Project no.:	**PIRSES-GA-2011-295261**
Project full title:	**Mobility between Europe and Argentina applying Logics to Systems**
Project Acronym:	**MEALS**
Deliverable no.:	**1.3 / 1**
Title of Deliverable:	**Distributed Probabilistic Input/Output Automata: Expressiveness, (Un)Decidability and Algorithms**

Contractual Date of Delivery to the CEC:	**30-Sep-2013**
Actual Date of Delivery to the CEC:	**30-Sep-2013**
Organisation name of lead contractor for this deliverable:	**UNC**
Author(s):	**Sergio Giro, Pedro R. D'Argenio, Luis María Ferrer Fioriti**
Participants(s):	**SAU, RWT, TUD, TUE, UNC, UBA**
Work package contributing to the deliverable:	**WP1**
Nature:	**R**
Dissemination Level:	**Public**
Total number of pages:	**31**
Start date of project:	1 Oct. 2011 **Duration:** 48 month

Abstract:

Probabilistic 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 over-estimations 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 introduced definitions that characterise which are the schedulers that properly capture the idea of distributed behaviour in probabilistic and nondeterministic systems modeled as a set of interacting components.
In this article, we provide an overview of the work we have done in the last years which includes: (1) the definitions of distributed and strongly distributed schedulers, providing motivation and intuition; (2) expressiveness results, comparing them to restricted versions such as deterministic variants or finite memory variants; (3) undecidability results —in particular the model checking problem is not decidable in general when restricting to distributed schedulers; (4) a counterexample guided refinement technique that, using standard probabilistic model checking, allows to increase precision in the actual bounds in the distributed setting; and (5) a revision of the partial order reduction technique for probabilistic model checking. We conclude with an extensive review of related work dealing with similar approaches to ours.

Note:

This deliverable is based on material that will be published in Theoretical Computer Science (http://dx.doi.org/10.1016/j.tcs.2013.07.017)

This project has received funding from the European Union Seventh Framework Programme (FP7 2007-2013) under Grant Agreement Nr. 295261.

Contents

1 Introduction

Markov decision processes (MDPs) are widely used in diverse fields ranging from ecology to computer science. They are useful to model and analyse systems in which both probabilistic and nondeterministic choices interact. MDPs can be automatically analysed using quantitative model checkers such as PRISM [24] or LiQuor [10].

Since MDPs contain nondeterministic choices (in addition to probabilistic steps), the model checking problem is to find out the largest or smallest probability of reaching a goal under any possible resolution of the nondeterministic choices, a concrete instance being "the probability of arrival of a package is at least 0.95 no matter how the package is routed". The resolution of such nondeterminism is given by the so called *schedulers* (called also adversaries or policies –see e.g. [4, 28]) which choose an enabled transition after each finite execution path of the system.

The available tools for model checking such as PRISM [24] or LiQuor [10] calculate the worst-case probability considering all possible schedulers. However, in distributed systems, some schedulers correspond to unrealistic resolutions of the nondeterminism (as we illustrate below) thus resulting in overly pessimistic worst-case probabilities. A restricted class of schedulers was proposed to cope with this problem in previous literature –see e.g. [13, 9, 8, 12, 16]. We call these schedulers *distributed schedulers*, since in these settings there is a *local* scheduler for each component and so the resolution of the nondeterminism is *distributed* among the different components.

In this paper, we investigate different subclasses of distributed schedulers in order to answer to which extent these subclasses are able to attain the worst-case probability. The subclasses we consider are strongly related to the development of techniques for MDP analysis. As an example, if the class of all schedulers is considered, worst-case probabilities of reachability properties are attained by schedulers that are both Markovian –i.e. the decision is based on the current state of the execution, disregarding the previous history– and deterministic –i.e. the schedulers themselves have no probabilistic choices, see [4]. The existence of this subclass ensures that the worst-case probability can be found by exhaustive search (notice that more efficient methods exist [4]). Hence, one may like to know to which extent these results hold in case the schedulers are restricted to be distributed.

1.1 Unrealistic worst cases and distributed schedulers

A scheduler is a function mapping paths to transitions (or, in the more general case, paths to distributions on transitions). Given that the execution up to some state s is known (namely, the history path), the scheduler "chooses" to perform one transition out of all transitions enabled in state s.

The following example illustrates the problem that motivates the introduction of distributed schedulers: a man tosses a coin and another one has to guess heads or tails. Figure 1 depicts the models of these two men in terms of MDPs. Man T, who tosses the coin, has only one transition which represents the toss of the coin: with probability $\frac{1}{2}$ he moves to state $heads_T$ and with probability $\frac{1}{2}$ he moves to state $tails_T$. Instead, man G has two possible transitions, each one representing his choice: $heads_G$ or $tails_G$. An *all-knowing* scheduler for this system may let

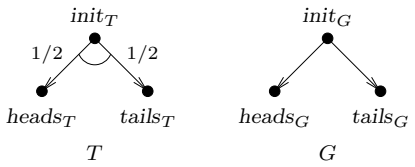


Figure 1: T tosses a coin and G has to guess

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 transition leading to heads if T tossed a head or the transition leading to tails if T tossed a tail. Therefore, the supremum probability of guessing obtained by quantifying over these all-knowing schedulers is 1, even if T is a smart player that always hides the outcome until G reveals his choice. As a consequence, quantitative model checkers based on [4], though safe, yield an overestimation of the correct value. In this example, in which T and G do not share all information, we would like that the supremum probability of guessing (i.e., of reaching any of the states $(heads_T, heads_G)$ or $(tails_T, tails_G)$) is $\frac{1}{2}$.

This observation is fundamental in distributed systems in which components share little information with each other, as well as in security protocols, where the possibility of information hiding is a fundamental assumption [6]. Similar phenomena to the one we illustrated have been observed in [28] from the point of view of compositionality and studied in [12, 13, 9] in different settings. Distributed schedulers are also related to the partial-information policies of [12].

In order to avoid considering these unrealistic behaviours, *distributed* schedulers were proposed in previous literature. *Local* schedulers for each component of the system are defined in the usual way (that is, the choices are based on the complete history of the component) and distributed schedulers are defined to be the schedulers that can be obtained by composing these local schedulers. We remark that the "all-knowing" scheduler of the example would not be a valid scheduler in this new setting since the choice for G depends on information which is external to (and not observable by) G. In contrast, a local scheduler for G takes the decision having no information about the actual state of T, and so the choice cannot conveniently vary according to the outcome of T.

Previous work in the area either deals with nondeterminism in a unique manner (regardless whether it originates from local choices or from the interleaving) or simply focuses on local choices avoiding the resolution of interleaving nondeterminism (either by assumming full synchronization [13] or by model construction [9]; see Sec. 7 for a detailed comparison). If we allow interleaving nondeterminism, the schedulers can also be restricted to handle this nondeterminism in a realistic way. So, we motivate a restriction to distributed schedulers in this direction, and define the *strongly distributed* schedulers as the schedulers complying with such restriction.

1.2 Overview of the paper

This article surveys the state of the art of model checking for distributed probabilistic systems modeled as a network of interconnecting probabilistic I/O automata. It collects and summarizes the work that we have done in the last years since we discovered the undecidability of the model

checking problem in the general setting [16, 20, 17, 21, 18, 15, 22]. The article spans from the basic motivations and definitions to expressiveness, undecidability issues and model checking algorithms. More precisely:

(1) We introduce the concept of distributed scheduler which is constructed from two types of schedulers: local schedulers, which randomly resolve the non-determinism of a single component based only on the local knowledge of the execution, and the interleaving scheduler, which randomly determines the next component to produce an output transition. We impose restrictions on the interleaving scheduler to capture the exact notion of distributed behaviour. Formal definitions, motivation and intuition are discussed.

(2) It is known that memoryless deterministic schedulers suffice to verify reachability properties in classical probabilistic model checking. We show that this is not the case for (strongly) distributed schedulers and analyze comparatively the expressiveness of restricted variants including deterministic and finite memory schedulers.

(3) We discuss several undecidability issues. In particular the reachability problem under (strongly) distributed schedulers is neither decidable nor approximable within a given error bound (and hence neither is the model checking problem in general). Not even is decidable the problem to verify if a set of states is reachable with probability 1. Besides, the problem of model checking under bounded memory schedulers (i.e. schedulers that decide based on the last nth steps, for a fix n) is shown to be NP-hard.

(4) We discuss a counterexample guided refinement algorithm based on classical techniques that in each successive refinement increases the precision on the bounds of the actual probability bounds of LTL and PCTL* properties under distributed schedulers. The algorithm verifies the property using classical probabilistic model checking techniques [4]. If the property is violated, the counterexample scheduler is analyzed. If it is a distributed scheduler then the property is indeed false. If it does not meet the conditions to be distributed, the counterexample is consider spurious and the model is refined accordingly.

(5) We present a revision of the partial order reduction technique for probabilistic model checking. This technique was introduced in [2, 11] where Peled's original conditions were extended with an extra condition that guarantees that the results of classical probabilistic model checking are preserved. We show that this new condition is specially tailored to preserve the values of non-distributed schedulers and can be weakened for distributed schedulers, or even eliminated if we want to preserve only strongly distributed schedulers.

To have a complete overview, we also discuss related work, surveying on other approaches to distributed schedulers or to partial information models, their use on security, and new algorithms for model checking restricted sets of properties under (strongly) distributed schedulers.

A generative structure with two transitions A reactive structure with two transitions

Figure 2: Reactive and generative structures

2 Interleaved Probabilistic Input/Output Automata

We present a framework based on the Switched PIOA [9] (see Sec. 7 for a detailed comparison). It uses reactive and generative structures (see [23, 29]). For a finite set S, we denote by $\mathsf{Dist}(S)$ the set of all the probability distributions over the set S. Given a set ActLab of action labels and a set St of states, the set of generative transitions T_G on $(\mathsf{St}, \mathsf{ActLab})$ is $\mathsf{Dist}(\mathsf{St} \times \mathsf{ActLab})$, and the set T_R of reactive transitions is $\mathsf{Dist}(\mathsf{St})$. A generative structure on $(\mathsf{St}, \mathsf{ActLab})$ is a function $G : \mathsf{St} \to \mathcal{P}(T_G)$ and a reactive structure on $(\mathsf{St}, \mathsf{ActLab})$ is a function $R : \mathsf{St} \times \mathsf{ActLab} \to T_R$. Figure 2 depicts an example of these structures. Generative transitions model both communication and state change. The component executing a generative transition chooses both a label a to output (the ! indicates that the label is output) and a new state s according to a given distribution. Reactive transitions specify how a component reacts to a given input (the ? represents input). Since the input is not chosen, reactive transitions are simply distributions on states.

In our framework, a system is obtained by composing several *probabilistic I/O atoms*. Each atom is a probabilistic automaton having reactive and generative transitions.

Definition 1. A probabilistic I/O atom is a 5-tuple $(\mathsf{St}, \mathsf{ActLab}, G, R, \mathsf{init})$, where St is a finite set of states, ActLab is a finite set of actions labels, and G and R are a generative and a reactive structure on $(\mathsf{St}, \mathsf{ActLab})$, respectively. $\mathsf{init} \in \mathsf{St}$ is the initial state. Since $R : \mathsf{St} \times \mathsf{ActLab} \to T_R$ is a (total) function, an atom is input deterministic and input enabled.

An interleaved probabilistic I/O system (IPIOA) P is a finite set $\mathrm{Atoms}(P)$ of probabilistic I/O atoms $A_1, \cdots, A_N$. The set of states of the system is $\prod_i \mathsf{St}_i$, and the initial state of the system is $\mathsf{init} = (\mathsf{init}_1, \cdots, \mathsf{init}_N)$. We often write St_i to denote the set of states of an atom A_i and similarly for the other elements of the 5-tuple. In addition, we write T_{G_i} and T_{R_i} for the set of generative and reactive transitions on $(\mathsf{St}_i, \mathsf{ActLab}_i)$, respectively.

In order to define how the system evolves, we define *compound transitions*, which are the transitions performed by the system as a whole. In such compound transitions, all the atoms having the same action label in their alphabet must synchronise and *exactly one* of them must participate with an output (generative) transition (thus modelling multicasting). Formally, a compound transition is a tuple $c = (g_i, a, r_{j_1}, \cdots, r_{j_m})$ (we require $i \neq j_k$ and $j_k \neq j_{k'}$ for all $k \neq k'$) where g_i is a generative transition in the atom A_i (the *active* atom), $a \in \mathsf{ActLab}_i$ is an action label, the r_{j_k} are reactive transitions in the atoms A_{j_k} (the *reactive* atoms) and $\{A_i, A_{j_1}, \cdots, A_{j_m}\}$ is equal to the set $\{A_j \mid a \in \mathsf{ActLab}_j\}$. We denote this set by $\mathsf{Inv}(c)$ and say that A_i, A_{j_1}, …, A_{j_m} are the atoms *involved* in the compound transition. A compound transition $(g_i, a, r_{j_1}, \cdots, r_{j_m})$ is enabled in a given state $(s_1, \cdots, s_N)$ if $g_i \in G_i(s_i)$ and $R_{j_k}(s_{j_k}, a) = r_{j_k}$. The action label a of a compound transition c is indicated by $\mathsf{label(c)}$. The probability $c(s, s')$ of reaching a state $s' = (s'_1, \cdots, s'_N)$ from a state $(s_1, \cdots, s_N)$ using a compound transition $c = (g_i, a, r_{j_1}, \cdots, r_{j_m})$

is $g_i(s'_i, a) \cdot \prod_{k=1}^{m} r_{j_k}(s'_{j_k})$ if $s_t = s'_t$ for every atom not involved in the transition. Otherwise, $c(s, s') = 0$.

To avoid technical complications, we assume that at every state in the system P there is some compound transition enabled. Therefore, for every reachable state s of P, there is an atom A_j such that $G_j(s_j) \neq \emptyset$.

A path σ of P is a sequence $s_1.c_1.s_2.c_2 \cdots c_{n-1}.s_n$ where each s_i is a (compound) state and each c_i is a compound transition such that $s_1 = \mathsf{init}$, c_i is enabled in s_i and $c_i(s_i, s_{i+1}) > 0$. A path can be finite or infinite. For a finite path σ as before, the *set of extensions* (denoted by $[\sigma]$) contains all the infinite paths starting with σ. In addition, we define $\text{last}(\sigma) = s_n$ and $\text{len}(\sigma) = n$.

In the remaining of the paper, we suppose that input-enabled atoms $A_1, \ldots, A_N$ are given, and we are considering the system P comprising all the atoms A_i. We call this system "the compound system". The states of the compound system are called global states and the states of each atom are called local states. Similarly, we use the notion of global and local paths whenever we refer to paths of the compound system or path of an atom, respectively.

3 Schedulers

The probability of a set of executions depends on how the nondeterminism is resolved. A scheduler transforms a nondeterministic choice into a probabilistic choice by assigning probabilities to the available transitions. Given a system and a scheduler, the probability of a set of executions is completely determined.

In the usual MDP setting, schedulers assign probabilities to the enabled transitions taking into account the complete history of the system, and hence history-dependent schedulers are defined as functions mapping paths to distributions on transitions. As we have seen it may be unrealistic to assume that the schedulers are able to see the full history of all the components in the system. In the following, we define a restricted class of schedulers in order to avoid considering unrealistic behaviours.

3.1 Distributed schedulers

In a distributed setting as the one we are introducing, different kinds of nondeterministic choices need to be resolved. We need to decide what is the next atom to execute a generative transition. And each atom needs a corresponding *output* scheduler to choose among generative transitions enabled. Notice that, since atoms are input enabled and input deterministic, there is no need to schedule reactive transitions. They will only take place if another atom produces the corresponding output. An output scheduler is able to make its decisions based only on the local history of the atom. So, we need the notion of *projection*.

Given a path σ, the projection $\sigma[i]$ of the path σ over an atom A_i is defined inductively as follows: **(1)** $(\mathsf{init}_1, \cdots, \mathsf{init}_N)[i] = \mathsf{init}_i$, **(2)** $\sigma.c.s[i] = \sigma[i]$ if A_i is not involved in c, and **(3)** $\sigma.c.s[i] = \sigma[i]\,.\mathsf{label(c)}.\pi_i(s)$, otherwise (where π_i denotes the i-th projection of a tuple). The set of all the projections of paths over an atom A_i is denoted by $\mathsf{Proj}_i(P)$. We say that these projections are the *local paths* of A_i.

An output scheduler for the atom A_i is a function $\Theta_i : \mathsf{Proj}_i(P) \to \mathsf{Dist}(T_{G_i})$ such that, if $G_i(\text{last}(\sigma[i])) \neq \emptyset$ then $\Theta_i(\sigma[i])(g) > 0 \implies g \in G_i(\text{last}(\sigma[i]))$. Note that, if the output scheduler Θ_i fixes a generative transition for a given local path σ, then the actions in the generative transition can be executed in every global path whose projection to i is σ, since we require the atoms to be input-enabled.

We still need to resolve the nondeterministic choice concerning the next atom to perform an output. An interleaving scheduler is a map that probabilistically chooses an active atom for each (global) history. This atom will be the next to execute a generative transition (this transition, in turn, is chosen according to the output scheduler). Formally, an *interleaving scheduler* is a function $\mathcal{I} : \mathsf{Paths}(P) \to \mathsf{Dist}(\{1, \cdots, N\})$ such that $\mathcal{I}(\sigma)(i) > 0 \implies G_i(\text{last}(\sigma[i])) \neq \emptyset$. Note that, even if interleaving schedulers are unrestricted, compound schedulers for the compound system are still restricted, since the output schedulers can only see the portion of the history corresponding to the component.

A scheduler for the compound system is obtained by the appropriate composition of the interleaving scheduler and the output scheduler of each atom.

Definition 2. Given an interleaving scheduler $\mathcal{I}$ and output schedulers Θ_i for each atom i, the distributed scheduler η obtained by composing $\mathcal{I}$ and Θ_i is defined as:

$$\eta(\sigma)(g_i) = \mathcal{I}(\sigma)(i) \cdot \Theta_i(\sigma[i])(g_i)$$

Unless stated otherwise, when referring to schedulers for IPIOA we are considering these distributed schedulers.

The probability of the sets of the form $[\sigma]$ is inductively defined as follows:

$$\begin{aligned} \Pr^\eta([\mathsf{init}]) &= 1 \\ \Pr^\eta([\sigma.c.s)]) &= \Pr^\eta(\sigma) \cdot \eta(\sigma)(g_i) \cdot c(\text{last}(\sigma), s) \quad \text{if } c = (g_i, a, r_{j_1}, \cdots, r_{j_m}) \end{aligned}$$

Note that, if $c = (g_i, a, r_{j_1}, \cdots, r_{j_m})$, then

$$\eta(\sigma)(c) \cdot c(s, s') \;=\; \mathcal{I}(\sigma)(i) \cdot \Theta_i(\sigma[i])(g_i) \cdot g_i(s'_i, a) \cdot \textstyle\prod_{k=1}^{m} r_{j_k}(s'_{j_k}) \;\;,$$

which implies $\sum_{c,s'} \eta(\sigma)(c) \cdot c(\text{last}(\sigma), s') = 1$. This probability can be extended to the least σ-field containing all the sets of extensions in the standard way (see e.g. [4]). We say that the sets in such σ-field are *measurable*. Given a measurable set S, we are interested in the value $\sup_\eta \Pr^\eta(S)$. By calculating this amount it can be answered, for instance, whether or not "the probability of a package loss is at most 0.05 no matter how the package is routed". This property, in particular, is what we call a *reachability property*: we are interested in the set of paths in which some states are reached (namely, the states in which the package has been lost). Given a set U of states, we denote by $\Pr^\eta(\mathsf{F}\, U)$ the probability of reaching any state in U.

Deterministic schedulers We defined schedulers so that they map into distributions on transitions. We say that a scheduler is *deterministic* if all the choices in the interleaving and output schedulers choose one atom and one generative transition with probability 1, respectively. A

scheduler that is not deterministic is said to be *randomized*. That is, for all finite path σ and atom A_i, $\Theta_i(\sigma[i])(g_i) > 0 \implies \Theta_i(\sigma[i])(g_i) = 1$, and $\mathcal{I}(\sigma)(i) > 0 \implies \mathcal{I}(\sigma)(i) = 1$.

Given a deterministic output scheduler Θ we write $\Theta(\sigma) = g$ to indicate that $\Theta(\sigma)(g) = 1$, and similarly for the interleaving scheduler.

Compared to general schedulers, deterministic schedulers are easier to manipulate in proofs. Theorem 4 ensures that it suffices to consider deterministic schedulers. The proof of the theorem uses limit schedulers, as defined below.

Limit schedulers Several properties of schedulers can be proven using *limit schedulers*. Given a sequence $Q = \eta^1, \eta^2, \cdots$ of distributed schedulers, a limit for Q is a scheduler η^* such that for every N, there exist infinitely many k such that η^k and η^* coincide for all paths σ with length less than or equal to N, in symbols:

$$\forall N\colon \exists k^1, k^2, \cdots\colon \forall i\colon \forall \sigma\colon \ \mathrm{len}(\sigma) \leq N \Longrightarrow \eta^{k^i}(\sigma) = \eta^*(\sigma)\ . \tag{1}$$

Theorem 1. *Every sequence $Q = \eta^1, \eta^2, \cdots$ of deterministic schedulers has a limit.*

Proof. We start by constructing a sequence $Q^1 = \eta^{k^1}, \eta^{k^2}, \cdots$ such that there exists g such that $\eta^{k^x}(\mathsf{init}) = g$ for all k^x. The construction of Q^1 shows how to construct a sequence Q^{N+1} from Q^N. Finally, these sequences are used to construct η^*.

To construct Q^1, fix an atom A, and consider the output schedulers Θ^k_A corresponding to η^k. In particular, consider the choices $\Theta^k_A(\mathsf{init}_A)$. Since there are finitely many generative transitions enabled, there exists some g_A and infinitely many k^x such that $\Theta^{k^x}_A(\mathsf{init}_A) = g_A$. We can take the subsequence comprising all such k^x, and repeat the same reasoning for each different atom thus yielding a subsequence such that, for all atoms A we have $\Theta^{k^x}_A(\mathsf{init}_A) = g_A$. We can repeat the reasoning for the interleaving scheduler. As a result, we have a sequence Q^1 such that $\eta^{k^x}(\mathsf{init}) = g_A$ for all x. Since init is the only path of length 1, we can use the sequence Q^1 in Eqn. 1 for $N = 1$, if we define $\eta^*(\mathsf{init}) = g_A$. Starting from Q^1 we can construct Q^2, which satisfies the condition for $N = 2$ and in general, Q^{N+1} is obtained from Q^N.

We define η^* as $\eta^*(\sigma) = \eta^{\mathrm{len}(\sigma)}(\sigma)$ where, for all N, $\eta^{\mathrm{len}(\sigma)}$ is some scheduler in Q^N. The condition for limits holds: for all N, the sequence Q^N provides the $k^1, k^2, \cdots$ for Eqn. 1. □

3.2 Strongly distributed schedulers

Distributed schedulers model the fact that components can only look at their local history to choose the next transition to perform. However, under distributed schedulers, it is still possible that the hidden state of a component affects the behaviour of an unrelated group of components.

We explain how this leak of information occurs using atoms depicted in Fig. 3. Consider the system P having atoms T, Z, A, B. In this system, T is a process that tosses a coin. For the labels $h!$ and $t!$ corresponding to heads and tails, we have $h!, t! \notin \mathsf{ActLab}_Z \cup \mathsf{ActLab}_A \cup \mathsf{ActLab}_B$. So, according to this model, T keeps the outcome as a secret (coins whose output are assumed to be secrets can be found in probabilistic security protocols such as the solution to the dining cryptographers problem, see [7]). Atom Z models an attacker trying to guess the outcome of the coin. Atoms A and B are two processes that Z is able to observe.

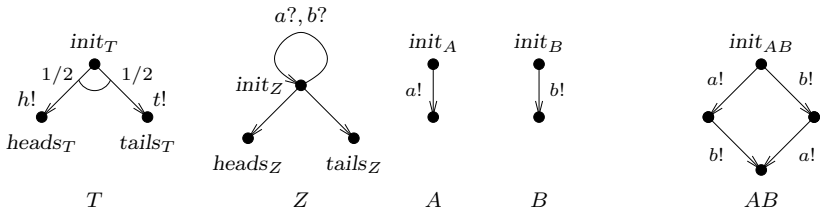


Figure 3: Motivating strongly distributed schedulers

Consider the maximum probability that attacker Z guesses the outcome (that is, the probability of reaching a state of the form $(heads_T, heads_Z, \cdots)$ or $(tails_T, tails_Z, \cdots)$). Since the attacker is able to see only the actions of A and B (and these atoms cannot, in turn, see the outcome of T) the attacker has no information about T, and so the maximum probability should be $1/2$. Unfortunately, there exists a distributed scheduler that yields probability 1: the interleaving scheduler chooses T in the first place, and then it chooses either (A and then B) or (B and then A), according to the outcome of the probabilistic transition. Finally, the interleaving scheduler chooses Z. The order in which $a!$ and $b!$ were output is part of the local history of Z, so the output scheduler for Z can always choose the transition agreeing with the outcome of the coin.

Note that the leak of information arises from the fact that the interleaving scheduler can look at the complete history of the system. In the following we derive restrictions on interleaving schedulers that prevent the leak presented above. Then, *strongly distributed schedulers* are defined as distributed schedulers whose interleaving scheduler complies with such condition.

In the example above, the state of T affects the execution of atoms A and B. Distributed schedulers were defined in such a way that the state of an atom cannot affect the execution of another atom. Note that, if we regard the parallel composition between A and B as a single component AB (with the composition as in, for instance, [8, p. 99]), we end up in a situation very similar to the one depicted in Fig. 1: in the case in which the coin lands heads AB chooses to perform the transition $a!$, while in the other case it chooses to perform the transition $b!$. In fact, if we consider the system P' such that $\text{Atoms}(P') = \{T, Z, AB\}$, no output scheduler for AB can be defined in such a way that the order of execution of $a!$ and $b!$ depends on the outcome of T (since the outcome of T does not affect the state of AB). Then, there is no distributed scheduler for P' that can simulate the behaviour in P in which Z guesses all the time. Therefore, we would like that the new scheduler works just like distributed schedulers would do when A and B are considered as a single atom.

Let P be a compound system containing atoms A and B. Let AB be a single atom representing the composition of A and B and P' another compound system such that $\text{Atoms}(P') = (\,\text{Atoms}(P) \setminus \{A, B\}\,) \cup \{AB\}$. In general, we want to restrict to interleaving schedulers such that, for every distributed scheduler η on P complying to such restriction, there is a distributed scheduler η' on P' that defines the same probabilistic behaviour.

To motivate the restriction, consider a scheduler for the system P with T, A and B in Fig. 3. Consider a distributed scheduler η whose interleaving scheduler complies $\mathcal{I}(\mathsf{init}) = (\frac{1}{2}T + \frac{2}{6}A + \frac{1}{6}B)$. We seek a restriction on $\mathcal{I}$ s.t. it is possible to find a distributed scheduler for P' containing atoms T and AB in Fig. 3. When AB is in state $(\mathsf{init}_A, \mathsf{init}_B)$, the output scheduler Θ_{AB} chooses

a distribution on $\{a!, b!\}$. To respect the choice of $\mathcal{I}$ in P, it must hold that $\Theta_{AB}(\mathsf{init}_{AB})(a!) = 2 \cdot \Theta_{AB}(\mathsf{init}_{AB})(b!)$, since, according to $\mathcal{I}$, the probability of executing $a!$ is twice the probability of executing $b!$. Then,

$$\Theta_{AB}(\mathsf{init}_{AB})(a!) = \tfrac{2}{3} \quad \text{and} \quad \Theta_{AB}(\mathsf{init}_{AB})(b!) = \tfrac{1}{3} \ . \tag{2}$$

Consider the path $\sigma = (\mathsf{init}_T, \mathsf{init}_A, \mathsf{init}_B) \xrightarrow{t!} (\mathit{heads}_T, \mathsf{init}_A, \mathsf{init}_B)$ in P. The corresponding path in P' is $\sigma' = (\mathsf{init}_T, \mathsf{init}_{AB}) \xrightarrow{t!} (\mathit{heads}_T, \mathsf{init}_{AB})$.

Since $\sigma'[AB] = \mathsf{init}_{AB} = (\mathsf{init}_T, \mathsf{init}_{AB})[AB]$, we have that

$$\Theta_{AB}((\mathsf{init}_T, \mathsf{init}_{AB})[AB])(a!) = \Theta_{AB}(\sigma'[AB])(a!) = \Theta_{AB}(\mathsf{init}_{AB})(a!) = \tfrac{2}{3}$$

and similarly for $b!$. Therefore $\Theta_{AB}(\sigma'[AB])(a!) = 2\Theta_{AB}(\sigma'[AB])(b!)$. This relation has to be maintained in P by $\mathcal{I}(\sigma)$. That is, whichever is the probabilistic choice in $\mathcal{I}(\sigma)$ w.r.t. other atoms, the relation $\mathcal{I}(\sigma)(a!) = 2 \cdot \mathcal{I}(\sigma)(b!)$ has to be maintained.

This suggests that, in the general case, for two executions that cannot be distinguished by any of the two atoms A and B, the *relative probabilities* of choosing A over B (or B over A) should be the same. Or better stated: conditioned to the fact that the choice is between atoms A and B, the probability should be the same in two executions that cannot be distinguished by any of the two atoms.

Formally, given any two atoms A, B of a system P, for all σ, σ' s.t. $\sigma[A] = \sigma'[A]$ and $\sigma[B] = \sigma'[B]$, it must hold that

$$\frac{\mathcal{I}(\sigma)(A)}{\mathcal{I}(\sigma)(A) + \mathcal{I}(\sigma)(B)} = \frac{\mathcal{I}(\sigma')(A)}{\mathcal{I}(\sigma')(A) + \mathcal{I}(\sigma')(B)} \tag{3}$$

provided that $\mathcal{I}(\sigma)(A) + \mathcal{I}(\sigma)(B) \neq 0$ and $\mathcal{I}(\sigma')(A) + \mathcal{I}(\sigma')(B) \neq 0$.

Definition 3. A scheduler η is *strongly distributed* iff η is distributed and equation (3) holds on the interleaving scheduler $\mathcal{I}$ that defines η. The set of strongly distributed schedulers of P is denoted by $\mathsf{SDist}(P)$.

We emphasize that strongly distributed schedulers are useful depending on the particular model under consideration. In case we are analysing an agreement protocol and each atom models an independent node in a network, then the order in which nodes A and B execute cannot depend on information not available to none of them, and so strongly distributed schedulers give more realistic worst-case probabilities. However, in case the interleaving scheduler represents an entity that is able to look at the whole state of the atoms (for instance, if the atoms represent processes running on the same computer, and the interleaving scheduler plays the role of the kernel scheduler), then the restriction above may rule out valid behaviours, and so general distributed schedulers should be considered.

The following theorem is the generalization of the fact that, for every strongly distributed scheduler η on $P = \{T, Z, A, B\}$ as in Fig. 3 there is a distributed scheduler η' on $P' = \{T, Z, AB\}$ that defines the same probabilistic behaviour.

Theorem 2. *Let P be a IPIOA such that $A, B \in \text{Atoms}(P)$. Consider the system P' such that $\text{Atoms}(P') = (\text{Atoms}(P) \setminus \{A, B\}) \cup \{AB\}$, where AB is the usual parallel composition of A and B (as in, for instance, [8, p. 99]). Then, for every strongly distributed scheduler η for P, there exists a strongly distributed scheduler η' for P' yielding the same probability distribution on paths as η.*

One may wonder what happens if, instead of considering two atoms A and B in (3), two *disjoint sets* $\mathcal{A}, \mathcal{B}$ of atoms are considered. The (apparently more general) condition on sets holds whenever condition (3) on atom holds.

Theorem 3. *Let $\mathcal{A} = \{A_1, \cdots, A_n\}$, $\mathcal{B} = \{B_1, \cdots, B_m\}$ be disjoint sets of atoms. Then, if $\mathcal{I}$ is the interleaving scheduler that defines a strongly distributed scheduler η (i.e., it satisfies Eqn. 3), then*

$$\frac{\sum_i \mathcal{I}(\sigma)(A_i)}{\sum_i \mathcal{I}(\sigma)(A_i) + \sum_j \mathcal{I}(\sigma)(B_j)} = \frac{\sum_i \mathcal{I}(\sigma')(A_i)}{\sum_i \mathcal{I}(\sigma')(A_i) + \sum_j \mathcal{I}(\sigma')(B_j)}$$

holds whenever $\sigma[A] = \sigma'[A]$ for all $A \in \mathcal{A} \cup \mathcal{B}$, and the denominators are diferent from 0.

The proof of the theorem uses a standard argument for conditional probabilities, see [20].

4 Subclasses of distributed schedulers

Next, we discuss the expressive power of several subclasses of distributed schedulers. While the results in this section are not exactly about model checking, they are useful to prove undecidability and correctness of algorithms.

4.1 Power of deterministic schedulers

In the following, we investigate to which extent we can restrict to deterministic schedulers in order to get worst-case probabilities. Fortunately, for every system P, the class of deterministic distributed schedulers (denoted by $\mathsf{DetDist}(P)$) is equally expressive as the class of all distributed schedulers (denoted by $\mathsf{Dist}(P)$) if we aim to find the supremum (or infimum) probability of a given measurable set of infinite paths. This result holds for total information schedulers and PCTL* formulae [4], and so the result for partial information schedulers. However, it holds because of the way in which distributed schedulers restrict the use of the information. In fact, we will see that the restriction we impose on the interleaving scheduler for strongly distributed schedulers causes randomized schedulers to add extra power.

Theorem 4. *For any set S of infinite traces, S being measurable, we have that*

$$\inf_{\eta \in \mathsf{DetDist}(P)} \Pr^{\eta}(S) = \inf_{\eta \in \mathsf{Dist}(P)} \Pr^{\eta}(S)$$

and

$$\sup_{\eta \in \mathsf{DetDist}(P)} \Pr^{\eta}(S) = \sup_{\eta \in \mathsf{Dist}(P)} \Pr^{\eta}(S)$$

Proof. (Sketch. For details see [20].) The proof of this theorem proceeds by first proving that

$$\inf_{\eta \in \mathsf{DetDist}(P)} \Pr^{\eta}(S) = \inf_{\eta \in \mathsf{Dist}(P)} \Pr^{\eta}(S)$$

whenever S is a *finite-horizon* set of the form:

$$\biguplus_{i=1}^{n} [\sigma_i] \tag{4}$$

where $\{\sigma_i\}_{i=1}^{n}$ is a set of finite paths. The proof for these sets relies on the fact that the probability of a finite-horizon set depends on a finite number of choices (the choices for the paths whose length is greater than $\max_i \mathrm{len}(\sigma_i)$ are irrelevant). It is possible to show that each choice of $\Theta(\sigma_i)$ and $\mathcal{I}(\sigma)$ can be transformed into a deterministic choice without increasing the probabilities: an optimal scheduler for a finite-horizon set is then constructed by changing finitely many choices. Using these optimal schedulers, the proof then resorts to a limit argument: given any scheduler η, the probability of a measurable set S under η can be approximated by the probabilities of finite-horizon sets under η. Near-optimal schedulers for S are obtained as limits of optimal schedulers for finite-horizon sets (the existence of limits being ensured by Theorem 1). □

Unfortunately, if in the statement of Theorem 4 we consider *strongly distributed schedulers* the same claim is false. Consider the example in Fig. 4. Atoms A, B and C need to be "activated" by labels e_A, e_B and e_C, respectively. The atom E tosses a coin and activates A, B and C if the output of the coin is l, or B and C if the output of the coin is r. The atom R "remembers" the order in which the other atoms execute. The objective of the scheduler is to reach some state in R marked with a smile. It is clear that any deterministic scheduler yields a probability of 0, $1/2$ or 1. We verify that there is no deterministic strongly distributed scheduler η reaching a smile with probability 1. Towards a contradiction, suppose η reaches a smile both after l and r. In order to succeed in case the first output is l, η must choose the transitions whose outputs are e_A, e_B and e_C. Then, η should choose either a, b and c (in this order) or b, a and c. In order to succeed when r is chosen, η must choose the transitions whose outputs are e_B and e_C. Note that the projections of atoms A and B after r, e_B and e_C are the same as the projections after l, e_A, e_B and e_C. Since b must be chosen before c in case the first output is l, and η is strongly distributed, then η must choose b before c in case the first output is r. After B, R should output w, and E should output e_A. At this point, both A and C are active, and the projections of these atoms are the same as in case the first output is l. Since η is strongly distributed and a must be chosen before c in case the first output is l, a must be chosen before c also when the first output is r. However, choosing a before c does not lead to a state marked with a smile. Hence, there is no deterministic strongly distributed scheduler yielding probability 1, and so the supremum quantifying over deterministic strongly distributed schedulers is $1/2$. Nevertheless, consider the scheduler in which:

1. If there is a transition enabled in E, then the transition in E is chosen (i.e. the interleaving scheduler chooses E with probability 1)

2. If there is a transition enabled in R, then the transition in R is chosen (note that it cannot be the case that there are transitions enabled in both E and R)

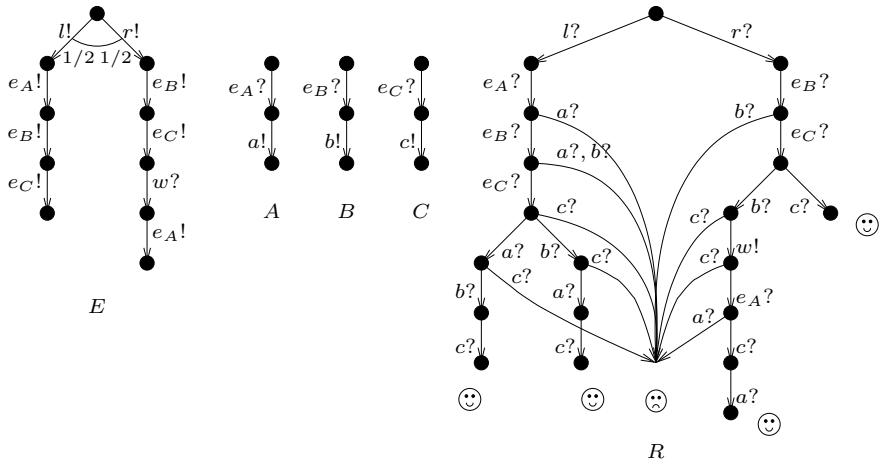


Figure 4: Example showing that randomization adds power to strongly distributed schedulers

3. If there are neither transitions enabled in E nor in R, then the scheduler chooses uniformly among the transitions a, b and c. That is, if a, b and c are enabled, choose each one with probability $1/3$, and, if b and c are enabled, choose each one with probability $1/2$.

This scheduler is strongly distributed, and yields the probability $13/24$, which is larger than the maximum probability over deterministic strongly distributed scheduler (which we have before found to be $1/2$). Therefore, this example shows that *randomized choices add power to strongly distributed schedulers*.

The same example can be used to show that there are systems for which deterministic strongly distributed schedulers cannot emulate *rate schedulers*. Such schedulers are introduced in [17], and resolve non-determinism by assigning rates to each local path. Given a rate scheduler, probabilities are then calculated as for continuous time Markov chains or the probabilistic automata of [29]. For this same example, rate schedulers yielding probabilities arbitrarily close to $13/24$ can be obtained by replacing arbitrarily high rates for the Dirac distributions (in which an atom is chosen deterministically) and equal rates for the uniform distributions.

4.2 On the (in)existence of a scheduler yielding the supremum probability

For traditional all-knowing schedulers, for every reachability property there exists a Markovian deterministic scheduler attaining the supremum probability. Consider the system comprising atoms T and G in Fig. 5. For this system, we show that there is no distributed scheduler maximizing the probability of reaching s_w. The behaviour of this system can be seen as a game: T tosses a coin without communicating the outcome to G, but communicating that the coin has been tossed (this is represented by $t!$). Atom T moves to state s_2 once the coin lands tails. Atom G can stop the game. The aim of G is to stop the game only if the coin has landed tails at least once. If G outputs n, then the coin is tossed again and the game continues. If G believes that the coin has landed tails sometime before, then it outputs g. If T is in state s_2 and G outputs g, then the objective state s_w is reached. Otherwise, if T receives g in state s_1, the undesirable state s_l is reached. In the following we calculate the supremum probability of reaching s_w. If G waits

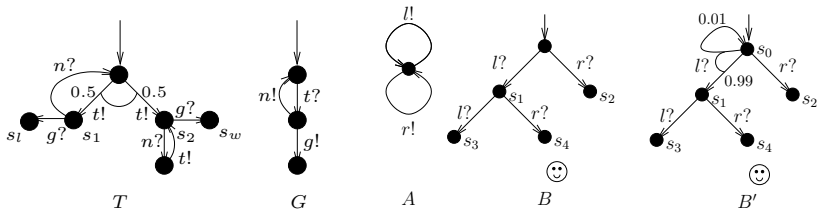


Figure 5: Atoms used in our examples

for one occurrence of t before communicating through g, then the probability of reaching s_w is $1/2$. However, G may decide to wait for two occurrences of t, thus having a probability of $3/4$. In general, waiting for k occurrences of t yields a probability of $1 - (1/2)^k$. In addition, it is easy to see that there is no randomized scheduler yielding probability 1. In conclusion, although the supremum is 1, there is no scheduler yielding such probability.

4.3 Finite-memory (and Markovian) schedulers

A scheduler is Markovian if it chooses the next transition according to the last state, regardless of the past history. In case traditional all-knowing schedulers are considered, *Markovian* schedulers attain the supremum probability for reachability properties [4].

In our setting, one may think of two types of Markovian distributed schedulers: a *globally Markovian* (distributed) scheduler should comply $\eta(\sigma)(g_i) = \eta(\sigma')(g_i)$ whenever $\text{last}(\sigma) = \text{last}(\sigma')$, while a *locally Markovian* (distributed) scheduler should choose the same local transitions whenever the local states coincide. In order to define locally Markovian schedulers, we say that an output scheduler is *Markovian* iff, for all g, it holds that $\Theta(\sigma)(g) = \Theta(\sigma')(g)$ whenever $\text{last}(\sigma) = \text{last}(\sigma')$. Similarly, an interleaving scheduler is *Markovian* iff $\mathcal{I}(\sigma)(A) = \mathcal{I}(\sigma')(A)$ whenever $\text{last}(\sigma) = \text{last}(\sigma')$. We say that a scheduler is locally Markovian if it can be obtained by composing Markovian schedulers. Markovian schedulers are a particular case of a more general class: the N-Markovian schedulers. A scheduler is globally N-Markovian if $\eta(\sigma\,\sigma') = \eta(\sigma')$ for all σ' of length N. Note that globally Markovian schedulers coincide with globally 1-Markovian schedulers. Similarly, locally N-Markovian schedulers can be defined. A simple example shows that locally Markovian schedulers do not attain supremum probabilities. Consider the system comprising atoms A and B in Fig. 5. First, we consider deterministic schedulers. A deterministic locally Markovian scheduler must output the same label in every path. So, if we quantify over *deterministic* locally Markovian schedulers, the supremum probability of reaching a smile is 0. The supremum quantifying over locally Markovian schedulers is 0.25, and is obtained by the scheduler that chooses $l!$ with probability 0.5 and $r!$ with probability 0.5 for all σ. This implies that *given a fixed amount of memory N, randomization adds power to N-Markovian schedulers*.

For the same example, note that *globally Markovian* schedulers obtain probability 1. However, in the following we use atoms A and B' in Fig. 5 to show an unnatural aspect of globally Markovian schedulers. Again, the aim of the scheduler is to reach a smile. Consider any deterministic globally Markovian scheduler η. In the initial state (s, s_0), atom A must output l. The label l must also be output in the path $(s, s_0).l!.(s, s_0)$, since the scheduler is globally Markovian.

Then, we have $\Theta_A((s.l!.s)) = l!$. This implies that l is also output in the path $(s, s_0).l!.(s, s_1)$. The same reasoning allows to conclude that $\Theta_A(\sigma) = l!$ for every A-path σ. So, the existence of the loop in s_0 implies that the choices of the scheduler should coincide for every path. In conclusion, although the system comprising atoms A and B is very *similar* to the system comprising atoms A, B', the power of globally Markovian schedulers is significantly *different*.

We say that a scheduler has local (global, resp.) finite memory if it is locally (globally, resp.) N-Markovian for some N. We denote the set of local (global, resp.) finite-memory distributed schedulers of a system P by $\mathsf{LFinMem}(P)$ ($\mathsf{GFinMem}(P)$, resp.) and the set of deterministic finite-memory schedulers by $\mathsf{DetLFinMem}(P)$ ($\mathsf{DetGFinMem}(P)$, resp.) It is easy to find examples showing the limitations of finite-memory schedulers: for instance, consider atom A in Fig. 5. Suppose that we are interested in the probability of the path having the sequence of labels $lrlrrlrrr\cdots$, that is, each l is followed by a sequence of r's, and the amount of r's is exactly the previous amount plus 1. There are no finite-memory schedulers yielding probabilities arbitrarily close to 1 for this path. Intuitively, an optimal scheduler should remember how much r's were in the previous sequence, and the amount of r's grows arbitrarily. (Note that, since we are considering a single atom, local finite-memory schedulers and global finite-memory schedulers coincide.)

We have seen that locally Markovian schedulers cannot attain worst-case probabilities even for simple reachability properties, and we have seen that finite-memory schedulers do not attain optimal probabilities for every property. However, if we consider only reachability properties, we obtain the following theorem.

Theorem 5. *For any set of goal states U,*

$$\sup_{\eta\in\mathsf{Dist}(P)} \Pr^{\eta}(\mathsf{F}\, U) = \sup_{\eta\in\mathsf{DetLFinMem}(P)} \Pr^{\eta}(\mathsf{F}\, U)\,.$$

Proof. Given, $\epsilon > 0$, let η^s be a scheduler such that $\sup_{\eta\in\mathsf{Dist}(P)} \Pr^{\eta}(\mathsf{F}\, U) - \Pr^{\eta^s}(\mathsf{F}\, U) < \epsilon/2$. We denote the set of paths reaching some element in U before the N-th step as $\mathsf{F}^N(U)$. Let N^* be such that $\Pr^{\eta^s}(\mathsf{F}\, U) - \Pr^{\eta^s}(\mathsf{F}^{N^*}\, U) < \epsilon$. The set F^{N^*} can be written as a disjoint union of set of extensions $[\sigma_k]$ where the length of the σ_k is at most N^*. Then, by Theorem 4, we know that there exists a deterministic scheduler η^d yielding the supremum probability for F^{N^*}. Let Θ_i^d and $\mathcal{I}^d$ be the schedulers that define η^d. Then, we can consider the (uniquely defined) N^*-Markovian schedulers Θ_i^m and $\mathcal{I}^m$ that coincide with the schedulers for η^d upto the N^*. The scheduler η^m obtained by composing Θ_i^m and $\mathcal{I}^m$ is N^*-Markovian, and it holds $\sup_{\eta\in\mathsf{Dist}(P)} \Pr^{\eta^m}(\mathsf{F}\, U) - \Pr^{\eta^s}(\mathsf{F}\, U) < \epsilon$. □

The statement of Theorem 5 is false in case strongly distributed schedulers are considered: the example in Fig. 4 is also a counterexample for such a statement. Theorem 5 can also be contrasted with the fact that, given a fixed amount of memory, nondeterministic schedulers are needed, as we have seen for atoms A and B in Fig. 5.

5 (Un)decidability and NP-hardness

In this section we consider the complexity of model checking under different subsets of distributed schedulers. Unfortunately, the results are negative in the sense that we find the problems

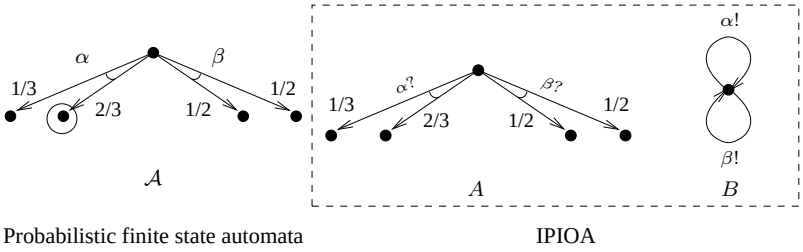


Figure 6: From PFA to IPIOA

undecidable or NP-hard.

5.1 General distributed schedulers

The probabilistic model checking problem is undecidable in case the schedulers are restricted to be distributed, in the sense that the supremum probability of a reachability probability cannot be approximated. This is stated in the following theorem.

Theorem 6. *Given* $0 < \epsilon < 1$*, there is no algorithm such that, for all IPIOA P, for all sets U, the algorithm computes r such that*

$$\Big| \sup_{\eta \in \mathsf{Dist}(P)} \Pr^{\eta}(\mathsf{F}\, U) - r \Big| \leq \epsilon \ .$$

This theorem was proven in [16]. The proof just points out that Probabilistic Finite Automata (PFA, for which the supremum probability is known to be undecidable [25]) are a special case of IPIOA under deterministic schedulers. Since Theorem 4 establishes that the expressive power of deterministic schedulers is the same as of distributed schedulers, undecidability for distributed schedulers follows. In a PFA $\mathcal{A}$, for each symbol α in the alphabet Σ, for each state s, there is exactly one transition labelled with α in s. PFA semantics define the probability of accepting a word $\alpha_1 \cdots \alpha_n$ as the probability of reaching an accepting state by successively taking the transitions labelled with $\alpha_1 \cdots \alpha_n$ at each state.

Figure 6 shows a simple PFA and its corresponding IPIOA atoms. The IPIOA we construct has two atoms A and B. The set of labels of both atoms is Σ. The sets of states of atom A is the set of states of $\mathcal{A}$. Moreover, A encodes the transition function of $\mathcal{A}$ using reactive transitions. Atom B is the one that outputs labels and introduces the nondeterminism. Notice that A is deterministic in the sense that, at every state, each label uniquely determines the transition to execute. Hence, a word w over Σ is equivalent to the deterministic scheduler for B that chooses the symbols in w. Let $U = \{(s,t) \mid s$ is accepting in $\mathcal{A}$ and t is the only state of $B\}$. The probability of reaching a state in U under a deterministic scheduler is the probability of reaching an accepting state in $\mathcal{A}$.

Theorem 6 concerns the *quantitative* problem of approximating the supremum. The *qualitative* problems of computing whether the supremum is 1, and whether there exists a scheduler yielding probability 1 are also undecidable (these problems are not necessarily equivalent, as we have seen in Subsection 4.2).

Theorem 7. *There is no algorithm that, for all IPIOA P, for all sets U, the algorithm decides whether or not*

$$\sup_{\eta \in \mathsf{Dist}(P)} \Pr^{\eta}(\mathsf{F}\, U) = 1 \ .$$

There is no algorithm that, for the same input as above, decides whether or not there exists $\eta \in \mathsf{Dist}(P)$ such that $\Pr^{\eta}(\mathsf{F}\, U) = 1$.

This theorem, and versions of Theorems 6 and 7 for strongly distributed schedulers are proven in [18].

5.2 Finite memory distributed schedulers

Theorem 5 implies that the model checking problem is undecidable even if we restrict to finite-memory schedulers. Moreover, if we want to restrict to deterministic schedulers having at most N memory, the amount of memory N needed in order to get a bounded approximation of the probability cannot be calculated. Formally, let $\mathsf{DetLFinMem}_N(P)$ be the set of deterministic locally N-Markovian schedulers for P. Then:

Theorem 8. *Given $\epsilon > 0$, there is no algorithm computing N such that*

$$\sup_{\eta \in \mathsf{Dist}(P)} \Pr^{\eta}(\mathsf{F}\, U) - \sup_{\eta \in \mathsf{DetLFinMem}_N(P)} \Pr^{\eta}(\mathsf{F}\, U) < \epsilon \, .$$

Proof. Suppose, towards a contradiction, that the problem is decidable. Since $\mathsf{DetLFinMem}_N(P)$ is finite, then there exists an algorithm to find a value r such that $\sup_{\eta \in \mathsf{Dist}(P)} \Pr^{\eta}(\mathsf{F}\, U) - r < \epsilon$. Such algorithm simply computes N and then performs an exhaustive search on $\mathsf{DetLFinMem}_N(P)$. However, the existence of such algorithm contradicts Theorem 6. □

Since Theorem 6 holds also if we restrict to systems in which only one atom has generative transitions, we cannot compute N even under such restriction. Hence, the result holds also for strongly distribute schedulers.

Even if a reasonable bound for the memory of the schedulers can be calculated somehow, then the problem is still complex, as stated in the following theorem.

Theorem 9. *For all*

$$\mathcal{S} \in \{\mathsf{LFinMem}_1(P), \mathsf{DetLFinMem}_1(P), \mathsf{GFinMem}_1(P), \mathsf{DetGFinMem}_1(P)\} \ ,$$

the problem of computing $\sup_{\eta \in \mathcal{S}} \Pr^{\eta}(\mathsf{F}\, U)$ is NP-hard.

In [20] we prove this theorem by presenting a reduction of the 3SAT problem to the supremum reachability problem. For every 3SAT instance, the system we construct has no cycles, and so the reduction is valid regardless of the memory of the schedulers considered.

6 Algorithms

The algorithms we introduce in this section are based on classic algorithms for MDPs. We present a simple MDP setting and show how it can be linked to the IPIOA we were considering so far.

Given a system P comprised by atoms $A_1, \dots A_N$, the induced MDP of P is $P_{\mathsf{ind}} = (\mathsf{St}, \mathrm{T}, \mathsf{en})$, where St is the set of global states and T is a set of probabilistic transitions defined as follows: for each $s_i \in \mathsf{St}_i$ and $g_i \in G_i(s_i)$ there is a transition $\alpha_{s_i,g_i} \in \mathrm{T}$ such that $\alpha_{s_i,g_i}(s, s') = \sum_{a \in \mathsf{ActLab}_i} g(s'[i], a) \prod_{j=0 j\neq i}^{N} r_{s[j],a}(s'[j])$. Where $r_{s[j],a} = R(s[j], a)$ if $a \in \mathsf{ActLab}_j$ and $1_{s[j]}$ otherwise. $\mathsf{en} : \mathsf{St} \mapsto \mathcal{P}(\mathrm{T})$ is the enabling function that satisfies $\alpha_{s_i,g_i} \in \mathsf{en}(s)$ iff $s[i] = s_i$. A sequence $\sigma = s_0, \alpha_0, s_1 \dots s_n$ is a path in P_{ind} if for all i, $s_i \in \mathsf{St}$, $\alpha_i \in \mathsf{en}(s_i)$ and $\alpha(s, s') > 0$. We say that a transition α is probabilistic if there exist s, s' such that $0 < \alpha(s, s') < 1$. A scheduler in P_{ind} is a function η from finite paths to transitions such that $\eta(\sigma) \in \mathsf{en}(last(\sigma))$. The set of all schedulers in an MDP P is written as $\mathsf{Sched}(P)$.

For simplicity, we assume that in a generative transition the state reached determines the label uniquely (any atom in which this does not hold can be transformed by adding the last label as part of the local state). Formally,

$$g(s, a) > 0 \wedge g(s, b) > 0 \implies a = b \,.$$

Given a scheduler η for P, it is easy to define a scheduler η_{ind} for P_{ind}:

$$\eta_{\mathsf{ind}}(s^1.\alpha^1_{g_1} \cdots \alpha^{n-1}_{g_{n-1}}.s^n)(\alpha_{g_i}) \\ \mathcal{I}(s^1.g_1 \cdots \alpha^{n-1}_{g_{n-1}}.s^n)(A_i) \cdot \Theta_i(s^1[i].a_1 \cdots a_{n-1}.s^n[i])(g_i) \,. \quad (5)$$

The particular labels $a_1, \cdots, a_{n-1}$ are irrelevant because of the previous assumption.

There is a natural correspondence between P and P_{ind}:

$$\Pr^{\eta_{\mathsf{ind}}}([s^1.\alpha^1_{g_1} \cdots \alpha^{n-1}_{g_{n-1}}.s^n]) \\ = \Pr^{\eta}(\biguplus_{a_1,\cdots a_{n-1}} [s^1.(g_1, a_1, \dots) \cdots (g_{n-1}, a_{n-1}, \dots).s^n]) \,. \quad (6)$$

In other words, in the MDP semantics all the paths that differ only with respect to labels are grouped together in a single path. Therefore, the analysis of properties that only concern states can be carried out interchangeably on the IPIOA or on its induced MDP.

Given a transition $\alpha_{g_i} \in \mathrm{T}$ the set of atoms that may be affected by the transition α_{g_i} is $\mathsf{Inv}(\alpha_{g_i}) = \{A_j \,|\, a \in \mathsf{ActLab}_j\}$. Notice that if the corresponding transition of α_{g_i} is executed in P the only local histories that may change are the ones in $\mathsf{Inv}(\alpha_{g_i})$. In addition, we denote with $\mathsf{GenAtom}(\alpha_{g_i})$ the only atom that produces the generative transition g_i in α_{g_i}.

In the rest of this section we will use interchangeably α to denote a transition in the induced MDP P_{ind} or its corresponding generative transition. In addition, we assume that properties depend only on states, that is, we are interested on the probabilities of sets $\mathcal{S}$ such that

$$s^1.(g_1, a_1, \dots) \cdots (g_{n-1}, a_{n-1}, \dots).s^n.\cdots \in \mathcal{S} \\ \iff \forall g'_1, \cdots, a'_1, \cdots : s^1.(g'_1, a'_1, \dots) \cdots (g'_{n-1}, a'_{n-1}, \dots).s^n.\cdots \in \mathcal{S} \,. \quad (7)$$

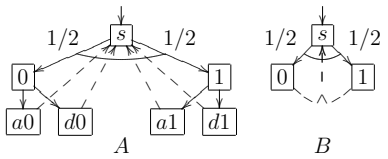


Figure 7: Player A tries to guess if the outcomes agree

6.1 Counterexample-guided refinements

Given an IPIOA P and a property, one can obtain the probability under total information schedulers by constructing P_{ind} and using standard algorithms for MDPs such as [4].

The probabilities under total information can be seen as a safe (although overly pessimistic) bound on the minimum/maximum probability. In this section, 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. 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.

This technique has been introduced in [22]. Here, we adapt the notation to the one in this paper and omit proofs. The reader is referred to [22] for details.

We illustrate the technique using the players A and B in Fig. 7. To simplify, we assume that they play a turn-based game. 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 the outcome of A 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. Note that, since we are assuming that the game is turn based, except for the synchronized transition, there is no interleaving nondeterminism and the only nondeterminism concerns the behaviour of A. Intuitively we can think that player A wants to prevent the system from reaching one of the states in which B wins. It is easy to see that, for every scheduler, B wins in the first round with probability $1/2$, and the probability that B has won after round N is $1 - (1/2)^N$. Hence, the

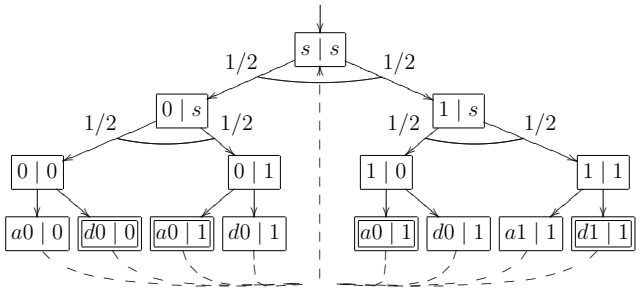


Figure 8: MDP obtained by composing A and B

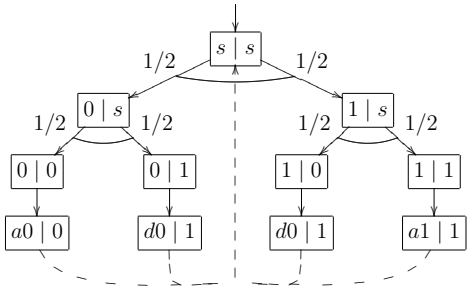


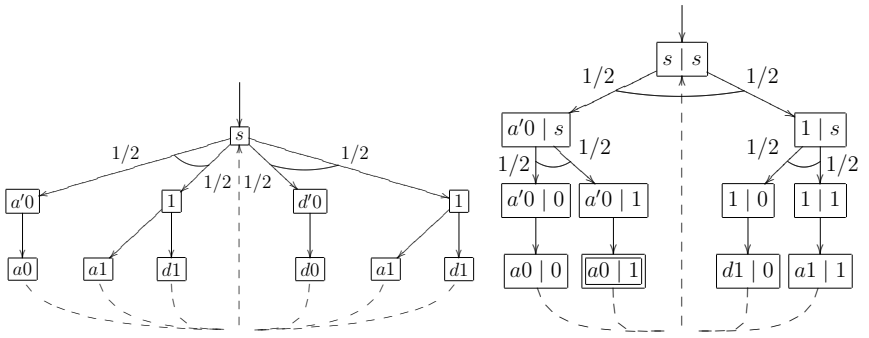
Figure 9: Unrealistic counterexample in the parallel composition

probability that B wins the game in some round is 1. The minimum probability that B wins, quantifying over all distributed schedulers, is then 1.

If we construct P_{ind} and analyse it under total information, a scheduler can simply guess an agreement in case an agreement happened, and a disagreement otherwise. (Which is unrealistic, since we assume that A is unable to see the outcome of B.) P_{ind} and the unrealistic scheduler are depicted in Figures 8 and 9, respectively. The probability under such scheduler is 0.

We can explain our counterexample-based transformation of P by following our previous example: we first detect that, in the counterexample in Fig. 9, 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 (called A') is shown in Fig. 10(a). Roughly speaking, the non-determinism at state 0 has been "pulled backwards". If we now consider P'_{ind} (the MDP induced by $\{A', B\}$), we still have some unrealistic counterexamples, as A can still cheat in state 1. One of such unrealistic counterexamples is shown in Fig. 10(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



(a) Refined model of A

(b) A counterexample after the first refinement

Figure 10: Refined system and counterexample

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.

6.1.1 Detection of partial-information counterexamples

Under total information, for minimum/maximum reachability it suffices to consider only globally Markovian schedulers. Moreover, the verification of general LTL and PCTL* formulae is carried out by reducing the original problem to problems for which globally Markovian schedulers are sufficient [4].

We address the problem of checking whether a globally Markovian scheduler η for P_{ind} corresponds to a scheduler η' in P, that is, whether $\eta'_{\mathsf{ind}} = \eta$. If such an η' exists, then η corresponds to a valid partial information scheduler P. Otherwise, the scheduler under total information cannot be produced under partial information, and we apply the refinement described in Subsection 6.1.2.

The method for checking whether a scheduler complies with the partial information restrictions resembles the well-known technique of *self-composition* [3]. We denote by $\eta^{>0}(s)$ the value of $\eta(\sigma)$ for all σ such that $\mathrm{last}(\sigma) = s$. Note that a scheduler $\eta \in \mathsf{GMarkovian}$ 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 $\eta^{>0}(s) = \zeta^{>0}(s)$, then they yield the same probabilities for all paths.

We reduce the problem to that of checking whether η has *conflicting paths*. Given a $\eta \in \mathsf{GMarkovian}$, we say that two states s, t with $s[i] = t[i]$ are η-*conflicting* for the atom A_i iff $\eta^{>0}(s) = \alpha_{g_i}$ and $\eta^{>0}(t) = \alpha_{g'_i}$ for some $g_i \neq g'_i$.

We say that two paths are η-*conflicting* for A_i if $\sigma[i] = \sigma'[i]$, $\mathrm{last}(\sigma) = s$, $\mathrm{last}(\sigma') = t$ and the states s, t are η-conflicting. From the definition of η-conflicting we can prove:

$$\exists \eta' : \eta = \eta'_{\mathsf{ind}} \iff \eta \text{ has no conflicting paths} . \tag{8}$$

Now we show how to check the (in)existence of conflicting paths for A_i. For all $s, t \in \mathsf{St}$, $r_i \in \mathsf{St}_i$, we define the relation $s \overset{r_i}{\rightsquigarrow} t$, that holds iff there exist paths σ, σ' such that $\mathrm{first}(\sigma) = \mathsf{init}$,

$\text{last}(\sigma) = s = \text{first}(\sigma')$, $\text{last}(\sigma') = t$ and $\sigma'[i] = r_i$. This relation can be extended naturally to the local paths $\sigma_i = r_i^1 \cdots r_i^k$, so we can write $s \overset{\sigma_i}{\rightsquigarrow} t$. By the definition of $\rightsquigarrow$, if $s \overset{\sigma_i}{\rightsquigarrow} t$ then there exists a path σ from s to t in η such that $\sigma[i] = \sigma_i$. Consider the non-deterministic finite automaton $\text{Nfa}_i(\eta)$ that represents the relation $\overset{r_i}{\rightsquigarrow}$. In this automaton, each word starting in s and ending in t corresponds to a σ_i such that $s \overset{\sigma_i}{\rightsquigarrow} t$.

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

$$\text{Nfa}_i^2(\eta) = \text{Nfa}_i(\eta) \times \text{Nfa}_i(\eta) \tag{9}$$

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

6.1.2 Refining a system for a conflict

According to Eq. (8), if a counterexample η does not comply with the partial-information constraints, there exist two η-conflicting states s and t for a module A_i. Since these states are conflicting, we have that $s[i] = t[i]$ and $\eta^{>0}(s) = \alpha_{g_i}$, $\eta^{>0}(t) = \alpha_{g_i'}$ for some $g_i \neq g_i'$. In words, two different transitions g_i, g_i' are chosen in A_i while, because of the partial-information constraints the choices must coincide. Next, we show how to refine the atom A_i. The refinement is modular, in the sense that only A_i is affected.

As illustrated in the example in the introduction, the idea is to split $s[i]$ in such a way that g_i and g_i' are not enabled in the same state. We assume that no input transition reaches $s[i]$ (we can always insert an intermediate state after an input transition and a generative transition after such state). This assumption is not needed in the original presentation of this technique: in the setting of [22], atoms (or modules) are allowed to have input non-determinism; for simplicity, we avoided such non-determinism in this paper, at the expense of introducing additional states.

Let $s_i = s[i]$. We call the new states s_i^1 and s_i^2. Given a generative transition h_i, we define two new transitions h_i^1, h_2^2, in such a way that, if $h_i(s_i, a) > 0$, then h_i^1 reaches s_i^1 and h_i^2 reaches s_i^2. We define $h_i^1(t_i, a) = h_i(t_i, a)$ if $t_i \neq s_i^1$ and $t_i \neq s_i^2$, $h_i^1(s_i^1, a) = h_i(s_i, a)$ and $h_i^1(s_i^2, a) = 0$. The transition h_i^2 is defined similarly, but we have $h_i^2(s_i^2, a) = h_i(s_i, a)$ and $h_i^2(s_i^1, a) = 0$. Note that if h_i does not reach s_i, then h_i^1 and h_i^2 coincide.

Definition 4. Given an atom A_i, a local state s_i, and two generative transitions g_i, g_i' the refined atom A_i' is defined as:

- $\mathsf{St}_i' = \mathsf{St}_i \setminus \{s_i\} \cup \{s_i^1, s_i^2\}$
- $G_i'(t_i) = \{h_i^1, h_i^2 \mid h_i \in G_i(t_i)\}$ if $t_i \neq s_i^1$ and $t_i \neq s_i^2$
- $G_i'(s_i') = \{h_i^1, h_i^2 \mid h_i \in G_i(s_i)\}$ if $s_i' = s_i^1$ or $s_i' = s_i^2$
- $R_i'(t_i, a) = R_i(t_i, a)$

Let P be the IPIOA comprising atoms $\text{Atoms}(P) \setminus \{A\} \cup \{A'\}$. Given a set of paths $\mathcal{S}$ in P, we can construct the corresponding set of paths $\mathcal{S}'$ in P' ($\mathcal{S}'$ has the same paths but with s_i^1 or s_i^2 instead of s_i).

We have then the following theorem:

Theorem 10. *For all IPIOA P, and set of path $\mathcal{S}$:*

$$\sup_{\eta \in \mathsf{Dist}(P)} \Pr^{\eta}(\mathcal{S}) = \sup_{\eta \in \mathsf{Dist}(P')} \Pr^{\eta}(\mathcal{S}')$$

The theorem ensures that the probabilities under distributed schedulers do not change. However, as shown in the introduction, we might have that the total information probabilities in P'_{ind} are more realistic than in P_{ind}. If in P' we find an optimal scheduler that complies with the partial information restrictions (as checked by the method in the previous sub-subsection), then the verification is finished. If the scheduler found in P' is still unrealistic, the refinement can be carried out on P' and so on.

6.1.3 Convergence

A scheduler might have several conflicts, and the choice of the conflicts can affect the probabilities in P'_{ind}.

In [22] we show that the conflicts can be chosen in such a way that, if the refined systems are $P^1, P^2, \cdots$ we have:

$$\lim_{n \to \infty} \inf_{\eta \in \mathsf{Sched}(P^n)} \Pr^{\eta}(\mathsf{F}\, U) = \inf_{\eta \in \mathsf{Dist}(P)} \Pr^{\eta}(\mathsf{F}\, U)\ .$$

That is, the infimum probability values in the refined systems under total information converge to the probability values under partial information in the original system.

The criterion for conflicts that ensures convergence does not perform well in the systems we analyzed. Another criterion (namely, to choose any conflict in one of the shortest local paths having a conflict) offers better performance in the three systems we analyzed, but examples can be constructed in which the probabilities do not converge to the infimum.

There is no similar convergence for upper bounds of supremum probabilities: together with the computable lower bounds $\lim_{N\to\infty} \sup_{\eta \in \mathsf{Dist}(P)} \Pr^{\eta}(\mathsf{F}^N)$ (which converge to $\sup_{\eta \in \mathsf{Dist}(P)} \Pr^{\eta}(\mathsf{F}\, U)$), such upper bounds would turn the approximation problem for the supremum decidable, thus contradicting Theorem 6.

As explained in [22], the convergence of our technique does not imply decidability for the case of the minimum, as in each step there is no way to know how close is the approximation to the actual value. The decidability of the approximation problem for the minimum is thus open. In the particular case of Probabilistic Finite Automata the problem is decidable [19], but we could not find an extension of the proof for IPIOA, where information is hidden more selectively.

6.2 Partial order reduction for $\mathsf{LTL}_{\setminus\{\text{next}\}}$

Given a system and a property, the technique of partial order reduction yields another system with less transitions. The reduced system is constructed by traversing the state space. When

expanding a given state, not all the transitions enabled are considered. An *ample set* $\mathsf{ample}(s)$ must be calculated for each state s, and only transitions in the ample set are considered during the search. POR techniques impose restrictions on the ample sets to ensure that, for each property, the reduced system complies with the property iff the original system does.

In this section, we show how the POR technique for probabilistic systems [11, 2] can be improved under the assumption that the schedulers are (strongly) distributed.

The reduction technique we discuss in the following guarantees the preservation of probability bounds for LTL properties not containing the next operator. Given a set AP of atomic propositions and a labeling function $\mathsf{L} : \mathsf{St} \to \mathcal{P}(\mathsf{AP})$, the set of $\mathsf{LTL}_{\setminus\{\text{next}\}}$ formulae are generated by the following grammar.

$$\phi ::= \text{True} \mid l \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \mathsf{U} \phi_2 \ ,$$

where True is a constant and $l \in \mathsf{AP}$. Intuitively, an infinite path ρ satisfies $\phi_1 \mathsf{U} \phi_2$ (denoted by $\rho \models \phi_1 \mathsf{U} \phi_2$) iff there is position in ρ in which ϕ_2 holds, and ϕ_1 holds in all intermediate positions of ρ from the beginning until the position in which ϕ_2 holds. As usual, we write $\mathsf{F}\phi$ for $\text{True}\,\mathsf{U}\,\phi$, and $\Pr^{\eta}(\phi)$ for $\Pr^{\eta}(\{\rho \mid \rho \models \phi\})$.

Restrictions to the ample sets are based on the notion of *independence*. We say that two transitions α, β are *independent* iff their generative transitions affect different atoms. Formally, $\mathsf{Inv}(\alpha) \cap \mathsf{Inv}(\beta) = \emptyset$.

Note that the order of execution is irrelevant and that neither of them can disable or enable the other. Notice also that this definition is of a more structural nature than those in [2, 11]. This is no surprise, since our improvements profit from the structure of the model.

We need some additional definitions before presenting the restrictions for POR. A transition α is *stutter* iff for all s such that $\alpha \in \mathsf{en}(s)$ and s' such that $\alpha(s, s') > 0$ implies that $\mathsf{L}(s) = \mathsf{L}(s')$.

An *end component* (EC) is a pair $(\mathring{S}, \mathring{T})$ where $\mathring{T} : \mathring{S} \to \mathcal{P}(\mathrm{T})$ and $\mathring{S}$ is a set of states such that: **(1)** $\emptyset \neq \mathring{T}(s) \subseteq \mathsf{en}(s)$ for all $s \in \mathring{S}$, **(2)** $\alpha(s, s') > 0$ implies $s' \in \mathring{S}$ for all $s \in \mathring{S}$, $\alpha \in \mathring{T}(s)$ **(3)** for every $s, s' \in \mathring{S}$ there exists a path from s to s' using only actions in $\mathring{T}$. Intuitively, an end component induces an scheduler that stays in $\mathring{S}$ with probability 1 and moreover visits all its states infinitely often.

The restrictions for the ample sets of [2] to preserve $\mathsf{LTL}_{\setminus\{\text{next}\}}$ properties under unrestricted full-history dependent schedulers are listed below. $\hat{\mathsf{St}}$ denotes the set of reachable states in the reduced system $\hat{P}$, which is constructed by taking $\mathsf{ample}(s)$ to be the set of enabled transitions in $s \in \hat{\mathsf{St}}$.

(**A1**) *For all states* $s \in \mathsf{St}$, $\emptyset \neq \mathsf{ample}(s) \subseteq \mathsf{en}(s)$,
(**A2**) *If* $s \in \hat{\mathsf{St}}$ *and* $\mathsf{ample}(s) \neq \mathsf{en}(s)$, *then each transition* $\alpha \in \mathsf{ample}(s)$ *is stutter,*
(**A3**) *For each path* $\sigma = s.\alpha_1.s_1.\alpha_2.\cdots.\alpha_n.s_n.\gamma$ *in* P *where* $s \in \hat{\mathsf{St}}$ *and* γ *is dependent on* $\mathsf{ample}(s)$ *there exists an index* $1 \leq i \leq n$ *such that* $\alpha_i \in \mathsf{ample}(s)$,
(**A4**) *If* $(\mathring{S}, \mathring{T})$ *is an EC in* $\hat{P}$ *and* $\alpha \in \bigcap_{s \in \mathring{S}} \mathsf{en}(s)$, *then* $\alpha \in \bigcup_{s \in \mathring{S}} \mathsf{ample}(s)$
(**A5**) *If* $s.\alpha_1.s_1.\alpha_2.s_2.\cdots.\alpha_n.s_n.\gamma.s_{n+1}$ *is a path in* P *where* $s \in \hat{\mathsf{St}}$, $\alpha_1, \cdots, \alpha_n$, $\gamma \notin \mathsf{ample}(s)$ *and* γ *is probabilistic then* $|\mathsf{ample}(s)| = 1$.

Conditions **A1**–**A3** are originaly from POR on non-probabilistic systems [26]. **A1** ensures that the reduced model is a submodel of the original one, and that it does not introduce terminal

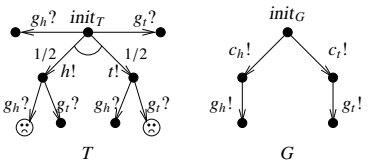


Figure 11: T tosses a coin, G guesses heads or tails

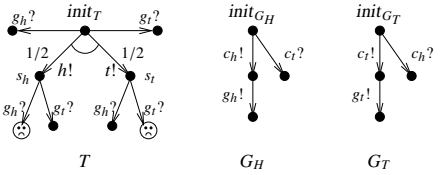


Figure 12: Two different guessers

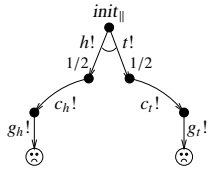


Figure 13: A dubious scheduling

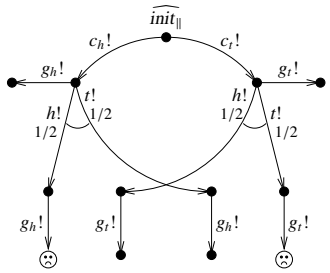


Figure 14: A POR based reduction

states (since the original model does not have either). **A3** enforces that any finite sequence of transitions leaving a state s that does not contain a transition in $\mathsf{ample}(s)$ can be extended with a transition from $\mathsf{ample}(s)$. Together with **A2**, they ensure that any execution in the original system can be mimicked by an observational equivalent trace in the reduced system. Besides, notice that **A3** is the only condition that is concretely related to the notion of (in)dependence. Condition **A4** is a probabilistic variant of Peled's cycle condition. Peled's condition ensure that if an action is enabled continuously along a path in the original system, then then that action is eventually enabled in the reduced system. Condition **A4** is weaker that Peled's condition and guarantees that the set of paths that disable actions forever has measure zero. Therefore, condition **A4** ensures that all fair paths are also represented in the reduced system. Condition **A5** is particular for probabilistic models. Contrarily to the other conditions, **A5** is technical and non-intuitive and has been introduced precisely to *not* eliminate the behaviour introduced by (non-distributed!) schedulers like the one of the example in Fig. 13. A probabilistic action can only be delayed only if the ample action in all the branches is the same. In case of the total information schedulers, it is only possible if the ample is a singleton. We remark that the fully expanded ample set $\mathsf{ample}(s) = \mathsf{en}(s)$ trivially satisfies conditions **A1**–**A5**. We also remark that if the model P is non-probabilistic, condition **A5** has no effect and condition **A4** reduces to Peled's original cycle condition. As a consequence, conditions **A1**–**A5** behave exactly in the same way as Peled's original conditions for POR on non-probabilistic models.

In case we assume that the schedulers are distributed, we can replace **A5** by

(**A5′**) *If* $s.\alpha_1.s_1.\alpha_2.s_2 \cdots \alpha_n.s_n.\gamma.s_{n+1}$ *is a path in* P *where* $s \in \hat{\mathsf{St}}$, $\alpha_1,\cdots,\alpha_n,\gamma \notin \mathsf{ample}(s)$ *and* γ *is a probabilistic transition, then* $\mathsf{GenAtom}(\beta) = \mathsf{GenAtom}(\beta')$, *for all* $\beta,\beta' \in \mathsf{ample}(s)$.

Condition **A5′** relaxes condition **A5**. Contrarily to **A5**, **A5′** does not requires $\mathsf{ample}(s)$ to be a singleton; instead, $\mathsf{ample}(s)$ may contain several transitions as long as they are generated by the same atom. The execution of probabilistic but independent transitions does not modify the local history of the atom in the ample, thus the output scheduler always chose the same transition regardless of the probabilistic behaviour.

The result is formalized in the following theorem.

Theorem 11. *Let ϕ be an $\mathsf{LTL}_{\setminus\{\text{next}\}}$ formula and P be an interleaved probabilistic I/O automata. Let $\hat{P}$ be a reduction of P complying with conditions* **A1**–**A4**, **A5**′. *Then,* $\sup_{\eta\in\mathsf{Dist}(P)} \Pr^{\eta}(\phi) \leq \sup_{\eta\in\mathsf{Sched}(\hat{P})} \Pr^{\eta}(\phi)$.

In case we assume *strongly distributed schedulers*, **A5** can be disregarded. The execution of independent transitions does not modify the history of any generative and/or reactive atom in the ample. Thus, the local histories coincide and it forces the interleaving scheduler to maintain the same relative distributions for the atoms in the ample.

Theorem 12. *Let ϕ, P be as in Theorem 11. Let $\hat{P}$ be a reduction of P complying with conditions* **A1**–**A4**. *Then,* $\sup_{\eta\in\mathsf{SDist}(P)} \Pr^{\eta}(\phi) \leq \sup_{\eta\in\mathsf{Sched}(\hat{P})} \Pr^{\eta}(\phi)$.

As an example, recall atoms T and G in Fig. 11 and the non-distributed scheduler η^w in Fig. 13. According to Theorem 11 the reduction in Fig. 14 is correct in case distributed schedulers are assumed. However, in the original system P we have $\Pr^{\eta^w}(\mathsf{F}\,☹) = 1$, while in $\hat{P}$ we have $\Pr^{\eta}(\mathsf{F}\,☹) \leq \frac{1}{2}$ for all η. This is due to the fact that η^w is not distributed. In fact, the supremum over all distributed schedulers in P is $\frac{1}{2}$, which coincides with $\sup_{\eta\in\mathsf{Sched}(\hat{P})} \Pr^{\eta}(\mathsf{F}\,☹)$. Recall now the example in Fig. 12 with atoms T, G_H and G_T. Notice that the scheduler of Fig. 13 *is* distributed in this setting. Call this scheduler η^d. If we assume strongly distributed schedulers, the reduction in Fig. 14 is allowed, and there is no scheduler yielding probability 1 in the reduced system. This is correct, since the scheduler η^d is *not* strongly distributed. However, if we want to preserve all distributed schedulers (even those that are *not* strongly distributed) then condition **A5**′ prevents the reduction in Fig. 14, since $c_h!$ and $c_t!$ are generated by atoms G_H and G_T, resp. This is exactly what we want, since the scheduler η^d is a valid distributed scheduler for T, G_H and G_T, and so a corresponding scheduler yielding probability 1 must exist in the reduced system.

Experimental results [21] show that the reduction profits from the fact that the conditions are weakened, saving up to 40% of verification time in a system with 4×10^9 states. A summary of the proof of Theorems 11 and 12 can be found in [21]. A full proof can be found in [15].

7 Related work

Our definition of strongly distributed schedulers is an important contribution, since it exactly captures the restrictions that the lack of information imposes to schedulers in asynchronous settings. In previous frameworks, there are no nondeterministic choices concerning the interleaving. In [12], the components are not specified explicitly (then, there are no interleaving issues) and the schedulers are restricted by imposing the condition that they must observe only a portion of every state in the history. In [13] a step of the whole system is obtained by taking a step in every component (thus, no interleaving is needed). The main difference between our framework and the PIOA framework in [9] is the concept of interleaving scheduler. In contrast, in the framework presented in [9] the different components have only input and output local schedulers, and a token is used in order to decide the next component to perform an output. The interleaving among different components is not resolved by the schedulers, since the way in which the token

is passed is specified by the components. Note that, because of the internal nondeterminism, the choice of the next component to execute is still nondeterministic, since there may be different transitions passing the token to different components. However, since internal nondeterminism is resolved according to the local history, the choice of the next component to execute is based on the history of the component that passes the token. In [8] it is suggested that a fictitious arbiter component can be added in order to specify interleaving policies. The components pass the token to the arbiter and the arbiter selects one of the components to which the token is passed. Using this schema, the information used to choose the next component can be restricted simply by restricting the information available to the arbiter. Although this approach is useful in order to keep some information hidden, such approach cannot be used to represent the restriction we impose to strongly distributed schedulers since, in our restriction, the lack of information depends on each pair of components and there is no information completely hidden. In [17], a mechanism is devised in such a way that the interleaving is determined using rates for each component, and these rates depend solely on the information available to the component.

In [1] a deterministic variant of our distributed scheduler is used, taking also into account that secret information needs to be hidden. The framework they present is simpler since components are required to be deterministic, and therefore there is no need for something like our output schedulers. Hence, the global scheduler can only choose components which are properly tagged in the transition. Thus, their global scheduler is quite similar to our interleaving scheduler.

Though similar in nature, a somewhat different approach to distributed scheduler is developed in [14]. The authors introduce a testing theory *a la* De Nicola-Hennessy that considers the distributed nature of the components. Their technique is based on labelling each nondeterministic transition considering only the local information of the component that produces it. This labelling is what lately drives the choices of the scheduler.

The example used to show that Markovian schedulers cannot attain worst-case probabilities resembles the well-known *partially observable* Markov decision processes (POMDPs). POMDPs are MDPs in which the scheduler cannot distinguish the states: for each state, a distribution on the possible *observations* is defined, and the scheduler chooses according to these observations. The way in which the information is hidden is a crucial difference with respect to PIOA, since the lack of information in PIOA is not "state based" but "transition based": in the PIOA framework, an atom is not aware of a state change unless the atom has synchronized in the transition leading to this state change. This difference suggests that care must be taken to translate results from the POMDP setting to the PIOA setting. Similarly, the hardness result in [12] is proved in a setting in which the lack of information is not necessarily a consequence of the existence of several components.

Finally, we notice that [5] shows that the verification of bounded time reachability properties under distributed schedulers and strongly distributed schedulers are actually decidable. The paper provides two algorithms that reduce the bounded reachability problem to a polynomial optimization problem. This was expected for distributed schedulers since there are finitely many deterministic distributed schedulers on finite systems not containing loops (which are sufficient by Theorem 4). However, this was not obvious for strongly distributed schedulers since they are strictly more expressive than its corresponding deterministic variant. (See Figure 4 and its explanatory text. Notice that the reachability property in this example is actually bounded by the

number of transitions in the longest path of component R.) This work was later extended to deal with information hiding in [27] aiming to analyse security protocols. This paper also provides optimizations to the original algorithms, making the derived polynomial optimization problem significantly more compact.

Acknowledgements We kindly thanks the recommendations and suggestions of the anonymous reviewers.

Bibliography

[1] Miguel E. Andrés, Catuscia Palamidessi, Peter van Rossum, and Ana Sokolova. Information hiding in probabilistic concurrent systems. *Theor. Comput. Sci.*, 412(28):3072 – 3089, 2011.

[2] C. Baier, M. Größer, and F. Ciesinski. Partial order reduction for probabilistic systems. In *QEST '04*, pages 230–239, Washington, DC, USA, 2004. IEEE CS.

[3] Gilles Barthe, Pedro R. D'Argenio, and Tamara Rezk. Secure information flow by self-composition. In *CSFW*, pages 100–114. IEEE Computer Society, 2004.

[4] A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In *Proc. of FSTTCS 95*, LNCS 1026, pages 288–299. Springer, 1995.

[5] Georgel Calin, Pepijn Crouzen, Pedro R. D'Argenio, E. Hahn, and Lijun Zhang. Time-bounded reachability in distributed input/output interactive probabilistic chains. In Jaco van de Pol and Michael Weber, editors, *Model Checking Software*, volume 6349 of *LNCS*, pages 193–211. Springer, 2010.

[6] Konstantinos Chatzikokolakis and Catuscia Palamidessi. A framework for analyzing probabilistic protocols and its application to the partial secrets exchange. *Theor. Comput. Sci.*, 389(3):512–527, 2007.

[7] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *J. Cryptology*, 1(1):65–75, 1988.

[8] L. Cheung. *Reconciling Nondeterministic and Probabilistic Choices*. PhD thesis, Radboud Universiteit Nijmegen, 2006.

[9] L. Cheung, N. Lynch, R. Segala, and F. Vaandrager. Switched Probabilistic PIOA: Parallel composition via distributed scheduling. *Theor. Comput. Sci.*, 365(1-2):83–108, 2006.

[10] F. Ciesinski and C. Baier. LiQuor: A tool for qualitative and quantitative linear time analysis of reactive systems. In *Proc. of QEST'06*, pages 131–132. IEEE CS Press, 2006.

[11] P. R. D'Argenio and P. Niebert. Partial order reduction on concurrent probabilistic programs. In *QEST '04*, pages 240–249, Washington, DC, USA, 2004. IEEE CS.

[12] L. de Alfaro. The verification of probabilistic systems under memoryless partial-information policies is hard. In *Proc. of PROBMIV 99*, pages 19–32. University of Birmingham, 1999.

[13] L. de Alfaro, T. A. Henzinger, and R. Jhala. Compositional methods for probabilistic systems. In *Proc. of CONCUR 01*, LNCS 2154, pages 351–365. Springer, 2001.

[14] Sonja Georgievska and Suzana Andova. Retaining the probabilities in probabilistic testing theory. In *FOSSACS*, volume 6014 of *LNCS*, pages 79–93. Springer, 2010.

[15] S. Giro. *On the automatic verification of distributed probabilistic automata with partial information*. PhD thesis, FaMAF, University of Cordoba, 2010. Available at `http://cs.famaf.unc.edu.ar/~sgiro/thesis.pdf`.

[16] S. Giro and P. R. D'Argenio. Quantitative model checking revisited: neither decidable nor approximable. In *Proc. of FORMATS'07*, LNCS 4763, pages 179–194. Springer, 2007.

[17] S. Giro and P.R. D'Argenio. On the verification of probabilistic I/O automata with unspecified rates. In *SAC '09: Proceedings of the 2009 ACM symposium on Applied Computing*, pages 582–586, New York, NY, USA, 2009. ACM.

[18] Sergio Giro. Undecidability results for distributed probabilistic systems. In Marcel Vinicius Medeiros Oliveira and Jim Woodcock, editors, *SBMF*, volume 5902 of *Lecture Notes in Computer Science*, pages 220–235. Springer, 2009.

[19] Sergio Giro. An algorithmic approximation of the infimum reachability probability for probabilistic finite automata. *CoRR*, abs/1009.3822, 2010.

[20] Sergio Giro and Pedro R. D'Argenio. On the expressive power of schedulers in distributed probabilistic systems. *Electr. Notes Theor. Comput. Sci.*, 253(3):45–71, 2009.

[21] Sergio Giro, Pedro R. D'Argenio, and Luis María Ferrer Fioriti. Partial order reduction for probabilistic systems: A revision for distributed schedulers. In Mario Bravetti and Gianluigi Zavattaro, editors, *CONCUR*, volume 5710 of *Lecture Notes in Computer Science*, pages 338–353. Springer, 2009.

[22] Sergio Giro and Markus N. Rabe. Verification of partial-information probabilistic systems using counterexample-guided refinements. In Supratik Chakraborty and Madhavan Mukund, editors, *ATVA*, volume 7561 of *Lecture Notes in Computer Science*, pages 333–348. Springer, 2012.

[23] R.J. van Glabbeek, S.A. Smolka, and B. Steffen. Reactive, generative, and stratified models of probabilistic processes. *Information and Computation*, 121:59–80, 1995.

[24] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In *Proc. of TACAS'06*, LNCS 3920, pages 441–444. Springer, 2006.

[25] Omid Madani, Steve Hanks, and Anne Condon. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artif. Intell.*, 147(1-2):5–34, 2003.

[26] D. Peled. All from one, one for all: On model checking using representatives. In *Proc. of 5th CAV*, LNCS 697, pages 409–423. Springer, 1993.

[27] Silvia Pelozo and Pedro R. D'Argenio. Security analysis in probabilistic distributed protocols via bounded reachability. In *Proceedings of the 7th International Symposium on Trustworthy Global Computing (TGC 2012)*, 2012. To appear.

[28] R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, Laboratory for Computer Science, MIT, 1995.

[29] S.-H. Wu, S. A. Smolka, and E. W. Stark. Composition and behaviors of probabilistic I/O automata. *Theor. Comput. Sci.*, 176(1-2):1–38, 1997.

MEALS Partner Abbreviations

SAU: Saarland University, D

RWT: RWTH Aachen University, D

TUD: Technische Universität Dresden, D

INR: Institut National de Recherche en Informatique et en Automatique, FR

IMP: Imperial College of Science, Technology and Medicine, UK

ULEIC: University of Leicester, UK

TUE: Technische Universiteit Eindhoven, NL

UNC: Universidad Nacional de Córdoba, AR

UBA: Universidad de Buenos Aires, AR

UNR: Universidad Nacional de Río Cuarto, AR

ITBA: Instituto Técnológico Buenos Aires, AR