On the Expressive Power of Schedulers in Distributed Probabilistic Systems

Sergio Giro[1,2]   P. R. D'Argenio[1,3]

FaMAF, UNC – CONICET
Córdoba, Argentina

Abstract

In this paper, we consider several subclasses of distributed schedulers and we investigate the ability of these subclasses to attain worst-case probabilities.

Based on previous work, we consider the class of distributed schedulers, and we prove that randomization adds no extra power to distributed schedulers when trying to attain the supremum probability of any measurable set, thus showing that the subclass of deterministic schedulers suffices to attain the worst-case probability. Traditional schedulers are a particular case of distributed schedulers. So, since our result holds for any measurable set, our proof generalizes the well-known result that randomization adds no extra power to schedulers when trying to maximize the probability of an ω-regular language. However, non-Markovian schedulers are needed to attain supremum probabilities in distributed systems.

We develop another class of schedulers (the strongly distributed schedulers) that restricts the nondeterminism concerning the order in which components execute. We compare this class against previous approaches in the same direction, showing that our definition is an important contribution. For this class, we show that randomized and non-Markovian schedulers are needed to attain worst-case probabilities.

We also discuss the subclass of finite-memory schedulers, showing the intractability of the model checking problem for these schedulers.

Keywords: probabilistic systems, distributed systems

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 [13] or LiQuor [7].

Since MDPs involve nondeterminism, the model checking problem is to find out the lowest 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 above the bound 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. [1,16]) which choose an enabled transition for each path of the system.

The available tools for model checking as PRISM [13] or LiQuor [7] calculate the worst-case probability considering all 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. [9,6,5,8,10]. 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 [1]. The existence of this subclass ensures that the worst-case probability can be found by exhaustive search [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 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 *almighty* scheduler for this system may let G guess the correct answer with probability 1 according to the following sequence: first, it lets T toss the coin, and then it chooses for G the 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 almighty 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 [1], though safe, yield an

---

[4] Although more efficient methods exist [1]
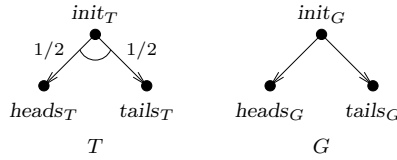
Fig. 1. T tosses a coin and G has to guess

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 [3]. The phenomenon we illustrated has been first observed in [16] from the point of view of compositionality and studied in [8,9,6] in different settings. Distributed schedulers are also related to the partial-information policies of [8].

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 "almighty" scheduler of the example would not be a valid scheduler in this new setting since the choice for G depends only on information which is external to (and not observable by) G. Then, a local scheduler for G takes the decision having no information about the actual state of T, and so the choice cannot be changed according to the outcome of T.

Roughly speaking, in previous literature there is no nondeterminism concerning the different interleavings in which the components execute (for a detailed comparison see Sec. 5). 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.

Contributions. The definition of strongly distributed schedulers we provide is the first one to capture the notion of partial information in asynchronous distributed systems in a general way, as discussed in Sec. 5.

As the familiar reader would expect, we found that Markovian schedulers fail to attain worst-case probabilities. Surprisingly, when considering *strongly distributed* schedulers, we found examples in which *deterministic* strongly distributed schedulers do not attain worst-case probabilities, that is, the schedulers that choose a distribution on the available transitions are more powerful than the schedulers that choose a single transition.

However, as an interesting result, we proved that deterministic *distributed* schedulers attain worst-case probabilities *for any measurable property*. Since traditional schedulers for MDPs are a particular case of distributed schedulers (just consider

A generative structure with two transitions   A reactive structure with two transitions
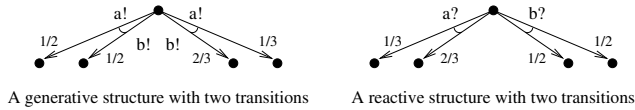
Fig. 2. Reactive and generative structures

a distributed system having only one component) we conclude that deterministic traditional schedulers attain extreme probabilities for any measurable set. In the setting of MDPs, this result has been proven only for ω-regular sets —see, e.g. [1,16]. As pointed out in [2], the generalization to measurable sets (only for the particular case of total information schedulers) can also be derived from very non-trivial results in Borel games [14]. Our proof is, however, much simpler and suited to the MDP setting (and also valid for distributed schedulers).

The model checking problem considering only distributed schedulers has been proven to be undecidable in general [10]. So, one may think that undecidability can be overcome by restricting the schedulers to have finite memory. In this case, an obvious question is how much memory the scheduler should have in order to accurately approximate the worst-case value. We show that the amount of memory needed to get an approximation of the worst-case value cannot be calculated. In addition, we show that nondeterministic schedulers are more powerful than deterministic schedulers given a fixed amount of memory. We also show that the problem of calculating the worst-case value among all Markovian distributed schedulers is NP-hard.

We expect these results and these examples to be useful for further developments on model checking of distributed probabilistic systems.

2   Interleaved Probabilistic Input/Output Automata

We present a framework based on the Switched PIOA [6] (see Sec. 5 for a detailed comparison). It uses reactive and generative structures (see [12,18]). 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 $R : \mathsf{St} \times \mathsf{ActLab} \to \mathcal{P}(T_R)$. 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 2.1 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 (R, resp.) is a generative (reactive, resp.) structure on $(\mathsf{St}, \mathsf{ActLab})$. init $\in \mathsf{St}$ is the initial state. We require the atoms to be input-enabled, so $R(s, a) \neq \emptyset$ for every $s \in \mathsf{St}$, $a \in \mathsf{ActLab}$.

An interleaved probabilistic I/O system P is a 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} (T_{R_i}, resp.) for the set of generative (reactive, resp.) transitions on $(\mathsf{St}_i, \mathsf{ActLab}_i)$.

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 synchronize and *exactly one* of them must participate with an output (generative) transition (thus modelling multicasting). Formally, a compound transition is a tuple $(g_i, a, r_{j_1}, \cdots, r_{j_m})$ (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 say that A_i, A_{j_1}, $\ldots$, 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 $r_{j_k} \in R_{j_k}(s_{j_k}, a)$. The action label a of a compound transition c is indicated by $label(c)$. 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, $c(s, s') = 0$.

In order to ease some definitions, we introduce a fictitious "stutter" compound transition ς. Intuitively, this transition is executed iff the system has reached a state in which no atom is able to generate a transition. The probability $\varsigma(s, s')$ of reaching s' from s using ς is 1, if $s = s'$, or 0, otherwise.

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 c_i is enabled in s_i and $c(s_i, s_{i+1}) > 0$. 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. We define $\mathrm{last}(\sigma) = s_n$ and $\mathrm{len}(\sigma) = n$.

In the following, 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 (paths, resp.) of the compound system are called global states (global paths, resp.) and the states (paths, resp.) of each atom are called local states (local paths, resp.).

3   Schedulers

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

In the usual MDP setting, schedulers assign probabilities to the available tran-sitions 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 nonde-terministic choices need to be resolved. An atom needs a corresponding *output* scheduler to choose the next generative transition. In addition, it may be the case that many reactive transitions are enabled for a single label in the same atom. Hence, for each atom we need an *input* scheduler in order to choose a reactive tran-sition for each previous history and for each label. Output and input schedulers are able to make their 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)** $\sigma.c.s[i] = \sigma[i] .label(c).\pi_i(s)$, 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 $G_i(\mathrm{last}(\sigma[i])) \neq \emptyset$ then $\Theta_i(\sigma[i])(g) > 0 \implies g \in G_i(\mathrm{last}(\sigma[i]))$. An input scheduler for an atom A_i is a function $\Upsilon_i : \mathsf{Proj}_i(P) \times \mathsf{ActLab}_i \to \mathsf{Dist}(T_{R_i})$ s.t. $\Upsilon_i(\sigma[i], a)(r) > 0 \implies r \in R_i(\mathrm{last}(\sigma[i]), a)$. 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 probabilisti-cally 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 ac-cording to the output scheduler). Formally, an *interleaving scheduler* is a func-tion $\mathcal{I} : \mathsf{Paths}(P) \to \mathsf{Dist}(\{1, \cdots, N\})$ such that, if there exists i such that $G_i(\mathrm{last}(\sigma[i])) \neq \emptyset$ (that is, if there is some atom being able to generate a transition) then $\mathcal{I}(\sigma)(i) > 0 \implies G_i(\mathrm{last}(\sigma[i])) \neq \emptyset$. Note that, even if interleaving schedulers are unrestricted, compound schedulers for the compound system are still restricted,

since the local 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 and input schedulers of each atom.

Definition 3.1 Given an interleaving scheduler $\mathcal{I}$, input schedulers Υ_i and output schedulers Θ_i for each atom i, the distributed scheduler η obtained by composing $\mathcal{I}$, Θ_i and Υ_i is defined as:

$$\eta(\sigma)(g_i, a, r_{k_1}, \cdots, r_{k_m}) = \mathcal{I}(\sigma)(i) \cdot \Theta_i(\sigma[i])(g_i) \cdot \prod_{j=1}^{m} \Upsilon_{k_j}(\sigma[k_j], a)(r_{k_j})$$

where A_{k_j} are all the atoms such that $a \in \mathsf{ActLab}_{k_j}$.

Usually, schedulers are defined to map into distributions on transitions. However, it may be the case that $\sum_c \eta(\sigma)(c) > 1$ for a distributed scheduler η. This is because action labels are *not* chosen by the scheduler (they are chosen by the generative transition). However, for every label a, $\sum_{\{c|label(c)=a\}} \eta(\sigma)(c) = 1$.

The probability of the sets of the form $[\sigma]$ is inductively defined as follows: the probability $\mathrm{Pr}^{\eta}([\mathsf{init}])$ of the extensions of the initial state is 1. If there exists i s.t. $G_i(\mathrm{last}(\sigma)) \neq \emptyset$, then the probability $\mathrm{Pr}^{\eta}([\sigma.c.s])$ is $\mathrm{Pr}^{\eta}(\sigma) \cdot \eta(\sigma)(c) \cdot c(\mathrm{last}(\sigma), s)$. If there is no such i, then the system cannot generate any transition. In this case, we let $\mathrm{Pr}^{\eta}([\sigma.c.s]) = \mathrm{Pr}^{\eta}([\sigma])$ if $c = \varsigma$ and $s = \mathrm{last}(\sigma)$, or 0 otherwise.

Note that, if $c = (g_i, a, r_{j_1}, \cdots, r_{j_m})$, then $\eta(\sigma)(c) \cdot c(s, s')$ is

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

which implies $\sum_{c,s'} \eta(\sigma)(c) \cdot c(\mathrm{last}(\sigma), s') = 1$. This probability can be extended to the least σ-field containing all the sets of extensions in the standard way. We say that the sets in such σ-field are *measurable*. Given a measurable set S, we are interested in the value $\sup_\eta \mathrm{Pr}^{\eta}(S)$. By calculating this amount it can be answered, for instance, whether or not "the probability of a package loss is less then 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 $\mathrm{Pr}^{\eta}(\mathrm{reach}(U))$ the probability of reaching any state in U.

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
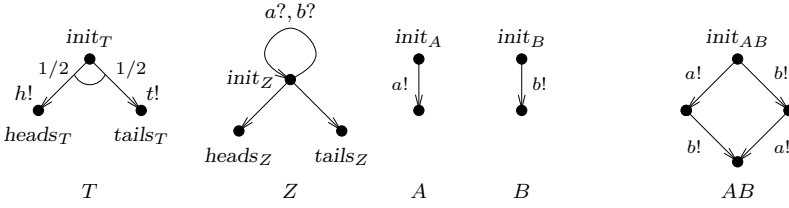
Fig. 3. Motivating strongly distributed schedulers

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 [4]). 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.

Consider the maximum probability that attacker Z guesses the outcome (i.e. the probability that a state of the form $(heads_T, heads_Z, \cdots)$ or $(tails_T, tails_Z, \cdots)$ is reached). 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 A and B as a single component AB, 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 $\text{Atoms}(P') = \big(\text{Atoms}(P) \setminus \{A, B\}\big) \cup \{AB\}$. 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,

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

Suppose $(\mathsf{init}_T, \mathsf{init}_A, \mathsf{init}_B) \xrightarrow{t!} (heads_T, \mathsf{init}_A, \mathsf{init}_B)$ in P. The corresponding path in P' is $(\mathsf{init}_T, \mathsf{init}_{AB}) \xrightarrow{t!} (heads_T, \mathsf{init}_{AB})$. Call both these paths σ_{heads} (ambiguity is resolved according to whether it is used in the context of P or P').

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

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

and similarly for $b!$. Therefore $\Theta_{AB}(\sigma_{heads}[AB])(a!) = 2\Theta_{AB}(\sigma_{heads}[AB])(b!)$. This relation has to be maintained in P by $\mathcal{I}(\sigma_{heads})$. That is, whichever is the probabilistic choice in $\mathcal{I}(\sigma_{heads})$ w.r.t. other atoms, the relation $\mathcal{I}(\sigma_{heads})(a!) = 2 \cdot \mathcal{I}(\sigma_{heads})(b!)$ 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

$$\frac{\mathcal{I}(\sigma)(A)}{\mathcal{I}(\sigma)(A) + \mathcal{I}(\sigma)(B)} = \frac{\mathcal{I}(\sigma')(A)}{\mathcal{I}(\sigma')(A) + \mathcal{I}(\sigma')(B)} \tag{2}$$

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

Definition 3.2 A scheduler η is *strongly distributed* iff η is distributed and equation (2) 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 3.3 *Let P be a system such that $A, B \in \text{Atoms}(P)$. Consider the system P' such that $\text{Atoms}(P') = \big(\text{Atoms}(P) \setminus \{A, B\}\big) \cup \{AB\}$, where AB is the usual cross-product of A and B (as in, for instance, [5, 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.*

Proof. We show that the condition imposed to the interleaving scheduler is sufficient to define an output scheduler for AB. Let σ_{AB} be a local path on AB, and let σ be a global path σ such that $\sigma[AB] = \sigma_{AB}$. Define

$$\Theta_{AB}(\sigma_{AB})(g_A) = \frac{\mathcal{I}(\sigma)(A)}{\mathcal{I}(\sigma)(A) + \mathcal{I}(\sigma)(B)} \, \Theta_A(\sigma[A]) \, .$$

Note that the condition imposed to $\mathcal{I}$ ensures that the particular σ chosen is not relevant. Let $\mathcal{I}'$ be the interleaving scheduler for P_{AB} such that $\mathcal{I}'(\sigma)(AB) = \mathcal{I}(\sigma)(A) + \mathcal{I}(\sigma)(B)$ and $\mathcal{I}'(\sigma)(C) = \mathcal{I}(\sigma)(C)$ for any other atom C. We have to prove that the scheduler η' for P_{AB} obtained from $\mathcal{I}'$ as interleaving scheduler and Θ_{AB} as output scheduler for AB yields the same behaviour as the original scheduler η for P. To see this, note that for a path σ, the probability assigned to a generative transition g_A of A is $p_{\sigma, g_A} = \mathcal{I}(\sigma)(A) \cdot \Theta_A(\sigma[A])(g_A)$. Multiplying and dividing by $\mathcal{I}(\sigma)(A) + \mathcal{I}(\sigma)(B)$ yields

$$p_{\sigma, g_A} = \big(\mathcal{I}(\sigma)(A) + \mathcal{I}(\sigma)(B)\big) \left(\frac{\mathcal{I}(\sigma)(A)}{(\mathcal{I}(\sigma)(A) + \mathcal{I}(\sigma)(B))} \Theta_A(\sigma[A])(g_A) \right) ,$$

which equals to $\mathcal{I}'(\sigma)(AB) \cdot \Theta_{AB}(\sigma[AB])(g_A)$, that is, the probability of p_{σ, g_A} in η'. The same reasoning allows to conclude a similar equality if atom B is considered instead of A. The input, output, and interleaving schedulers do not change in all other cases.   □

One may wonder what happens if, instead of considering two atoms A and B in (2), two *disjoint sets* $\mathcal{A}, \mathcal{B}$ of atoms are considered. The (apparently more general) condition on sets holds whenever condition (2) on atom holds.

Theorem 3.4 *Let $\mathcal{A} = \{A_1, \cdots, A_n\}$, $\mathcal{B} = \{B_1, \cdots, B_m\}$ be disjoint sets of atoms. Then, if Eqn. 2 holds, 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 $\sum_i \mathcal{I}(\sigma')(A_i) + \sum_j \mathcal{I}(\sigma')(B_j) \neq 0$.*

Proof. By induction on n. We prove the base case $n = 1$ by induction on m. If $m = 1$, the statement becomes Eqn. 2. For the inductive step, we need a preliminary equality. Note that, if $\mathcal{I}(\sigma)(A) \neq 0$ and $\mathcal{I}(\sigma')(A) \neq 0$ in Eqn. 2, then simple arithmetic gives

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

The inductive step is

$$\frac{\mathcal{I}(\sigma)(A_1)}{\mathcal{I}(\sigma)(A_1) + \sum_j \mathcal{I}(\sigma)(B_j)} = \frac{\mathcal{I}(\sigma')(A_1)}{\mathcal{I}(\sigma')(A_1) + \sum_j \mathcal{I}(\sigma')(B_j)} \ .$$

First, we prove the case $\mathcal{I}(\sigma)(A_1) = 0$. In this case, either $\mathcal{I}(\sigma)(B_j) = 0$ for all j (in this case the condition $\mathcal{I}(\sigma)(A_1) + \sum_j \mathcal{I}(\sigma)(B_j) \neq 0$ is false, then the equation is not required to hold) or $\mathcal{I}(\sigma')(A_1) = 0$. To see this, suppose towards the contradiction that $\mathcal{I}(\sigma')(A_1) \neq 0$. Then, by Eqn. 2 it must be

$$\frac{\mathcal{I}(\sigma)(A_1)}{\mathcal{I}(\sigma)(A_1) + \mathcal{I}(\sigma)(B_{j^*})} = \frac{\mathcal{I}(\sigma')(A_1)}{\mathcal{I}(\sigma')(A_1) + \mathcal{I}(\sigma')(B_{j^*})}$$

where j^* is an index such that $\mathcal{I}(\sigma)(B_{j^*}) > 0$ (we don't need $\mathcal{I}(\sigma')(B_{j^*}) \neq 0$, since $\mathcal{I}(\sigma')(A_1) \neq 0$). So, since $\mathcal{I}(\sigma)(A_1) = 0$ then it must be $\mathcal{I}(\sigma')(A_1) = 0$, thus reaching a contradiction. Therefore, the inductive step holds in case $\mathcal{I}(\sigma)(A_i) = 0$.

If $\mathcal{I}(\sigma)(A_1) \neq 0$, then either $\mathcal{I}(\sigma')(A_1) = 0$ and $\mathcal{I}(\sigma')(B_j) = 0$ for all j (and so the condition is not required to hold) or $\mathcal{I}(\sigma')(A_1) \neq 0$, and so we can use Eqn. 3 in the following calculation.

$$\frac{\mathcal{I}(\sigma)(A_1)}{\mathcal{I}(\sigma)(A_1) + \sum_j \mathcal{I}(\sigma)(B_j)}$$

$$= \ \{\text{Arithmetics}\}$$

$$\left(\frac{\mathcal{I}(\sigma)(B_m)}{\mathcal{I}(\sigma)(A_1)} + \frac{\mathcal{I}(\sigma)(A_1) + \sum_{j=1}^{m-1} \mathcal{I}(\sigma)(B_j)}{\mathcal{I}(\sigma)(A_1)} \right)^{-1}$$

$$= \ \{\text{Equation 3}\}$$

$$\left(\frac{\mathcal{I}(\sigma')(B_m)}{\mathcal{I}(\sigma')(A_1)} + \frac{\mathcal{I}(\sigma)(A_1) + \sum_{j=1}^{m-1} \mathcal{I}(\sigma)(B_j)}{\mathcal{I}(\sigma)(A_1)} \right)^{-1}$$

$$= \ \{\text{Inductive hypothesis}\}$$

$$\left(\frac{\mathcal{I}(\sigma')(B_m)}{\mathcal{I}(\sigma')(A_1)} + \frac{\mathcal{I}(\sigma')(A_1) + \sum_{j=1}^{m-1} \mathcal{I}(\sigma')(B_j)}{\mathcal{I}(\sigma')(A_1)} \right)^{-1}$$

$$= \ \{\text{Arithmetics}\}$$

$$\frac{\mathcal{I}(\sigma')(A_1)}{\mathcal{I}(\sigma')(A_1) + \sum_j \mathcal{I}(\sigma')(B_j)}$$

Then, the statement holds for $n = 1$. For the remaining inductive step, we calculate:

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

$$
= \frac{\sum_{i=1}^{n-1} \mathcal{I}(\sigma)(A_i) + \mathcal{I}(\sigma)(A_n)}{\sum_{i=1}^{n-1} \mathcal{I}(\sigma)(A_i) + \mathcal{I}(\sigma)(A_n) + \sum_j \mathcal{I}(\sigma)(B_j)}
$$

$$
= \frac{\sum_{i=1}^{n-1} \mathcal{I}(\sigma)(A_i)}{\sum_{i=1}^{n-1} \mathcal{I}(\sigma)(A_i) + \mathcal{I}(\sigma)(A_n) + \sum_j \mathcal{I}(\sigma)(B_j)} + \frac{\mathcal{I}(\sigma)(A_n)}{\sum_{i=1}^{n-1} \mathcal{I}(\sigma)(A_i) + \mathcal{I}(\sigma)(A_n) + \sum_j \mathcal{I}(\sigma)(B_j)}
$$

$$
= \{\text{Inductive hypothesis for } \{A_i\}_{i=1}^{n-1}, A_n \cup \{B_j\}_{j=1}^m\}
$$

$$
= \frac{\sum_{i=1}^{n-1} \mathcal{I}(\sigma')(A_i)}{\sum_{i=1}^{n-1} \mathcal{I}(\sigma')(A_i) + \mathcal{I}(\sigma')(A_n) + \sum_j \mathcal{I}(\sigma')(B_j)} + \frac{\mathcal{I}(\sigma)(A_n)}{\sum_{i=1}^{n-1} \mathcal{I}(\sigma)(A_i) + \mathcal{I}(\sigma)(A_n) + \sum_j \mathcal{I}(\sigma)(B_j)}
$$

$$
= \{\text{Base case with } \{A_n\}, \{B_i\}_{i=1}^m \cup \{A_i\}_{i=1}^{n-1}\}
$$

$$
= \frac{\sum_{i=1}^{n-1} \mathcal{I}(\sigma')(A_i)}{\sum_{i=1}^{n-1} \mathcal{I}(\sigma')(A_i) + \mathcal{I}(\sigma')(A_n) + \sum_j \mathcal{I}(\sigma')(B_j)} + \frac{\mathcal{I}(\sigma')(A_n)}{\sum_{i=1}^{n-1} \mathcal{I}(\sigma')(A_i) + \mathcal{I}(\sigma')(A_n) + \sum_j \mathcal{I}(\sigma')(B_j)}
$$

$\square$

4   Subclasses of distributed schedulers

Next, we discuss the expressive power of several subclasses of distributed schedulers.

4.1   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 all the input (output, interleaving, resp.) schedulers choose a reactive transition (generative transition, atom, resp.) with probability 1. That is, $\Upsilon_i(\sigma_i, a)(r_i) > 0 \implies \Upsilon_i(\sigma_i, a)(r_i) = 1$ (and similarly for output and interleaving schedulers).

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

Theorem 4.1 *For any set S of infinite traces, S being measurable, we have that*

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

The proof of this theorem is very long and so we split it in several lemmata.

First, we need some elements from probability theory. These definitions and the proofs not given here can be found at [17].

Definition 4.2 Given a set Σ, a semi-ring is a set $\mathcal{S} \subseteq \mathcal{P}(\Sigma)$ complying:
- $\emptyset \in \mathcal{S}$,

- $A, B \in \mathcal{S} \implies A \cap B \in \mathcal{S}$,
- $A, B \in \mathcal{S} \implies \exists n \geq 0, \exists A_i \in \mathcal{S} : A \setminus B = \biguplus_{i=1}^{n} A_i$.

A ring is a set $\mathcal{R} \subseteq \mathcal{P}(\Sigma)$ complying:

- $\emptyset \in \mathcal{R}$,
- $A, B \in \mathcal{R} \implies A \cup B \in \mathcal{S}$,
- $A, B \in \mathcal{S} \implies A \setminus B \in \mathcal{S}$.

The ring $\mathcal{R}(\mathcal{S})$ generated by a semi-ring $\mathcal{S}$ is the least ring containing $\mathcal{S}$.

It can be proven that each element in the ring generated by a semi-ring $\mathcal{S}$ is of the form $\biguplus_{i=1}^{n} A_i$ with $A_i \in \mathcal{S}$. The set of whose elements are all the sets $[\sigma]$ forms a semi-ring. In the following, we denote this semi-ring by $\mathcal{S}$, while $\mathcal{R}$ denotes the ring generated by $\mathcal{S}$.

The following lemma states that the probability of any measurable set can be approximated as the probability of a countable disjoint union of sets of extensions.

Lemma 4.3 *Let $\mathcal{C}^\omega$ be the set*

$$\left\{ \{A_i\}_{i=1}^{\infty} \mid \forall i, j, i \neq j \bullet A_i \in \mathcal{S} \wedge A_i \cap A_j = \emptyset \right\} .$$

For every measurable set of infinite paths S, we have

$$\Pr^\eta(S) = \inf_{\{C \in \mathcal{C}^\omega \mid S \subseteq \biguplus_{A \in C} A\}} \sum_{A \in C} \Pr^\eta(A) .$$

Proof. An $\mathcal{R}$-cover of a set S is a set $\{B_i\}_{i=1}^{\infty}$ where $B_i \in \mathcal{R}$ and $S \subseteq \bigcup_{n=1}^{\infty} B_i$. Let $\mathcal{P}(S)$ be the set of all the $\mathcal{R}$-covers of S. The probability of a measurable set S in the σ-algebra generated by the semi-ring $\mathcal{S}$ can be defined as

$$\inf_{\{B_i\}_{i=1}^{\infty} \in \mathcal{P}(S)} \sum_{i=1}^{\infty} \Pr^\eta(B_i)$$

(see [17]). Given an $\mathcal{R}$-cover $\{B_i\}$ for S where each B_i is of the form $\biguplus_{k=0}^{n_i} A_k^i$, we define an element C in $\mathcal{C}^\omega$ as follows: $A \in C$ iff $A = A_k^i$ for some i,k and there is no $A_{k'}^{i'}$ such that $A_k^i \subset A_{k'}^{i'}$. Since our semi-ring is the set of extension sets, in the construction of C we dropped the extensions $[\sigma']$ such that there exists $[\sigma]$ with σ being a prefix of σ'.

Then, we have

$$\sum_{n=1}^{\infty} \Pr^\eta(B_n) = \sum_{i=1}^{\infty} \sum_{k=0}^{n_i} \Pr^\eta(A_k^i) \geq \sum_{A \in C} \Pr^\eta(A)$$

In addition, C is an $\mathcal{R}$-cover of S, since in the construction of C we only dropped sets of extensions included in other sets of extensions.

So, for each $\mathcal{R}$-cover we found another $\mathcal{R}$-cover in $\mathcal{C}^\omega$ yielding less or equal probability, thus completing the proof. □

The following lemma concerns the infimum probability of "finite-horizon" properties of the form $\biguplus_{i=1}^{n}[\sigma_i]$. Note that the only choices affecting such probability are the choices for the paths having length less than $N = \max_i\{\text{len}(\sigma_i)\}$.

Lemma 4.4 *For all sequences of finite paths $\{\sigma_i\}_{i=1}^{n}$ such that $[\sigma_i] \cap [\sigma_j] = \emptyset$ for all $i \neq j$, there exists a deterministic distributed scheduler η^d such that*

$$\text{Pr}^{\eta^d}(\biguplus_{i=1}^{n}[\sigma_i]) = \inf_{\eta \in \text{Dist}(P)} \text{Pr}^{\eta}(\biguplus_{i=1}^{n}[\sigma_i]) .$$

Proof. Similarly as in Lemma 3 in [10], given any distributed scheduler η and any local path σ^* we obtain a deterministic distributed scheduler $\det(\eta, \sigma^*)$ such that η chooses deterministically for σ^* and $\det(\eta, \sigma^*)$ yields less probability than η. In order to obtain the deterministic scheduler η^d, we successively transform η to choose deterministically for all the local paths whose length is less than N, where $N = \max_i\{\text{len}(\sigma_i)\}$. That is, we consider the scheduler $\eta^N = \det(\det(\cdots \det(\eta, \sigma^1), \cdots), \sigma^N)$, where $\sigma^1 \cdots \sigma^N$ are all the local paths whose length is less than N. Given the scheduler η^N, we consider each local path of length greater than or equal to N, and for these paths we define the new scheduler η^d to deterministically choose a transition (the particular transition chosen is not relevant, since the choices for paths of length greater than or equal to N do not affect the value of $\text{Pr}^{\eta^d}(\biguplus_{i=1}^{n}[\sigma_i])$).

The existence of such η^d ensures that the infimum quantifying over deterministic schedulers is less than or equal to the infimum quantifying over possibly nondeterministic schedulers. In addition, we conclude that there exists a scheduler yielding the infimum probability, since there are only finitely many combinations of deterministic choices for the paths of length less than N.

The only difference with respect to the proof in [10] is that the choices must be made deterministic for every local path and for every input and output scheduler. In addition, the choices must be made deterministic for the interleaving scheduler, by considering every global path.

In order to show that our input/output mechanism does not introduce any issue, we illustrate how to transform the choices for the input schedulers by mimicking the proof in [10]. In the proof, we manipulate finite paths. In order to do this, for a path $\sigma = s_1.c_1.\cdots.c_{n-1}.s_n$ we define $\sigma(i) = s_i$ and $\sigma\langle i \rangle = c_i$. In addition $\sigma\downarrow_i = s_1.c_1 \cdots c_{i-1}.s_i$, $\text{last}(\sigma) = s_n$ and $\text{len}(\sigma) = n$.

Let σ^* be a path of an atom A_i and let $a \in \text{ActLab}_i$. We show how to make the choice deterministic for the input scheduler of A_i when a occurs in σ^*. Let r_{σ^*} be the set of all the paths in $\{\sigma_i\}_{i=1}^{n}$ such that "a occurs in σ^*", that is, there exists k_σ such that $\sigma\downarrow_{k_\sigma}[A_i] = \sigma^*$ and $\text{label}(\sigma\langle k_\sigma \rangle) = a$. The probabilities of the paths in r_{σ^*} are the only ones to be changed, since we are only changing $\Upsilon_i(\sigma^*, a)$. So, we show only that, for this set, the scheduler in which the choice is deterministic yields a probability less than or equal to the probability yielded by the original scheduler.

Let A_{g_σ} be the atom that generates the output a in $\sigma\downarrow_{k_\sigma}$ and g^σ be the corresponding generative transition. Let r_j^σ be the reactive transition executed by A_j

when a occurs in σ in the k_σ-th step. We will focus on $\Upsilon_i(\sigma^*, a)$. The probability of a path σ in r_{σ^*} is $\Upsilon_i(\sigma^*, a) \cdot r_i^\sigma(\sigma(k_\sigma + 1)) \cdot Q_\sigma$, where

$$
Q_\sigma = \Pr^\eta([\sigma\!\downarrow_{k_\sigma}]) \cdot \mathcal{I}(\sigma\!\downarrow_{k_\sigma})(A_{g_\sigma}) \cdot \Theta_{g_\sigma}(\sigma\!\downarrow_{k_\sigma}[g_\sigma])(g^\sigma)
$$
$$
\cdot \prod_{w \in \{1, \cdots, m\} \setminus \{i\}} \Upsilon_{j_w}(\sigma\!\downarrow_{k_\sigma}[j_w], a)(r_{j_w}^\sigma)
$$
$$
\cdot g^\sigma(\pi_{g_\sigma}(\sigma(k_\sigma + 1)), a)
$$
$$
\cdot \prod_{w \in \{1, \cdots, m\} \setminus \{i\}} r_{j_w}^\sigma(\pi_{j_w}(\sigma(k_\sigma + 1)))
$$
$$
\cdot \prod_{t=k_\sigma+1}^{\text{len}(\sigma)-1} \eta(\sigma\!\downarrow_t)(\sigma\langle t\rangle) \cdot \sigma\langle t\rangle(\sigma(t), \sigma(t+1))
$$

Now, we calculate,

$$
\sum_{\sigma \in r_{\sigma^*}} \Pr^\eta([\sigma])
$$

$=$ {Definition of probabilities for extensions}

$$
\sum_{\sigma \in r_{\sigma^*}} r_i^\sigma(\pi_i(\sigma(k_\sigma + 1))) \ \Upsilon_i(\sigma^*)(r_i^\sigma) \ Q_\sigma
$$

$=$ {Rearrange sums}

$$
\sum_{r_i} \sum_s \sum_{\{\sigma \in r_{\sigma^*} | r_i^\sigma = r_i \wedge \pi_i(\sigma(k_\sigma+1))=s\}} r_i^\sigma(\pi_i(\sigma(k_\sigma + 1))) \ \Upsilon_i(\sigma^*)(r_i^\sigma) \ Q_\sigma
$$

$=$ {Rearrange sums}

$$
\sum_{r_i} \sum_s \sum_{\{\sigma \in r_{\sigma^*} | r_i^\sigma = r_i \wedge \pi_i(\sigma(k_\sigma+1))=s\}} r_i(s) \ \Upsilon_i(\sigma^*)(r_i) \ Q_\sigma
$$

$= \sum_{r_i} \Upsilon_i(\sigma^*)(r_i) \sum_s r_i(s) \sum_{\{\sigma \in r_{\sigma^*} | r_i^\sigma = r_i \wedge \pi_i(\sigma(k_\sigma+1))=s\}} Q_\sigma$

Let

$$
r^* = \arg\min_{r_i} \sum_s r_i(s) \sum_{\{\sigma \in r_{\sigma^*} | r_i^\sigma = r_i \wedge \pi_i(\sigma(k_\sigma+1))=s\}} Q_\sigma .
$$

Since $\sum_{r_i} \Upsilon_i(\sigma^*)(r_i) = 1$, we have

$$
\sum_{\sigma \in r_{\sigma^*}} \Pr^\eta([\sigma]) \geq \sum_s r^*(s) \sum_{\{\sigma \in r_{\sigma^*} | r_i^\sigma = r^* \wedge \pi_i(\sigma(k_\sigma+1))=s\}} Q_\sigma ,
$$

which is the probability using the scheduler $\det(\eta)$ that mimics η except for Υ_i. The input scheduler for A_i in $\det(\eta)$ chooses $\Upsilon_i'(\sigma^*, a)(r^*) = 1$.

The choices for the output schedulers can be made deterministic in an easier way (since labels need not be considered).

With respect to the interleaving scheduler, let σ^* be a path of the system of length less than N. Let r_{σ^*} be the set of all the paths σ_i having σ^* as suffix. Let $k = \text{len}(\sigma^*)$. As before, for every $\sigma \in r_{\sigma^*}$, let g_σ be the atom that performs an output in the k-th step, and g^σ be the corresponding generative transition. Moreover, let a_σ be the label after the k-th step in σ and let r_j be the reactive transition executed

by A_j after the k-th step. Let

$$Q_\sigma = \mathrm{Pr}^\eta([\sigma{\downarrow}_k]) \cdot \prod_{w\in\{1,\cdots,m\}} \Upsilon_{j_w}(\sigma{\downarrow}_{k_\sigma}[j_w], a)(r_{j_w})$$
$$\cdot \prod_{w\in\{1,\cdots,m\}} r_{j_w}(\pi_{j_w}(\sigma(k_\sigma+1)))$$
$$\cdot \prod_{t=k+1}^{\mathrm{len}(\sigma)-1} \eta(\sigma{\downarrow}_t)(\sigma\langle t\rangle)\ \sigma\langle t\rangle(\sigma(t),\sigma(t+1))$$

Then, we proceed similarly as before:

$$\sum_{\sigma\in r_{\sigma^*}} \mathrm{Pr}^\eta([\sigma])$$
$$= \sum_{\sigma\in r_{\sigma^*}} \mathcal{I}(\sigma^*)(A_{g_\sigma})\ \Theta_{g_\sigma}(\sigma{\downarrow}_k[g_\sigma])(g^\sigma)\ g^\sigma(\pi_{A_{g_\sigma}}(\sigma(k+1)),a_\sigma)\ Q_\sigma$$
$$=$$

$$\sum_{A_i}\ \sum_{\sigma_i,g_i}\ \sum_{s_i,a}$$
$$\sum_{\{\sigma\in r_{\sigma^*}|A_{g_\sigma}=A_i\wedge\sigma{\downarrow}_k[i]=\sigma_i\wedge\pi_{A_i}(\sigma(k+1))=s_i\wedge g^\sigma=g_i\wedge a_\sigma=a\}}$$
$$\mathcal{I}(\sigma^*)(A_{g_\sigma})\ \Theta_{g_\sigma}(\sigma{\downarrow}_k[A_{g_\sigma}])(g^\sigma)\ g^\sigma(\pi_{A_{g_\sigma}}(\sigma(k+1)),a_\sigma)\ Q_\sigma$$

$$=$$

$$\sum_{A_i}\ \sum_{\sigma_i,g_i}\ \sum_{s_i,a}$$
$$\sum_{\{\sigma\in r_{\sigma^*}|A_{g_\sigma}=A_i\wedge\sigma{\downarrow}_k[i]=\sigma_i\wedge\pi_{A_i}(\sigma(k+1))=s_i\wedge g^\sigma=g_i\wedge a_\sigma=a\}}$$
$$\mathcal{I}(\sigma^*)(A_i)\ \Theta_{g_i}(\sigma{\downarrow}_k[A_i])(g_i)\ g_i(\pi_{A_i}(s),a)\ Q_\sigma$$

$$=$$

$$\sum_{A_i}\mathcal{I}(\sigma^*)(A_i) \sum_{\sigma_i,g_i}\Theta_i(\sigma_i)(g_i)\sum_{s_i,a} g_i(s_i,a)$$
$$\sum_{\{\sigma\in r_{\sigma^*}|A_{g_\sigma}=A_i\wedge\sigma{\downarrow}_k[i]=\sigma_i\wedge\pi_{A_{g_\sigma}}(\sigma(k+1))=s_i\wedge g^\sigma=g_i\wedge a_\sigma=a\}} Q_\sigma$$

As before, we take

$$A_{i^*} = \arg\min_{A_i} \sum_{\sigma_i,g_i}\Theta_i(\sigma_i)(g_i)\sum_{s_i,a} g_i(s_i,a)$$
$$\sum_{\{\sigma\in r_{\sigma^*}|g_\sigma=i\wedge\sigma{\downarrow}_k[i]=\sigma_i\wedge\pi_{g_\sigma}(\sigma(k+1))=s_i\wedge g^\sigma=g_i\wedge a_\sigma=a\}} Q_\sigma$$

and define $\mathcal{I}'(\sigma) = A_{i^*}$. $\qquad\qquad\square$

For convenience, sometimes we denote a deterministic scheduler η as a function mapping global paths to n-tuples of the form $(g_i, f_1, \cdots, f_N)$, where $f_j : \mathsf{ActLab}_j \to T_{R_j}$ (recall Def. 2.1). Each n-tuple of the form $(g_i, f_1, \cdots, f_{i-1}, f_{i+1}, \cdots, f_N)$ corresponds to several compound transitions: namely, there is one compound transition for each label in ActLab_i. Given an n-tuple as before and a label a, we obtain the compound transition $(g_i, a, f_{r_1}(a), \cdots, f_{r_k}(a))$, where $r_1, \cdots, r_k$ are the atoms that

react to a. Concretely, if η is obtained by composing $\mathcal{I}, \Theta_1, \cdots, \Theta_N, \Upsilon_1, \cdots, \Upsilon_N$ we write $\eta(\sigma) = (g_i, f_1, \cdots, f_{i-1}, f_{i+1}, \cdots, f_N)$ iff $\mathcal{I}(\sigma) = i$ and $\Theta_i(\sigma[i]) = g_i$ and, for all a, j such that $a \in \mathsf{ActLab}_j$, we have $\Upsilon_j(\sigma[j], a) = f_j(a)$.

Note that a function η mapping histories to n-tuples is not necessarily a distributed scheduler. In general, we call these functions *arbitrary schedulers*. Given an arbitrary scheduler η, η is a distributed scheduler iff for all i, σ, σ' s.t. $\sigma[i] = \sigma'[i]$, **(1)** $\eta(\sigma) = (g_i, f_1, \cdots, f_N)$ implies that $\eta(\sigma')$ is of the form $(g_i, f_1', \cdots, f_N')$ and **(2)** $\eta(\sigma) = (g_j, \cdots, f_i, \cdots)$ implies that $\eta(\sigma')$ is of the form $(g_{j'}', \cdots, f_i, \cdots)$. Since we focus on distributed schedulers, schedulers are supposed to be distributed, except when stated otherwise.

The following lemma concerns "infinite-horizon" properties of the form $\biguplus_{i=1}^{\infty}[\sigma_i]$, and shows how to construct an optimal scheduler for such properties using optimal schedulers for the "finite-horizon" approximations of $\biguplus_{i=1}^{\infty}[\sigma_i]$. This optimal scheduler will be used in the proof of Theorem 4.1. Our construction resembles the "limit construction" in [5, Sec. 4.3].

Lemma 4.5 *For all sequences of finite paths $\{\sigma_i\}_{i=1}^{\infty}$ such that $[\sigma_i] \cap [\sigma_j] = \emptyset$ let S_N be the set $\biguplus\{[\sigma_i] \mid \mathrm{len}(\sigma_i) \leq N\}$. If there exists a sequence $\{\eta_N\}_{N=1}^{\infty}$ of deterministic schedulers such that, for all N,*

$$\mathrm{Pr}^{\eta_N}(S_N) = \inf_{\eta} \mathrm{Pr}^{\eta}(S_N)$$

*then there exists a deterministic arbitrary scheduler η^d such that **(1)** for all N there exists $N' > N$ such that $\eta^d(\sigma) = \eta_{N'}(\sigma)$ for all path σ s.t. $\mathrm{len}(\sigma) \leq N$ and **(2)** $\eta^d = \inf_{\eta} \mathrm{Pr}^{\eta}(\biguplus_i[\sigma_i])$.*

Proof. In order to construct η^d, we will construct a sequence of schedulers $\{\eta^N\}_{N=0}^{\infty}$. Then, we simply define $\eta^d(\sigma) = \eta^{\mathrm{len}(\sigma)}(\sigma)$. The idea behind the construction of the schedulers η^N is that η^N must comply the following property: there exists a sequence $\{Z_i^N\}_{i=1}^{\infty}$ (the $\{\cdot\}_{i=1}^{\infty}$ indicates that the sequence is indexed by i) such that

$$\eta^N(\sigma) = \eta_{Z_i^N}(\sigma) \tag{4}$$

for all σ having length less than or equal to N, for all i.

The scheduler η^0 is simply η_1. The sequence $\{Z_i^0\}$ is the sequence $\{i\}_{i=1}^{\infty}$. It trivially complies with (4), since there are no paths of length 0 (init has length 1).

In order to construct the scheduler η^N from the scheduler η^{N-1}, we define schedulers $\eta^{N-1,Q}$, where Q is a set of paths of length N. In addition, each scheduler $\eta^{N-1,Q}$ has a corresponding sequence $\{Z_i^{N-1,Q}\}_{i=1}^{\infty}$. Once these schedulers are defined, we define $\eta^N = \eta^{N-1,Q_N}$ and $Z^N = Z^{N-1,Q_N}$, where Q_N is the set of *all* paths of length N. We will construct the schedulers $\eta^{N-1,Q}$ in such a way that $\eta^{N,Q}(\sigma) = \eta_{Z_i^{N,Q}}(\sigma)$ for all σ such that $\sigma \in Q$ or $\mathrm{len}(\sigma) \leq N-1$. The scheduler $\eta^{N,\{\}}$ is η^{N-1}. Now, we show how to construct $\eta^{N,Q \cup \{\sigma^*\}}$ from $\eta^{N,Q}$.

We consider the sequence $\{\eta_{Z_i^{N,Q}}(\sigma^*)\}_{i=1}^{\infty}$. In this sequence, at least one element a^* is repeated infinitely many times. We let $\eta^{N,Q \cup \{\sigma^*\}}(\sigma^*) = a^*$, and let

$Z^{N,Q\cup\{\sigma^*\}}$ be the infinite subsequence of $Z^{N,Q}$ complying with $\eta_{Z^{N,Q\cup\{\sigma^*\}}}(\sigma^*) = a^*$ (this infinite subsequence is ensured to exist since a^* appears infinitely many times in $\{\eta_{Z_i^{N,Q}}(\sigma^*)\}_{i=1}^{\infty}$).

Now, we prove the properties for η^d enounced in the theorem.

(i) Given any N, we take any N' in the sequence Z^N such that $N' > N$. So, the property for η^d is implied by the property (4) for η^N.

(ii) Suppose, towards a contradiction, that $\Pr^{\eta^d}(\biguplus_i \sigma_i) > \inf_\eta \Pr^\eta(\biguplus_i \sigma_i)$. Since $\Pr^{\eta^d}(\biguplus_i \sigma_i) = \sum_i \Pr^{\eta^d}(\sigma_i)$, there exists N such that

$$\Pr^{\eta^d}(\biguplus_i \{[\sigma_i] \mid \operatorname{len}(\sigma_i) \leq N\}) > \inf_{\eta \in \mathsf{Dist}(P)} \Pr^\eta(\biguplus_i \sigma_i) \tag{5}$$

Let $N' > N$ such that $\eta^d(\sigma) = \eta_{N'}(\sigma)$ for all paths σ such that $\operatorname{len}(\sigma) \leq N$ (its existence is ensured by the previous property) and let η^{inf} be such that $\Pr^{\eta^{\mathrm{inf}}}(\biguplus_i \sigma_i) < \Pr^{\eta^d}(\biguplus_i \{[\sigma_i] \mid \operatorname{len}(\sigma_i) \leq N\})$ (its existence is ensured because of (5)). Now, we reason

$$\begin{aligned}
&\Pr^{\eta^d}(\biguplus\{[\sigma_i] \mid \operatorname{len}(\sigma_i) \leq N\}) \\
&= \Pr^{\eta_{N'}}(\biguplus\{[\sigma_i] \mid \operatorname{len}(\sigma_i) \leq N\}) \\
&\leq \Pr^{\eta_{N'}}(\biguplus\{[\sigma_i] \mid \operatorname{len}(\sigma_i) \leq N'\}) .
\end{aligned} \tag{6}$$

In addition,

$$\begin{aligned}
&\Pr^{\eta^d}(\biguplus_i\{[\sigma_i] \mid \operatorname{len}(\sigma_i) \leq N\}) \\
&> \Pr^{\eta^{\mathrm{inf}}}(\biguplus_i \sigma_i) \\
&\geq \Pr^{\eta^{\mathrm{inf}}}(\biguplus\{[\sigma_i] \mid \operatorname{len}(\sigma_i) \leq N'\}) \\
&\geq \{\text{Optimality condition for } \eta_{N'} \text{ (see theorem statement)}\} \\
&\quad \Pr^{\eta_{N'}}(\biguplus\{[\sigma_i] \mid \operatorname{len}(\sigma_i) \leq N'\}) .
\end{aligned}$$

This contradicts (6).

$\square$

The following lemma simply combines Lemma 4.4 and Lemma 4.5 in order to show that deterministic schedulers are sufficient to obtain the infimum probability of an "infinite-horizon" property as before.

Lemma 4.6 *For all sequences of finite paths* $S = \{\sigma_i\}_{i=1}^{\infty}$ *s.t.* $[\sigma_i] \cap [\sigma_j] = \emptyset$ *for all* $i \neq j$, *there exists a deterministic distributed scheduler* η^* *such that* $\Pr^{\eta^*}(\biguplus_{\{\sigma \in S\}}[\sigma]) = \inf_{\eta \in \mathsf{Dist}(P)} \Pr^\eta(\biguplus_{\{\sigma \in S\}}[\sigma])$.

Proof. For each n, Lemma 4.4 ensures the existence of a deterministic distributed scheduler η_n such that $\Pr^{\eta_n}(\biguplus_{\{\sigma \in S\}}[\sigma]) = \inf_{\eta \in \mathsf{Dist}(P)} \Pr^\eta(\biguplus_{\{\sigma \in S \mid \operatorname{len}(\sigma_i) \leq n\}}[\sigma])$.

Then, Lemma 4.5 ensures the existence of an arbitrary scheduler η^d such that $\eta^d = \inf_\eta \Pr^\eta(\biguplus_{\{\sigma \in S | \text{len}(\sigma_i) \leq n\}} [\sigma])$.

Now, we prove that this arbitrary scheduler is indeed distributed. Suppose, towards a contradiction, that there exist two paths σ, σ' and an atom A_i complying $\sigma[i] = \sigma'[i]$ such that

- $\eta^d(\sigma) = (g_i, f_1, \cdots, f_N)$ and $\eta^d(\sigma') = (g_i', f_1', \cdots, f_N')$ with $g_i \neq g_i'$. Or
- $\eta^d(\sigma) = (g_j, \cdots, f_i, \cdots)$ and $\eta^d(\sigma') = (g_j', \cdots, f_i', \cdots)$ with $f_i \neq f_i'$.

Let $M = \max\{\text{len}(\sigma), \text{len}(\sigma')\}$. Then, by Lemma 4.5 there exists $M' > M$ such that $\eta_{M'}(\sigma) = \eta^d(\sigma)$ and $\eta_{M'}(\sigma') = \eta^d(\sigma')$. Therefore, $\eta_{M'}$ would not be distributed, contradicting the hypothesis for the schedulers η_n. □

Proof. [of Theorem 4.1] Given $\epsilon > 0$, we will find a deterministic distributed η^d such that $\sup_{\eta \in \mathsf{Dist}(P)} \Pr^\eta(S) - \Pr^{\eta^d}(S) < \epsilon$.

We use $\complement S$ to denote the complement of the set S.

Let η^s be such that $\sup_{\eta \in \mathsf{Dist}(P)} \Pr^\eta(S) - \Pr^{\eta^s}(S) < \epsilon/2$. By Lemma 4.3 (applied to the complement of $\complement S$), there exists a sequence $\{[\sigma_i]\}_{i=1}^\infty$ of disjoint extensions sets such that $(\complement S) \subseteq \biguplus_i [\sigma_i]$ and

$$\Pr^{\eta^s}(\biguplus_i [\sigma_i]) - \Pr^{\eta^s}(\complement S) < \epsilon/2 . \tag{7}$$

By Lemma 4.6, there exists a deterministic distributed scheduler η^d such that $\Pr^{\eta^d}(\biguplus_i [\sigma_i]) = \inf_{\eta \in \mathsf{Dist}(P)} \Pr^\eta(\biguplus_i [\sigma_i])$. In particular, $\Pr^{\eta^d}(\biguplus_i [\sigma_i]) \leq \Pr^{\eta^s}(\biguplus_i \sigma_i)$. Therefore, from (7) we have

$$\Pr^{\eta^d}(\biguplus_i [\sigma_i]) - \Pr^{\eta^s}(\complement S) < \epsilon/2 .$$

From which we obtain

$$1 - \Pr^{\eta^d}(\complement \biguplus_i [\sigma_i]) - (1 - \Pr^{\eta^s}(S)) < \epsilon/2 ,$$

this inequation being equivalent to

$$\Pr^{\eta^s}(S) - \Pr^{\eta^d}(\complement \biguplus_i [\sigma_i]) < \epsilon/2 . \tag{8}$$

Since $\complement S \subseteq \biguplus_i [\sigma_i]$ we have that $\complement \biguplus_i [\sigma_i] \subseteq S$. So, $\Pr^{\eta^d}(\complement \biguplus_i [\sigma_i]) \leq \Pr^{\eta^d}(S)$. From (8) we obtain $\Pr^{\eta^s}(S) - \Pr^{\eta^d}(S) < \epsilon/2$. Then,

$$\sup_\eta \Pr^\eta(S) - \Pr^{\eta^d}(S) = \sup_\eta \Pr^\eta(S) - \Pr^{\eta^s}(S) + \Pr^{\eta^s}(S) - \Pr^{\eta^d}(S) < \epsilon/2 + \epsilon/2 = \epsilon .$$

□

Unfortunately, if in the statement of Theorem 4.1 we consider *strongly distributed schedulers* the same claim is false. Consider the example in Fig. 4. Atoms A, B
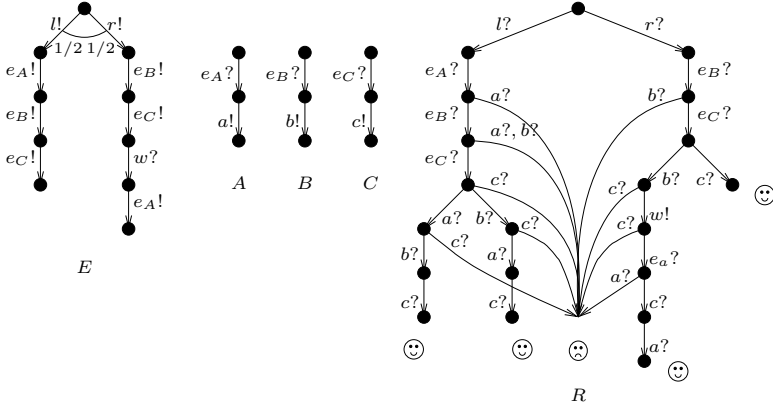
Fig. 4. Example showing that randomization adds power to strongly distributed schedulers

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. Let's see if there exists a deterministic strongly distributed scheduler η reaching a smile with probability 1. In order to yield a probability of 1, η must reach a smile for both 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 activated, 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 a Dirac distribution on E) **(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) **(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 a probability of $13/24 > 1/2$. 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 the *rate schedulers* in [11] (rate schedulers yielding probabilities arbitrarily close to 13/24 can be obtained by replacing arbitrarily high rates for the Dirac distributions and equal rates for the uniform distributions).

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

For traditional almighty 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 (initial states are enclosed in circles). 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. Let's see what the supremum probability of reaching s_w is. If G waits for one t before communicating g, then the probability of reaching s_w is 1/2. However, G may be smarter and wait for two t's, thus having a probability of 3/4. In general, waiting for k t's yields a probability of $1 - (1/2)^k$. In addition, it is easy to see that there is no nondeterministic 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-seing schedulers are considered, *Markovian* schedulers attain the supremum probability for reachability properties [1].

In our setting, one may think of two types of Markovian schedulers: a *globally Markovian* scheduler should comply $\eta(\sigma)(c) = \eta(\sigma')(c)$ whenever $\mathrm{last}(\sigma) = \mathrm{last}(\sigma')$, while a *locally Markovian* scheduler should choose the same local transitions whenever the local states coincide. In order to define locally Markovian schedulers, we say that an input scheduler is *Markovian* iff, for all a, r, it holds $\Upsilon(\sigma, a)(r) = \Upsilon(\sigma', a)(r)$ whenever $\mathrm{last}(\sigma) = \mathrm{last}(\sigma')$. Similarly, Markovian output schedulers can be defined. An interleaving scheduler is *Markovian* iff $\mathcal{I}(\sigma)(A) = \mathcal{I}(\sigma')(A)$ whenever $\mathrm{last}(\sigma) = \mathrm{last}(\sigma')$. 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, lo-
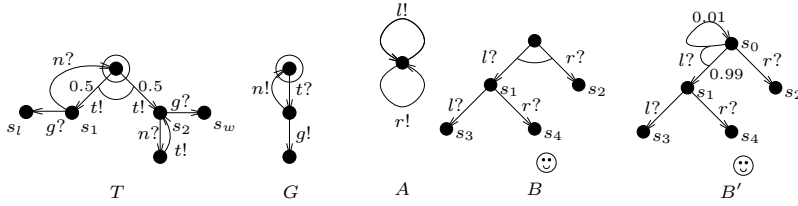
Fig. 5. Atoms used in our examples

cally 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 local (global, resp.) finite-memory distributed schedulers of a system P by LFinMem(P) (GFinMem(P), resp.) and the deterministic finite-memory schedulers by DetLFinMem(P) (DetGFinMem(P), resp.) We illustrate the limitations of finite-memory schedulers using 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 4.7 $\forall U : \sup_{\eta \in \mathsf{Dist}(P)} \Pr^{\eta}(\mathrm{reach}(U)) = \sup_{\eta \in \mathsf{DetLFinMem}(P)} \Pr^{\eta}(\mathrm{reach}(U))$.

Proof. Given, $\epsilon > 0$, let η^s be a scheduler such that $\sup_{\eta \in \mathsf{Dist}(P)} \Pr^{\eta}(\mathrm{reach}(U)) - \Pr^{\eta^s}(\mathrm{reach}(U)) < \epsilon/2$. We denote the set of paths that reach some element in U before the N-th step as $\mathrm{reach}_N(U)$. Let N^* be such that $\Pr^{\eta^s}(\mathrm{reach}(U)) - \Pr^{\eta^s}(\mathrm{reach}_{N^*}(U)) < \epsilon$. The set reach_{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 Lemma 4.4, we know that there exists a deterministic scheduler η^d yielding the supremum probability for reach_{N^*}. Let Θ_i^d, Υ_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, Υ_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, Υ_i^m and $\mathcal{I}^m$ is N^*-Markovian, and it holds $\sup_{\eta \in \mathsf{Dist}(P)} \Pr^{\eta^m}(\mathrm{reach}(U)) - \Pr^{\eta^s}(\mathrm{reach}(U)) < \epsilon$. □

The statement of Theorem 4.7 is false in case strongly distributed schedulers are considered: the example in Fig. 4 is also a counterexample for such a statement. Theorem 4.7 can also be contrasted with the fact that, given a fixed amount of memory, nondeterministic schedulers are needed.

The probabilistic model checking problem has been proven to be undecidable in case the schedulers are restricted to be distributed [10]. Theorem 4.7 shows that the problem is still undecidable 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 an accurate 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 4.8 *Given $\epsilon > 0$, there is no algorithm computing N such that* $\sup_{\eta \in \mathsf{Dist}(P)} \Pr^{\eta}(\mathrm{reach}(U)) - \sup_{\eta \in \mathsf{DetLFinMem}_N(P)} \Pr^{\eta}(\mathrm{reach}(U)) < \epsilon$.

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 $\sup_{\eta \in \mathsf{Dist}(P)} \Pr^{\eta}(\mathrm{reach}(U)) - r < \epsilon$. Such algorithm simply computes N and then performs an exhaustive search on $\mathsf{DetLFinMem}_N(P)$ (note that $\mathsf{DetLFinMem}_N(P)$ is a finite set). However, the existence of such algorithm contradicts Theorem 1 in [10]. □

Since Theorem 1 in [10] 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 shown by the following theorem.

Theorem 4.9 *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 $\sup_{\eta \in \mathcal{S}} \Pr^{\eta}(\text{reach}(U))$ is NP-hard.

Proof. We reduce the 3SAT problem to the supremum reachability problem. The following reduction was suggested by Peter Niebert [15]. Let $c_1 \wedge c_2 \wedge \cdots \wedge c_m$ be an instance of the 3SAT problem where each c_i is a clause of the form $l_i^1 \vee l_i^2 \vee l_i^3$ and each l_i^j is a literal (it is either a variable v_k or the negation $\neg v_k$). We construct two atoms C and V. Intuitively, C chooses a clause and a literal in the clause, and V chooses a variable and a value for this variable. Atoms C and V do not synchronize at all. The set of states of C is

$$\{\text{init}, c_1, \cdots, c_m, l_1^1, \ldots, l_1^3, \cdots, l_m^1, \ldots, l_m^3\} \ .$$

The set of states of V is

$$\{\text{init}, (v_1, \text{Undef}), \cdots, (v_n, \text{Undef}), (v_1, \text{True}), \cdots, (v_n, \text{True}),$$
$$(v_1, \text{False}), \cdots, (v_n, \text{False})\} \ .$$

In the initial state, atom C has enabled only one transition. Such a transition probabilistically chooses one of the clauses, and it outputs a label a not visible to V. We write this generative transition as

$$c = \frac{1}{m}(a, c_1) + \cdots + \frac{1}{m}(a, c_m) \ .$$

In addition in each of the states c_i there are transitions h_i^1, h_i^2, h_i^3 leading to the respective literals:

$$h_i^j = 1(a, l_i^j) \ .$$

The generative structure of C is thus given by $G_C(\text{init}) = \{c\}$, $G_C(c_i) = \{h_i^1, h_i^2, h_i^3\}$ and $G_C(s) = \{\}$ for all other s. Note that a scheduler for C defines a set of literals $l_1^{j_1}, \cdots, l_m^{j_m}$ (one for each clause c_j). Atom V chooses a variable probabilistically, and then nondeterministically assigns a value to this variable. We write the transition that chooses the variable as

$$v = \frac{1}{n}(b, (v_1, \text{Undef})) + \cdots + \frac{1}{n}(b, (v_n, \text{Undef})) \ .$$

The generative structure of V assigns this transition to the initial state: $G_V(\text{init}) = \{v\}$. For each state of the form (v_k, Undef) we have two transitions $\text{False}_k = 1(b, (v_k, \text{False}))$ and $\text{True}_k = 1(b, (v_k, \text{True}))$. Then, $G_V(v_k, \text{Undef}) = \{\text{False}_k, \text{True}_k\}$. Each output scheduler for V can be seen as a valuation for the set of variables. The set of states U is the set in which the value assigned to variable in V does not disagree with the literal chosen by C, that is,

$$U = \{(l_r^j, (v_k, \text{False})) \mid l_r^j \neq v_k\} \cup \{(l_r^j, (v_k, \text{True})) \mid l_r^j \neq \neg v_k\} \ .$$

Therefore, $\sup_{\eta} \Pr^{\eta}(\text{reach}(U)) = 1$ iff there exist a set of literals $l_1^{j_1}, \cdots, l_m^{j_m}$ and a valuation such that all the literals hold in the valuation (in other words, iff the

formula is satisfiable). Note that the number of states of the system comprising atoms C and V is polynomial in the number n of variables. Moreover, the system has no cycles, and so Markovian schedulers attain the supremum probability. Then, the problem is NP-hard. □

5   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 [8], 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 [9] 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 [6] is the concept of interleaving scheduler. In contrast, in the framework presented in [6] 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 [5] 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 each pair of components and there is no information completely hidden. In [11], 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.

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 synchro-

nized 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 [8] is proved in a setting in which the lack of information is not necessarily a consequence of the existence of several components.

Acknowledgement

References

[1] 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.

[2] K. Chatterjee, R. Majumdar, and M. Jurdzinski. On Nash equilibria in stochastic games. In *CSL '04*, pages 26–40, 2004.

[3] 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.

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

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] D. A. Martin. Borel determinacy. *Annals of Mathematics*, 102(2):363–371, 1975.

[15] Peter Niebert. Personal communication.

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

[17] Noel Vaillant. `probability.net`. Probability tutorials on line. Tutorial 2.

[18] 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.

Appendix

The following table summarizes some of the results in the paper. For each subclass, the table indicates whether or not the subclass attain the same optimal probability as the whole class. For example, the $\sqrt{}$ corresponding to "Distributed", "Infinite Memory" and "Deterministic" indicates that deterministic distributed schedulers are as powerful as distributed schedulers.

		Deterministic	Nondeterministic
Distributed	N-Markovian	$\times$	$\times$
	Finite memory	$\times/\sqrt{}^{*}$	$\times/\sqrt{}^{*}$
	Infinite memory	$\sqrt{}$	$\sqrt{}^{\dagger}$
Strongly distributed	N-Markovian	$\times$	$\times$
	Finite memory	$\times$	$\times$
	Infinite memory	$\times$	$\sqrt{}^{\dagger}$

*: $\sqrt{}$ for reachability properties, $\times$ for general properties.

†: trivially true. This subclass is the class of all distributed (strongly distributed, resp.) schedulers.