulers are also infinite state automata, and the problem for these schedulers is decidable (see [5]). These facts suggest that the tractability of this problem is likely to vary from a formalism to another, although the formalisms under consideration may seem to be similar at first sight.

Unfortunately, our result also holds for Switched Probabilistic Input/Output Automata (Switched PIOA) [3,2] and for the schema of partial information presented in [6].

In the Switched PIOA formalism, the different components have input and output local schedulers, and a token is used in order to decide the next component to execute. The interleaving between different components is not resolved by the schedulers, since the way in which the token is passed is specified by the components. If a component has the token, its local output scheduler chooses a transition from a generative structure. Otherwise, its local input scheduler chooses a transition from a reactive structure, thus "reacting" to actions performed by the other components. (For definitions of reactive and generative structures see e.g. [8,16].)

The probabilistic finite-state automata in [12] can be seen as components having only reactive structures. In addition, the input scheduler for these components is uniquely determined, since each symbol uniquely determines the probability distribution for the next state. This fact allows to prove undecidability by composing the PFA (seen as a component of the Switched PIOA) with an automaton having only one state which chooses the next action to perform using generative structures (which are Dirac distributions for the sole state of the component). This component has the token during all the course of the execution. So, while this latter component chooses any action, the former component reacts to this choice as the PFA would do. Hence, these two components can simulate a probabilistic module as the one described in Def. 13. Note that we are working with a very strict subset of the Switched PIOA: there are no nondeterministic choices for the inputs, all generative structures are Dirac distributions and the token is owned by the same component during all the course of the execution. It is also worth noting that schedulers as presented in [3,2] are always deterministic because they can choose any transition in a generative or reactive structure, but they cannot choose convex combinations of these transitions. Thus, Lemma 4, and hence Lemma 3, are not needed for Switched PIOA as presented originally, and undecidability can be proved almost directly using Corollary 3.4 in [12]. Our result indicates that the problem remains undecidable even if we extend Switched PIOA with nondeterministic schedulers.

Though composition in [6] is not an issue, it presents schedulers for Markov decision processes which can observe partial portions of the states. The goal is to obtain better bounds for the probability of temporal properties. Markov decision process are defined using a set of actions in such a way that each pair

(s, a) determines the probability distribution for the next state. The schedulers are restricted as follows: given an equivalence relation $\sim$ over the set of states ($s \sim s'$ denoting that the scheduler cannot distinguish s and s'), the relation $\sigma \sim \sigma'$ over traces is defined to hold iff $\text{len}(\sigma) = \text{len}(\sigma')$ and for all i, $\sigma(i) \sim \sigma'(i)$. Then, a partial-information scheduler is required to satisfy $\eta(\sigma, a) = \eta(\sigma', a)$ if $\sigma \sim \sigma'$. Note that, by taking $\sim$ such that $s \sim s'$ for all s, the scheduler must decide the next action to perform based solely on the amount of actions chosen before. Then, using such a relation $\sim$, a scheduler in [6] is equivalent to a scheduler for the atom B as Def. 13, and hence the problem of finding the maximum reachability probability is equivalent to the problem of finding the maximum reachability probability for a module as in Def. 13.

We remark that [6] defines a model checking algorithm, but it calculates the supremum corresponding to *Markovian* partial-information policies, i.e., to the subset of partial-information policies restricted to choose (distributions on) actions by reading only the (corresponding portion of the) current state rather than the full past history. Quantitative model checking on MDP was originally introduced in [1,5] but for arbitrary schedulers. In particular, [5] proves that the maximum reachability problem under *arbitrary global* schedulers has an equivalent solution under *deterministic global* schedulers (in fact, they are also *Markovian* in the sense that only depend on the last state and not of the full trace, see Theorem 3.5 in [5]). Though this result is similar to Lemma 4, the proof of [5] has no connection to ours. In fact, the construction of the deterministic scheduler in [5] is also the proof that the problem for the general case is decidable.

6 Conclusion

We have argued that usual quantitative model checkers yields overestimations of the extremum probabilities in distributed programs and proposed to address model checking under the restriction of schedulers that are compatible with the expected behaviour of distributed systems. We showed that it is undecidable to compute the maximum probability of reaching a state when restricted to distributed schedulers, hence making the proposal unfeasible in its generality. Morever, we showed that such value cannot even be aproximated.

On proving undecidability, we needed to prove additional lemmas. In particular, we believe that the result of Lemma 4 has to be remarked, but mostly, that its proof technique, including the constructive proof of Lemma 3, is of relevance and can be reused in searching similar results.

The combination of our undecidability result and the NP-hardness result of [6] is not encouraging on seeking algorithms for model checking under distributed or partial-information schedulers. Yet, the observation that arbitrary schedulers yield overestimations of probability values remains valid. The question then is whether it is possible to find a proper subset of schedulers (surely including all distributed schedulers) that yields a tighter approximation of extremum probabilities while keeping tractability of the model checking problem.

Another question is to which extent the calculation (or approximation) of the *minimum* probability of reaching a state is also undecidable. Though this

is the dual problem to that in Def. 12, we could not obtain a straightforward dualization of our proof.

Acknowledgement. We would like to thank Peter Niebert who has been an excelent sparring during the development of these ideas, and Holger Hermanns for his most valuable feedback.

References

1. Bianco, A., de Alfaro, L.: Model checking of probabilistic and nondeterministic systems. In: Thiagarajan, P.S. (ed.) Foundations of Software Technology and Theoretical Computer Science. LNCS, vol. 1026, pp. 288–299. Springer, Heidelberg (1995)
2. Cheung, L.: Reconciling Nondeterministic and Probabilistic Choices. PhD thesis, Radboud Universiteit Nijmegen (2006)
3. Cheung, L., Lynch, N., Segala, R., Vaandrager, F.W.: Switched Probabilistic PIOA: Parallel composition via distributed scheduling. Theoretical Computer Science 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: Proc. of QEST'06, pp. 131–132. IEEE Computer Society Press, Los Alamitos (2006)
5. de Alfaro, L.: Formal Verification of Probabilistic Systems. PhD thesis, Stanford University (1997)
6. de Alfaro, L.: The verification of probabilistic systems under memoryless partial-information policies is hard. In: Proc. of PROBMIV 99. Tech. Rep. CSR-99-8, pp. 19–32. Univ. of Birmingham (1999)
7. 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)
8. van Glabbeek, R.J., Smolka, S.A., Steffen, B.: Reactive, generative, and stratified models of probabilistic processes. Information and Computation 121, 59–80 (1995)
9. Hinton, A., Kwiatkowska, M., Norman, G., Parker, D.: PRISM: A tool for automatic verification of probabilistic systems. In: Hermanns, H., Palsberg, J. (eds.) TACAS 2006 and ETAPS 2006. LNCS, vol. 3920, pp. 441–444. Springer, Heidelberg (2006)
10. Jeannet, B., D'Argenio, P.R., Larsen, K.G.: Rapture: A tool for verifying Markov Decision Processes. In: I. Cerna, editor, Tools Day'02, Brno, Czech Republic, Technical Report. Faculty of Informatics, Masaryk University Brno (2002)
11. Kemeny, J.G., Snell, J.L., Knapp, A.W.: Denumerable Markov Chains. Van Nostrand Company (1966)
12. 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)
13. Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T.: Numerical Recipes in FORTRAN: The Art of Scientific Computing, 2nd edn. pp. 343–347. Cambridge University Press, Cambridge (1992)
14. Puterman, M.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley, Chichester (1994)
15. Segala, R.: Modeling and Verification of Randomized Distributed Real-Time Systems. PhD thesis, Massachusetts Institute of Technology (1995)

16. Sokolova, A., de Vink, E.P.: Probabilistic automata: system types, parallel composition and comparison. In: Baier, C., Haverkort, B., Hermanns, H., Katoen, J.-P., Siegle, M. (eds.) Validation of Stochastic Systems. LNCS, vol. 2925, pp. 1–43. Springer, Heidelberg (2004)
17. Stoelinga, M.: Alea jacta est: Verification of Probabilistic, Real-time and Parametric Systems. PhD thesis, Katholieke Universiteit Nijmegen (2002)
18. Tripakis, S.: Undecidable problems of decentralized observation and control. In: Proc. 40th IEEE Conference on Decision and Control, vol. 5, pp. 4104–4109 (2001)
19. Varacca, D., Nielsen, M.: Probabilistic Petri nets and mazurkiewicz equivalence 2003 (Unpublished draft)