SOBRE LA VERIFICACIÓN AUTOMÁTICA DE AUTÓMATAS PROBABILISTAS DISTRIBUIDOS CON INFORMACIÓN PARCIAL

POR SERGIO GIRO

Presentado ante la Facultad de Astronomía, Matemática y Física como parte de los requerimientos para la obtención del grado de Doctor en Ciencias de la Computación de la

UNIVERSIDAD NACIONAL DE CÓRDOBA

Marzo, 2010

Director: Dr. Pedro R. D'Argenio

Sergio Giro: *On the automatic verification of Distributed Probabilistic Automata with Partial Information,* Computer Science, PhD, © March 2010

To my family, according to the (not certainly broad) sense of family: "*Ohana* means family.
Family means nobody gets left behind, or forgotten."

— Lilo & Stitch

RESUMEN

En esta tesis desarrollamos algoritmos y técnicas de análisis basadas en model checking para analizar la corrección de sistemas distribuidos con características aleatorias y no-deterministas.

Una contribución importante es la demostración de que no existe un algoritmo que resuelva el problema de verificación de forma totalmente automática, esto es: no existe un algoritmo tal que, dados cualesquiera sistema distribuido y propiedad, el algoritmo decide si el sistema cumple con la propidad.

A pesar de este resultado, presentamos algoritmos que, si bien no pueden determinar la corrección para todos los sistemas y propiedades, sirven para detectar que ciertos sistemas son correctos o incorrectos.

Uno de los impedimentos más frecuentes a la hora de verificar PDMs es el *problema de la explosión de estado*. Este problema, bien conocido y atacado en model checking, se agrava en el ámbito de model checking cuantitativo (i.e. model checking de propiedades cuantificadas probabilísticamente). Los algoritmos de model checking cuantitativo, además de almacenar los estados en memoria, deben resolver un sistema de optimización lineal donde cada variable está asociada a un estado, y cada desigualdad a una transición probabilística. Existen trabajos previos que, con el fin de atacar este problema, presentan adaptaciones de las técnicas de reducción orden parcial para model checking cualitativo al caso cuantitativo.

En esta tesis presentamos una nueva adaptación de la técnica de reducción de orden parcial. Nuestra adaptación aprovecha el hecho de que las componentes de un sistema concurrente tienen acceso limitado a la información sobre el estado global del sistema. Usando nuestra técnica se obtienen reducciones más efectivas que las existentes para el caso cuantitativo.

Concluimos la tesis con casos de estudio que muestran las mejoras de nuestros algoritmos y nuestra técnica de orden parcial con respecto a sus contrapartes para PDMs.

ABSTRACT

We study concurrent systems involving probabilities and non-determinism. Specifically, we focus on the automatic verification of *distributed* systems, in which each component can access only a limited portion of the information in the system.

Although model checking algorithms for Markov decision processes (MDPs) can be applied to distributed systems, such algorithms assume that all components in the system have access to all the information. As a consequence, some correct distributed systems are deemed incorrect when we analyse them using algorithms for MDPs.

In this thesis, we present model checking algorithms for distributed systems involving probabilities and nondeterminism.

A relevant contribution is the result that there exists no algorithm to solve the model checking problem in a completely automated fashion. That is,

there exist no algorithm so that, for all distributed systems and properties, the algorithm decides whether the property holds or not.

Despite of this result, we present two algorithms: one of these algorithms is able to detect that some systems are correct, while the other detects incorrect ones.

In addition, we present a new adaptation of the POR technique. Our adaptation profits from the fact that a component in a concurrent system has limited access to the information stored by other components. Our technique provides more effective reductions than those obtained using existing techniques for MDPs.

We conclude the thesis by presenting case studies in which our algorithms yield better results when compared to their counterparts for MDPs.

PUBLICATIONS

Several of the results in this thesis appeared in the following publications:

Sergio Giro and Pedro R. D'Argenio. Quantitative model checking revisited: neither Decidable nor Approximable. In J.-F. Raskin and P.S. Thiagarajan, editors, *FORMATS*, volume 4763 of *Lecture Notes in Computer Science*, pages 179–194. Springer, 2007.

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.

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.

Sergio Giro and Pedro R. D'Argenio. On the verification of probabilistic I/O automata with unspecified rates. In Sung Y. Shin and Sascha Ossowski, editors, *SAC*, pages 582–586. ACM, 2009.

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.

Lasciate ogne speranza, voi ch'intrate.
Dante, *Divina Commedia*, Inferno, Canto Terzo

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Here, we introduce the mathematical notation we use throughout the thesis. In particular, the notation here concerns usual mathematical concepts such as sequences, equivalences relations, etc.

This thesis contains also a glossary in page 211. Such a glossary is intended to be a reminder for the symbols and notations specific to the thesis. *Glossary*

- π_j denotes the j-th projection: $\pi_j(a_1, \cdots, a_j, \cdots, a_n) = a_j$.
- The cardinality of a set S is denoted by $\left| S \right|$.
- The complement of S is denoted by $\overline{S}$.
- $\{a_i\}_{i=M}^{N}$ (with possibly $N = \infty$). Sequence $a_M, \cdots, a_N$. If the index and the bound are obvious we may omit them. In some other cases, the index is useful to avoid confusion. For instance, $\{a_j^i\}_{i=1}^{n}$ denotes the sequence $a_j^1, \cdots, a_j^n$.
- $\{a_i\}_{i\in S}$, sequence/set indexed by elements in S. For instance, if $S = \{M, \cdots, N\}$, then $\{a_i\}_{i\in S} = \{a_i\}_{i=M}^{N}$. If $a_i \in A$ for all i, then $\{a_i\}_{i\in S}$ can be seen as the set $\{f \mid f : S \to A\}$ comprising all functions from S to A .
- $s \mathcal{R} s'$, the pair (s, s') is in the relation $\mathcal{R}$.
- $s \not\mathcal{R} s'$, the pair (s, s') is not in the relation $\mathcal{R}$.
- $S/_{\mathcal{R}}$, set whose elements are the equivalence classes of $\mathcal{R}$. $\mathcal{R}$ is assumed to be an equivalence relation on S .
- A σ-algebra $\mathcal{F}$ on S is a set $\mathcal{F} \subseteq \mathcal{P}(S)$ such that: *(1)* $S \in \mathcal{F}$ and *(2)* $A \in \mathcal{F} \implies S \setminus A \in \mathcal{F}$ and *(3)* $\{A_n\}_{n=1}^{\infty} \subseteq \mathcal{F} \implies \bigcup_{n=1}^{\infty} A_i \in \mathcal{F}$.
- Given a set S and a σ-algebra $\mathcal{F}$ on S, a probability distribution on S is a function $p : \mathcal{F} \to \mathbb{R}_{\geqslant 0}$ such that $p(S) = 1$ and, if $A_n \in \mathcal{F}$ for all n, then $p\left(\biguplus_{n=1}^{\infty} A_n\right) = \sum_{n=1}^{\infty} p(A_n)$.
- A probability distribution on S is said to be *discrete* if $\mathcal{F} = \mathcal{P}(S)$ and $p(A) = \sum_{x\in A} p(\{x\})$. We denote by $\textsc{prob}(S)$ the set of all discrete probability distributions over the set S. We adopt the following notation: given $p \in \textsc{prob}(\{x_1, \cdots, x_n\})$, we may write p as $p(\{x_1\}){:}x_1 + \cdots + p(\{x_n\}){:}x_n$. Moreover, we may omit the terms in which $p(v_i) = 0$. For instance, if $p \in \textsc{prob}(\{x_1, x_2, x_3, x_4\})$ and $p(\{x_1\}) = 1/2$, $p(\{x_2\}) = 1/4$, $p(\{x_3\}) = 1/4$ and $p(\{x_4\}) = 0$, we write p as $1/2{:}x_1 + 1/4{:}x_2 + 1/4{:}x_3$.

0 INTRODUCTION

Model checking [56, 14] is a well-established technique to verify the correct behaviour of systems. Given a *specification* of the system (that is, a set of properties that the system must comply with) and a model of the behaviour of such system, model checking algorithms can be used to automatically check whether the system complies with its specification or not.

Model checking is a very valuable tool to save time in the development process, since models of the system can be constructed and checked at early stages of development. As a consequence, errors can be corrected before they affect a larger part of the system.

The first works on model checking started in the early 80's [53, 127]. More than twenty years later, we have several available tools implementing model checking techniques, such as SPIN [99], BLAST [24], NuSMV [51] and Java Pathfinder [92].

In practice, model checking tools are useful in diverse fields such as hardware design [26], web services [6], biological systems [36] and security protocols [55], just to name a few. It was thanks to a model checking tool (namely, SMV) that formal methods could spot an error in an IEEE standard for the first time [54].

As a consequence of the success of model checking, by the mid-90's this technique was adapted to deal with systems involving probabilities, thus giving birth to *probabilistic* model checking [25, 18, 21]. There are several tools for model checking of probabilistic systems: notable examples include PRISM [97], LiQuor [50] and MRMC [105]. The use of probabilistic model checking also spans several fields such as robotics [148], power management [112] and communication protocols [72].

In this thesis, we focus on model checking of *distributed* probabilistic systems, in which there are several entities that behave in an independent way. The framework we propose allows us to model multiple types of entities including (but not limited to) nodes in a network, computational processes and user interactions. In general, by "entity" we mean anything that can be described as a transition system (possibly including probabilities).

Existing tools for probabilistic model checking assume that all information is available to all entities. In other words, the system is verified under a *total-information* assumption. In case the entities of the system under consideration do not share all information, an accurate verification must take into account that each entity has access to a limited portion of the information in the system [45, 46, 42, 41]. In other words, the system must be verified under *partial information*. Although algorithms for total information can be applied to the partial information setting, the total-information assumption becomes fictitious. As we shall see, such fictitious availability of information results in fictitious behaviours in which the system does not necessarily complies with its specification. As a consequence, it is possible that a program is deemed incorrect by a total-information algorithm, while all the behaviours that violate the specification according to the algorithm (that is, the coun-

terexamples) are fictitious. The bottom line is that systems that are correct may be deemed incorrect by existing algorithms.

In this thesis, we explore the adaptation of model checking techniques for probabilistic systems to the distributed setting. The ideal adaptation should be an algorithm to automatically check whether or not a system satisfies a certain property *under partial information*. However, we show that the correctness of a probabilistic system under partial information is algorithmically undecidable (in other words, such an ideal algorithm does not exist).

Despite of this negative result, we develop two algorithms: one of these algorithms can be used to show that a system is *correct*. It computes an overestimation of the probability that a system fails: if this overestimation is below the maximum acceptable probability, then the system is correct. However, it might be the case that the overestimation obtained is not acceptable, while the exact value is. Hence, the algorithm is inconclusive if the overestimation exceeds the maximum acceptable value. The other algorithm can be used to show that a system with partial information is *incorrect*. Based on certain criteria, it explores a finite amount of behaviours of the system. Such behaviours can only occur in a partial information setting. In case the algorithm finds a non-acceptable behaviour, then the system is incorrect even under the assumption of partial information.

Model checking is not only about algorithms to decide whether a property holds or not. In fact, it owes part of its success to the techniques to reduce the size of the models. Giving a model, these techniques obtain a smaller one, in such a way that the counterexamples in the original model are preserved in the reduced one. In the setting of probabilistic systems, existing techniques include partial order reduction [12, 57] (POR) and symmetry reduction [113].

We introduce a variation on the POR technique. Our variation profits from the assumption of partial information, thus yielding better results than the existing technique for probabilistic systems under total information.

In the following, we take a deeper look on the motivations for the problems we tackle in this thesis.

0.1 MOTIVATIONS

"Who writes a dissertation should have no hope on its use beyond bookshelf decoration."
Pedro R. D'Argenio. [61], *Stellingen X*

Our work is based on three fundamental concepts: probability, nondeterminism and information. First, we show why probabilities are useful to solve current problems in software development. Next, we show the role of nondeterminism. Finally, we show why it is important to take into account the availability of information.

0.1.1 *The relevance of probabilities*

In the last years, the development of computer products has faced new challenges. We are far from the days in which computer systems were used only to perform calculations. Nowadays, computers are embedded in cars, phones, and music players. This new role of computer devices being embedded in everyday life has added new requirements. In the first place,

computer systems tend to be as small as possible: some years ago, it was difficult to imagine a computer device that fits in the pocket and provides all of the functionality that actual mobile phones do. Such quest for compactness influenced the problem of the cost: making a portable device is easier in case the cost is not a problem, but developments in which costs are not a problem are hard to imagine. Another challenge related to embedded devices is power consumption: tiny devices can carry little power with them. Interoperability is another important aspect: currently, devices must be able to communicate and exchange information in a reliable way. The interoperability of mobile and embedded devices, in turn, introduces new difficulties: a mobile device is always able to leave the network without sending any notification. Clearly, a protocol that has a high performance penalty in this scenario is not suitable for mobile devices.

Many of the solutions to the problems described above are related to probabilistic behaviour. For instance, if the size of a device is affected by the size of one of its components, we can consider a smaller "probabilistically reliable" replacement for this component, such replacement being ensured to work correctly "at least 95% of the times". The same applies with respect to the cost and the power consumption of components. Of course, the behaviour of the device under consideration should be acceptable even if some components do not work correctly every time they are required to.

Probabilities are also a useful tool when designing communication protocols: some fault-tolerant protocols can only be implemented in case randomization is introduced. In such implementations, the participants of the protocol toss coins in order to decide how to continue. In the case of a well-known consensus protocol [49] the validity of the protocol can only be ensured in case the possibility of failure does not depend on the outcome of the coin toss [47]. This is a case in which the availability of information makes a significant difference: we must assume that the outcome is hidden from the environment that causes the failure.

Also in cryptography, anonymity protocols may benefit of the ability of tossing fair coins. In the dining cryptographers protocol [44], anonymity holds only in case we assume that the outcome of the coin remains hidden from every potential adversary.

Those are just some examples in which the use of probabilities makes a significant difference. However, as we shall see, probabilities are not sufficient, and we also need the notion of *nondeterminism* in order to accurately model distributed systems.

0.1.2 *Probabilities and nondeterminism*

Even in case that some of the changes the system exhibits are driven by probabilistic events, some other changes cannot be correctly described using probabilities. This is illustrated in the following example.

Consider we are analyzing a system A that receives numbers n ranging from 1 to 10 from an external source S. The goal of A is to communicate one given message to another system B. System A is acceptable only if the probability that B receives the message is 0.9. The system A uses the (potentially infinite) sequence ξ, provided by S in order to calculate the precise moment to send the message. It is not known how S chooses the numbers in ξ, and

so the goal must be achieved with an acceptable probability (that is, with probability greater than or equal to 0.9) for all ξ. Once A decides to send the message, it is sent through a channel that fails to deliver the message with probability 0.05. For simplicity, we assume that A simply stops after sending the message, without checking if it got lost.

Since the system under consideration involves probabilities, one might be tempted to model the input of a number as a probabilistic choice, in which each of the values is chosen with probability 0.1. However, this model is not a suitable representation of the system. Suppose that A sends the message when it has received the value 2 after the value 1. In case the input is modelled as a probabilistic choice, a verification on this model will indicate that the system achieves its objective with an acceptable probability of 0.95: given that the choice is probabilistic, the subsequence 12 eventually appears in the sequence ξ with probability 1. Then, the system A sends the message, which is lost with probability 0.05. However, in the real system the probability that the message arrives to B might be far less than 0.95: the external source S might send the sequence 9876543219876..., or the sequence 19283746551928..., or only zeroes, or only ones. Moreover, it might be the case that S chooses the number at random but, if the previous output was 1 and the chosen value is 2, then S does not output 2, and selects another number instead. Each of these behaviours of S causes A to delay the communication of the message forever, and so the probability that the goal is achieved under these behaviours is 0. Nonetheless, these behaviours are not considered if the input is modelled as a probabilistic choice, and this is why a fully probabilistic model is not suitable.

We refer to choices that cannot be described using probabilities as *nondeterministic choices*. For these choices, we must consider that any of the options can be taken every time the choice arises. Nondeterministic choices are, thus, analogous to the branches found in the verification of non-probabilistic systems and, in fact, the verification of a system having only nondeterministic choices reduces to the verification of a conventional transition system.

Formalisms with probabilistic and nondeterministic choices consider *probabilistic transitions*. Each transition defines a probability distribution on the set of states. Such distribution models the probability with which each state is reached after the current one in case this transition is executed. In order to model nondeterminism, several of these transitions may be enabled in each state.

In this kind of formalism, the verification problem is to find out the smallest probability that the system behaves correctly, quantifying over all possible resolutions of the nondeterministic choices. As a concrete instance, suppose we are verifying a networking protocol and the nondeterministic choices correspond to routing decisions that are not specified. Moreover, suppose that the correct behaviours are those in which no packages are lost, and that we are able to prove that the smallest probability that a package is lost is 0.05, no matter how the nondeterministic choices are resolved. Then, we can state that "the probability that no package is lost is above the bound 0.95 no matter how the packages are routed".

The resolution of nondeterminism is given by the so called *schedulers* (called also adversaries, policies or strategies —see e.g. [133, 25, 126, 38]). A scheduler is a function mapping paths to transitions (or, in the more general case, paths to distributions on transitions). Metaphorically, we can think that

the scheduler "chooses" to perform one transition out of all transitions enabled in state s. The choice of the scheduler is based on the path that led the system to s. This metaphorical meaning also justifies the term "adversary", since the scheduler can be seen as an evil player trying to make the system behave as bad as possible by choosing the (un)appropriate transitions. The term "policy" is related to planning problems, in which the aim is to find the best plan (or the best *policy*) to accomplish a given goal. The term "strategy" applies both to the planning and the verification settings and, in addition, it is used often in game theory [38, 66]. Sometimes our examples have a "verification" flavour, some other times they may have a "game-theoretic" or "planning" flavour. The essence of the problems is the same: to find out the smallest/greatest probability that an event occurs, taking into account all schedulers/adversaries/policies/strategies.

There are efficient tools implementing algorithms to perform automatic verification [97, 50] on probabilistic and nondeterministic systems. However, the algorithms underlying existing tools do not take into account that the entities in the system might not share all the information. In some cases, this causes the tool to deem some correct systems as incorrect, as explained in the following section.

0.1.3 *The role of information*

If we consider a distributed system as a whole (disregarding the fact that the system comprises several independent entities) some schedulers correspond to unrealistic resolutions of the nondeterminism. As a consequence, it may be the case that overly pessimistic worst-case probabilities are computed during the verification. The following example illustrates the problem: a man tosses a coin and another one tries to guess heads or tails. We study the example from the point of view of T, and so we consider it inconvenient that G guesses the outcome. Figure 0.1 depicts models of these men. Man T, who tosses the coin, has only one transition which represents the toss of the coin: with probability 1/2 he moves to state $heads_T$ and with probability 1/2 he moves to state $tails_T$. Instead, man G has two possible transitions, each one representing his choice: $heads_G$ or $tails_G$.

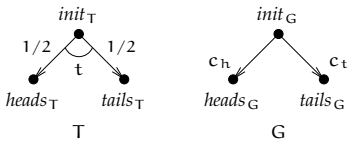


Figure 0.1: T tosses a coin and G has to guess

In the standard "compose-and-schedule" approach, the verification of the system comprising T and G considers a compound model. The way in which the compound model is defined corresponds to the product of labelled transition systems. The compound model comprises all the possible interleavings for the executions of the components. In Fig. 0.2 we depict the compound model for T and G.

Following the compose-and-schedule approach, the verification of the compound model is carried out by considering all of its schedulers. However, a scheduler for the compound model may let G guess the correct answer with

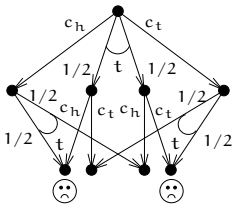


Figure 0.2: Compound model for T and G

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. This behaviour is depicted in Fig. 0.3. Therefore, the supremum probability of guessing obtained by

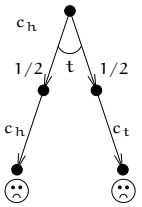


Figure 0.3: A fictitious behaviour in the compound model

quantifying over these almighty schedulers is 1, even if T is a smart player that always hides the outcome until G reveals his choice.

Note that, from the point of view of T, this is a very pessimistic result, since T loses all the times [†]. If we were analysing a strategy to avoid being predicted all the time, and our model checking tool tell us that our choice will be guessed with probability 1, then we would feel very disappointed about our strategy.

Our simple example shows that quantitative model checkers based on the compose-and-schedule approach, though safe, yield an overestimation of the correct value. Since 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 entities share little information with each other, as well as in security protocols, where the possibility of information hiding is a fundamental assumption [40]. The phenomenon we illustrated has been first observed in [133] from the point of view of compositionality and studied in [64, 46, 43] in different settings. Distributed schedulers are also related to the partial-information policies of [63].

In order to avoid considering these unrealistic behaviours, previous works introduce *distributed* schedulers. *Local* schedulers for each entity of the system are defined in the usual way (that is, the choices are based on the complete history of the entity) 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

[†]Moreover, note that he gets *very* sad in case he loses

for G takes the decision without any information about the actual state of T, and so the choice cannot be decided according to the outcome of T.

The nondeterministic choice between heads and tails is *internal* to G, in the sense that G is the only entity involved in this decision. Local schedulers, having access to the internal information of G, are thus suitable for resolving these choices. Asynchronous concurrent systems introduce another kind of nondeterministic choice. Such choices arise every time that several entities have transitions enabled, as only one of these entities must be chosen to perform the next transition. These choices determine the way in which the transitions of different entities are interleaved, and so we refer to them as the *interleaving* nondeterminism. Roughly speaking, previous work on distributed probabilistic systems do not consider interleaving nondeterminism (for a comparison see Sec. 2.4). This nondeterminism is one of the most difficult aspects to capture in a framework for asynchronous distributed systems, as it concerns a global decision that is based on local information. During this thesis, we consider several mechanisms to express that the interleaving is also chosen in a distributed fashion. These mechanisms are based on subtle considerations that are far beyond the scope of an introduction, and so we postpone the discussion on interleaving nondeterminism until Chapter 2.

0.2 A SURVEY OF RELATED WORK

> *"...y que ni el interés ni el miedo, el rencor ni la afición, no les haga torcer del camino de la verdad, cuya madre es la historia, émula del tiempo, depósito de las acciones, testigo de lo pasado, ejemplo y aviso de lo presente, advertencia de lo porvenir."*
>
> *Cervantes*. Don Quijote de la Mancha, *9*

The paper *Probabilistic Algorithms* by Rabin [128] is one of the earliest and most important contributions to the field of probabilistic algorithms. This paper presents two algorithms in which coins are tossed. The problems solved by these algorithms are: *Nearest Neighbours* (a problem in computational geometry) and *Primality Testing* (test whether a number is prime or composite). Moreover, these probabilistic algorithms were fastest than any other algorithm known at the time. In fact, a provably efficient deterministic algorithm for primality testing could not be devised until 2002 [2] (for a survey on primality testing see [70]). Since then, randomization was used in several algorithms as a useful tool to improve performance and/or save resources. An interesting example is the distributed randomized algorithm to ensure mutual exclusion presented in [129]. This algorithm uses a test-and-set shared variable with $O(\log n)$ possible values, while in [31] it is proven that $\Omega(n)$ values are necessary for a non-randomized distributed algorithm. The improvement is more dramatical in the case of distributed consensus: no deterministic algorithm can solve the problem of distributed consensus with faulty processes [74], but a randomized algorithm has been devised [49][†]. Several surveys on randomized algorithms are available: [104, 88, 101, 33].

[†]Such algorithm ensures *probabilistic* termination, that is, the algorithm is ensured to work correctly with probability 1

With respect to the verification of probabilistic systems, different research directions arose for non-probabilistic systems such as process algebra [10], labeled transition systems [108] and model checking [56, 14].

Several *probabilistic* algebra were devised [75, 82, 11, 4]. In [96], performance and process algebra were combined for the first time, giving rise to *stochastic* process algebra. An important class of algebra are Markovian process algebra. Such algebra were introduced to take advantage of the analytical framework provided by continuous time Markov chains. Some examples are IMC [93], TIPP [94], PEPA [76] and EMPA [23]. Other works considered the case of general (not only markovian) distributions [59, 60, 28, 91] (a survey on this topic can be found in [29]). In [42, 39, 43], an algebraic approach is used to model systems with restricted schedulers.

Labeled transition systems can be extended with probabilities in several ways. In [133], the model of *probabilistic automata* is defined. In such model, there are several transitions available at each state. Each transition defines the probabilities of both the label and the next state to be reached. However, most of the results of [133] are restricted to *simple* probabilistic automata, in which a label is assigned to each transition. So, in simple automata, transitions assign probabilities to states, while the label is fixed for each transition. The restriction to simple automata is needed to define a suitable composition operator. An important improvement over simple probabilistic automata is achieved by the probabilistic I/O automata in [147]. Such automata allow transitions assigning probabilities to labels, but behaviours are restricted in such a way that, in every synchronisation, only one of the participating transitions is allowed to assign probabilities to labels. As we shall see in Chapter 1, this restriction has an intuitive explanation in terms of input and output. The probabilistic Input/Output automata in [147] assume exponential distributions for the time that entities delay in a given state. Moreover, the exponential distribution is fixed for each state. This mechanism resembles the Markovian process algebra mentioned before. The Probabilistic I/O Automata model in [46], introduces a token-based mechanism in which the entity that owns the token is the only one able to perform an output. In addition, transitions specify whether or not the owner of the token changes after the transition, and also specify the next entity that receives the token. The introduction of the token eliminates the need for a delay mechanism but, as we shall see in Sec. 2.4, nondeterminism cannot be handled in a satisfactory way in all cases. So, we introduce an *interleaving scheduler* that decides the next entity to execute a transition. The introduction of this interleaving scheduler does not come for free, and we show that the interleaving scheduler needs to be restricted.

Probabilistic automata can be seen as Markov decision processes [126] in that they are Markov chains extended with the ability to choose among several distributions at a given state. In fact, existing algorithms for model checking probabilistic systems use the Bellman equations for Markov decision process [25]. An algorithm to check bisimulation was introduced in [15]. Interestingly, bisimulation can be checked also under the so-called *demonic schedulers* [42].

In addition to the algorithms to determine correctness, model checking also requires techniques to alleviate the state explosion problem. The technique of partial order reduction (POR) [125, 52, 83] was adapted to the probabilistic setting in [12, 57]. Other techniques adapted include symme-

try reduction [113] and abstraction [58, 5, 107]. In Chapter 7 we propose improvements to the technique of POR. As we shall see, the key ingredient to prove that such improvements are correct is the fact that entities do not share all information.

Several other approaches have been devised to deal with partial information. Partially Observable Markov Decision Processes (POMDPs [135, 35, 114]) and *Decentralised POMDPs* [135] have been heavily used in areas such as Artificial Intelligence and Planning. However, they have received little attention in recent research on verification of probabilistic systems, and so we preferred to adhere to the trend of Probabilistic I/O Automata. POMDPs use the notion of *observation*: in addition to the probabilities concerning the next state to be reached, the transition defines probabilistically how the state "looks like" to the observer, by defining a distribution on a set of observations. So, the choices are based on observations of the history of the system, and such observations represent the uncertainty about the actual state affecting decision making. We compare our approach to POMDPs more deeply in Sec. 1.4.

A formalism that considers partial information can also be found in [63]. Given a specified relation, two states of the system are meant to be indistinguishable for the decision maker iff they are related. The choices are then restricted to coincide on indistinguishable histories, and so the equivalence classes of the relation resemble the observations found in POMDPs.

Another formalism that deserves attention is the one in [64]. In this paper, the entities execute in a completely synchronous fashion (that is, each time that the system performs a step, all the entities perform a step). The states of the system are modelled as valuations over a set of variables, and the information available to each entity can be modelled by restricting the variables that it is able to read. The model is primarily intended for compositional reasoning. We preferred the Probabilistic I/O Automata model since it is more suitable for asynchronous systems.

0.3 OUTLINE

Chapter 1 presents the formalism of Interleaved Probabilistic I/O Automata (IPIOA) used in this thesis. It is based on the Switched PIOA of [45]. We present a general approach to partial information by considering arbitrary *projections* for each of the components. A projection is a function restricting the information available: two different executions are distinguished only if the component's projection maps the executions to different observations.

Chapter 2 discusses several restrictions on the interleaving scheduler. This scheduler resolves the nondeterminism concerning the different options to interleave the executions of the components. Given that this scheduler is not related to a particular component, it is not obvious how the information available to each of the components relates to information observable by the interleaving scheduler. The restrictions we propose ensure that the interleaving of two executions of components A, B does not depend on information hidden by another component C.

In Chapter 3 we show that some sequences of schedulers permit the construction of *limit schedulers*. Limits are constructed in such a way that, if all the schedulers in the sequence comply with a given property, then the limits

also do. Since this construction (as well as several other properties associated to limit schedulers) are reused many times along this thesis, we isolated the fundamental results in this chapter.

Chapter 4 compares different sets of schedulers with respect to their *expressive power*. A set of schedulers $\mathbb{S}$ has more expressive power than a set $\mathbb{S}'$ iff the worst-case probability that the system fails under $\mathbb{S}$ is greater than the probability under $\mathbb{S}'$. The results in Chapters 3 and 4 are generalizations of those presented in *On the Expressive Power of Schedulers in Distributed Probabilistic Systems* (Giro, D'Argenio [79]).

Chapter 5 presents several undecidability results concerning the calculation of worst-case probabilities. The maximum probability that a set of states is reached cannot be calculated. Moreover, there is no algorithm to approximate such probability within an error threshold ϵ. Some of the results in this chapter appeared in *Quantitative model checking revisited: neither Decidable nor Approximable* (Giro, D'Argenio [78]), while others appeared in *Undecidability Results for Distributed Probabilistic Systems* (Giro [77]).

Chapter 6 presents two algorithms. One of them calculates an overestimation of the maximum probability that the system fails. The other one exhaustively explores the set of non-randomized distributed Markovian schedulers, in order to look for schedulers in which the probability of a failure is not acceptable. We present a branch-and-bound technique to elide some subsets of schedulers during the exploration.

Chapter 7 introduces a variation on the technique of partial order reduction (POR) for probabilistic systems. The assumption that components can observe only a partial amount of information allows us to improve the technique, thus obtaining smaller systems for which the verification is faster.

Chapter 8 presents a case study concerning a protocol to anonymously serve two clients. One of the algorithms in Chapter 6 is used to analyze whether or not the protocol ensures that the clients are served in a fair fashion. The algorithm and the case study were introduced in *On the verification of probabilistic I/O automata with unspecified rates* (Giro, D'Argenio [80]).

Chapter 9 presents an interpretation of models in the PRISM language into IPIOA. This interpretation allows us to implement our POR technique into PRISM. We also present two examples showing how our implementation performs in practice. The POR technique in Chapter 7 and the examples in this chapter were presented in *Partial Order Reduction for Probabilistic Systems: A Revision for Distributed Schedulers* (Giro, D'Argenio, Ferrer Fioriti [81]).

The conclusion in Chapter 10 explores the thesis in a retrospective view and proposes further research directions.

Part I

PROBABILISTIC SYSTEMS AND SCHEDULERS

1

A FRAMEWORK FOR DISTRIBUTED SYSTEMS WITH PROBABILITIES AND NONDETERMINISM

We present a modelling framework based on the Switched Probabilistic I/O Automata [46]. It is called *Interleaved* Probabilistic I/O Automata (IPIOA), since we eliminate the "switching" semantics in [46] (in which the control of the outputs is switched using a token-based mechanism) and follow an approach closer to usual interleaving semantics. For the sake of simplicity, we split the presentation of our formalism into two sections. Section 1.1 starts with a simple framework, which is similar to the one in [46]. These automata are called *simple* IPIOA. In order to give the semantics of our automata, this section introduces notions of projections and schedulers, which resemble the ones in [46, 64]. Section 1.2 revisits several aspects of the framework, and defines *extended* IPIOA. The simple automata in Sec. 1.1 are a particular case of these ones. We expect that our presentation helps the reader familiar with PIOA, since he will be able to link our extended formalism to the existing one.

In Section 1.3 we generalize the notions of projections and schedulers. These generalizations apply to simple as well as to extended IPIOA, and we find them useful when developing algorithms and techniques for verification.

1.1 SIMPLE INTERLEAVED PROBABILISTIC I/O AUTOMATA

1.1.1 *Input/Output Transitions*

In process algebras such as CSP, processes synchronize on common actions. In order to avoid unrealistic behaviours, it may be useful to specify which entity takes the initiative to perform the action (for instance, which entity decides to send a message through several channels) and which entities simply react to the action initiated (for instance, the channels react by queuing the message). This fact is illustrated using the following example.

EXAMPLE 1.1. Consider a process P that sends data messages and control messages over the channel C. The channel C may fail during the startup. It fails with probability 0.01 and, in this case, the channel appears to be active but the messages are not transmitted. Models for P and C are depicted in Fig. 1.1. The label d (label c, resp.) represents the action in which P tries to send a data message (a control message, resp.) We need to model the fact that P is the entity that chooses between sending a control or a data message, otherwise, the model may be misinterpreted as follows: if the channel fails during startup, then C takes the initiative to execute c. Otherwise, C takes the initiative to execute d. Note that, in this behaviour, control messages are never transmitted. However, if P is not able to see whether or not C has

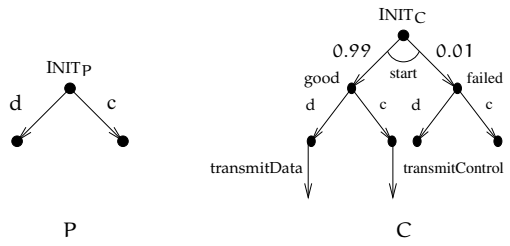


Figure 1.1: Unrealistic choices in synchronizations

failed during startup, one expects the probability that a message is lost to be 0.01, independently of the type of the message.

We use the symbol ! after a label to indicate that the label's entity chooses to perform the action. We say that the entity *generates* the label, and that the label is *output*. In addition, we use the symbol ? after a label to indicate that the entity *reacts* to the action. That is, although the action changes the actual state of this entity, the decision about whether to execute this action is not up to this entity. In this case, the label indicates an *input*. Figure 1.2 shows a modified version of Fig. 1.1.

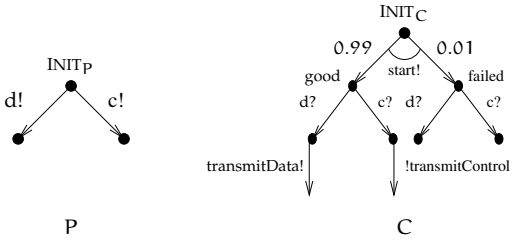


Figure 1.2: Fixing unrealistic choices in synchronizations

If an entity is able to perform several actions, the choice among these actions may be probabilistic. We can modify the previous example in such a way that data messages and control messages are sent with some particular probability, as illustrated in Fig. 1.3. In this figure, action labels c and d

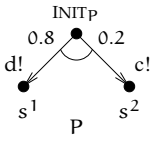


Figure 1.3: Generative transitions with several action labels

.

occur in the same *transition*. This transition is *enabled* in the state INIT$_P$, and it specifies that either d is output and P changes its state to s^1, or c is output and P changes its state to s^2. This is an example of a *generative transition*.

If an entity reacts to an input, the state of the entity input may change probabilistically. Figure 1.4 shows a modified version of the channel in Example 1.1. In this version, an external entity S starts the channel up. The probabilistic choices reflect the fact that the channel may fail during startup. This is an example of a *reactive transition*.

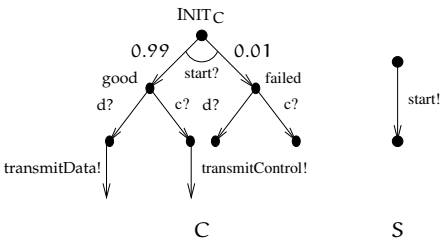


Figure 1.4: A reactive transition that probabilistically chooses between two states

.

The component executing a generative transition chooses both a label a to output and a new state s according to a given distribution. Reactive transitions specify how a component reacts to a given input. Therefore, reactive transitions are simply distributions on states.

DEFINITION 1.1. Given a set ACTLAB of action labels and a set S of states, the set of generative transitions T_G on (S, ACTLAB) is $\text{PROB}(\text{ACTLAB} \times S)$, and the set T_R of reactive transitions is $\text{PROB}(S)$.

PROB(·) is introduced in the basic notation, p. xv.

Generative and reactive *structures* [82, 136] provide the means to specify the transitions enabled in each state. Note that, in the presence of nondeterminism, a state might have several output transitions enabled. In addition, for each state and each label, we allow several input transitions to be enabled. This flexibility allows to specify that the entity may react to an input in several different ways. We call these structures *local*, in contrast to the *global* ones we present later in Sec. 1.2.

DEFINITION 1.2. A local *generative structure* on (S, ACTLAB) is a function $G : S \to \mathcal{P}(T_G)$. A local *reactive structure* on (S, ACTLAB) is a function $R : S \times \text{ACTLAB} \to \mathcal{P}(T_R)$. We restrict to finite structures, that is, $G(s)$ and $R(s, a)$ are finite for all s, a.

Figure 1.5 depicts an example of local generative/reactive structures.

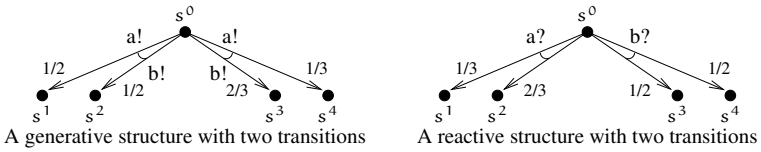


Figure 1.5: Generative and reactive structures

In the example,

$$G(s^0) = \{1/2{:}(a, s^1) + 1/2{:}(b, s^2)\ ,\ 2/3{:}(b, s^3) + 1/3{:}(a, s^4)\}$$

and

The notation $p_1{:}x_1 + \cdots + p_n{:}x_n$ is described in p. xv.

$$R(s^0, a) = \{1/3{:}s^1 + 2/3{:}s^2\}$$

$$R(s^0, b) = \{1/2{:}s^3 + 1/2{:}s^4\}\ .$$

Note that, if a generative transition g is enabled in two different states s^1 and s^2, then the probability that g outputs a and reaches a certain s' is the same in both s^1 and s^2 (namely, it is $g(a, s')$). In the extended version of

probabilistic I/O automata presented later in Sec. 1.2, the probability may change according to the source state. Moreover, in the case of input transitions, the probability depends on the source state and label. Then, the following notation allows us to treat transitions in both versions in a uniform way.

NOTATION 1.1. Given $s_j, s'_j \in S$, $a \in \text{ACTLAB}$, we define

$$\begin{aligned} g(s_j, a, s'_j) &= g(a, s'_j)\,, && \text{if } g \in G(s_j) \\ &= \text{undefined} && \text{otherwise} \end{aligned}$$

and

$$\begin{aligned} r(s_j, a, s'_j) &= r(s'_j)\,, && \text{if } r \in R(s_j, a) \\ &= \text{undefined} && \text{otherwise} \end{aligned}$$

1.1.2 *Modelling symmetric choices*

The communication mechanism we presented before is based on input/output, and thus *asymmetric*: our synchronizations distinguish the entity that decides to output the label from the entities that react to that decision. Sometimes synchronizations are fully symmetric, in the sense that, if there are several common labels enabled, then the decision concerning the label to execute is up to all the entities sharing the label.

Next, we show how symmetric choices can be modelled within our framework.

EXAMPLE 1.2. A boy from Colombia and a girl from Argentina are introduced in an informal meeting. In both countries it is usual to give a little kiss to a girl being introduced. However, they are not sure that in the other one's country such a kiss is usual, and they think that maybe they should shake their hands. On the other hand, shaking hands may seem very formal for this meeting... Note that in this example both the boy and the girl are choosing what to do, and that both need to synchronize to do it. Figure 1.6 illustrates how this choice can be modelled using input/output labels. In

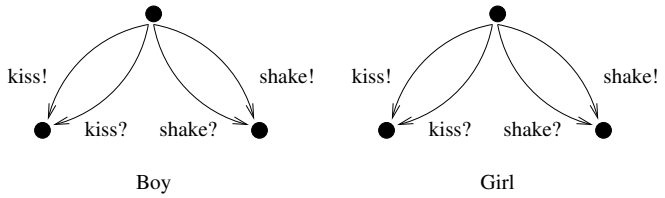


Figure 1.6: Modelling symmetric choices using input/output labels

these cases, we can abstract away the input/output qualifiers and simply consider the fact that they can kiss or shake hands with any probability. For instance, consider the behaviour in which they kiss with probability 1/2. This behaviour corresponds to several behaviours of our model. One of such behaviours is the one in which the boy decides first and he chooses to kiss with probability 1/2. Note that, in this case, the choice of the girl is irrelevant. In the converse case, the girl decides first and the choice of the boy is

irrelevant. Note that, in this example, we also deal with the nondeterminism concerning the one that "decides first". Such nondeterminism can be also resolved probabilistically. In fact, it may be the case that they note each other indecision and the boy says: "OK, let's flip a fair coin. If the coin lands heads, then I decide" [†]. Moreover, suppose that, if the boy decides, then he chooses to kiss with probability 1, while the girl, in case the coin lands tails, chooses to shake hands. The probability that they kiss is the probability that the boy decides *and* he chooses to kiss, that is, $1/2 \cdot 1 = 1/2$.

In the next section, we present a framework of probabilistic automata that uses the input/output mechanism we have described.

1.1.3 *Simple Interleaved Probabilistic I/O Automata*

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 1.3. A simple probabilistic I/O atom is a tuple

$$(S, \text{ACTLAB}, G, R, \text{INIT}) ,$$

where

- S is a finite set of states,
- ACTLAB is a finite set of actions labels,
- G is a generative structure on (S, ACTLAB), G_i
- R is a reactive structure on (S, ACTLAB) and R_i
- $\text{INIT} \in S$ is the initial state.

We require atoms to be *input-enabled*:

$$\forall s \in S, a \in \text{ACTLAB} : R(s, a) \neq \emptyset . \tag{1.1}$$

We write S_i to denote the set of states of an atom A_i and similarly for the other elements of the tuple. In addition, we write T_{G_i} (T_{R_i}, resp.) for the set of generative (reactive, resp.) transitions on (S_i, ACTLAB_i) (Def. 1.1). T_{G_i} T_{R_i}

The input-enabledness requirement is standard, and it is already present in the first works introducing I/O automata [117].

A path of an atom A_i is a sequence $s_i^1.a^1.\cdots.a^{n-1}.s_i^n$ such that $s_i^k \in S_i$, $a^k \in \text{ACTLAB}_i$ and $g(s_i^k, a, s_i^{k+1}) > 0$ for some $g \in G_i(s_i^k)$. The set of paths in A_i is denoted by $\text{APATHS}(A_i)$. *Path in an atom* $\text{APATHS}(A_i)$

An *interleaved probabilistic I/O system* P is a set $\text{ATOMS}(P)$ of probabilistic I/O atoms $A_1, \cdots, A_N$. The set of states of the system is $S_P = \prod_i S_i$, and the initial state of the system is $\text{INIT} = (\text{INIT}_1, \cdots, \text{INIT}_N)$. During this thesis we use $\mathcal{N}$ to denote the number of atoms in the system under consideration. $\mathcal{N}$

The *parallel composition* of two systems P, Q (denoted by $P \parallel Q$) is the system having $\text{ATOMS}(P \parallel Q) = \text{ATOMS}(P) \cup \text{ATOMS}(Q)$. Given two atoms A and B, we $\parallel$

[†] OK, I pushed the example too far. It cannot be such a case.

denote by $A \parallel B$ the parallel composition of the systems P with $\text{ATOMS}(P) = \{A\}$ and Q with $\text{ATOMS}(Q) = \{B\}$.

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 $c = (g_i, a, r_{j_1}, \cdots, r_{j_m})$ (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), $a \in \text{ACTLAB}_i$ is an action label, the r_{j_k} are reactive transitions in the atoms A_{j_k} (the *reactive* atoms) having a in their alphabet, i. e., the set $\{A_i, A_{j_1}, \cdots, A_{j_m}\}$ is equal to $\{A_j \mid a \in \text{ACTLAB}_j\}$. We say that A_i, A_{j_1}, ..., A_{j_m} are the atoms *involved* in the compound transition. We also say that an atom A_k *participates* in a compound transition if $A_k \in \{A_i, A_{j_1}, \cdots, A_{j_m}\}$.

$\text{LABEL}(c)$ The action label a of a compound transition c is indicated by $\text{LABEL}(c)$.

$\text{ACTIVE}(c)$ We denote the active atom A_i by $\text{ACTIVE}(c)$ and the set of reactive atoms $\{A_{j_1}, \cdots, A_{j_m}\}$ by $\text{REACTIVE}(c)$. Note that $\text{REACTIVE}(c) = \{A_i \mid \text{LABEL}(c) \in \text{ACTLAB}_i\} \setminus \{\text{ACTIVE}(c)\}$. A compound transition is an internal transition of an atom A_i if $\text{LABEL}(c) \in \text{ACTLAB}_i$ and for all $j \neq i$ we have $\text{LABEL}(c) \notin \text{ACTLAB}_j$.

$\text{REACTIVE}(c)$

We say that a compound transition $c = (g_i, a, r_{j_1}, \cdots, r_{j_m})$ is *enabled* in state $s = (s_1, \cdots, s_N)$ (denoted by $c \in \text{ENABLED}(s)$) if $g_i \in G_i(s_i)$ and $r_{j_k} \in R_{j_k}(s_{j_k}, a)$ for all A_{j_k} such that $a \in \text{ACTLAB}_{j_k}$.

DEFINITION 1.4. Given a compound transition $c = (g_i, a, r_{j_1}, \cdots, r_{j_m})$ and

$c(s, s')$ states $s = (s_1, \cdots, s_N)$, $s' = (s'_1, \cdots, s'_N)$, the probability $c(s, s')$ of reaching a state s' from a state s using c is

$$\frac{g_i(s, a, s'_i)}{\sum_{s''} g_i(s, a, s''_i)} \cdot \prod_{k=1}^{m} r_{j_k}(s, a, s'_{j_k})$$

if $s_l = s'_l$ for every atom not involved in the transition. Otherwise, the probability is 0.

The factor $\frac{1}{\sum_{s''} g_i(s, a, s''_i)}$ is introduced since $c(s, s')$ is the probability that s' is reached *conditioned to the event that* a *is output*. The following lemma ensures that that $c(s, \cdot)$ can be seen as a discrete probability distribution.

LEMMA 1.1. *For all states* s *and compound transitions* c *such that* $c \in \text{ENABLED}(s)$*:*

$$\sum_{s'} c(s, s') = 1 \quad .$$

Note that, in this proof, we make intensive use of Notation 1.1.

Proof. Let $c = (g_i, a, r_{j_1}, \cdots, r_{j_m})$ and $s = (s_1, \cdots, s_N)$. Then,

$$\begin{aligned}
&\sum_{s'} g_i(s_i, a, \pi_i(s')) \cdot \prod_{k=1}^{m} r_{j_k}(s_{j_k}, a, \pi_{j_k}(s')) \\
&= \sum_{s'_i} \sum_{s'_{j_1}} \cdots \sum_{s'_{j_m}} g_i(s_i, a, s'_i) \cdot \prod_{k=1}^{m} r_{j_k}(s_{j_k}, a, s'_{j_k}) \\
&= \sum_{s'_i} g_i(s_i, a, s'_i) \cdot \Big(\sum_{s'_{j_1}} \cdots \sum_{s'_{j_m}} \prod_{k=1}^{m} r_{j_k}(s_{j_k}, a, s'_{j_k}) \Big) \\
&= \sum_{s'_i} g_i(s_i, a, s'_i) \cdot \Big(\sum_{s'_{j_1}} r_{j_1}(s_{j_1}, a, s'_{j_1}) \cdot \Big(\sum_{s'_{j_2}} \cdots \sum_{s'_{j_m}} \prod_{k=1}^{m} r_{j_k}(s_{j_k}, a, s'_{j_k}) \Big) \Big)
\end{aligned}$$

$$
\begin{aligned}
&= \cdots \\
&= \sum_{s_i'} g_i(s_i, a, s_i') \cdot \Big(\sum_{s_{j_1}'} r_{j_1}(s_{j_1}, a, s_{j_1}') \cdot \Big(\cdots \sum_{s_{j_{m-1}}'} r_{j_{m-1}}(s_{j_{m-1}}, a, s_{j_{m-1}}') \\
&\qquad \cdot \Big(\sum_{s_{j_m}'} r_{j_m}(s_{j_m}, a, s_{j_m}') \Big) \cdots \Big) \Big) \\
&= \sum_{s_i'} g_i(s_i, a, s_i') \cdot \Big(\sum_{s_{j_1}'} r_{j_1}(s_{j_1}, a, s_{j_1}') \cdot \Big(\cdots \sum_{s_{j_{m-1}}'} r_{j_{m-1}}(s_{j_{m-1}}, a, s_{j_{m-1}}') \\
&\qquad \cdot 1 \cdots \Big) \Big) \\
&= \sum_{s_i'} g_i(s_i, a, s_i') \cdot \Big(\sum_{s_{j_1}'} r_{j_1}(s_{j_1}, a, s_{j_1}') \cdot \Big(\cdots \sum_{s_{j_{m-1}}'} r_{j_{m-1}}(s_{j_{m-1}}, a, s_{j_{m-1}}') \cdots \Big) \Big) \\
&= \cdots \\
&= \sum_{s_i'} g_i(s_i, a, s_i')
\end{aligned}
$$

From this calculation, we obtain:

$$
\begin{aligned}
&\sum_{s'} c(s, s') \\
&= \frac{1}{\sum_{s''} g_i(s_i, a, s_i'')} \cdot \sum_{s'} \Big(g_i(s_i, a, \pi_i(s')) \cdot \prod_{k=1}^{m} r_{j_k}(s_{j_k}, a, \pi_{j_k}(s')) \Big) \\
&= \frac{1}{\sum_{s''} g_i(s_i, a, s_i'')} \sum_{s_i'} g_i(s_i, a, s_i') \\
&= 1 \,.
\end{aligned}
$$

□

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. ς

DEFINITION 1.5. For all states s such that $\forall_{A_i} G_i(\pi_i(s)) = \emptyset$, we let

$$\text{ENABLED}(s) = \{\varsigma\} \,.$$

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 $s^1.c^1.s^2.c^2 \cdots c^{n-1}.s^n$ such that

$$c^i \text{ is enabled in } s^i \text{ and } c^i(s^i, s^{i+1}) > 0 \text{ for all } i \,. \tag{1.2}$$

A path can be finite or infinite. Paths of the system P are called *global paths* to disambiguate them from the paths of the atoms. *Global path*

For a finite path σ as before, we define:

- $\sigma(k) = s^k$, $\sigma(k)$
- $\sigma\langle k \rangle = c^k$, $\sigma\langle k \rangle$
- $\text{LAST}(\sigma) = s^n$, $\text{LAST}(\sigma)$
- $\text{LEN}(\sigma) = n$ $\text{LEN}(\sigma)$

$\sigma_{\downarrow k}$

- $\sigma_{\downarrow k} = s^1.c^1 \cdots c^{k-1}.s^k$. If k is negative, $s^1.c^1 \cdots c^{\text{LEN}(\sigma)+k-1}.s^{\text{LEN}(\sigma)+k}$

$\sigma\restriction^k$

- $\sigma\restriction^k = s^k.c^1 \cdots c^{n-1}.s^n$.

$\sigma' \sqsubseteq \sigma$

- $\sigma' \sqsubseteq \sigma$ if $\sigma' = \sigma_{\downarrow k}$ for some k,

$\sigma \cdot \sigma'$

- $\sigma \cdot \sigma' = s^1.c^1 \cdots c^{n-1}.s^n.d^2.t^2 \cdots d^{m-1}.t^m$ if $\sigma' = t^1.d^2.t^2 \cdots d^{m-1}.t^m$ and $t^1 = s^n$

$(\sigma)^\uparrow$

- the *cylinder* generated by σ (denoted by $(\sigma)^\uparrow$) comprises all the infinite paths ω that extend σ, that is, $\sigma \sqsubseteq \omega$. It is called also the set of *extensions* to σ.

EXAMPLE 1.3 (Guess heads or tails). We can use the IPIOA to present the toy example in Subsection 0.1.3 in a formal setting. The atoms corresponding to T and G are depicted in Fig. 1.7.

In general, in the pictures we omit input transitions required by Eqn. (1.1) if they are irrelevant

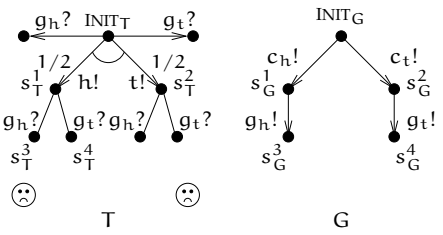


Figure 1.7: T tosses a coin and G has to guess

Actions g_h and g_t communicate the choice of G to T. An intuitive meaning is that G *c*hooses heads or tails using c_h and c_t and then it *g*uesses heads or tails accordingly using g_h and g_t.

We have $\text{ACTLAB}_T = \{h, t, g_h, g_t\}$ and $\text{ACTLAB}_G = \{c_h, c_t, g_h, g_t\}$. Later on, we define our semantics in such a way that

$$h \notin \text{ACTLAB}_G \ \wedge \ t \notin \text{ACTLAB}_G \ ,$$

implies that the outcome of the coin toss is not visible to G.

The model does not specify the order in which transitions of T and G are interleaved. It may be the case that T flips the coin immediately, while G delays for some time before deciding. In this case, atom T has some probability to lose. If G were allowed to see the outcome of the coin, such probability would be 1, since G can choose g_h! if he observes h, and g_t! if he observes t. Since G is *not* allowed to see the outcome of the coin, such probability is the probability of guessing a random value chosen uniformly among two options, that is, 1/2.

1.1.4 *Distributed schedulers*

In this subsection, we explain mechanisms to resolve nondeterminism. Although the mechanism to resolve nondeterministic choices among *transitions* are very similar to ones presented before [46, 64] (in particular, our input and output schedulers are similar to the ones in [46]), the mechanism to resolve choices among atoms was proposed by us [79].

Although we use the metaphor of "games and adversaries" in some explanations, for the formal definitions we prefer the term "scheduler" to "adversary", since "scheduler" is preferred in recent research in distributed systems [46, 45, 42, 64].

In a distributed setting as IPIOA, we need to resolve different kinds of nondeterministic choices. In the first place, it might be the case that several atoms have transitions enabled. In addition, each atom might have several output transitions enabled. And there is a third kind: it may be the case that several reactive transitions are enabled for the same label in the same state. These kinds of nondeterminism are resolved by three kinds of schedulers: the interleaving, output and input schedulers, respectively.

We start by explaining *output* schedulers. For each atom A_i, there is an output scheduler Θ_i. Such a scheduler chooses one of the enabled generative transitions in A_i. More generally, the scheduler may choose a probability distribution on the enabled generative transitions. We can see this scheduler as an adversary that tosses a (possibly biased) coin to decide which generative transition to pick up. The choice of the transition (or the probability distribution) must depend solely on the information available to the output scheduler, Given a global path σ, we model the available information as the local path traversed by A_i during the execution of σ. The function $[\![\cdot]\!]_i : \textsc{paths}(P) \to \textsc{apaths}(A_i)$ strips a global path to obtain a path of A_i.

Definition 1.6. For all atoms A_i, the function $[\![\cdot]\!]_i$ is defined inductively as follows:

- $[\![(\textsc{init}_1, \cdots, \textsc{init}_N)]\!]_i = \textsc{init}_i$ and
- $[\![\sigma.c.(s_1, \cdots, s_N)]\!]_i = [\![\sigma]\!]_i.\textsc{label}(c).s_i$ if $\textsc{label}(c) \in \textsc{actlab}_i$ and
- $[\![\sigma.c.(s_1, \cdots, s_N)]\!]_i = [\![\sigma]\!]_i$ if $\textsc{label}(c) \notin \textsc{actlab}_i$.

An output scheduler Θ_i for atom A_i is then a function

$$\Theta_i : \textsc{apaths}(A_i) \to \textsc{prob}(T_{G_i}) .$$

We restrict the schedulers so that they can only choose enabled transitions, and so we require

$$\Theta_i([\![\sigma]\!]_i)(g_i) > 0 \implies g_i \in G_i(\pi_i(\textsc{last}(\sigma))) \tag{1.3}$$

for all σ such that $\left| G_i(\pi_i(\textsc{last}(\sigma))) \right| > 0$, that is, for all σ in which A_i has enabled transitions. Because of the way in which we give semantics to schedulers, the value of Θ_i is irrelevant in case A_i has no enabled transitions.

The *input* scheduler chooses a reactive transition for each state s and action label a. Following the same argument as for output schedulers, an input scheduler Υ_i is a function

$$\Upsilon_i : \textsc{apaths}(A_i) \times \textsc{actlab}_i \to \textsc{prob}(T_{R_i})$$

such that

$$\Upsilon_i([\![\sigma]\!]_i, a)(r_i) > 0 \implies r_i \in R_i(\pi_i(\textsc{last}(\sigma)), a) .$$

Notice that $\left| R_i(\pi_i(\textsc{last}(\sigma)), a) \right| > 0$ by the input-enabledness condition (1.1).

We still need to resolve the nondeterministic choice concerning the next atom to perform an output. We use an *interleaving scheduler* to resolve such nondeterminism. Note that, so far, we were using $\textsc{apaths}(A_i)$ as the argument to schedulers. However, the interleaving scheduler is not related to a particular atom A_i. In our first attempt, we take a permissive approach, and our definition allows the interleaving scheduler to see the global path (later on, in Sec. 2.1, we introduce restrictions on the interleaving scheduler). An interleaving scheduler is thus a function:

$$\mathcal{I} : \textsc{paths}(P) \to \textsc{prob}(\{A_1, \cdots, A_N\}) \tag{1.4}$$

such that

$$\mathcal{I}(\sigma)(A_i) > 0 \implies G_i(\pi_i(\textsc{last}(\sigma))) \neq \emptyset\,. \tag{1.5}$$

The last restriction ensures that the atoms chosen by the interleaving scheduler are able to generate a transition.

A composition of interleaving, output and input schedulers forms a scheduler for the whole system, as formally defined below.

Definition 1.7. A distributed scheduler is a tuple

$$(\mathcal{I}, \{\Theta_i\}_{i=1}^N, \{\Upsilon_i\}_{i=1}^N)$$

$\mathcal{I}$ is an interleaving scheduler, Θ_i is an output scheduler and Υ_i is an input scheduler for each $A_i \in \textsc{atoms}(P)$.

$\textsc{dist}_P$ Given a system P, we denote by $\textsc{dist}_P$ the set of all distributed schedulers for P.

An important subset of schedulers is that of non-randomized schedulers. Intuitively, these schedulers correspond to adversaries that, when facing a nondeterministic choice, pick one of the options instead of selecting one of them at random.

Non-randomized schedulers

Definition 1.8. We say that a scheduler is *non-randomized* iff $\mathcal{I}(\sigma)(A_i) > 0 \implies \mathcal{I}(\sigma)(A_i) = 1$, $\Theta_i(\sigma_i)(g_i) > 0 \implies \Theta(\sigma_i)(g_i) = 1$ and $\Upsilon_i(\sigma_i, a)(r_i) > 0 \implies \Upsilon(\sigma_i, a)(r_i) = 1$.

The set of *non-randomized* distributed schedulers is denoted by $\textsc{nrdist}(P)$.

Each scheduler defines a probability measure on the set of infinite paths. It does so by defining the probability that a compound transition c occurs, given that the global finite path σ has occurred. This probability is denoted by $\eta(\sigma)(c)$. After the intuitive explanations, we prove that the function defined below is a discrete probability distribution (Lemma 1.2).

Definition 1.9. Let $\mathcal{C}$ be the set of all compound transitions for system P. For all $\eta \in \textsc{dist}_P$, $\sigma \in \textsc{paths}(P)$, the function $\eta(\sigma)(\cdot) \colon \mathcal{C} \to [0,1]$ is defined as:

$$\begin{aligned}\eta(\sigma)(g_i, a, r_{j_1}, \cdots, r_{j_m}) = \mathcal{I}(\sigma)(A_i) \cdot \Theta_i(\llbracket\sigma\rrbracket_i)(g_i) \\ \cdot \sum_{s_i} g_i(\pi_i(\textsc{last}(\sigma)), a, s_i) \\ \cdot \prod_{k=1}^{m} \Upsilon_{j_k}(\llbracket\sigma\rrbracket_{j_k}, a)(r_{j_k})\end{aligned}$$

if $\left| G_i(\pi_i(\textsc{last}(\sigma))) \right| > 0$ for some A_i. Otherwise, $\eta(\sigma)(\varsigma) = 1$.

Intuitively, the event "$(g_i, a, r_{j_1}, \cdots, r_{j_m})$ occurs" is the intersection of the events:

- the interleaving scheduler $\mathcal{I}$ chooses A_i
- Θ_i chooses g_i
- g_i outputs a
- each of the atoms A_{j_k} chooses r_{j_k}.

We assume all these events to be independent, and so the probability assigned to the compound transition $(g_i, a, r_{j_1}, \cdots, r_{j_m})$ is the product of the events' probabilities.

Recall that, in the definition of output schedulers, we did not impose the restriction

$$\Theta_i([\![\sigma]\!]_i)(g_i) > 0 \implies g_i \in G_i(\pi_i(\text{LAST}(\sigma))) \,.$$

for the paths σ such that $\left| G_i(\pi_i(\text{LAST}(\sigma))) \right| = 0$, on the basis that, for such σ, the value of $\Theta_i([\![\sigma]\!]_i)$ would be irrelevant in our semantics. For all transitions $g_i \in T_{G_i}$ note that, regardless of the value $\Theta([\![\sigma]\!]_i)$, we have $\eta(\sigma)(g_i, \cdots) = 0$, since $\mathcal{I}(\sigma)(A_i) = 0$ by (1.4).

Similarly as we did for Def. 1.4, we show that $\eta(\sigma)(\cdot)$ can be seen as a discrete probability distribution on the set $\{c \mid c \in \text{ENABLED}(\text{LAST}(\sigma))\}$.

LEMMA 1.2. *For all distributed schedulers η, paths σ, compound transitions c, we have*

$$\sum_{c \in \text{ENABLED}(\text{LAST}(\sigma))} \eta(\sigma)(c) = 1 \,.$$

Proof. Let $s = (s_1, \cdots, s_{\mathcal{N}}) = \text{LAST}(\sigma)$.

$$\begin{aligned}
&\sum_{(g_i, a, r_{j_1}, \cdots, r_{j_m}) \in \text{ENABLED}(\text{LAST}(\sigma))} \eta(\sigma)((g_i, a, r_{j_1}, \cdots, r_{j_m})) \\
&= \sum_{A_i \in \text{ATOMS}(P)} \sum_{g_i \in G_i(s_i)} \sum_{a \in \text{ACTLAB}_i} \sum_{r_{j_1} \in R_{j_1}(s_{j_1}, a)} \cdots \sum_{r_{j_m} \in R_{j_m}(s_{j_m}, a)} \\
&\qquad \mathcal{I}(\sigma)(A_i) \cdot \Theta_i([\![\sigma]\!]_i)(g_i) \cdot \Big(\sum_s g_i(\text{LAST}(\sigma), a, s) \Big) \cdot \prod_{k=1}^{m} \Upsilon_{j_k}([\![\sigma]\!]_i)(r_{j_k}) \\
&= \sum_{A_i} \mathcal{I}(\sigma)(A_i) \sum_{g_i} \sum_{a} \sum_{r_{j_1}} \cdots \sum_{r_{j_m}} \Theta_i([\![\sigma]\!]_i)(g_i) \\
&\qquad\qquad \cdot \sum_s g_i(\text{LAST}(\sigma), a, s) \cdot \prod_{k=1}^{m} \Upsilon_{j_k}([\![\sigma]\!]_i)(r_{j_k}) \\
&= \sum_{A_i} \mathcal{I}(\sigma)(A_i) \sum_{g_i} \Theta_i([\![\sigma]\!]_i)(g_i) \cdot \Big(\sum_a \big(\sum_s g_i(\text{LAST}(\sigma), a, s) \big) \\
&\qquad\qquad \cdot \Big(\sum_{r_{j_1}} \cdots \sum_{r_{j_m}} \prod_{k=1}^{m} \Upsilon_{j_k}([\![\sigma]\!]_i)(r_{j_k}) \Big) \Big) \\
&= \sum_{A_i} \mathcal{I}(\sigma)(A_i) \sum_{g_i} \Theta_i([\![\sigma]\!]_i)(g_i) \cdot \Big(\sum_a \big(\sum_s g_i(\text{LAST}(\sigma), a, s) \big) \\
&\qquad\qquad \cdot \Big(\sum_{r_{j_1}} \Upsilon_{j_1}([\![\sigma]\!]_i)(r_{j_1}) \Big(\sum_{r_{j_2}} \cdots \sum_{r_{j_m}} \prod_{k=2}^{m} \Upsilon_{j_k}([\![\sigma]\!]_i)(s_{j_k}) \Big) \Big) \Big)
\end{aligned}$$

$$
\begin{aligned}
&= \cdots \\
&= \sum_{A_i} \mathcal{I}(\sigma)(A_i) \sum_{g_i} \Theta_i([\![\sigma]\!]_i)(g_i) \cdot \Big(\sum_a (\sum_s g_i(\textsc{last}(\sigma), a, s)) \\
&\qquad \cdot \Big(\sum_{r_{j_1}} \Upsilon_{j_1}([\![\sigma]\!]_i)(r_{j_1}) \\
&\qquad \cdot \Big(\cdots \sum_{r_{j_{m-1}}} \Upsilon_{j_{m-1}}([\![\sigma]\!]_i)(r_{j_{m-1}}) \\
&\qquad \cdot \Big(\sum_{r_{j_m}} \Upsilon_{j_m}([\![\sigma]\!]_i)(s'_{j_m}) \Big) \cdots \Big) \Big) \Big) \\
&\quad \sum_{A_i} \mathcal{I}(\sigma)(A_i) \sum_{g_i} \Theta_i([\![\sigma]\!]_i)(g_i) \cdot \Big(\sum_a (\sum_s g_i(\textsc{last}(\sigma), a, s)) \\
&\qquad \cdot \Big(\sum_{r_{j_1}} \Upsilon_{j_1}([\![\sigma]\!]_i)(r_{j_1}) \Big(\cdots \sum_{r_{j_{m-1}}} \Upsilon_{j_{m-1}}([\![\sigma]\!]_i)(r_{j_{m-1}}) \\
&\qquad \cdot 1 \cdots \Big) \Big) \Big) \\
&= \cdots \\
&= \sum_{A_i} \mathcal{I}(\sigma)(A_i) \sum_{g_i} \Theta_i([\![\sigma]\!]_i)(g_i) \cdot \Big(\sum_a (\sum_s g_i(\textsc{last}(\sigma), a, s)) \cdot 1 \Big) \\
&= \{ \textstyle\sum_a \sum_s g(a, s) = 1 \text{ (definition of generative transition)} \} \\
&\quad \sum_{A_i} \mathcal{I}(\sigma)(A_i) \sum_{g_i} \Theta_i([\![\sigma]\!]_i)(g_i) \cdot 1 \\
&= \sum_{A_i} \mathcal{I}(\sigma)(A_i) \\
&= 1
\end{aligned}
$$

□

The probability distribution in Def. 1.9, induces a probability measure on the set of paths.

Definition 1.10 (Probability of a set of paths). For a cylinder $(\sigma)^\uparrow$, the probability measure $\Pr^\eta$ is inductively defined by:

$$
\begin{aligned}
\Pr^\eta((\textsc{init})^\uparrow) &= 1 \\
\Pr^\eta((\sigma.c.s)^\uparrow) &= \Pr^\eta((\sigma)^\uparrow) \cdot \eta(\sigma)(c) \cdot c(\textsc{last}(\sigma), s)
\end{aligned}
$$

$\Pr^\eta$ uniquely extends to least σ-field containing all cylinders in the standard way (namely, by resorting to the Carathéodory extension theorem [109]).

Reachability set $\textsc{reach}(U)$

Although in the general case we deal with arbitrary measurable sets, for some results we restrict to *reachability sets*. Given a set a set of states U, let $\textsc{reach}(U)$ denote the set of all in infinite paths σ such that $\omega(k) \in U$ for some k.

Example 1.4. Consider again the guess-heads-or-tails example. In Fig. 1.9 we present a graphical representation of the system $P = T \parallel G$. Figure 1.10 depicts a scheduler $\eta \in \textsc{dist}_P$. Given the enabledness restrictions we impose to schedulers (the interleaving scheduler must choose atoms with enabled transitions, etc.) such a scheduler is completely determined by the definitions: $\mathcal{I}(\textsc{init}) = 1{:}T$ and $\Theta_G(\textsc{init}_G) = 1{:}c_h!$. As we can see in the graphical

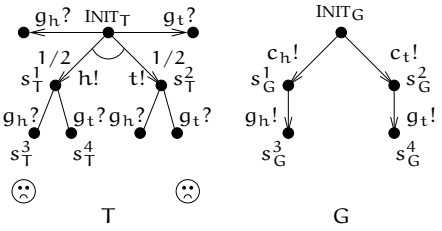


Figure 1.8: T tosses a coin, G guesses heads or tails

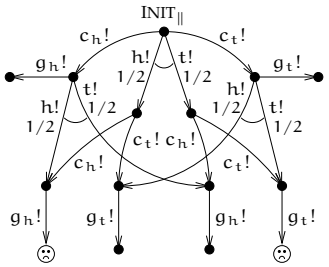


Figure 1.9: System $P = T \parallel G$

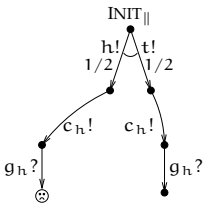


Figure 1.10: Scheduler $\eta \in \text{DIST}_P$

representation, the choice is not changed according to the outcome of the coin toss: c_h is chosen for both s_T^1 and s_T^2.

Note that, by Def. 1.9, we have $\eta(\sigma)(\varsigma) = 1$ for all σ such that $\pi_2(\text{LAST}(\sigma)) = s_G^3$ or $\pi_2(\text{LAST}(\sigma)) = s_G^4$, since in these paths there are no enabled transitions. In particular, note that

$$\forall \sigma, \sigma' : \eta(\sigma) = \eta'(\sigma)$$

holds for all η, η' differing only wrt. the value of $\Theta_G(\text{INIT}_G.c_h!.s_G^1.g_h!.s_G^3)$.

1.2 EXTENDED INTERLEAVED PROBABILISTIC I/O AUTOMATA

In this section, we present several extensions to the definitions in the previous section. We need these extensions to deal with existing formalisms such as the PRISM language [97]. Subsections 1.2.1 and 1.2.2 present extensions to transitions and structures, resp. Subsection 1.2.3 summarizes the link between the systems defined in Sec. 1.1 and the ones in this section.

1.2.1 *Extended transitions*

Suppose that a generative transition g is enabled in two states s and s'. According to Def. 1.1, the probability of generating a and reaching s'' is the same in both s and s' (namely $g(a, s'')$). This definition of transition is not straightforwardly compatible with guarded command languages, where the state of the system is given by a valuation over a set of variables. As an

example, consider a system whose state consists of two variables s and t ranging over $\{0,1\}$. Moreover, suppose that we have a command $[a]s = 0 \to s = 1$, whose intended meaning is "if s is 0, then output a and assign 1 to s". Then, in the state $(s = 0, t = 0)$ this command leads to $(s = 1, t = 0)$ with probability 1, and to $(s = 1, t = 1)$ with probability 0. Conversely, in the state $(s = 0, t = 1)$, it leads to $(s = 1, t = 0)$ with probability 0, and to $(s = 1, t = 1)$ with probability 1. In short, the state reached after the command depends on what the actual state is. Then, we generalize generative transitions so that the probabilities depend on the actual state. With respect to reactive transitions, we extend them so that probabilities depend both in the actual state and the label to which the atom reacts.

DEFINITION 1.11. An (extended) generative transition for atom A_i is a function $g_i : S_i \to \text{PROB}(\text{ACTLAB}_i \times S_i)$. The set comprising all generative transitions for atom A_i is denoted by T_{G_i}. An (extended) reactive transition in atom A_i is a function $r_i : S_i \times \text{ACTLAB}_i \to \text{PROB}(S_i)$. The set comprising all reactive transitions for atom A_i is denoted by T_{R_i}.

Of course, if g_i is not enabled in a given state s, then the value $g_i(s)$ is irrelevant. We could have defined the domain of g_i as the set of states in which g_i is enabled, but this makes no difference and we prefer to keep the definition simple. Note that the definition of T_G clashes with that of the original definition (Def. 1.1). This causes no harm as long as it is clear whether the transitions we are considering are extended or not.

NOTATION 1.2. We write $g_i(s_i, a, s_i')$ for $g_i(s_i)(a, s_i')$ and $r_i(s_i, a, s_i')$ for $r_i(s_i, a)(s_i')$.

Together with Notation 1.1, this notation allows us to abstract whether the transitions are extended or not.

1.2.2 *Global enabledness conditions*

We have defined the generative structure of an atom A_i as a function $G_i : S_i \to T_{G_i}$. Then, it suffices to look to the *local* state in order to see if a transition is enabled. This is possible since we require input-enabledness (Eqn. (1.1)): otherwise, it might be the case that g_i is enabled in s_i, $g_i(a, s_i') > 0$ for some a, s' and a is in the alphabet of an atom A_j such that $R_j(s_j, a) = \emptyset$. In other words, g_i is enabled, but it generates an action a while A_j blocks this action.

As a result, when interpreting languages without input-enabledness into IPIOA, our definition of generative structures happens to be inappropriate. In general, if $g_i(a, s_i') > 0$, then we would like g_i to be enabled only if $R_j(s_j, a) \neq \emptyset$ for all A_j such that $a \in \text{ACTLAB}_j$, $A_j \neq A_i$. The following definition helps us to achieve this goal.

Generative structure
Reactive structure

DEFINITION 1.12. A (global) generative structure for atom A_i is a function $G_i : \prod_{i=1}^{\mathcal{N}} S_i \to \mathcal{P}(T_{G_i})$. A (global) reactive structure for atom A_i is a function $R_i : \prod_{i=1}^{\mathcal{N}} S_i \times \text{ACTLAB} \to \mathcal{P}(T_{R_i})$. (For the definition of T_{G_i} and T_{R_i}, see Def. 1.11.) We restrict to finite structures, that is, $G_i(s)$ and $R_i(s, a)$ are finite for all s, a.

In order to see the usefulness of Def. 1.12, suppose that we have an IPIOA such that $g_i(a, s_i') > 0$, $g_i \in G_i(s_i)$ and $R_j(s_j, a) = \emptyset$ for some s_j, a. Then, we can use the local generative structure G_i of atom A_i to define a global generative structure G_i'. This global structure is defined as:

$$G_i'(s) = \{g_i \mid g_i \in G_i(\pi_i(s)) \\ \wedge\ \forall A_j \neq A_i : g_i(a, s_i') > 0 \wedge a \in \text{ACTLAB}_j \implies R_j(\pi_j(s), a) \neq \emptyset\}$$

That is, if $g_i \in G_i(\pi_i(s))$ and no atom blocks an action generated by g_i, then $g_i \in G_i'(s)$.

An important benefit of this encoding is that we do not need to resign the input-enabledness assumption: in fact, if $R_j(s, a) = \emptyset$, then we can define $R_j(s, a)$ arbitrarily, since (by definition of G_i') the transitions in $R_j(s, a)$ will not be executed in s_j.

We finish this subsection by noting that local structures can be seen as a particular case of global ones: in fact, given a global structure G_i' complying with

$$\forall s, t : \pi_i(s) = \pi_i(t) \implies G_i'(s) = G_i'(t)\ , \tag{1.6}$$

we can define a local structure $G_i(\pi_i(s)) = G_i(s)\ (= G_i(t)\ )$. Conversely, given a local structure G_i, we can define a global structure G_i' as $G_i'(s) = G_i(\pi_i(s))$. Similarly, a local reactive structure can be seen as a global structure R_i' complying

$$\forall a : \forall s, t : \pi_i(s) = \pi_i(t) \implies R_i'(s, a) = R_i'(t, a)\ . \tag{1.7}$$

Most of the time we find it useful to abstract whether the atoms have local or global structures, and so we assume the structures to be global. Some other times we want to show that our results are valid specifically in the case of local structures (for instance, undecidability results are stronger if they hold for specific classes of systems). In case the structures are local (Def. 1.2) we say that the system has *local enabledness conditions*. If we are under Def. 1.12, we say that the system has *global enabledness conditions*.

Local enabledness conditions
Global enabledness conditions

1.2.3 *Extended systems*

We define an extended probabilistic I/O atom as a tuple $(S, \text{ACTLAB}, G, R, \text{INIT})$. The only difference with respect to the atoms defined before is that G and R are structures as in Def. 1.12.

An (extended) IPIOA is a set of extended atoms, and the composition of two systems P and Q comprises the atoms of both P and Q. Similarly, all other definitions in Subsection 1.1.3 map straightforwardly to extended systems.

Each atom $A_i = (S, \text{ACTLAB}, G, R, \text{INIT})$ as in Def. 1.3 can be seen as an extended atom $A' = (S, \text{ACTLAB}, G', R', \text{INIT})$ complying with the following properties:

Notation 1.1 results particularly helpful to understand the last two conditions.

- G_i' complies with Eqn. (1.6),
- R_i' complies with Eqn. (1.7),
- $g_i(s_i, a, s_i') = g_i(t_i, a, s_i')$ for all g_i, s_i, a, s_i', t_i and

- $r_i(s_i, a, s_i') = r_i(t_i, b, s_i')$ for all r_i, s_i, a, s_i', b, t_i.

The first two conditions reflect the fact that the structures in A are as in Definition 1.2, while the last two conditions reflect the fact the transitions in A are as in Definition 1.1.

1.3 GENERALIZED PROJECTIONS AND SCHEDULERS

In this section, we generalize the projections and schedulers introduced in Subsection 1.1.4. These generalized versions apply to simple as well as to extended IPIOA.

1.3.1 *Projections*

Definition 1.6 in Subsection 1.1.4 introduces the function $[\![\cdot]\!]_i$. It transforms a global path into an local path of atom A_i. This function is used in order to evaluate the scheduler in a local path instead of a global one: when defining the probability $\eta(\sigma)(c)$ (Def. 1.9), we faced the factor $\Theta_i([\![\sigma]\!]_i)$. Since $\Theta_i : \text{APATHS}(A_i) \to \text{PROB}(\text{T}_{G_i})$, we have $\Theta_i([\![\sigma]\!]_i) = \Theta_i([\![\sigma']\!]_i)$ for all σ, σ' such that $[\![\sigma]\!]_i = [\![\sigma']\!]_i$.

The transformation from global to local paths hides information to the scheduler. In order to illustrate this, we can consider functions other than $[\![\cdot]\!]_i$ and, particularly, two extreme cases:

- If we consider the function $f_i : \text{PATHS}(P) \to \{\text{INIT}_i\}$ defined as $f_i(\sigma) = \text{INIT}_i$ for all σ, then we can define our output schedulers as functions $\Theta_i : \{\text{INIT}_i\} \to \text{PROB}(\text{T}_{G_i})$. In this case, $\Theta_i(f_i(\sigma)) = \Theta_i(f_i(\sigma'))$ for all σ, σ'. Here, Θ_i has no information about what the global path is, and so it is forced to choose the same distribution in all paths (to ease explanation, we assume all transitions to be enabled in all states, thus disregarding restriction (1.3)).
- If we consider the identity function $\text{ID} : \text{PATHS}(P) \to \text{PATHS}(P)$ and the schedulers $\Theta_i : \text{PATHS}(P) \to \text{PROB}(\text{T}_{G_I})$, then $\Theta_i(\text{ID}(\sigma)) = \Theta_i(\sigma)$. Here, it may be the case that Θ_i chooses a different transition for each global path. In other words, Θ_i has perfect information about what the global path is, and then Θ_i can choose according to the global path.

These examples motivate the following general definition.

DEFINITION 1.13. Given a system $P = \|_{i=1}^{N} A_i$, a projection $[\cdot]$ is a family of functions $\{ [\cdot]_i : \text{PATHS}(P) \to O^{[\cdot]_i} \}_i$. The set of all projections for P is denoted by $\text{PROJECTIONS}(P)$. For the particular case of $[\![\cdot]\!]_i$, we have $O^{[\cdot]_i} = \text{APATHS}(A_i)$.

The range of $[\cdot]_i$ is denoted by $\text{LOCALPATHS}_i^{[\cdot]}$.

Usually, we do not care too much for the precise definition of the set $O^{[\cdot]_i}$, as we can often infer it from the definition of $[\cdot]_i$. We have $\text{LOCALPATHS}_i^{[\cdot]} \subseteq O^{[\cdot]_i}$, as that $O^{[\cdot]_i}$ is the *co-domain* of $[\cdot]$, while $\text{LOCALPATHS}_i^{[\cdot]}$ is the *range*. Then, for all $\sigma_i \in \text{LOCALPATHS}_i^{[\cdot]}$ there exists σ such that $[\sigma]_i = \sigma_i$.

$[\cdot]^f$ $[\cdot]^{ID}$

From the two examples above, we can construct two projections $[\cdot]^f$ and $[\cdot]^{ID}$ defined by $[\sigma]_i^f = \text{INIT}_i$ and $[\sigma]_i^{ID} = \sigma$ for all σ.

When introducing the function f_i, we assumed that all transitions were enabled in all states. We need this assumption in order to ensure that $\Theta_i(f_i(\sigma))$

is well defined, in other words, it must comply with the following analogous of (1.3):

$$\Theta_i(f_i(\sigma))(g_i) > 0 \implies g_i \in G_i(\text{LAST}(\sigma)) \,. \tag{1.8}$$

Next, we look for conditions on the projections and the generative transitions so that well-defined schedulers are ensured to exist. In case there are two paths σ, σ' such that $[\sigma]_i = [\sigma']_i = \sigma_i$, we require $G_i(\pi_i(\text{LAST}(\sigma))) = G_i(\pi_i(\text{LAST}(\sigma')))$. To illustrate this, suppose $[\sigma]_i \neq [\sigma']_i$, $G_i(\pi_i(\text{LAST}(\sigma))) = \{g_i\}$ and $G_i(\pi_i(\text{LAST}(\sigma'))) = \{g_i'\}$, with $g_i \neq g_i'$. In this system, no output scheduler for A_i can be defined, since condition (1.8) fails for either σ or σ'. In short, the generative transitions enabled at the end of two indistinguishable paths must coincide: if this restriction does not hold, then the scheduler would not know whether it can choose a certain transition or not, since it might be the case that a transition is enabled in some global path σ, but it is disabled in other global paths having the same projection as σ.

Hence, we require,

$$[\sigma]_i = \left[\sigma'\right]_i \implies G_i(\text{LAST}(\sigma)) = G_i(\text{LAST}(\sigma')) \,. \tag{1.9}$$

Note that we write $G_i(\text{LAST}(\sigma)) = G_i(\text{LAST}(\sigma'))$, instead of $G_i(\pi_i(\text{LAST}(\sigma))) = G_i(\pi_i(\text{LAST}(\sigma')))$, since in the general case we deal with global transition structures (see Def. 1.12). The projection $[\![\cdot]\!]$ complies with this requirement for all simple IPIOA: if $[\![\sigma]\!]_i = [\![\sigma']\!]_i = s_i^1.a^1.\cdots.s_i^k$, then $\pi_i(\text{LAST}(\sigma)) = \pi_i(\text{LAST}(\sigma')) = s_i^k$. Since simple IPIOA have local transition structures (that is, its transitions structures are as in Def. 1.2), we obtain $G_i(\pi_i(\text{LAST}(\sigma))) = G_i(\pi_i(\text{LAST}(\sigma'))) = G_i(s_i^k)$, as desired.

Similarly, we require input schedulers to satisfy:

$$[\sigma]_i = \left[\sigma'\right]_i \implies R_i(\text{LAST}(\sigma), a) = R_i(\text{LAST}(\sigma'), a) \tag{1.10}$$

for all $a \in \text{ACTLAB}_i$.

We postpone the proof that these requirements ensure that a well-defined scheduler exists until we have presented the definition of generalized schedulers (Theorem 1.1).

In addition to requirements (1.9) and (1.10), we assume an additional property. The motivation for this assumption is that the property is very natural, and all the projections we present comply with it (in addition, it is quite tiresome to repeat the property in the hypotheses of all theorems). We assume:

$$\forall \sigma, \sigma' : \left(\exists k : \text{LABEL}(\sigma'\langle k\rangle) \in \text{ACTLAB}_i\right) \implies [\sigma]_i \neq \left[\sigma \cdot \sigma'\right]_i \,. \tag{1.11}$$

This assumption is best explained by showing why it holds for $[\![\cdot]\!]$. Note that $\text{LABEL}(\sigma'\langle k\rangle) \in \text{ACTLAB}_i$ means that A_i participates in the transition after the k-th state in σ'. W. l. o. g. , we consider the least such a k. Let $a = \text{LABEL}(\sigma'\langle k\rangle)$. Then, $[\![\sigma \cdot \sigma']\!]_i = [\![\sigma]\!]_i.a.\sigma_i' \neq [\![\sigma]\!]_i$, where σ_i' is a local path. That is, after $\sigma \cdot \sigma'$ the atom is able to see the label in which it synchronized after the k-th step in σ'. In the general case, the assumption is even weaker since, by requiring $[\sigma]_i \neq [\sigma \cdot \sigma']_i$, we just enforce that, after participating in a transition, the scheduler has a different information than it had previous to the transition. Intuitively, the scheduler notices that "something has happened".

EXAMPLE 1.5. Suppose that a system comprises several components, each of which is modelled by an atom A_i. These components share a common resource, which is modelled as a separate atom A_r. We assume that the model of the resource is completely deterministic: the components perform operations on the resource which univocally determine its next state; moreover, the resource receives inputs from the components, and it does not generate any output. Each component is allowed to see its local state, plus the state of the resource. However, the component is not aware of a change in the state of the resource until it performs an operation on it: in case the operation changes the state of the resource, the state observed is the updated one. For the atoms A_i modelling components, the projection $[\cdot]_i$ capturing the information available to an atom at each point of the execution is:

- $[(\text{INIT}_1, \cdots, \text{INIT}_N)]_i = \text{INIT}_i$
- $[\sigma.c.(s_1, \cdots, s_N)]_i = [\sigma]_i.\text{LABEL}(c).(s_i, s_r)$

  If $\text{LABEL}(c) \in \text{ACTLAB}_i \cap \text{ACTLAB}_r$. Here, s_r is the local state of atom A_r, and $\text{LABEL}(c) \in \text{ACTLAB}_i \cap \text{ACTLAB}_r$ means that c is a compound transition involving both the component and the resource.
- $[\sigma.c.(s_1, \cdots, s_N)]_i = [\sigma]_i.\text{LABEL}(c).s_i$

  If $\text{LABEL}(c) \in \text{ACTLAB}_i$ and $\text{LABEL}(c) \notin \text{ACTLAB}_r$. In this case, the transition involves the component but not the resource, and so the projection gives the same information as $[\![\cdot]\!]$.
- $[\sigma.c.(s_1, \cdots, s_N)]_i = [\sigma]_i$ if $\text{LABEL}(c) \notin \text{ACTLAB}_i$.

  In this case, the atom does not obtain new information from c.

With respect to atom A_r, the projection can be defined arbitrarily: projections capture the information used to resolve the nondeterministic choices, and we assumed that there are no such choices in A_r. For simplicity, let $[\sigma]_r = [\![\sigma]\!]_r$.

The restrictions (1.9) and (1.10) indicate that, if a component other than A_i performs an operation on the resource, this operation does not affect the enabledness of the transitions in A_i, until the next time A_i operates on the resource.

In order to give a concrete example, assume there are two components modelled by atoms A_1 and A_2, while the resource is modelled by A_r. Let $a \in \text{ACTLAB}_1 \cap \text{ACTLAB}_r$, $b \in \text{ACTLAB}_1$, $b \notin \text{ACTLAB}_r$, $c \notin \text{ACTLAB}_1$. Then,

$$[\overbrace{(s_1^1, s_2^1, s_r^1).(g^1, a, r^1).(s_1^2, s_2^1, s_r^2).(g^2, b).(s_1^3, s_2^1, s_r^2).(g^3, c, r^3).(s_1^3, s_2^2, s_r^3)}^{\sigma}]_1 = s_1^1.a.(s_1^2, s_r^2).b.s_1^3 \ .$$

Furthermore, suppose that $g^3(s_2^2, d, s_2^3) > 0$ for some $d \notin \text{ACTLAB}_r$. That is, by executing g^3, atom A_2 can output not only c, but also a label d that is not an operation on A_r. Then,

$$[\overbrace{(s_1^1, s_2^1, s_r^1).(g^1, a, r^1).(s_1^2, s_2^1, s_r^2).(g^2, b).(s_1^3, s_2^1, s_r^2).(g^3, d).(s_1^3, s_2^2, s_r^2)}^{\sigma'}]_1 = s_1^1.a.(s_1^2, s_r^2).b.s_1^3 = [\sigma]_1 \ .$$

The fact that $[\sigma]_1 = [\sigma']_1$ reflects that A_1 is not able to see whether the state of the resource has changed since the last operation on it. Atom A_1 only

knows that, after the last time A_1 performed an operation, the state of the resource was s_r^2.

An order on projections

We say that a projection $[\cdot]'$ gives at least the same information as $[\cdot]$ (written $[\cdot] \sqsubseteq [\cdot]'$) if

$$\forall A_i \in \text{ATOMS}(P), \sigma, \sigma' : [\sigma]_i \neq [\sigma']_i \implies [\sigma]'_i \neq [\sigma']'_i \tag{1.12}$$

that is, all the paths distinguished by $[\cdot]$ are distinguished by $[\cdot]'$ as well.

As an example, for the projection $[\cdot]$ in Example 1.5, we have $[\![\cdot]\!] \sqsubseteq [\cdot]$. Intuitively, $[\![\cdot]\!]$ only allows to see the local state and the action labels in ACTLAB_i, while $[\cdot]$ also allows to see the state of the shared resource after a synchronization. In order to prove $[\![\cdot]\!] \sqsubseteq [\cdot]$, we can prove the contrapositive of (1.12), namely

$$\forall \sigma, \sigma' : [\sigma]_i = [\sigma']_i \implies [\![\sigma]\!]_i = [\![\sigma']\!]_i .$$

In case $A_i = A_r$, the result follows trivially from $[\cdot]_r = [\![\cdot]\!]_r$. For atoms $A_i \neq A_r$, the implication can be proven by induction, considering the four cases in the definition of $[\sigma]_i$ in Example 1.5. A concrete example gives us more insight: if $[\sigma^1]_i = \text{INIT}_1.a^1.(s_1^1, s_r^1).a^2.s_1^2$, then $[\![\sigma^1]\!]_i = \text{INIT}_1.a^1.s_1^1.a^2.s_1^2$. In general, $[\![\sigma]\!]_i$ can be obtained by removing the A_r-states from $[\sigma]_i$. Hence, for all σ, σ' such that $[\sigma]_i = [\sigma']_i$, by removing the A_r-states in $[\sigma]_i$ and $[\sigma']_i$, we obtain $[\![\sigma]\!]_i = [\![\sigma']\!]_i$ as well.

We say that two projections are equivalent (denoted by $[\cdot] \equiv [\cdot]'$) iff

$$\forall \sigma, \sigma' : [\sigma]_i \neq [\sigma']_i \iff [\sigma]'_i \neq [\sigma']'_i .$$

Note that $[\cdot] \equiv [\cdot]'$ iff $[\cdot] \sqsubseteq [\cdot]'$ and $[\cdot]' \sqsubseteq [\cdot]$.

We can obtain more insight on the relations $\sqsubseteq$ and $\equiv$ by considering the kernel of the projections. The *kernel* of a function f (denoted by KER_f) is an equivalence relation defined as:

$$a \,\text{KER}_f\, b \iff f(a) = f(b) .$$

Hence,

$$[\cdot] \equiv [\cdot]' \iff \forall A_i : \text{KER}_{[\cdot]_i} = \text{KER}_{[\cdot]'_i} \tag{1.13}$$

In addition, $[\cdot] \sqsubseteq [\cdot]'$ iff

$$\forall A_i : \forall \sigma, \sigma' : \sigma \,\text{KER}_{[\cdot]'_i}\, \sigma' \implies \sigma \,\text{KER}_{[\cdot]_i}\, \sigma' .$$

If we see the relation $\text{KER}_{[\cdot]_i}$ as a set of ordered pairs, then

$$[\cdot] \sqsubseteq [\cdot]' \iff \text{KER}_{[\cdot]'_i} \subseteq \text{KER}_{[\cdot]_i} \tag{1.14}$$

(in terms of relations, $\text{KER}_{[\cdot]_i}$ is coarser than or equal to $\text{KER}_{[\cdot]'_i}$).

The equivalences (1.13) and (1.14) imply that $\sqsubseteq$ defines a partial order on $\text{PROJECTIONS}(P)/_{\equiv}$. The results in this thesis do not profit from this property of $\sqsubseteq$, and we point it out just to justify the notation. The only property of the order $\sqsubseteq$ we use in the thesis is the lemma below, which states that $[\cdot]^{\text{ID}}$ is a top element of the order.

LEMMA 1.3. *For all projections* $[\cdot]$, *we have* $[\cdot] \sqsubseteq [\cdot]^{ID}$.

Proof. We have to prove

$$[\sigma]_i \neq \left[\sigma'\right]_i \implies [\sigma]_i^{ID} \neq [\sigma']_i^{ID} .$$

By definition of $[\cdot]^{ID}$ this is equivalent to

$$[\sigma]_i \neq \left[\sigma'\right]_i \implies \sigma \neq \sigma'$$

which is true since $[\cdot]_i$ is a function. □

1.3.2 *Schedulers*

Using the generalized projections, we can generalize the distributed schedulers. In order to introduce this generalization, we illustrate how output schedulers (as in Subsection 1.1.4) can be seen as functions whose domain is the set of *global* paths, instead of the local ones. Consider a function $f : \text{PATHS}(P) \to \text{PROB}(T_{G_i})$. If

$$\forall \sigma, \sigma' : \left([\![\sigma]\!]_i = [\![\sigma']\!]_i \implies f(\sigma) = f(\sigma') \right) \tag{1.15}$$

then we can define an output scheduler Θ_i^f as $\Theta_i^f([\![\sigma]\!]_i) = f(\sigma)$, where σ is any global path such that $[\![\sigma]\!]_i = \sigma_i$. Equation (1.15) ensures that the definition of $\Theta_i^f(\sigma_i)$ does not depend on the particular σ chosen (as long as $[\![\sigma]\!]_i = \sigma_i$). The functions f and Θ_i^f are related by

$$\forall \sigma : \Theta_i^f([\![\sigma]\!]_i) = f(\sigma) . \tag{1.16}$$

Conversely, given a scheduler Θ_i, we can define a function f^Θ as $f^\Theta(\sigma) = \Theta_i([\![\sigma]\!]_i)$. This function complies with (1.15), as well as with the following analogue of Equation (1.16):

$$\forall \sigma : \Theta_i([\![\sigma]\!]_i) = f^\Theta(\sigma) . \tag{1.17}$$

In conclusion, the set of output schedulers is in one-to-one correspondence with the set of functions f complying with (1.15). The existence of this correspondence means that we could have defined output schedulers as functions $\Theta_i : \text{PATHS}(P) \to \text{PROB}(T_{G_i})$ complying with Eqn. (1.15). By substituting an arbitrary projection $[\cdot]$ for $[\![\cdot]\!]$ in Eqn. (1.15), we get the general definition.

Given a projection $[\cdot]$, an output scheduler under $[\cdot]$ is a function

$$\Theta_i : \text{PATHS}(P) \to \text{PROB}(T_{G_i})$$

complying

$$\Theta_i(\sigma)(g_i) > 0 \implies g_i \in G_i(\text{LAST}(\sigma))$$

for all σ such that $\left| G_i(\text{LAST}(\sigma)) \right| > 0$ and

$$\forall \sigma, \sigma' : \left([\sigma]_i = \left[\sigma'\right]_i \implies \Theta_i(\sigma) = \Theta_i(\sigma') \right) . \tag{1.18}$$

Similarly, an input scheduler under $[\cdot]$ is a function

$$\Upsilon_i : \text{PATHS}(P) \times \text{ACTLAB}_i \to \text{PROB}(T_{R_i})$$

such that

$$\Upsilon_i(\sigma, a)(a_i) > 0 \implies r_i \in R_i(\sigma, a)$$

and

$$\forall a \in \text{ACTLAB}_i, \sigma, \sigma' : \left([\sigma]_i = \left[\sigma'\right]_i \implies \Upsilon_i(\sigma, a) = \Upsilon_i(\sigma', a) \right) . \quad (1.19)$$

Recall that, when defining projections, we imposed the restrictions (1.9) and (1.10), with the aim to ensure the existence of output and input schedulers. Moreover, we can ensure the existence of non-randomized schedulers.

THEOREM 1.1. *For all atoms A_i, projections $[\cdot]$, a non-randomized output (input, resp.) scheduler for A_i exists.*

Proof. We prove the case for output schedulers (the case for input schedulers follows in the same way). In order to do so, we construct an output scheduler Θ_i. For all $\sigma_i \in \text{LOCALPATHS}_i^{[\cdot]}$, let $Q(\sigma_i)$ be a global path such that $[Q(\sigma_i)]_i = \sigma_i$, and let $T(\sigma_i)$ be a generative transition enabled in $\text{LAST}(Q(\sigma_i))$. For all σ such that $[\sigma]_i = \sigma_i$, define

$$\Theta_i(\sigma)(T([\sigma]_i)) = 1$$

If $[\sigma]_i = [\sigma']_i$, then

$$\Theta_i(\sigma) = 1 : T([\sigma]_i) = 1 : T(\left[\sigma'\right]_i) = \Theta_i(\sigma')$$

as desired.

In addition, $\Theta_i(\sigma)(g_i) > 0 \implies g_i = T([\sigma]_i)$ and so, by definition of $T([\sigma]_i)$, we obtain $g_i \in G_i(\text{LAST}(Q([\sigma]_i)))$. Since (by definition of $Q(\sigma_i)$) we have $[\sigma]_i = [Q([\sigma]_i)]_i$, requirement (1.9) implies

$$G_i(\text{LAST}(\sigma)) = G_i(\text{LAST}(Q([\sigma]_i))) ,$$

and hence $g_i \in G_i(\text{LAST}(\sigma))$. □

EXAMPLE 1.6. In order to exemplify how schedulers and projections interact, we consider the projections $[\cdot]^f$ and $[\cdot]^{\text{ID}}$ presented at the beginning of Subsection 1.3.1. We show that the restriction imposed to schedulers yields the intended meaning we gave to these projections in Subsection 1.3.1. Again, in order to ease explanation, we assume that all transitions are enabled in all states. If Θ_i is distributed under $[\cdot]^f$, then it must be $\Theta_i(\sigma) = \Theta_i(\sigma')$ for all σ, σ' such that $[\sigma]_i^f = [\sigma']_i^f$. Since $[\sigma]_i^f = [\sigma']_i^f = \text{INIT}_i$, we have $\Theta_i(\sigma) = \Theta_i(\sigma')$ for all σ, σ'. That is, the scheduler chooses the same (distribution on) transition(s) for all σ, σ'. This corresponds to the intended meaning in Subsection 1.3.1, since we get an output scheduler whose resolution of nondeterminism is the same in all paths.

If Θ_i is distributed under $[\cdot]^{\text{ID}}$, then the equality is required for all σ, σ' such that $[\sigma]_i^{\text{ID}} = [\sigma']_i^{\text{ID}}$, that is, for all σ, σ' such that $\sigma = \sigma'$. Of course, the requirement $\forall \sigma = \sigma' : \Theta_i(\sigma) = \Theta_i(\sigma')$, holds no matter how we define Θ_i. Then, the schedulers distributed under $[\cdot]^{\text{ID}}$ are not restricted at all. They can be seen as schedulers that have access to all the information, and are thus able to make any arbitrary decision based on the full history of system.

Recall the projection $[\cdot]$ in Example 1.5, and the paths σ, σ' defined therein. In the example, we have $[\sigma]_1 = [\sigma']_1$. The restriction $\Theta_1(\sigma) = \Theta_1(\sigma')$ indicates that the resolution of the nondeterminism in A_1 cannot be changed according to whether A_2 has performed an operation on the shared resource or not. This constraint on the scheduler captures the fact that, in a distributed setting, A_1 is not able to see the operations of A_2 until some communication occurs via the shared resource.

So far, we have extended output and input schedulers. The definition of the interleaving scheduler is almost unchanged with respect to the one in Subsection 1.1.4: an interleaving scheduler is a function

$$\mathcal{I} : \textsc{paths}(P) \to \textsc{prob}(\{A_1, \cdots, A_N\})$$

such that

$$\mathcal{I}(\sigma)(A_i) > 0 \implies G_i(\textsc{last}(\sigma)) \neq \emptyset \,. \tag{1.20}$$

Note that the only change wrt. Subsection 1.1.4 is that the implication (1.20) concerns $\textsc{last}(\sigma)$, while the implication (1.5) concerns $\pi_i(\textsc{last}(\sigma))$. The reason for this change is that, in the general case, we deal with global transition structures (Def. 1.12).

A distributed scheduler under $[\cdot]$ is a tuple

$$(\mathcal{I}, \{\Theta_i\}_{i=1}^N, \{\Upsilon_i\}_{i=1}^N)$$

Θ_i is an output scheduler and Υ_i is an input scheduler under $[\cdot]$ for each A_i, and $\mathcal{I}$ is an interleaving scheduler.

Given a system P, we denote by $\textsc{dist}_P([\cdot])$ the set of all distributed schedulers for P under $[\cdot]$.

The following notation allows us to adapt the definitions and results for the projection $[\![\cdot]\!]$ (and its respective schedulers introduced in Subsection 1.1.4) to generalized projections and schedulers.

NOTATION 1.3. Given an output scheduler Θ_i under $[\cdot]$, and a local path σ_i in $\textsc{localpaths}_i^{[\cdot]}$ (recall Def. 1.13), we define $\Theta_i(\sigma_i) = \Theta_i(\sigma)$, where σ is any global path such that $[\sigma]_i = \sigma_i$. Equation (1.18) ensures that the existence of several such σ does not introduce any ambiguity. Similarly, we write $\Upsilon_i(\sigma_i, a)$ for $\Upsilon_i(\sigma, a)$ for any input scheduler Υ_i.

This notation is useful in calculations, since it allows us to write

$$\sum_{\{\sigma \,|\, [\sigma]_i = \sigma_i\}} \Theta_i(\sigma) \cdot f(\sigma) = \Theta_i(\sigma_i) \cdot \sum_{\{\sigma \,|\, [\sigma]_i = \sigma_i\}} f(\sigma)$$

instead of the more verbose

$$\sum_{\{\sigma \,|\, [\sigma]_i = \sigma_i\}} \Theta_i(\sigma) \cdot f(\sigma) = \Theta_i(\sigma^*) \cdot \sum_{\{\sigma \,|\, [\sigma]_i = \sigma_i\}} f(\sigma) \quad \text{for some } \sigma^* \text{ s.t. } [\sigma^*]_i = \sigma_i \;\; .$$

In the light of the notation, we notice that an alternative definition for output schedulers could be $\Theta_i : \textsc{localpaths}_i^{[\cdot]} \to \textsc{prob}(T_{G_i})$. The problem with this definition is that, given Θ_i under $[\cdot]$ and Θ_i' under $[\cdot]' \neq [\cdot]$, these output schedulers are different mathematical entities (since the domain of Θ_i is

$\text{LOCALPATHS}_i^{[\cdot]}$, while the domain of Θ_i' is $\text{LOCALPATHS}_i^{[\cdot]'}$) even if they resolve nondeterminism in the same way for all paths, that is, even if $\Theta_i([\sigma]_i) = \Theta_i'([\sigma]_i')$ for all σ. In particular, the following theorem could not be stated so concisely.

THEOREM 1.2.

$$[\cdot] \sqsubseteq [\cdot]' \implies \text{DIST}_P([\cdot]) \subseteq \text{DIST}_P([\cdot]')$$

In words, if $[\cdot]'$ gives more information than $[\cdot]$, then the schedulers have more freedom to resolve nondeterminism under $[\cdot]'$ than under $[\cdot]$[†].

Proof. Let $\eta = (\mathcal{I}, \{\Theta_i\}_i, \{\Upsilon_i\}_i)$ be distributed under $[\cdot]$. We show that the schedulers Θ_i are also distributed under $[\cdot]'$. Since the same argument applies to the schedulers Υ_i, this implies that η is distributed under $[\cdot]'$.

For all σ, σ', we have:

$$\begin{aligned}
& [\sigma]_i' = [\sigma']_i' \\
\implies & \{ \text{ Definition of } \sqsubseteq \ \} \\
& [\sigma]_i = [\sigma']_i \\
\implies & \{ \ \Theta_i \text{ is distributed under } [\cdot] \ \} \\
& \Theta_i(\sigma) = \Theta_i(\sigma')
\end{aligned}$$

The implication $[\sigma]_i' = [\sigma']_i' \implies \Theta_i(\sigma) = \Theta_i(\sigma')$ is precisely the requirement for a scheduler to be distributed under $[\cdot]'$. □

The next corollary follows from Lemma 1.3.

COROLLARY 1.1. *For all projections* $[\cdot]$, $\text{DIST}_P([\cdot]) \subseteq \text{DIST}_P([\cdot]^{\text{ID}})$.

Hence, the set $\text{DIST}_P([\cdot]^{\text{ID}})$ encompasses all schedulers for P, for all projections. We denote this set by SCHED_P SCHED_P

An alternative interpretation of Theorem 1.2 is that, if a strategy can be carried out with a certain amount of information (namely, the information provided by $[\cdot]$) then the same strategy can be carried out with more information and, in particular, the information provided by $[\cdot]'$. This is illustrated by the following example.

EXAMPLE 1.7. Recall the system in Example 1.5. Consider two distinct paths

$$\sigma^1 = (s_1^1, s_2^1, s_r^1).(g^1, a, r^1).(s_1^1, s_2^2, s_r^2).(g^2, b, r^2).(\, s_1^2 \,, s_2^2, s_r^3)$$

and

$$\sigma^2 = (s_1^1, s_2^1, s_r^1).(g^1, a, r^1).(s_1^1, s_2^2, s_r^2).(g^2, b, r^2).(\, s_1'^2 \,, s_2^2, s_r^3)$$

where $a \in \text{ACTLAB}_2 \cap \text{ACTLAB}_r$, $a \notin \text{ACTLAB}_1$ and $b \in \text{ACTLAB}_1 \cap \text{ACTLAB}_r$. The only difference between σ^1 and σ^2 is that, in σ^1, the transition g^2 leads A_1 to s_1^2 while, in σ^2, it leads A_1 to $s_1'^2 \neq s_1^2$.

Let Θ_1, be an output scheduler such that

$$\Theta_1(\sigma^1)(g_1) = 1$$
$$\Theta_1(\sigma^2)(g_1') = 1$$

[†]Kind of "The information will set you free"

with $g_1 \neq g'_1$. Since

$$[\![\sigma^1]\!]_1 = s_1^1.b.s_1^2 \neq s_1^1.b.s_1'^2 = [\![\sigma^2]\!]_1$$

Θ_1 is distributed under $[\![\cdot]\!]$, as far as σ^1 and σ^2 are concerned: formally, Θ_1 complies with (1.18) for $[\![\cdot]\!]$, σ^1, σ^2. The restriction (1.18) is also satisfied under the projection $[\cdot]$ in Example 1.5, since

$$\left[\sigma^1\right]_1 = s_1^1.b.(s_1^2, s_r^3) \neq s_1^1.b.(s_1'^2, s_r^3) = \left[\sigma^2\right]_1 .$$

Roughly speaking, if the scheduler for A_i resolves nondeterminism according to the state of A_i, then it can resolve nondeterminism according the state of A_i and the state of A_r.

Now, consider the path

$$\sigma^3 = (s_1^1, s_2^1, s_r^1).(g^1, c, r^1).(s_1^1, s_2^2, s_r'^2).(g^2, b, r^2).(s_1^2, s_2^2, s_r'^3)$$

where $c \in \text{ACTLAB}_2 \cap \text{ACTLAB}_r$. Note that the difference wrt. σ^1 is that g^1 outputs c instead of a, and the label c leads the resource to state $s_r'^2$ instead of s_r^2. In addition, when the resource is in $s_r'^2$ and receives the label b, it moves to state $s_r'^3$, instead of s_r^3. An output scheduler Θ_i complying with $\Theta_1(\sigma^1) = g_1$, $\Theta_1(\sigma^3) = g'_1$ satisfies (1.18) under the projection $[\cdot]$, since

$$\left[\sigma^1\right]_1 = s_1^1.b.(s_1^2, s_r^3) \neq s_1^1.b.(s_1^2, s_r'^3) = \left[\sigma^3\right]_1 .$$

However, it is not a distributed scheduler under the projection $[\![\cdot]\!]$, since

$$[\![\sigma^1]\!]_1 = s_1^1.b.s_1^2 = [\![\sigma^3]\!]_1 .$$

For any scheduler η having Θ_1 as output scheduler for A_1, we have $\eta \notin \text{DIST}_P([\![\cdot]\!])$. This reflects the fact that, if the information about the state of the resource is not available to A_1, then (the scheduler of) A_1 cannot resolve its nondeterminism based on such information.

Definitions 1.9 and 1.10, as well as Lemma 1.2 can be straightforwardly adapted to generalized schedulers and projections.

DEFINITION 1.14. Let $\eta \in \text{SCHED}_P$. The discrete probability distribution on compound transitions $\eta(\sigma)(\cdot)$ is defined as

$$\begin{aligned}\eta(\sigma)(g_i, a, r_{j_1}, \cdots, r_{j_m}) = {} & \mathcal{I}(\sigma)(A_i) \cdot \Theta_i([\sigma]_i)(g_i) \\ & \cdot \sum_{s_i} g_i(\pi_i(\text{LAST}(\sigma)), a, s_i) \\ & \cdot \prod_{k=1}^{m} \Upsilon_{j_k}([\sigma]_{j_k}, a)(r_{j_k})\end{aligned}$$

if $\left| G_i(\text{LAST}(\sigma)) \right| > 0$ for some A_i. Otherwise, $\eta(\sigma)(\varsigma) = 1$.

LEMMA 1.4. *For all schedulers η distributed under $[\cdot]$, paths σ, and compound transitions c, we have*

$$\sum_{c \in \text{ENABLED}(\text{LAST}(\sigma))} \eta(\sigma)(c) = 1 .$$

The proof of the lemma is the same as that of Lemma 1.2, replacing $[\![\cdot]\!]$ by $[\cdot]$.

DEFINITION 1.15 (Probability of a set of paths). Let η be a scheduler distributed under $[\cdot]$. For a cylinder $(\sigma)^\uparrow$, the probability measure $\Pr^\eta$ is inductively defined by:

$$\Pr^\eta((\text{INIT})^\uparrow) = 1$$
$$\Pr^\eta((\sigma.c.s)^\uparrow) = \Pr^\eta((\sigma)^\uparrow) \cdot \eta(\sigma)(c) \cdot c(\text{LAST}(\sigma), s)$$

$\Pr^\eta$ uniquely extends to least σ-field containing all cylinders.

1.4 COMPARISON WITH EXISTING APPROACHES

PROBABILISTIC SYSTEMS In [133], a general framework of probabilistic automata is presented. The composition defined in this framework does not preserve the structure concerning the constituent entities, and an ad-hoc equivalence is required to consider partial information. Similarly, [63] considers simple MDPs equipped with an equivalence relation $\sim$ on states. Two paths σ, σ' are equivalent iff they have the same length and $\sigma(i) \sim \sigma'(i)$ for all i.

In contrast to these approaches, we introduce a framework with the aim to represent the uncertainty that is present in distributed systems. Existing frameworks in which partial information is an essential characteristic include the *probabilistic modules* in [64], and the different versions of *probabilistic I/O automata* [147, 46, 34]. It is no coincidence that these formalisms were devised with the aim to develop techniques of compositional reasoning.

The probabilistic I/O automata in [147] are I/O deterministic, but there is nondeterministic choice concerning the order in which components execute. The probability that a given entity executes before another one depends solely on the local states of these entities. The switched probabilistic I/O automata in [46] are similar to ours, and we borrow the input and output schedulers from this approach. A state of the system comprises the state of each automaton being composed, plus the state of a *token*. During the execution, the token is assigned to exactly one of the automata. The automaton holding the token is the only one able to execute generative transitions. Generative transitions also specify which automaton is the next one to receive the token, and so the interleaving is restricted in the specification. Interleaving nondeterminism arises from several transitions passing the token to different automata. The recent task probabilistic I/O automata in [34] go back to the original approach in [133] by considering equivalence relations on transitions.

PROJECTIONS The concept of projection resembles the *observations* in Partially Observable Markov Decision Processes (POMDPs), but a crucial difference is that our projections are *path based* instead of *state based*: in POMDPs (see [135, 35, 114]), the model specifies a distribution $p_{t,s}$ on observations for each pair transition/state. This models the fact that, when state s is reached through transition t, the POMDP can obtain different observations o, each one with probability $p_{t,s}(o)$. The observations on pairs induce observations on paths in the natural way (two paths are observed as equal if each of the

pairs transition/state are observed as equal). This is a good model to deal with environments in which there is some amount of uncertainty. In such environments, the execution of a transition can yield different observations, and the probability that a certain observation is perceived might depend on the particular state and the particular transition. In Decentralized POMDPs, several entities are considered. The steps of the system are obtained by performing a step in each of the entities and, for each step, each entity may have a different distribution on observations. Again, this a good model in case several entities are placed in an environment that introduces some kind of uncertainty.

Briefly speaking, Decentralized POMDPs are about several entities evolving in an environment that introduces uncertainty, while Interleaved Probabilistic I/O Automata are about several entities that are uncertain with respect to each other's state. So, although we also deal with unavailability of information, our uncertainty is caused exclusively by the information that is not shared among components. Since we deal with this kind of uncertainty, two important differences arise: in the first place, the uncertainty does not need to be modelled separately, since it is derived from the entities' models. For instance, suppose that an entity tosses a balanced coin. Moreover, suppose that the entity communicates to all other entities the fact that the coin has been tossed, but the outcome is kept as a secret. Then all of the other entities are equally uncertain about whether the coin landed heads or tails, and they know that, with probability $1/2$, the coin has landed heads (and tails, resp.) Note that this probability distribution does not need to be modelled separately, since it is implied by the distribution on coin outcomes. Another important implication is that, given that the entities are distributed, an entity may perform a step in such a way that another entity does not even notice it. So, not every step yields an observation for every entity, and so the notion of observation on states cannot be extended to paths in a straightforward way.

SCHEDULERS The idea of having separate schedulers for each atom is already used in [64, 46]. However, this is not the only possible approach. In POMDPS, as well as in [63], the schedulers are required to satisfy $\eta(\sigma) = \eta(\sigma')$ whenever $\sigma \sim \sigma'$. In [133], partial information is modelled using an equivalence relation $\equiv$ on paths and a family F of functions mapping transitions to transitions (one function $f_{\sigma,\sigma'}$ for each pair (σ, σ') in the equivalence relation). The pair $(\equiv, F)$ is called an *oblivious relation*. A scheduler is then said to be with partial information if, whenever two paths σ, σ' are equivalent with respect to $\equiv$, then $\eta(\sigma) = f_{\sigma,\sigma'}(\eta(\sigma'))$. The problem with this approach is that the equivalence relation does not allow to express that two paths are equivalent from the point of view of a certain atom, but distinguishable by another one. In addition, using our schema of partial information we can obtain more precise results about the expressive power of randomized schedulers. In [133, p. 99], a discussion explains that adversaries must be length sensitive in order to prevent randomized schedulers to be more powerful than non-randomized schedulers. In Chapter 4 (more precisely, in Corollary 4.2), we sharpen this statement and find a more precise criterion. In particular, the equivalence between randomized and non-randomized schedulers is preserved by some projections that are not fully sensitive to the path length.

2

RESTRICTIONS ON THE INTERLEAVING SCHEDULER

The definition of distributed schedulers introduced in the last chapter does not restrict the interleaving schedulers with respect to the availability of information. The fact that the domain of interleaving schedulers is the set of *global* paths introduces unrealistic behaviours similar to those that motivated the introduction of distributed schedulers, and so this chapter is devoted to impose restrictions on the interleaving scheduler with the aim to eliminate such behaviours.

The first restriction we propose is based on conditional probabilities, and is the more permissive of the restrictions we present in this chapter. The schedulers complying with such restriction are called *strongly distributed schedulers*. The second restriction can be seen as a scheduling assumption, namely, that the sojourn time of an atom after a given local path is distributed according to an exponential distribution, whose mean depends on the particular local path. The set of *rate-based* schedulers comprises the schedulers that can arise under this assumption. We use the word *rate* since the parameters of exponential distributions are often interpreted as rates, and the schedulers in this set resolve the interleaving nondeterminism by setting such rates according to the local path. The third restriction can be thought of as another assumption, namely, that the atoms are given a certain priority according to the local path traversed so far, and the next atom to execute is the one having the highest priority. These priorities are modelled using a total order on the local paths, and so these schedulers are called *total order-based* schedulers. These schedulers can occur if one of the entities coordinates the execution of the others. In this case, the local paths of the atoms reflect the information that the coordinator is able to see. These schedulers might seem not applicable in all cases, since the systems we consider do not necessarily have a coordinator. However, if the projection fulfils certain conditions (detailed in Subsection 4.2.2), the extremal probabilities quantifying over rate-based or strongly distributed schedulers equal those obtained by quantifying over total order-based schedulers. These equalities are useful in proofs, since total order-based schedulers are technically easier to manipulate than general strongly distributed schedulers.

2.1 STRONGLY DISTRIBUTED SCHEDULERS

Distributed schedulers model the fact that, when facing a nondeterministic choice, the components can only look at their local history. 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. 2.1. Consider the system $T \parallel Z \parallel A \parallel B$. In this system, T is a process that tosses a coin. For the labels $h!$ and $t!$, corresponding to heads and tails, we have $h!, t! \notin \text{ACTLAB}_Z \cup \text{ACTLAB}_A \cup \text{ACTLAB}_B$. So, if we consider the usual

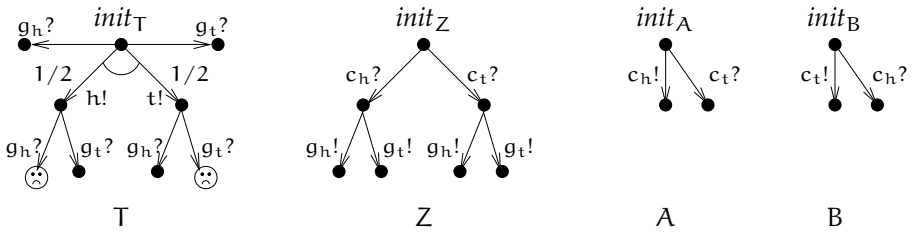


Figure 2.1: Motivating strongly distributed schedulers

projection $[\![\cdot]\!]$, the intended meaning is that T keeps the outcome as a secret [†]. 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 ☹ 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 η^d 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 $c_h!$ and $c_t!$ 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. The scheduler η^d is depicted in Fig. 2.2.

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 (see Fig. 2.3), we end up in a situation very similar to the guess-heads-or-tails example: in the case in which the coin lands heads, AB chooses to perform the transition $c_h!$. In case the coin lands tails, AB chooses to perform the transition $c_t!$. In fact, note that the graphical representation of the unrealistic scheduler η^d coincides with the unrealistic

[†]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 [44].

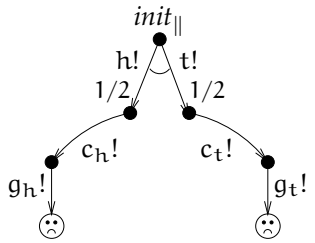


Figure 2.2: An unrealistic distributed scheduler

scheduler of the guess-heads-or-tails example in Fig. 0.3. If we consider the system $T \parallel Z \parallel AB$, no output scheduler for AB can be defined in such a way that the order of execution of $c_h!$ and $c_t!$ 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 $T \parallel Z \parallel AB$ being able to simulate η^d. Therefore, in $T \parallel Z \parallel A \parallel B$ we would like to consider only the schedulers having a corresponding distributed scheduler in $T \parallel Z \parallel AB$.

In the general case, 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 $\text{ATOMS}(P') = \big(\text{ATOMS}(P) \setminus \{A, B\}\big) \cup \{AB\}$. We want to restrict interleaving schedulers in such a way 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 distributed scheduler η for the system $T \parallel A \parallel B$ such that $\mathcal{I}(\text{INIT}) = (\frac{1}{2}T + \frac{2}{6}A + \frac{1}{6}B)$. We seek a restriction on $\mathcal{I}$ s.t. it is possible to find a distributed scheduler for $T \parallel AB$. When AB is in state $(\text{INIT}_A, \text{INIT}_B)$, the output scheduler Θ_{AB} chooses a distribution on $\{c_h!, c_t!\}$. To respect the choice of $\mathcal{I}$ in $T \parallel A \parallel B$, it must hold that $\Theta_{AB}(\text{INIT}_{AB})(c_h!) = 2 \cdot \Theta_{AB}(\text{INIT}_{AB})(c_t!)$, since, according to $\mathcal{I}$, the probability of executing $c_h!$ is twice the probability of executing $c_t!$. Then,

$$\Theta_{AB}(\text{INIT}_{AB})(c_h!) = \tfrac{2}{3} \quad \text{and} \quad \Theta_{AB}(\text{INIT}_{AB})(c_t!) = \tfrac{1}{3}\,. \tag{2.1}$$

Suppose $(\text{INIT}_T, \text{INIT}_A, \text{INIT}_B) \xrightarrow{t!} (\mathit{heads}_T, \text{INIT}_A, \text{INIT}_B)$ in $T \parallel A \parallel B$. The corresponding path in $T \parallel AB$ is $(\text{INIT}_T, \text{INIT}_{AB}) \xrightarrow{t!} (\mathit{heads}_T, \text{INIT}_{AB})$. Call both these paths σ_{heads} (ambiguity is resolved according to whether it is used in the context of $T \parallel A \parallel B$ or $T \parallel AB$).

Since $[\![\sigma_{\mathit{heads}}]\!]_{AB} = \text{INIT}_{AB} = [\![(\text{INIT}_T, \text{INIT}_{AB})]\!]_{AB}$, we have that

$$\begin{aligned}\Theta_{AB}([\![(\text{INIT}_T, \text{INIT}_{AB})]\!]_{AB})(c_h!) &= \Theta_{AB}([\![\sigma_{\mathit{heads}}]\!]_{AB})(c_h!) \\ &= \Theta_{AB}(\text{INIT}_{AB})(c_h!) = \frac{2}{3}\end{aligned}$$

and similarly for $c_t!$. Therefore

$$\Theta_{AB}([\![\sigma_{\mathit{heads}}]\!]_{AB})(c_h!) = 2 \cdot \Theta_{AB}([\![\sigma_{\mathit{heads}}]\!]_{AB})(c_t!)\,.$$

This relation has to be maintained in $T \parallel A \parallel B$ by $\mathcal{I}(\sigma_{\mathit{heads}})$. That is, whichever is the probabilistic choice in $\mathcal{I}(\sigma_{\mathit{heads}})$ w.r.t. other atoms, the relation

$$\mathcal{I}(\sigma_{\mathit{heads}})(c_h!) = 2 \cdot \mathcal{I}(\sigma_{\mathit{heads}})(c_t!)$$

has to be maintained.

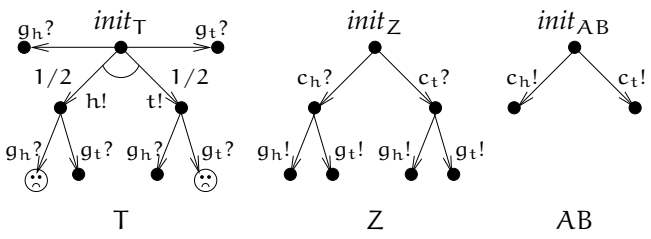


Figure 2.3: Regarding A and B as a single component

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 and a projection $[\cdot]$, the examples above motivate the following general restriction for $\mathcal{I}$: for all σ, σ' s.t. $[\sigma]_A = [\sigma']_A$ and $[\sigma]_B = [\sigma']_B$:

$$\frac{\mathcal{I}(\sigma)(A)}{\mathcal{I}(\sigma)(A)+\mathcal{I}(\sigma)(B)} = \frac{\mathcal{I}(\sigma')(A)}{\mathcal{I}(\sigma')(A)+\mathcal{I}(\sigma')(B)} \tag{2.2}$$

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

Strongly distributed scheduler

DEFINITION 2.1. A scheduler $\eta = (\mathcal{I}, \{\Theta_i\}_i, \{\Upsilon_i\}_i)$ is *strongly distributed* under projection $[\cdot]$ iff $\eta \in \text{DIST}_P([\cdot])$ and

$$\frac{\mathcal{I}(\sigma)(A)}{\mathcal{I}(\sigma)(A)+\mathcal{I}(\sigma)(B)} = \frac{\mathcal{I}(\sigma')(A)}{\mathcal{I}(\sigma')(A)+\mathcal{I}(\sigma')(B)} \tag{2.3}$$

holds for all σ, σ' such that $[\sigma]_A = [\sigma']_A$, $[\sigma]_B = [\sigma']_B$, $\text{PR}^\eta((\sigma)^\uparrow) > 0$, $\text{PR}^\eta((\sigma')^\uparrow) > 0$ and $\mathcal{I}(\sigma)(A)+\mathcal{I}(\sigma)(B) \neq 0$, $\mathcal{I}(\sigma')(A)+\mathcal{I}(\sigma')(B) \neq 0$. The set comprising these schedulers is denoted by $\text{SDIST}_P([\cdot])$.

Note that we do not require condition (2.3) to hold unless both σ and σ' have positive probability under η. By doing so, we allow some schedulers for which (2.3) does not hold (for such scheduler s, we have either $\text{PR}^\eta((\sigma)^\uparrow) = 0$ or $\text{PR}^\eta((\sigma')^\uparrow) > 0$). However, we profit from this relaxation in several proofs.

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 distributed schedulers should be considered.

The following theorem is the generalization of the fact that, for every strongly distributed scheduler η on $T \parallel Z \parallel A \parallel B$ as in Fig. 2.1 there is a distributed scheduler η' on $T \parallel Z \parallel AB$ that defines the same probabilistic behaviour.

THEOREM 2.1. *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, [45, 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 such that $[\sigma]_{AB} = \sigma_{AB}$. Define

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

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 $p_{\sigma,g_A} = \mathcal{I}(\sigma)(A) \cdot \Theta_A([\sigma]_A)(g_A)$. Then, p_{σ,g_A} equals

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

which in turn equals $\mathcal{I}'(\sigma)(AB) \cdot \Theta_{AB}([\sigma]_{AB})(g_A)$, 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.

Given reactive transitions $r_A \in R_A(s,a)$ and $r_B \in R_B(s,a)$, the atom AB has a reactive transition $(r_A, r_B) \in R_{AB}(s,a)$. This transition is defined as $(r_A, r_B)(t_A, t_B) = r_A(t_A) \cdot r_B(t_B)$. The input scheduler for AB is defined as $\Upsilon_{AB}(\sigma,a)(\,(r_A,r_B)\,) = \Upsilon_A(\sigma,a)(r_A) \cdot \Upsilon_B(\sigma,a)(r_B)$.

The input and output schedulers for the atoms other than A, B are the same as in the scheduler for the original system. □

One may wonder what happens if, instead of considering two atoms A and B in (2.3), two *disjoint sets* $\mathcal{A}, \mathcal{B}$ of atoms are considered. The (apparently more general) condition on sets holds whenever the condition (2.3) on atom pairs holds, as formalized in the following theorem.

THEOREM 2.2. *Let $\mathcal{A} = \{A_1, \cdots, A_n\}$, $\mathcal{B} = \{B_1, \cdots, B_m\}$ be disjoint sets of atoms. If Eqn. (2.3) 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 $\sum_i \mathcal{I}(\sigma')(A_i) + \sum_j \mathcal{I}(\sigma')(B_j) \neq 0$ and $[\sigma]_A = [\sigma']_A$ for all $A \in \mathcal{A} \cup \mathcal{B}$.

Proof. By induction on n. We prove the base case $n = 1$ by induction on m. If $m = 1$, the statement becomes Eqn. (2.3). 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. (2.3), then simple arithmetic gives

$$\frac{\mathcal{I}(\sigma)(B)}{\mathcal{I}(\sigma)(A)} = \frac{\mathcal{I}(\sigma')(B)}{\mathcal{I}(\sigma')(A)} \,. \tag{2.4}$$

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 explore the case $\mathcal{I}(\sigma)(A_1) = 0$. If $\mathcal{I}(\sigma)(B_j) = 0$ for all j, condition $\mathcal{I}(\sigma)(A_1) + \sum_j \mathcal{I}(\sigma)(B_j) \neq 0$ is false, and so the equation is not required to hold. If $\mathcal{I}(\sigma)(B_{j^*}) > 0$ for some B_{j^*}, we show that $\mathcal{I}(\sigma')(A_1) = 0$ and so the equation holds. Suppose, towards a contradiction, that $\mathcal{I}(\sigma')(A_1) \neq 0$. Then, $\mathcal{I}(\sigma')(A_1) + \mathcal{I}(\sigma')(B_{j^*}) \neq 0$. In addition, $\mathcal{I}(\sigma)(A_1) + \mathcal{I}(\sigma)(B_{j^*}) > 0$. Hence, Eqn. (2.3) ensures

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

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. (2.4) in the following calculation.

$$\begin{aligned}
&\frac{\mathcal{I}(\sigma)(A_1)}{\mathcal{I}(\sigma)(A_1) + \sum_j \mathcal{I}(\sigma)(B_j)} \\
=&\{\ \text{Arithmetics}\ \} \\
&\left(\frac{\mathcal{I}(\sigma)(B_m)}{\mathcal{I}(\sigma)(A_1)} + \frac{\mathcal{I}(\sigma)(A_1) + \sum_{j=1}^{m-1} \mathcal{I}(\sigma)(B_j)}{\mathcal{I}(\sigma)(A_1)}\right)^{-1} \\
=&\{\ \text{Equation (2.4)}\ \} \\
&\left(\frac{\mathcal{I}(\sigma')(B_m)}{\mathcal{I}(\sigma')(A_1)} + \frac{\mathcal{I}(\sigma)(A_1) + \sum_{j=1}^{m-1} \mathcal{I}(\sigma)(B_j)}{\mathcal{I}(\sigma)(A_1)}\right)^{-1} \\
=&\{\ \text{Inductive hypothesis}\ \} \\
&\left(\frac{\mathcal{I}(\sigma')(B_m)}{\mathcal{I}(\sigma')(A_1)} + \frac{\mathcal{I}(\sigma')(A_1) + \sum_{j=1}^{m-1} \mathcal{I}(\sigma')(B_j)}{\mathcal{I}(\sigma')(A_1)}\right)^{-1} \\
=&\{\ \text{Arithmetics}\ \} \\
&\frac{\mathcal{I}(\sigma')(A_1)}{\mathcal{I}(\sigma')(A_1) + \sum_j \mathcal{I}(\sigma')(B_j)}
\end{aligned}$$

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

$$\begin{aligned}
&\frac{\sum_i \mathcal{I}(\sigma)(A_i)}{\sum_i \mathcal{I}(\sigma)(A_i) + \sum_j \mathcal{I}(\sigma)(B_j)} \\
=&\frac{\sum_{i=1}^{n-1} \mathcal{I}(\sigma)(A_i) + \mathcal{I}(\sigma)(A_n)}{\sum_{i=1}^{n-1} \mathcal{I}(\sigma)(A_i) + \mathcal{I}(\sigma)(A_n) + \sum_j \mathcal{I}(\sigma)(B_j)} \\
=&\frac{\sum_{i=1}^{n-1} \mathcal{I}(\sigma)(A_i)}{\sum_{i=1}^{n-1} \mathcal{I}(\sigma)(A_i) + \mathcal{I}(\sigma)(A_n) + \sum_j \mathcal{I}(\sigma)(B_j)} \\
&\quad + \frac{\mathcal{I}(\sigma)(A_n)}{\sum_{i=1}^{n-1} \mathcal{I}(\sigma)(A_i) + \mathcal{I}(\sigma)(A_n) + \sum_j \mathcal{I}(\sigma)(B_j)} \\
=&\{\ \text{Inductive hypothesis for } \{A_i\}_{i=1}^{n-1},\ A_n \cup \{B_j\}_{j=1}^{m}\ \} \\
&\frac{\sum_{i=1}^{n-1} \mathcal{I}(\sigma')(A_i)}{\sum_{i=1}^{n-1} \mathcal{I}(\sigma')(A_i) + \mathcal{I}(\sigma')(A_n) + \sum_j \mathcal{I}(\sigma')(B_j)}
\end{aligned}$$

$$
\begin{aligned}
&\quad + \frac{\mathcal{I}(\sigma)(A_n)}{\sum_{i=1}^{n-1}\mathcal{I}(\sigma)(A_i)+\mathcal{I}(\sigma)(A_n)+\sum_j \mathcal{I}(\sigma)(B_j)} \\
&= \{\ \text{Base case with } \{A_n\}, \{B_i\}_{i=1}^m \cup \{A_i\}_{i=1}^{n-1}\ \} \\
&= \frac{\sum_{i=1}^{n-1}\mathcal{I}(\sigma')(A_i)}{\sum_{i=1}^{n-1}\mathcal{I}(\sigma')(A_i)+\mathcal{I}(\sigma')(A_n)+\sum_j \mathcal{I}(\sigma')(B_j)} \\
&\quad + \frac{\mathcal{I}(\sigma')(A_n)}{\sum_{i=1}^{n-1}\mathcal{I}(\sigma')(A_i)+\mathcal{I}(\sigma')(A_n)+\sum_j \mathcal{I}(\sigma')(B_j)} \\
&= \frac{\sum_i \mathcal{I}(\sigma')(A_i)}{\sum_i \mathcal{I}(\sigma')(A_i)+\sum_j \mathcal{I}(\sigma')(B_j)}
\end{aligned}
$$

□

The analogous of Theorem 1.2 holds for strongly distributed schedulers.

THEOREM 2.3.

$$[\cdot] \sqsubseteq [\cdot]' \implies \text{SDIST}_P([\cdot]) \subseteq \text{SDIST}_P([\cdot]')$$

Proof. Let $\eta \in \text{SDIST}_P([\cdot])$. We have to prove $\eta \in \text{SDIST}_P([\cdot]')$. Since $\eta \in \text{DIST}_P([\cdot])$, by Theorem 1.2 we have $\eta \in \text{DIST}_P([\cdot]')$. So, we need to prove

$$\frac{\mathcal{I}(\sigma)(A)}{\mathcal{I}(\sigma)(A)+\mathcal{I}(\sigma)(B)} = \frac{\mathcal{I}(\sigma')(A)}{\mathcal{I}(\sigma')(A)+\mathcal{I}(\sigma')(B)} \tag{2.5}$$

holds for all σ, σ' such that $[\sigma]'_A = [\sigma']'_A$, $[\sigma]'_B = [\sigma']'_B$, $\text{PR}^\eta((\sigma)^\uparrow) > 0$, $\text{PR}^\eta((\sigma')^\uparrow) > 0$ and $\mathcal{I}(\sigma)(A)+\mathcal{I}(\sigma)(B) \neq 0$, $\mathcal{I}(\sigma')(A)+\mathcal{I}(\sigma')(B) \neq 0$. For any such σ, σ', the hypothesis $[\cdot] \sqsubseteq [\cdot]'$ yields

$$[\sigma]'_A = [\sigma']'_A \wedge [\sigma]'_B = [\sigma']'_B \implies [\sigma]_A = \left[\sigma'\right]_A \wedge [\sigma]_B = \left[\sigma'\right]_B\ . \tag{2.6}$$

By $\eta \in \text{SDIST}_P([\cdot])$, Eqn. (2.5) holds for all σ, σ' complying with the consequent of (2.6). Hence, Eqn. (2.5) holds for all σ, σ' such that $[\sigma]'_A = [\sigma']'_A$, $[\sigma]'_B = [\sigma']'_B$. □

For the study of strongly distributed schedulers, the interesting systems are those in which the interleaving scheduler resolves at least one nondeterministic choice, that is, the systems P such that there exists $\sigma^* \in \text{PATHS}(P)$ such that $G_i(\text{LAST}(\sigma^*)) \neq \emptyset$ and $G_j(\text{LAST}(\sigma^*)) \neq \emptyset$ for some A_i, A_j with $A_i \neq A_j$. The following theorem holds for all of such systems.

THEOREM 2.4. *Let P be a system such that there exists $\sigma^* \in \text{PATHS}(P)$ such that $G_i(\text{LAST}(\sigma^*)) \neq \emptyset$ and $G_j(\text{LAST}(\sigma^*)) \neq \emptyset$ for some A_i, A_j with $A_i \neq A_j$. Then, there exists a randomized interleaving scheduler for P complying with Eqn. (2.3).*

Proof. Let $\mathcal{I}(\sigma)(A_k) = 1 / \left|\{A_m \mid G_m(\text{LAST}(\sigma)) \neq \emptyset\}\right|$ for all A_k such that $G_k(\text{LAST}(\sigma)) \neq \emptyset$. This scheduler is randomized since, by hypothesis,

$$\left|\{A_m \mid G_i(\text{LAST}(\sigma^*))\}\right| \geqslant 2\ ,$$

and so $0 < \mathcal{I}(\sigma^*)(A_i) \leqslant 1/2$. We prove that $\mathcal{I}$ complies with Eqn. (2.3). Suppose $\mathcal{I}(\sigma)(A_k) > 0$ and $[\sigma]_k = [\sigma']_k$. Then $G_k(\text{LAST}(\sigma)) \neq \emptyset$ and Eqn. (1.9)

imply $G_k(\text{LAST}(\sigma')) \neq \emptyset$. Similarly, if $[\sigma]_{k'} = [\sigma']_{k'}$, we have $G_{k'}(\text{LAST}(\sigma)) \neq \emptyset \iff G_{k'}(\text{LAST}(\sigma')) \neq \emptyset$. Hence, if $G_{k'}(\text{LAST}(\sigma)) = \emptyset$, we have

$$\frac{\mathcal{I}(\sigma)(A_k)}{\mathcal{I}(\sigma)(A_k)+\mathcal{I}(\sigma)(A_{k'})} = \frac{\mathcal{I}(\sigma)(A_k)}{\mathcal{I}(\sigma)(A_k)} = 1 = \frac{\mathcal{I}(\sigma')(A_k)}{\mathcal{I}(\sigma')(A_k)+\mathcal{I}(\sigma')(A_{k'})}\,.$$

If $G_{k'}(\text{LAST}(\sigma)) \neq \emptyset$, we have

$$\begin{aligned}
&\frac{\mathcal{I}(\sigma)(A_k)}{\mathcal{I}(\sigma)(A_k)+\mathcal{I}(\sigma)(A_{k'})}\\
&= \frac{1/\left|\{A_m \mid G_m(\text{LAST}(\sigma)) \neq \emptyset\}\right|}{1/\left|\{A_m \mid G_m(\text{LAST}(\sigma)) \neq \emptyset\}\right| + 1/\left|\{A_m \mid G_m(\text{LAST}(\sigma)) \neq \emptyset\}\right|}\\
&= \frac{1}{2}\\
&= \frac{\mathcal{I}(\sigma')(A_k)}{\mathcal{I}(\sigma')(A_k)+\mathcal{I}(\sigma')(A_{k'})}
\end{aligned}$$

□

THEOREM 2.5. *For all systems* P, *projections* $[\cdot]$, *there exist a non-randomized scheduler* $\eta \in \text{SDIST}_P([\cdot])P$.

Proof. Suppose that $\text{ATOMS}(P) = \{A_1, \cdots, A_N\}$. By Theorem 1.1, there exist input and output schedulers under $[\cdot]$. Hence, it suffices to construct an interleaving scheduler complying with Eqn. (2.3). Let $\mathcal{I}(\sigma) = 1{:}A_j$ where $A_j = \min\{A_i \mid G_i(\text{LAST}(\sigma)) \neq \emptyset\}$. Suppose, towards a contradiction, that Eqn. (2.3) does not hold. Then, since the scheduler is non-randomized, we have $\mathcal{I}(\sigma)(A_i) = 1$, $\mathcal{I}(\sigma')(A_j) = 1$ for some A_i, A_j such that $A_i \neq A_j$, $[\sigma]_i = [\sigma']_i$ and $[\sigma]_j = [\sigma']_j$. W.l.o.g., we can assume $i < j$. Since $[\sigma]_i = [\sigma']_i$, Eqn. (1.9) ensures $G_i(\text{LAST}(\sigma')) \neq \emptyset$. Then, $A_i \in \{A_k \mid G_k(\text{LAST}(\sigma')) \neq \emptyset\}$. Hence, $\mathcal{I}(\sigma') = 1{:}A_k$ for some $k \leqslant i$, thus contradicting $\mathcal{I}(\sigma') = 1{:}A_j$ with $j > i$. □

2.2 RATE SCHEDULERS

In this section, we propose another restriction on interleaving schedulers. The restriction we impose is based on a mechanism to resolve interleaving choices for probabilistic I/O automata.

In [147], interleaving choices are resolved in a distributed fashion: the model assigns a *rate*[†] $\text{RATE}(s_i)$ to each state s_i in each atom A_i. Using such rate, the choice among all the entities that are able to perform a transition is transformed into a probabilistic choice as follows: when an atom arrives in state s, it draws a random delay time from an exponential distribution with parameter $\text{RATE}(s)$ (i.e. an exponential distribution whose mean is $1/\text{RATE}(s)$). This delay describes the length of time the atom will remain in state s before executing a transition. So, the atom having the least delay time is the next one to perform a transition. In [147] it is explained that, according to this interpretation of the rate, a definite probability can be assigned to the event in which a given atom is the next one to perform a transition in a given state. If each atom A_i is in state s_i, the probability that A_j is the next atom

[†]Rates are called *delays* in the paper introducing these automata [147]. However, we prefer the term *rate* used in subsequent works [138, 137, 139, 140].

to perform a transition is $\text{RATE}(s_j) \,/\, \sum_i \text{RATE}(s_i)$. The same mechanism is used also in [138, 137, 139, 140]. As a simple example, if atom A_1 (A_2, resp.) is in state s_1 (s_2, resp.) and $\text{RATE}(s_1) = 1$ and $\text{RATE}(s_2) = 2$, the probability that A_1 executes first is $1/(1+2)$ while the probability that A_2 executes first is $2/(1+2)$. Note that the rate of s_2 is twice the rate of s_1, and this is reflected in the probabilities. In general, the rate can be seen as a "likeliness factor" that indicates how likely is an atom to execute with respect to another.

We can straightforwardly adapt this mechanism to our nondeterminism setting. Since we consider history dependent schedulers, it is natural to consider that the rate is a function of the history (instead of the state). Then, a *rate scheduler* for atom A_i is a function $\text{RATE}_i : \text{LOCALPATHS}_i^{[\cdot]} \to \mathbb{R}_{\geqslant 0}$ such that

Rate scheduler

$$\text{RATE}_i(\sigma_i) = 0 \iff G_i(\text{LAST}(\sigma_i)) = \emptyset \tag{2.7}$$

(this condition is inherited from [147]). For a path σ, the probability that A_i is the next atom to execute can be calculated according to the interpretation given before as:

$$\frac{\text{RATE}_i([\sigma]_i)}{\sum_j \text{RATE}_j([\sigma]_j)} .$$

We then restrict to the set of interleaving schedulers that can be obtained using this mechanism. Later on, we will prove that these schedulers are strongly distributed, thus justifying the notation introduced in the following definition.

DEFINITION 2.2. An interleaving scheduler $\mathcal{I}$ is rate-based for projection $[\cdot]$ iff there exist rate schedulers $\{\text{RATE}_i\}_{i=1}^{N}$ such that

Rate-based interleaving scheduler

$$\mathcal{I}(\sigma)(A_i) = \frac{\text{RATE}_i([\sigma]_i)}{\sum_j \text{RATE}_j([\sigma]_j)} .$$

A scheduler η is rate-based if the interleaving scheduler the defines η is rate-based. We denote by $\text{SDIST}_P([\cdot], \text{RATE})$ the set of rate-based schedulers for P.

Rate-based scheduler

THEOREM 2.6.

$$\text{SDIST}_P([\cdot], \text{RATE}) \subseteq \text{SDIST}_P([\cdot])$$

Proof. Let σ, σ' be such that $[\sigma]_A = [\sigma']_A$ and $[\sigma]_B = [\sigma']_B$. In the following calculation, $\mathcal{I}$ is the interleaving scheduler that defines $\eta \in \text{SDIST}_P([\cdot], \text{RATE})$ and RATE_i are the rate schedulers that define $\mathcal{I}$.

$$\begin{aligned}
&\frac{\mathcal{I}(\sigma)(A)}{\mathcal{I}(\sigma)(A) + \mathcal{I}(\sigma)(B)} \\
=&\frac{\frac{\text{RATE}_A([\sigma]_A)}{\sum_j \text{RATE}_j([\sigma]_j)}}{\frac{\text{RATE}_A([\sigma]_A)}{\sum_j \text{RATE}_j([\sigma]_j)} + \frac{\text{RATE}_B([\sigma]_B)}{\sum_j \text{RATE}_j([\sigma]_j)}} \\
=&\frac{\frac{\text{RATE}_A([\sigma]_A)}{\sum_j \text{RATE}_j([\sigma]_j)}}{\frac{\text{RATE}_A([\sigma]_A) + \text{RATE}_B([\sigma]_B)}{\sum_j \text{RATE}_j([\sigma]_j)}} \\
=&\frac{\text{RATE}_A([\sigma]_A)}{\text{RATE}_A([\sigma]_A) + \text{RATE}_A([\sigma]_B)}
\end{aligned}$$

$$
\begin{aligned}
&= \frac{\textsc{Rate}_A([\sigma']_A)}{\textsc{Rate}_A([\sigma']_A) + \textsc{Rate}_B([\sigma']_B)} \\
&= \frac{\frac{\textsc{Rate}_A([\sigma']_A)}{\sum_j \textsc{Rate}_j([\sigma']_j)}}{\frac{\textsc{Rate}_A([\sigma']_A)}{\sum_j \textsc{Rate}_j([\sigma']_j)} + \frac{\textsc{Rate}_B([\sigma']_B)}{\sum_j \textsc{Rate}_j([\sigma']_j)}} \\
&= \frac{\mathcal{I}(\sigma')(A)}{\mathcal{I}(\sigma')(A) + \mathcal{I}(\sigma')(B)}
\end{aligned}
$$

Hence, $\mathcal{I}$ complies with Eqn. (2.3), and so $\eta \in \textsc{SDist}_P([\cdot])$. □

In general, the inclusion in the previous theorem is strict. In fact, consider a system having a path σ such that $G_i(\textsc{Last}(\sigma)) \neq \emptyset$ and $G_j(\textsc{Last}(\sigma)) \neq \emptyset$. By (2.7), we have $\mathcal{I}(\sigma)(A_i) > 0$, $\mathcal{I}(\sigma)(A_j) > 0$ for all rate-based interleaving schedulers. Then, for all systems in which two atoms are enabled at the end of some path, we have that all rate-based schedulers are randomized. By Theorem 2.5 there is at least one non-randomized strongly distributed scheduler, and hence

$$\textsc{SDist}_P([\cdot], \textsc{Rate}) \subsetneq \textsc{SDist}_P([\cdot]) \,. \tag{2.8}$$

We finish this section by proving the analogous of Theorem 1.2.

THEOREM 2.7.

$$[\cdot] \sqsubseteq [\cdot]' \implies \textsc{SDist}_P([\cdot], \textsc{Rate}) \subseteq \textsc{SDist}_P([\cdot]', \textsc{Rate})$$

Proof. Given $\eta \in \textsc{SDist}_P([\cdot], \textsc{Rate})$, we prove that $\eta \in \textsc{SDist}_P([\cdot]', \textsc{Rate})$. It suffices to find rate schedulers $\textsc{Rate}'_i \colon \textsc{LocalPaths}_i^{[\cdot]'} \to \mathbb{R}_{\geqslant 0}$ such that

$$\mathcal{I}(\sigma)(A_j) = \textsc{Rate}'_j([\sigma]'_j) / \sum_i \textsc{Rate}'_i([\sigma]'_j) \,.$$

We construct such $\textsc{Rate}'_i$ using the schedulers $\textsc{Rate}_i \colon \textsc{LocalPaths}_i^{[\cdot]} \to \mathbb{R}_{\geqslant 0}$ whose existence is ensured by $\eta \in \textsc{SDist}_P([\cdot], \textsc{Rate})$.

In order to define $\textsc{Rate}'_i$ in terms of $\textsc{Rate}_i$, for all $\sigma'_i \in \textsc{LocalPaths}_i^{[\cdot]'}$ let $h(\sigma'_i)$ be a global path such that $[h(\sigma'_i)]'_i = \sigma'_i$. Then, we can define:

$$\textsc{Rate}'_i(\sigma'_i) = \textsc{Rate}_i(\left[h(\sigma'_i)\right]_i) \,.$$

We check that these rate schedulers do indeed generate $\mathcal{I}$. Let $\mathcal{I}'$ be the rate-based scheduler generated by $\textsc{Rate}'_i$. We prove that $\mathcal{I}' = \mathcal{I}$ by using the following property of h:

$$\forall A_i \,:\, [h([\sigma]'_i)]'_i = [\sigma]'_i \,.$$

This property holds by definition of h and (since $[\cdot] \sqsubseteq [\cdot]'$) implies

$$\forall A_i \,:\, \left[h([\sigma]'_i)\right]_i = [\sigma]_i \,. \tag{2.9}$$

Now we are able to prove $\mathcal{I}' = \mathcal{I}$. For all σ, A_j,

$$
\begin{aligned}
& \mathcal{I}'(\sigma)(A_j) \\
= & \ \{ \text{ Definition of rate-based scheduler } \} \\
& \mathrm{RATE}'_j([\sigma]'_j)(A_j) \,/\, \sum_i \mathrm{RATE}'_i([\sigma]'_i)(A_i) \\
= & \ \{ \text{ Definition of } \mathrm{RATE}'_i \ \} \\
& \mathrm{RATE}_j(\left[h([\sigma]'_j)\right]_j)(A_j) \,/\, \sum_i \mathrm{RATE}_i(\left[h([\sigma]'_i)\right]_i)(A_i) \\
= & \ \{ \text{ Equation (2.9) } \} \\
& \mathrm{RATE}_j([\sigma]_j)(A_j) \,/\, \sum_i \mathrm{RATE}_i([\sigma]_i)(A_i) \\
= & \ \{ \text{ Definition of rate-based scheduler } \} \\
& \mathcal{I}(\sigma)(A_j)
\end{aligned}
$$

□

2.3 TOTAL ORDER-BASED SCHEDULERS

In this section, we introduce yet another mechanism to resolve interleaving choices. This mechanism resembles the oblivious schedulers for task PIOA [34]. Each oblivious scheduler is simply a sequence of *tasks*: once the scheduler has been fixed, the same sequence is applied regardless of the probabilistic outcomes of the actual execution.

We adapt this mechanism so that the order in which the interleaving of two atoms is resolved depends on the local path observed by these atoms. The intuitive idea is that, whenever atom A has observed σ_A and atom B has observed σ_B, a given scheduler η always schedules A before B (or it always schedules B before A), regardless of the information not present in σ_A or σ_B.

Total order-based interleaving scheduler

DEFINITION 2.3. An interleaving scheduler $\mathcal{I}$ is total order-based with projection $[\cdot]$ iff there exists a total order $\leqslant^{\mathcal{I}}$ on $\bigcup_i \mathrm{LOCALPATHS}_i^{[\cdot]}$ such that, for all σ, we have $\mathcal{I}(\sigma) = 1{:}A_i$, where $A_i = \arg\min^{\leqslant^{\mathcal{I}}}_{A_i}[\sigma]_i$.

Total order-based scheduler

A scheduler η is total order-based iff the interleaving scheduler that defines η is total order-based. We denote by $\mathrm{SDIST}_P([\cdot], \leqslant)$ the set of total order-based schedulers for P.

THEOREM 2.8.

$$\mathrm{SDIST}_P([\cdot], \leqslant) \subseteq \mathrm{SDIST}_P([\cdot])$$

Proof. Let σ, σ' be two paths such $[\sigma]_A = [\sigma']_A$ and $[\sigma]_B = [\sigma']_B$ and $\mathcal{I}$ be a total order-based scheduler. If $\mathcal{I}(\sigma) = 1{:}C$, where $C \neq A$ and $C \neq B$, then $\mathcal{I}(\sigma)(A) = \mathcal{I}(\sigma)(B) = 0$ and so Eqn. (2.3) is not required to hold for σ, σ'. The same argument can be applied in case $\mathcal{I}(\sigma') = 1{:}C$.

If $\mathcal{I}(\sigma) = 1{:}A$, then $[\sigma]_A \leqslant^{\mathcal{I}} [\sigma]_B$. Then, either $\mathcal{I}(\sigma') = 1{:}C$ (with $C \neq A$ and $C \neq B$) or $\mathcal{I}(\sigma') = 1{:}A$. In the latter case:

$$\frac{\mathcal{I}(\sigma)(A)}{\mathcal{I}(\sigma)(A) + \mathcal{I}(\sigma)(B)} = 1 = \frac{\mathcal{I}(\sigma')(A)}{\mathcal{I}(\sigma')(A) + \mathcal{I}(\sigma')(B)} \,.$$

The case in which $\mathcal{I}(\sigma) = 1{:}B$ is exactly the same. □

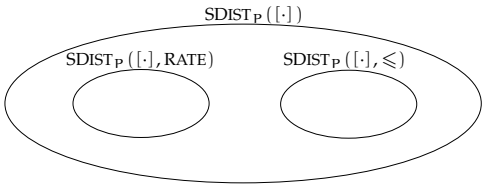


Figure 2.4: Inclusion relations among schedulers with restricted interleaving

As in the case of rate-based schedulers, the inclusion is strict. In fact, all total order-based interleaving schedulers are non-randomized and Theorem 2.4 ensures the existence of a randomized scheduler, and so

$$\text{SDIST}_{\text{P}}([\cdot], \leqslant) \subsetneq \text{SDIST}_{\text{P}}([\cdot]) \tag{2.10}$$

for all systems with interleaving nondeterminism.

Moreover, since in these systems rate-based schedulers are randomized, we have

$$\text{SDIST}_{\text{P}}([\cdot], \text{RATE}) \cap \text{SDIST}_{\text{P}}([\cdot], \leqslant) = \emptyset\ . \tag{2.11}$$

Figure 2.4 summarizes the inclusion relations among the sets of schedulers presented in this chapter. We construct this figure from (2.8), (2.10), (2.11) and Theorems 2.6, 2.8.

2.4 COMPARISON WITH EXISTING APPROACHES

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 [63], the components are not specified explicitly (then, there are no interleaving issues). In [64] a step of the whole system is obtained by taking a step in every component (and so no interleaving is needed). The main difference between our framework and the Switched PIOA framework in [46] is the concept of interleaving scheduler. In contrast, in the framework presented in [46] 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 [45] 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 the arbiter schema, the information used to choose the next component can be restricted by specifying that some information is not available to the arbiter. Although this approach is useful to specify that some information is not used at all when resolving the interleaving, such an 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.

3

LIMIT SCHEDULERS

"El número de todos los átomos que componen el universo es, aunque desmesurado, finito, y sólo capaz como tal de un número finito (aunque desmesurado también) de permutaciones. En un tiempo infinito, el número de las permutaciones posibles debe ser alcanzado, y el universo tiene que repetirse. De nuevo nacerás de un vientre, de nuevo crecerá tu esqueleto, de nuevo arribará esta misma página a tus manos iguales, de nuevo cursarás todas las horas hasta la de tu muerte increíble."

Jorge Luis Borges. Historia de la eternidad, *La doctrina de los ciclos*

Several proofs in this thesis involve transformations on schedulers. For instance, in order to show that there exists a scheduler complying with a certain property P, one may prove that every scheduler η can be transformed into a scheduler η^* complying with the property. Quite often, these transformations proceed in a step-wise fashion: the n-th step changes the choices of the scheduler for a finite set of paths, and ensures that the resulting scheduler complies with a certain property P^n. The schedulers resulting from each of the steps form a (possibly infinite) sequence $\{\eta^n\}$, the scheduler η^* yielded by the transformation is constructed using the schedulers in this sequence, and the validity of P is derived from the validity of the properties P^n.

In this chapter, we introduce the notion of *limit scheduler*. A sequence of schedulers may have several limits, and certain properties for the limits are implied by properties of the sequence's schedulers. Hence, in step-wise transformations the resulting scheduler η^* can be taken to be a limit, and the properties for η^* can be deduced from the properties for the schedulers η^n.

We introduce a simple finiteness condition ensuring the existence of a limit. Moreover, we give a sufficient condition to characterize the set of schedulers $\mathbb{S}$ that are *closed under limits*, in the sense that, if the schedulers of the sequence are in $\mathbb{S}$, then every limit is also in $\mathbb{S}$.

3.1 LIMIT SCHEDULERS

In order to express the choices of a scheduler $\eta = (\mathcal{I}, \{\Theta_i\}_{i=1}^{\mathcal{N}}, \{\Upsilon_i\}_{i=1}^{\mathcal{N}}) \in \text{SCHED}_P$ in a more succinct way, we define the function $\eta[\cdot]$. For a given path σ, this function returns a tuple containing all the resolutions of nondeterminism after σ according to η. In other words, this tuple comprises the choice of the interleaving schedulers, the choices of the atoms having generative transitions enabled, and the choices of all input schedulers for each one of the labels in the alphabet of the corresponding atom.

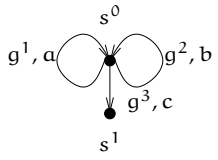


Figure 3.1: A simple example to illustrate limits

Formally:

$$\eta[\sigma] = (\, \mathcal{I}(\sigma)\,,\, \Theta_{i_1}(\sigma), \cdots \Theta_{i_n}(\sigma)\,,\, \Upsilon_1(\sigma, a^1_1), \ldots, \Upsilon_1(\sigma, a^1_{m_1}), \cdots,\, \Upsilon_{\mathcal{N}}(\sigma, a^{\mathcal{N}}_1), \ldots, \Upsilon_{\mathcal{N}}(\sigma, a^{\mathcal{N}}_{m_{\mathcal{N}}})\,)$$

if there exists A_j such that $G_j(\text{LAST}(\sigma)) \neq \emptyset$.

In this definition, $\{A_{i_1}, \cdots, A_{i_n}\}$ is the set of atoms such that $G_{i_k}(\text{LAST}(\sigma)) \neq \emptyset$. In addition, $\{a^i_1, \cdots, a^i_{m_i}\} = \text{ACTLAB}_i$ for all $A_i \in \text{ATOMS}(P)$.

If there exists no A_k such that $G_k(\text{LAST}(\sigma)) \neq \emptyset$, let

$$\eta[\sigma] = \varsigma\ .$$

(Recall we also denote by ς the compound transition that the system executes in case there is no atom with generative transitions enabled, see p. 19.)

The co-domain of $\eta[\cdot]$ is a bit difficult to express formally. We write it explicitly just because we give it a name to use it later.

$$\eta[\cdot] : \text{PATHS}(P) \to \bigcup_{S \subseteq \text{ATOMS}(P)} \Big(\prod_{A_i \in S} \text{PROB}(T_{G_i}) \times \prod_{A_i \in \text{ATOMS}(P)} \prod_{a \in \text{ACTLAB}_i} \text{PROB}(T_{R_i}) \Big) \cup \{\varsigma\}$$

(where T_{G_i} is the set of generative transitions of atom A_i and T_{R_i} is the set of reactive transitions).

$\mathbb{C}_P$ We write the co-domain of $\eta[\cdot]$ as $\mathbb{C}_P$.

DEFINITION 3.1. Given a sequence $\mathcal{E} = \{\eta_k\}_{k=1}^{\infty}$, the scheduler η^* is called a *limit* of the sequence if, for every finite set S of finite paths, there exists a subsequence $\mathcal{E}' = \{\eta_{k_j}\}_{j=1}^{\infty}$ complying

$$\forall \sigma \in S\ :\ \text{PR}^{\eta^*}((\sigma)^\uparrow) > 0 \implies \forall j\ :\ \eta^*[\sigma] = \eta_{k_j}[\sigma]\ .$$

Among other issues, the example below clarifies why we require the equality only in case $\text{PR}^{\eta^*}((\sigma)^\uparrow) > 0$.

EXAMPLE 3.1. Consider the system whose sole atom is depicted in Fig. 3.1. Consider the sequence of schedulers $\mathcal{E} = \{\eta^n\}_{n=1}^{\infty}$ whose output schedulers are defined as

$$\Theta^n(\overbrace{\text{INIT}.a.\cdots.a.\text{INIT}}^{n \text{ times}})(g^1) = 1$$

for all $k \leqslant n$ and

$$\Theta^n(\overbrace{\text{INIT}.a.\cdots.a.\text{INIT}}^{n \text{ times}})(g^3) = 1\ .$$

In words, Θ^n delays the execution of g^3 for n steps.

Recall that, by Def. 1.9, for the paths σ such that $\textsc{last}(\sigma) = s^1$ we have $\eta(\sigma)(\varsigma) = 1$ for all η, regardless of the value of $\Theta([\sigma]_i)$ (we already faced this property in Example 1.4). Hence, η^n delays the execution of g^3 for n steps; then, it chooses g^3, and it keeps choosing ς in all paths of the form

$$\overbrace{\textsc{init}.(g^1,a).\cdots.\textsc{init}}^{n \text{ times}}.(g^3,c).s^1.\varsigma.\cdots.s^1 \ .$$

We show that the scheduler η^* choosing g^1 in all paths is a limit of $\mathcal{E}$. Given a finite set S of finite paths with positive probability under η^*, let $M_S = \max_{\sigma\in S} \textsc{len}(\sigma)$. In order to prove that η^* is a limit, we must find a subsequence of $\mathcal{E}$ such that $\eta^*[\sigma] = \eta^n[\sigma]$, for all η^n in the subsequence, For this particular system, this is equivalent to $\Theta^*(\sigma) = \Theta^n(\sigma)$ for all paths $\sigma \in S$. Note that, every path σ such that $\textsc{pr}^{\eta^*}((\sigma)^\uparrow) > 0$ has the form

$$\overbrace{\textsc{init}.(g^1,a).\cdots.\textsc{init}}^{k}$$

with $k \leqslant M_S$. Then, the subsequence we need to find comprises all the η^n such that $n > M_S$. In fact, for all $n > M_S$, we have

$$\Theta^*(\overbrace{\textsc{init}.(g^1,a).\cdots.\textsc{init}}^{k}) = \Theta^n(\overbrace{\textsc{init}.(g^1,a).\cdots.\textsc{init}}^{k}) = 1{:}g^1$$

for all $k \leqslant n$ and, in particular, for all $k \leqslant M_S$. In conclusion, for all S the subsequence $\{\eta^n\}_{n=M_S+1}^{\infty}$ complies with $\Theta^*(\sigma) = \Theta^n(\sigma)$ for all σ such that $\textsc{len}(\sigma) \leqslant M_S$ (in particular, for all $\sigma \in S$) and $\textsc{pr}^{\eta^*}((\sigma)^\uparrow) > 0$.

Note that $\textsc{pr}^{\eta^n}(\textsc{reach}(\{s^1\})) = 1$ for all n, while $\textsc{pr}^{\eta^*}(\textsc{reach}(\{s^1\})) = 0$.

We also use this example to show why the equality $\eta^*(\sigma) = \eta^n(\sigma)$ is required only for the paths σ such that $\textsc{pr}^{\eta^*}((\sigma)^\uparrow) > 0$. If the restriction were $\eta^*(\sigma) = \eta^n(\sigma)$ for all $\sigma \in S$, then S might contain the path

$$\textsc{init}.(g^1,a).\textsc{init}.(g^2,b).\textsc{init} \ .$$

Hence, whether η^* is a limit or not would depend on the values that the schedulers η^n assign to the path $\textsc{init}.(g^1,a).\textsc{init}.(g^2,b).\textsc{init}$, which has probability 0 in all of the schedulers η^n. This is clearly undesirable since the choices for the paths with probability 0 should be irrelevant.

In order to emphasize that our definition of limits only considers equalities, we present another example concerning the sequence $\mathcal{E}$ whose schedulers are defined by:

$$\eta^n(\textsc{init}) = \frac{1}{2^n}{:}g^1 + (1-\frac{1}{2^n}){:}g^2$$
$$\eta^n(\textsc{init}.(g^1,a).\textsc{init})(g^3) = 1$$
$$\eta^n(\textsc{init}.(g^2,b).\textsc{init})(g^3) = 1$$

The larger the n, the higher the probability that g^2 is chosen in $\textsc{init}$. However, the scheduler η^q defined by $\Theta^q(\textsc{init})(g^2) = 1$, $\Theta^q(\textsc{init}.(g^2,b).\textsc{init})(g^3) = 1$ is *not* a limit of $\mathcal{E}$. In fact, we can show that $\mathcal{E}$ has no limits: $\textsc{pr}^{\eta}((\textsc{init})^\uparrow) = 1 > 0$ for all η, and there are no two different schedulers $\eta^n, \eta^{n'} \in \mathcal{E}$ such that $\eta^n[\textsc{init}] = \eta^{n'}[\textsc{init}]$. So, for any S including the path $\textsc{init}$ we cannot have a subsequence (not even two schedulers) coinciding for all paths in S.

The following theorem gives a sufficient condition for the existence of a limit.

THEOREM 3.1. *Given a sequence of schedulers* $\mathcal{E} = \{\eta_k\}_{k=1}^{\infty}$*, if*

$$\forall \sigma : \{\eta_k[\sigma] \mid 1 \leqslant k \leqslant \infty\} \text{ is finite.}$$

then $\mathcal{E}$ *has at least one limit.*

The proof requires an ancillary lemma.

LEMMA 3.1. *Let* $\{\eta_k\}_{k=1}^{\infty}$ *be a sequence of schedulers such that,*

$$\forall \sigma : \{\eta_k[\sigma] \mid 1 \leqslant k \leqslant \infty\} \text{ is finite.}$$

Then, there exists a subsequence $\{\eta^N\}_{N=0}^{\infty}$ *of* $\{\eta_k\}_k$ *such that, for all* N*, there exists a sequence* $Z^N = \{Z_k^N\}_{k=1}^{\infty}$ *complying with:*

$$\eta^N[\sigma] = \eta_{Z_k^N}[\sigma] \tag{3.1}$$

for all k *and* σ *such that* $\text{LEN}(\sigma) \leqslant N$.

Proof. The scheduler η^0 is simply η_1. The sequence $Z^0 = \{Z_k^0\}_k$ is the sequence $\{k\}_{k=1}^{\infty}$. It trivially complies with (3.1), since there are no paths of length 0 (INIT has length 1).

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

We consider the sequence $\{\eta_{Z_k^{N,Q}}[\sigma^*]\}_{k=1}^{\infty}$. The hypothesis ensures that, in such sequence, at least one element d^* is repeated infinitely many times. Let I be first index such that $\eta_{Z_I^{N,Q}}[\sigma^*] = d^*$. We define $\eta^{N,Q\cup\{\sigma^*\}} = \eta_{Z_I^{N,Q}}$, and take $Z^{N,Q\cup\{\sigma^*\}}$ to be the infinite subsequence of $Z^{N,Q}$ complying with $\eta_{Z_k^{N,Q\cup\{\sigma^*\}}}[\sigma^*] = d^*$ for all $k \leqslant 1$ (this subsequence is ensured to exist since d^* appears infinitely many times in $\{\eta_{Z_k^{N,Q}}(\sigma^*)\}_{k=1}^{\infty}$). □

Proof (of Theorem 3.1). Consider the sequence $\{\eta^N\}_{N=0}^{\infty}$ whose existence is ensured by the previous lemma. We define η^* as

$$\eta^*[\sigma] = \eta^{\text{LEN}(\sigma)}[\sigma] \ .$$

We prove that η^* is a limit of the sequence. For each finite S, let $M_S = \max_{\sigma\in S} \text{LEN}(\sigma)$. Then, by Eqn. (3.1), we have $\eta^*[\sigma] = \eta_{Z_k^{M_S}}[\sigma]$ for all $\sigma \in S$, for all k. □

3.2 FINITELY FALSIFIABLE SETS AND CLOSURE UNDER LIMITS

"falsify: 1 : to prove or declare false"
Merriam-Webster dictionary

The goal of this subsection is to establish a sufficient condition ensuring that a set of schedulers $\mathbb{S} \subseteq \text{SCHED}_P$ is closed under limits. By closure under limits we mean that, if the elements of a sequence are drawn from a set $\mathbb{S}$, then the limits of the sequence (if any) are also in $\mathbb{S}$. We present a general theorem to ensure closure under limits. This theorem requires the set of schedulers to be *finitely falsifiable*.

DEFINITION 3.2. We say that a set of schedulers $\mathbb{S} \subseteq \text{SCHED}_P$ is finitely falsifiable iff there exists a predicate

$$F\colon \mathcal{P}(\text{PATHS}(P) \times \mathbb{C}_P) \to \{\text{TRUE}, \text{FALSE}\}$$

(recall that $\mathbb{C}_P$ is the range of $\eta[\cdot]$) such that

$$\eta \notin \mathbb{S} \iff \exists S \subseteq \{\sigma \mid \text{PR}^{\eta}((\sigma)^{\uparrow}) > 0\} : S \text{ is finite} \wedge F(\{(\sigma, \eta[\sigma]) \mid \sigma \in S\}) .$$

The set S is said to be a *witness*.

Intuitively, a set $\mathbb{S}$ is finitely falsifiable if the statement "η is a member of $\mathbb{S}$" can be falsified by looking at the choices of η for a finite number (namely, $|S|$) of paths.

EXAMPLE 3.2. Let $d \in \mathbb{C}_P$, $s \in S_P$, $n \in \mathbb{N}$. Consider the set $\mathbb{S}$ comprising all the schedulers η such that the number of paths ending in s in which c is chosen is at most n. We show that $\mathbb{S}$ is finitely falsifiable. First, we give an alternative definition of $\mathbb{S}$: $\eta \notin \mathbb{S}$ iff there exist paths $\{\sigma^1, \cdots, \sigma^{n+1}\}$ such that $\text{LAST}(\sigma^k) = s$ and $\eta[\sigma^k] = c$ for all $1 \leqslant k \leqslant n+1$. The predicate is thus defined by

$$\begin{aligned} &F(\{(\sigma^1, d^1), \cdots, (\sigma^q, d^q)\}) \\ &\qquad \iff \exists j_1, \cdots, j_{n+1} : \forall_{1 \leqslant k \leqslant n+1} : \text{LAST}(\sigma^{j_k}) = s \wedge d^{j_k}(c) > 0 . \end{aligned}$$

Note that the witness set S is only restricted to be finite, and so the predicate must be defined for sets with any cardinality q. Given a set $S = \{\sigma^1, \cdots, \sigma^q\}$, the specialization of F for $\{(\sigma, \eta(\sigma)) \mid \sigma \in S\}$ yields

$$\begin{aligned} &F(\{(\sigma^1, \eta[\sigma^1]), \cdots, (\sigma^q, \eta[\sigma^q])\}) \\ &\qquad \iff \exists j_1 < \cdots < j_{n+1} : \forall_{1 \leqslant k \leqslant n+1} : \text{LAST}(\sigma^{j_k}) = s \wedge \eta[\sigma^{j_k}] = c . \end{aligned}$$

If $\eta \notin \mathbb{S}$, then we have a witness set S with $n+1$ elements; in this case any set $S' \supseteq S$ with $|S'| = q$ is also a witness. If $\eta \in \mathbb{S}$, there are no witnesses of any cardinality.

The set $\mathbb{S}$ comprising all schedulers such that $\eta[\sigma] = c$ for some σ is *not* finitely falsifiable (except for the boring case in which the number of paths in the system P is finite). Intuitively, in order to see that a scheduler η does not belong to $\mathbb{S}$, we must test all the paths in the system, in order to check that η does not choose c. In Example 3.1 we have shown a sequence of schedulers such that all the schedulers choose the compound transition g^3 in some of the paths, while the limit does not. So, the set $\mathbb{S}$ is not closed under limits. This example shows that, in the following theorem, the hypothesis of finite falsifiability cannot be disregarded.

THEOREM 3.2. *If a set $\mathbb{S}$ is finitely falsifiable and $\mathcal{E} = \{\eta_i\}_{i=1}^{\infty}$ is a sequence of elements from $\mathbb{S}$, then the limits of $\mathcal{E}$ are also in $\mathbb{S}$.*

Proof. Suppose, towards a contradiction, that η is a limit and $\eta \notin \mathbb{S}$. Then, there exist a predicate F and a finite set S such that $F(\{(\sigma, \eta(\sigma)) \mid \sigma \in S\})$ holds. However, since η is a limit of $\mathcal{E}$, there exist infinitely many η_i such that $\eta_i[\sigma] = \eta[\sigma]$ for all $\sigma \in S$, and so

$$F(\{(\sigma, \eta[\sigma]) \mid \sigma \in S\}) = F(\{(\sigma, \eta_i[\sigma]) \mid \sigma \in S\}) = \text{TRUE}$$

for all such η_i, thus contradicting the fact that $\eta_i \in \mathbb{S}$. □

3.3 DISTRIBUTED SCHEDULERS ARE CLOSED UNDER LIMITS

In this section, we show that the different sets of distributed schedulers defined in previous chapters are finitely falsifiable and, hence, closed under limits.

THEOREM 3.3. *For all $[\cdot]$, the set $\text{DIST}_P([\cdot])$ is finitely falsifiable.*

Proof. Given $d^j \in \mathbb{C}_P$, we define I^j, O_i^j and $Y_{i,a}^j$ to be the probability distributions such that $d^j = (I^j, \{O_i^j\}_i, \{Y_{i,a}^j\}_{i,a})$. We show that it suffices to consider the predicate $F(\{(\sigma^1, d^1), \cdots, (\sigma^n, d^n)\})$ defined as

$$\exists 1 \leqslant j < k \leqslant n, A_i \in \text{ATOMS}(P), a \in \text{ACTLAB}_i : \\ [\sigma^j]_i = [\sigma^k]_i \wedge (O_i^j \neq O_i^k \vee Y_{i,a}^j \neq Y_{i,a}^k) \quad (3.2)$$

Suppose $\eta = (\mathcal{I}, \{\Theta_i\}_i, \{\Upsilon_i\}_i)$. Recall that, in Def. 3.2, the predicate F occurs evaluated in the set $\{(\sigma, \eta[\sigma]) \mid \sigma \in S\}$, and so $d^j = \eta[\sigma^j]$, $d^k = \eta[\sigma^k]$. Moreover, $O_i^j = \Theta_i(\sigma^j)$ and $Y_{i,a}^j = \Upsilon_i(\sigma^j, a)$. Hence, the predicate (3.2) is equivalent to:

$$\exists 1 \leqslant j < k \leqslant n, A_i \in \text{ATOMS}(P), a \in \text{ACTLAB}_i : \\ [\sigma^j]_i = [\sigma^k]_i \wedge (\Theta_i(\sigma^j) \neq \Theta_i(\sigma^k) \vee \Upsilon_i(\sigma^j, a) \neq \Upsilon_i(\sigma^k, a))$$

Replacing σ^j by σ and σ^k by σ', we get

$$\exists \sigma, \sigma', A_i, a \in \text{ACTLAB}_i : \\ [\sigma]_i = [\sigma']_i \wedge (\Theta_i(\sigma) \neq \Theta_i(\sigma') \vee \Upsilon_i(\sigma, a) \neq \Upsilon_i(\sigma', a)) \quad (3.3)$$

If no such σ, σ' exist, then (1.18) and (1.19) hold, thus implying $\eta \in \text{DIST}_P([\cdot])$. If there exist σ, σ' complying with (3.3), then (1.18) or (1.19) do not hold for η, and so $\eta \notin \text{DIST}_P([\cdot])$.

Therefore $\eta \in \text{DIST}_P([\cdot])$ iff (3.2) does not hold. □

We prove the same result for strongly distributed schedulers.

THEOREM 3.4. *For all $[\cdot]$, the set $\text{SDIST}_P([\cdot])$ is finitely falsifiable.*

Proof. A scheduler is strongly distributed iff the following predicate is false

$$\exists \sigma, \sigma', A_i, A_j : \frac{\mathcal{I}(\sigma)(A_i)}{\mathcal{I}(\sigma)(A_i) + \mathcal{I}(\sigma)(A_j)} \neq \frac{\mathcal{I}(\sigma')(A_i)}{\mathcal{I}(\sigma')(A_i) + \mathcal{I}(\sigma')(A_j)}$$

Similarly as in Theorem 3.3, this predicate can be derived from a predicate of the form $F(\{(\sigma^1, d^1), \cdots, (\sigma^n, d^n)\})$ where $d^k = \eta[\sigma^k]$. □

The last set of schedulers for which we prove this property is the set of total order-based schedulers in Sec. 2.3.

THEOREM 3.5. *The set* $\mathrm{SDIST}_{\mathrm{P}}([\cdot], \leqslant)$ *is finitely falsifiable.*

Proof. We prove that the following predicate is suitable to falsify a scheduler.

$$
\begin{aligned}
&\exists A_i, \sigma \in S : 0 < \mathcal{I}(\sigma)(A_i) < 1 \\
\vee\quad &\exists A_1, \cdots, A_n \in \mathrm{ATOMS}(P), \sigma^1, \cdots, \sigma^n \in S : \quad \mathcal{I}(\sigma^1) = 1{:}A_1 \\
&\qquad\qquad \wedge\ [\sigma^1]_2 = [\sigma^2]_2 \\
&\qquad\qquad \wedge\ \mathcal{I}(\sigma^2) = 1{:}A_2 \\
&\qquad\qquad \wedge\ [\sigma^2]_3 = [\sigma^3]_3 \\
&\qquad\qquad \wedge\ \mathcal{I}(\sigma^3) = 1{:}A_3 \\
&\qquad\qquad \wedge\ \cdots \\
&\qquad\qquad \wedge\ \mathcal{I}(\sigma^n) = 1{:}A_n \\
&\qquad\qquad \wedge\ [\sigma^n]_1 = [\sigma^1]_1 \\
&\qquad\qquad \wedge\ A_1 \neq A_n
\end{aligned}
\tag{3.4}
$$

Note that, if $0 < \mathcal{I}(\sigma)(A) < 1$, then the scheduler cannot be based on a total order. If there exists $\sigma^1, \cdots, \sigma^n$ as in the predicate suppose, towards a contradiction, that $\mathcal{I}$ is based on a total order $\leqslant$. By Def. 2.3,

$$\left(\mathcal{I}(\sigma^k) = 1{:}A_k \wedge \left[\sigma^k\right]_{k+1} = \left[\sigma^{k+1}\right]_{k+1}\right) \implies \left[\sigma^k\right]_k < \left[\sigma^{k+1}\right]_{k+1}$$

and

$$\left(\mathcal{I}(\sigma^n) = 1{:}A_n \wedge [\sigma^n]_1 = [\sigma^1]_1\right) \implies [\sigma^n]_n < \left[\sigma^1\right]_1$$

Hence

$$\left[\sigma^1\right]_1 < \left[\sigma^2\right]_2 < \cdots < [\sigma^n]_n < \left[\sigma^1\right]_1 .$$

This contradicts the assumption that $\leqslant$ is a total order. Therefore, if η is total order-based, then (3.4) does not hold.

In order to prove that η is total order-based whenever the predicate does not hold, we define a relation $R^{\leqslant}$. Then, under the assumption that (3.4) does not hold, we prove that $R^{\leqslant}$ is a total order on the local paths and $\mathcal{I}$ behaves according to $R^{\leqslant}$. We start the construction of $R^{\leqslant}$ with a relation R defined as follows: $\sigma_i \, R \, \sigma_j$ iff there exists σ such that $[\sigma]_i = \sigma_i$ and $[\sigma]_j = \sigma_j$ and $\mathcal{I}(\sigma) = 1{:}A_i$. Let R^+ be the transitive closure or R. In order to complete our definition, we also consider the equivalence closure of R, denoted by R^*. The relation R^* has (at most) countably many equivalence classes $\{C_k\}_{k=1}^{\infty}$. Note that, if $\sigma_i \not R^* \sigma_j$, then σ_i and σ_j are not related by R^+. We define

$$R^{\leqslant} = R^+ \cup \{(\sigma_i, \sigma'_{i'}) \mid \sigma_i \in C_k \wedge \sigma'_{i'} \in C_m \wedge k < m\} .$$

Since $k < m$, we have $C_k \neq C_m$, and so $\sigma_i \not R^* \sigma'_{i'}$, which implies $\sigma_i \not R^+ \sigma'_{i'}$. By construction, $R^{\leqslant}$ is total and transitive. To prove that $R^{\leqslant}$ is antisymmetric, suppose $\sigma_i \, R^{\leqslant} \, \sigma_j$ and $\sigma_j \, R^{\leqslant} \, \sigma_i$ since $R^{\leqslant}$ only adds pairs that are not related in R^+, we have $\sigma_i \, R^+ \, \sigma_j$ and $\sigma_j \, R^+ \, \sigma_i$. By definition of R^+, there exists $\sigma_{i_1}, \cdots, \sigma_{i_n}, \sigma_{i'_1}, \cdots, \sigma_{i'_n}$, such that

$$\sigma_i \, R \, \sigma_{i_1} \wedge \forall_k : \sigma_{i_k} \, R \, \sigma_{i_{k+1}} \wedge \sigma_{i_n} \, R \, \sigma_j$$

and

$$\sigma_j \, R \, \sigma_{i'_1} \wedge \forall_k : \sigma_{i'_k} \, R \, \sigma_{i'_{k+1}} \wedge \sigma_{i'_{n'}} \, R \, \sigma_i \, .$$

By definition of R, there exists

$$S_\sigma = \{\sigma^{i,i_1}, \sigma^{i_1,i_2}, \cdots, \sigma^{i_n,j}, \sigma^{j,i'_1}, \sigma^{i'_1,i'_2}, \cdots, \sigma^{i'_{n'},i}\}$$

such that, for all p, q, r, we have $\sigma_q = [\sigma^{p,q}]_q = [\sigma^{q,r}]_q$ and $\mathcal{I}(\sigma^{p,q}) = 1{:}A_p$. Then, the assumption that (3.4) does not hold yields $i'_{n'} = i$. Hence, $\sigma_i = [\sigma^{i,i'_{n'}}]_i = [\sigma^{i,i'_{n'}}]_{i'_{n'}} = \sigma_{i'_{n'}}$. Similarly, by using different permutations of S_σ (more particularly, by "rotating" S_σ), we can prove $\sigma_p = \sigma_q$ for all $\sigma^{p,q} \in S_\sigma$. By transitivity of $=$, we have $\sigma_i = \sigma_j$ as desired. Therefore,

$$\sigma_i \, R^{\leqslant} \, \sigma_j \wedge \sigma_j \, R^{\leqslant} \, \sigma_i \implies \sigma_i = \sigma_j \, ,$$

and so $R^{\leqslant}$ is antisymmetric.

Let $\sigma_i \leqslant \sigma_j$ iff $\sigma_i \, R^{\leqslant} \, \sigma_j$. If $\mathcal{I}(\sigma) = 1{:}A_i$, by definition of R we have $[\sigma]_i \, R \, [\sigma]_j$ for all $i \neq j$, and so $i = \arg\min^{\leqslant}[\sigma]_i$. Then, $\mathcal{I}$ is based on the total order $\leqslant$. □

By considering Theorems 3.3, 3.4 and 3.5 together with Theorem 3.2, we obtain the following corollary.

COROLLARY 3.1. *For all* $[\cdot]$, *the sets* $\text{DIST}_P([\cdot])$, $\text{SDIST}_P([\cdot])$ *and* $\text{SDIST}_P([\cdot], \leqslant)$ *are closed under limits.*

3.4 DISCUSSION AND FURTHER WORK

Along this chapter, we used the name *limit* without providing a justification for it. We consider that an interesting research direction is to study whether limit schedulers are actually limit points (also called accumulation points) under some metric. If so, in such space we have that finitely falsifiable are closed sets. This would allow us to use the existing machinery for metric spaces in order to explore further properties for sets of schedulers.

The argument used to show the existence of limits in Theorem 3.1 (namely, to obtain successive subsequences of the original one) is usual in the field of Petri nets. In fact, this argument is used to show the existence of *covering trees* [32] for these nets. In a more abstract setting, this argument is used to show the existence of increasing sequences in well-quasi-orderings [111] and, more generally, in Ramsey theory [85]. This suggests a connection between limit schedulers and order theory. In fact, the "limit construction" in [45, Sec. 4.3] is about partially defined schedulers, which are ordered in the sense that "the greater the scheduler, the more defined it is".

4

ON THE EXPRESSIVE POWER OF DIFFERENT CLASSES OF SCHEDULERS

In the verification of probabilistic systems, the key problem is to find the "worst-case probability", that is, the maximum probability that the system fails to correctly achieve its goal (or the minimum probability that it works properly) quantifying over all possible schedulers. For full-history dependent schedulers, it is well-known that the worst-case probability equals the maximum probability quantifying over non-randomized schedulers (introduced in Def. 1.8). This fact often comes in hand to simplify correctness proofs: we can prove that a value ν is greater than the probabilities yielded for all schedulers (that is, ν is an overestimation of the worst-case probability), by proving that ν is greater than the value yielded for all *non-randomized* schedulers. Since these schedulers can be thought as functions mapping histories to transitions, they happen to be technically easier to manipulate than the arbitrary schedulers, which map to probability distributions. Given two sets of schedulers $\mathbb{S} \subseteq \mathbb{S}'$, we say that a set $\mathbb{S}$ is (at least) *as expressive* as the set $\mathbb{S}'$ iff the worst-case probability over $\mathbb{S}$ coincides with the worst-case probability over $\mathbb{S}'$: intuitively, we want the schedulers in $\mathbb{S}$ to be "evil enough" so that they can make the system fail with the worst probability. Note that, because of the set inclusion, the worst-case probability over $\mathbb{S}$ cannot be worse than that over probability over $\mathbb{S}'$. In case $\mathbb{S}'$ is clear from the context (for instance, if we are focusing on the set schedulers $\mathbb{S}' = \textsc{dist}_{\mathrm{P}}(\llbracket \cdot \rrbracket)$), we simply say that the subset $\mathbb{S}$ is *sufficiently expressive*.

In this chapter, we consider subsets of distributed/strongly distributed schedulers in order to study whether they are sufficiently expressive. After considering non-randomized schedulers, we present *Markovian* schedulers, in which the choice of the scheduler is based solely on the actual state. The domain of these schedulers can be thought as the set of states of the system, instead of the set of paths. We generalize Markovian schedulers to N-Markovian schedulers, which choose according to the last N states in the history, thus "forgetting" the states traversed before the last N steps ago. Further generalization leads to the subset of finite memory schedulers, will comprises all N-Markovian schedulers for all N.

Some of the questions we address in this chapter are: "Does randomization add expressiveness to Markovian schedulers?", "Are schedulers with finite memory sufficiently expressive?", "Given a system, can we find a suitable N such that N-Markovian schedulers are sufficiently expressive?". Many of these questions are addressed only for the projection $\llbracket \cdot \rrbracket$, since it is the projection that has been used in previous works to model distributed systems. In fact, several of the results presented are of a "negative" nature, and we only aim to show that the natural projection $\llbracket \cdot \rrbracket$ yields different results to the ones obtained by assuming that all information is available.

We start considering how randomization affects the power of distributed schedulers. Then, we consider the restriction to finite memory.

4.1 NON-RANDOMIZED DISTRIBUTED SCHEDULERS

For the usual schedulers full-information schedulers, the worst-case probability that a failure state is reached is attained by the set of schedulers that are both Markovian and non-randomized [25]. Then, when dealing with full-information schedulers and reachability properties, the schedulers which are functions mapping states to transitions are sufficiently expressive.

Unfortunately, these results do not hold in general for distributed schedulers, since Markovian schedulers are not sufficiently expressive (not even for the usual projection $[\![\cdot]\!]$ and reachability properties), as we shall see in Sec. 4.4. Moreover, in the setting of *strongly distributed* schedulers under $[\![\cdot]\!]$, non-randomized schedulers are not sufficiently expressive, as we show later on in Subsection 4.2.1.

In contrast to these negative results, non-randomized *distributed* schedulers are sufficiently expressive for all projections (and, in particular, for $[\![\cdot]\!]$).

THEOREM 4.1. *For any set S of infinite traces, S being measurable, we have*

$$\sup_{\eta \in \mathrm{NRDIST_P}([\cdot])} \mathrm{PR}^{\eta}(S) = \sup_{\eta \in \mathrm{DIST_P}([\cdot])} \mathrm{PR}^{\eta}(S) \,.$$

COROLLARY 4.1.

$$\sup_{\eta \in \mathrm{NRDIST_P}([\![\cdot]\!])} \mathrm{PR}^{\eta}(S) = \sup_{\eta \in \mathrm{DIST_P}([\![\cdot]\!])} \mathrm{PR}^{\eta}(S) \,.$$

The proof of Theorem 4.1 is quite long and, in fact, the remaining of this section is devoted to it.

NOTATION 4.1. Given a non-randomized output scheduler Θ we write $\Theta(\sigma) = g$ to indicate that $\Theta(\sigma)(g) = 1$, and similarly for input and interleaving schedulers.

First, we need some elements from probability theory. These definitions and the proofs not given here can be found in [145].

DEFINITION 4.1. Given a set Σ, a semi-ring is a set $\mathcal{S} \subseteq \mathcal{P}(\Sigma)$ complying:

- $\emptyset \in \mathcal{S}$,
- $S, T \in \mathcal{S} \implies S \cap T \in \mathcal{S}$,
- $S, T \in \mathcal{S} \implies \exists n \geqslant 0, Q_i \in \mathcal{S} : S \setminus T = \biguplus_{i=1}^{n} Q_i$.

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

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

The ring $\mathcal{R}(\mathcal{S})$ generated by a semi-ring $\mathcal{S}$ is the smallest 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 $\biguplus_{i=1}^{n} S_i$ with $S_i \in \mathcal{S}$.

The set comprising all the sets $(\sigma)^\uparrow$ and the empty set is a semi-ring. In the following, we denote this semi-ring by $\mathcal{S}$. Since we only consider this semi-ring, we write $\mathcal{R}$ for $\mathcal{R}(\mathcal{S})$.

In the definition of PR^η (Definition 1.10), we mentioned that, by defining a probability measure on extension sets, the probability of any measurable set is uniquely defined. Next, we give an explicit definition of the probability of a measurable set using the elements in $\mathcal{R}$. Afterwards, we give a lemma with an alternative definition that relies directly on the elements in $\mathcal{S}$.

DEFINITION 4.2. An $\mathcal{R}$-cover of a set S is a set $T = \{T_i\}_{i=1}^\infty$ where $T_i \in \mathcal{R}$ and $S \subseteq \bigcup_{n=1}^\infty T_i$. Let $\mathcal{Z}(S)$ 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 $\mathcal{S}$ is defined as

The union $\bigcup_{n=1}^\infty T_i$ is not required to be disjoint (see [145, Exercise 19])

$$\inf_{\{T_i\}_{i=1}^\infty \in \mathcal{Z}(S)} \sum_{i=1}^{\infty} \text{PR}^\eta(T_i) \,.$$

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 4.1. *Let $\mathcal{C}^\omega$ be the set*

$$\{\{U_i\}_{i=1}^\infty \mid \forall i,j, i \neq j : U_i \in \mathcal{S} \wedge U_j \in \mathcal{S} \wedge U_i \cap U_j = \emptyset\} \,.$$

For every measurable set of infinite paths S, we have

$$\text{PR}^\eta(S) = \inf_{\{C \in \mathcal{C}^\omega \mid S \subseteq \biguplus_{U \in C} U\}} \sum_{U \in C} \text{PR}^\eta(U) \,.$$

Proof. Let $T = \{T_i\}_{i=0}^\infty$ be an $\mathcal{R}$-cover for S where each T_i is of the form $\biguplus_{k=0}^{n_i} T_k^i$ with $T_k^i \in \mathcal{S}$ (recall the definition of $\mathcal{R}(\mathcal{S})$). We define $C_T \in \mathcal{C}^\omega$ as follows: $U \in C_T$ iff $U = T_k^i$ for some i,k and there is no $T_{k'}^{i'}$ such that $T_k^i \subset T_{k'}^{i'}$. Since our semi-ring is the set of extension sets, in the construction of C_T we dropped the sets $T_k^i = (\sigma)^\uparrow$ such that there exists $T_{k'}^{i'} = (\sigma')^\uparrow$ with $\sigma' \sqsubset \sigma$.

Then, we have

$$\sum_{i=1}^{\infty} \text{PR}^\eta(T_i) = \sum_{i=1}^{\infty} \sum_{k=0}^{n_i} \text{PR}^\eta(T_k^i) \geqslant \sum_{U \in C_T} \text{PR}^\eta(U)$$

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

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

Let η be a scheduler, σ_{i^*} be a local path and $a \in \text{ACTLAB}_{i^*}$ such that $0 < \Upsilon_{i^*}(\sigma_{i^*}, a) < 1$ (where Υ_{i^*} is the input scheduler for atom A_{i^*} in η). Next, we show that we can transform η into another scheduler η' in which the resolution for the input a in σ_{i^*} is not randomized. Formally, the value of $\Upsilon'_{i^*}(\sigma_{i^*}, a)$ in the resulting scheduler η' is a Dirac distribution, while all the other resolutions of nondeterministic choices in η' remain the same as in η. Moreover, given a set of the form $S = \biguplus_{m=1}^M (\sigma^m)^\uparrow$, we can find a reactive transition r^* in such a way that, by defining $\Upsilon'(\sigma_{i^*}, a)(r^*) = 1$, the probability of S under the resulting scheduler η' is not greater than the probability under η.

DEFINITION 4.3. Given $\eta = (\mathcal{I}, \{\Theta_1, \cdots, \Theta_N\}, \{\Upsilon_1, \cdots, \Upsilon_N\}) \in \text{DIST}_P([\cdot])$, the scheduler $\text{NR}(\eta, \sigma_{i^*}, a, r^*)$ is defined as

$$(\mathcal{I}, \{\Theta_1, \cdots, \Theta_N\}, \{\Upsilon_1, \cdots, \Upsilon'_{i^*}, \cdots, \Upsilon_N\})$$

where $\Upsilon'_{i^*}(\sigma_{i^*}, a)(r^*) = 1$ and $\Upsilon'_{i^*}(\sigma'_{i^*}, a') = \Upsilon_{i^*}(\sigma'_{i^*}, a')$ for all $(\sigma'_{i^*}, a') \neq (\sigma_{i^*}, a)$. Note that, for all r^*, $\text{NR}(\eta, \sigma_{i^*}, a, r^*) \in \text{DIST}_P([\cdot])$.

LEMMA 4.2. *For all* $\eta \in \text{DIST}_P([\cdot])$, $\sigma_{i^*}, a \in \text{ACTLAB}_{i^*}$ *such that* $0 < \Upsilon_{i^*}(\sigma_{i^*}, a) < 1$ *(where* Υ_{i^*} *is the input scheduler for atom* A_{i^*} *in* η*), for all* $S = \biguplus_{m=1}^{M} (\sigma^m)^\uparrow$, *there exists* r^* *such that* $\text{NR}(\eta, \sigma_{i^*}, a, r^*)$ *complies with*

$$\text{PR}^{\text{NR}(\eta, \sigma_{i^*}, a, r^*)}(S) \leqslant \text{PR}^{\eta}(S) \, .$$

Proof. Let T be the set of all the paths in $\{\sigma^m\}_{m=1}^{M}$ such that a occurs in σ_{i^*}, that is, $T \subseteq \{\sigma^m\}_{m=1}^{M}$ and, if there exists $k_\sigma < \text{LEN}(\sigma)$ such that $[\sigma_{\downarrow k_\sigma}]_{A_{i^*}} = \sigma_{i^*}$ and $\text{LABEL}(\sigma\langle k_\sigma \rangle) = a$, then $\sigma \in T$. W.l.o.g., let k_σ be the least such number. The probabilities of the paths in T are the only ones that change (since η and $\text{NR}(\eta, \sigma_{i^*}, a, r^*)$ differ only in the resolution of the input a in σ_{i^*}). Hence, the core of the proof is to find r^* such that:

$$\text{PR}^{\text{NR}(\eta, \sigma_{i^*}, a, r^*)}\Big(\biguplus_{\sigma \in T} (\sigma)^\uparrow\Big) \leqslant \text{PR}^{\eta}\Big(\biguplus_{\sigma \in T} (\sigma)^\uparrow\Big) \, . \tag{4.1}$$

Once we have been able to find such an r^*, the lemma follows from the following calculation:

$$\begin{aligned}
&\text{PR}^{\eta}(S) \\
&= \text{PR}^{\eta}\Big(\biguplus_{\sigma \in T} (\sigma)^\uparrow\Big) + \text{PR}^{\eta}\Big(\biguplus_{\sigma \in \{\sigma^m\}_{m=1}^{M} \setminus T} (\sigma)^\uparrow\Big) \\
&= \text{PR}^{\eta}\Big(\biguplus_{\sigma \in T} (\sigma)^\uparrow\Big) + \text{PR}^{\text{NR}(\eta, \sigma_{i^*}, a, r^*)}\Big(\biguplus_{\sigma \in \{\sigma^m\}_{m=1}^{M} \setminus T} (\sigma)^\uparrow\Big) \\
&\geqslant \text{PR}^{\text{NR}(\eta, \sigma_{i^*}, a, r^*)}\Big(\biguplus_{\sigma \in T} (\sigma)^\uparrow\Big) + \text{PR}^{\text{NR}(\eta, \sigma_{i^*}, a, r^*)}\Big(\biguplus_{\sigma \in \{\sigma^m\}_{m=1}^{M} \setminus T} (\sigma)^\uparrow\Big) \\
&= \text{PR}^{\text{NR}(\eta, \sigma_{i^*}, a, r^*)}(S)
\end{aligned}$$

In the following, we find r^* such that Eqn. (4.1) holds. Given $\sigma \in T$, let A_{g_σ} be $\text{ACTIVE}(\sigma\langle k_\sigma \rangle)$, g^σ be the corresponding generative transition, and for all A_j such that $a \in \text{ACTLAB}_j$ let $s_{\sigma,j} = \pi_j(\sigma(k_\sigma))$, $s'_{\sigma,j} = \pi_j(\sigma(k_\sigma + 1))$ For all $A_j \in \text{REACTIVE}(\sigma\langle k_\sigma \rangle)$, 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 $\Upsilon_i(\sigma_{i^*}, a)$. The following calculation helps us to isolate $\Upsilon_i(\sigma_{i^*}, a)$ from the rest of the factors in the probability of a given path $\sigma \in T$.

$$\begin{aligned}
&\text{PR}^{\eta}((\sigma)^\uparrow) \\
= \;& \text{PR}^{\eta}((\sigma_{\downarrow k_\sigma})^\uparrow) \\
& \cdot \eta(\sigma_{\downarrow k_\sigma})(\overbrace{\sigma\langle k_\sigma \rangle}^{c}) \cdot c\,(\,\sigma(k_\sigma), \sigma(k_\sigma + 1)\,) \\
& \cdot \prod_{t=k_\sigma+1}^{\text{LEN}(\sigma)-1} \eta(\sigma_{\downarrow t})(\sigma\langle t \rangle) \cdot \sigma\langle t \rangle(\sigma(t), \sigma(t+1))
\end{aligned}$$

$$
\begin{aligned}
= \;& \mathrm{PR}^{\eta}((\sigma_{\downarrow k_\sigma})^{\uparrow}) \\
& \cdot \mathcal{I}(\sigma_{\downarrow k_\sigma})(A_{g_\sigma}) \cdot \Theta_{g_\sigma}([\sigma_{\downarrow k_\sigma}]_{g_\sigma})(g^\sigma) \\
& \cdot \prod_{A_j \in \mathrm{REACTIVE}(\sigma\langle k_\sigma\rangle)} \Upsilon_j([\sigma_{\downarrow k_\sigma}]_j, a)(r_j^\sigma) \\
& \cdot g^\sigma(s_{\sigma,g_\sigma}, a, s'_{\sigma,g_\sigma}) \\
& \cdot \prod_{A_j \in \mathrm{REACTIVE}(\sigma\langle k_\sigma\rangle)} r_j^\sigma(s_{\sigma,j}, a, s'_{\sigma,j}) \\
& \cdot \prod_{t=k_\sigma+1}^{\mathrm{LEN}(\sigma)-1} \eta(\sigma_{\downarrow t})(\sigma\langle t\rangle) \cdot \sigma\langle t\rangle(\sigma(t), \sigma(t+1)) \\
= \;& \Upsilon_{i^*}([\sigma_{\downarrow k_\sigma}]_{i^*}, a)(r_{i^*}^\sigma) \cdot r_{i^*}^\sigma(s_{\sigma,i^*}, a, s'_{\sigma,i^*}) \\
& \cdot \mathrm{PR}^{\eta}((\sigma_{\downarrow k_\sigma})^{\uparrow}) \\
& \cdot \mathcal{I}(\sigma_{\downarrow k_\sigma})(A_{g_\sigma}) \cdot \Theta_{g_\sigma}([\sigma_{\downarrow k_\sigma}]_{g_\sigma})(g^\sigma) \\
& \cdot \prod_{A_j \in \mathrm{REACTIVE}(\sigma\langle k_\sigma\rangle)\setminus\{A_{i^*}\}} \Upsilon_j([\sigma_{\downarrow k_\sigma}]_j, a)(r_j^\sigma) \\
& \cdot g^\sigma(s_{\sigma,g_\sigma}, a, s'_{\sigma,g_\sigma}) \\
& \cdot \prod_{A_j \in \mathrm{REACTIVE}(\sigma\langle k_\sigma\rangle)\setminus\{A_{i^*}\}} r_j^\sigma(s_{\sigma,i^*}, a, s'_{\sigma,i^*}) \\
& \cdot \prod_{t=k_\sigma+1}^{\mathrm{LEN}(\sigma)-1} \eta(\sigma_{\downarrow t})(\sigma\langle t\rangle) \cdot \sigma\langle t\rangle(\sigma(t), \sigma(t+1))
\end{aligned}
$$

Hence, we have,

$$
\begin{aligned}
\forall \eta = (\mathcal{I}, \{\Theta_1, \cdots, \Theta_N\}, \{\Upsilon_1, \cdots, \Theta_N\}) : \forall \sigma \in T : \\
\mathrm{PR}^{\eta}((\sigma)^{\uparrow}) = \Upsilon_{i^*}(\sigma_{i^*}, a)(r_{i^*}^\sigma) \cdot r_{i^*}^\sigma(s_{\sigma,i^*}, a, s'_{\sigma,i^*}) \cdot Q_\sigma^\eta \quad , \quad (4.2)
\end{aligned}
$$

where

$$
\begin{aligned}
Q_\sigma^\eta = \;& \mathrm{PR}^{\eta}((\sigma_{\downarrow k_\sigma})^{\uparrow}) \\
& \cdot \mathcal{I}(\sigma_{\downarrow k_\sigma})(A_{g_\sigma}) \cdot \Theta_{g_\sigma}([\sigma_{\downarrow k_\sigma}]_{g_\sigma})(g^\sigma) \\
& \cdot \prod_{A_j \in \mathrm{REACTIVE}(\sigma\langle k_\sigma\rangle)\setminus\{A_{i^*}\}} \Upsilon_j([\sigma_{\downarrow k_\sigma}]_j, a)(r_j^\sigma) \\
& \cdot g^\sigma(s_{\sigma,g_\sigma}, a, s'_{\sigma,g_\sigma}) \\
& \cdot \prod_{A_j \in \mathrm{REACTIVE}(\sigma\langle k_\sigma\rangle)\setminus\{A_{i^*}\}} r_j^\sigma(s_{\sigma,i^*}, a, s'_{\sigma,i^*}) \\
& \cdot \prod_{t=k_\sigma+1}^{\mathrm{LEN}(\sigma)-1} \eta(\sigma_{\downarrow t})(\sigma\langle t\rangle) \cdot \sigma\langle t\rangle(\sigma(t), \sigma(t+1))
\end{aligned}
$$

Note that, although one occurrence of the factor $\Upsilon_{i^*}(\sigma_{i^*}, a)$ in $\mathrm{PR}^{\eta}((\sigma)^{\uparrow})$ has been left out of Q_σ^η, other occurrences of the factor might appear in Q_σ^η, since there might be several k' such that $[\sigma_{\downarrow k'}]_{i^*} = \sigma_{i^*}$ and $\mathrm{LABEL}(\sigma\langle k'\rangle) = a$. We use property (1.11) (see p. 29) to show that no $k' \neq k_\sigma$ exists. Since k_σ is

minimal, it should be $k' > k_\sigma$ for every such k', and so $\Upsilon_{i^*}(\sigma_{i^*}, a)$ might appear in

$$\prod_{t=k_\sigma+1}^{\text{LEN}(\sigma)-1} \eta(\sigma_{\downarrow_t})(\sigma\langle t\rangle) \cdot \sigma\langle t\rangle(\sigma(t), \sigma(t+1))$$

(more precisely, in $\eta(\sigma_{\downarrow_{k'}})(\sigma\langle k'\rangle)$). However, by property (1.11), it must be $[\sigma_{\downarrow_{k'}}]_{i^*} \neq [\sigma_{\downarrow_{k_\sigma}}]_{i^*}$: in effect, note that $\sigma_{\downarrow_{k'}}$ can be written as $\sigma_{\downarrow_{k_\sigma}} \cdot \sigma^q$ where σ^q is a path such that $\text{LABEL}(\sigma^q\langle 1\rangle) = a \in \text{ACTLAB}_{i^*}$. In conclusion, property (1.11) implies that the factor $\Upsilon_{i^*}(\sigma_{i^*}, a)$ cannot appear in Q^η_σ. Since η and $\text{NR}(\eta, \sigma_{i^*}, a, r)$ differ only with respect to the value of $\Upsilon_{i^*}(\sigma_{i^*}, a)$, we have

$$\forall \sigma \in T : Q^\eta_\sigma = Q^{\text{NR}(\eta,\sigma_{i^*},a,r)}_\sigma \,. \tag{4.3}$$

Now, we calculate:

$$\begin{aligned}
& \textstyle\sum_{\sigma\in T} \text{PR}^\eta((\sigma)^\uparrow) \\
= \;& \{\text{ Definition of probabilities for cylinders }\} \\
& \textstyle\sum_{\sigma\in T} \Upsilon_{i^*}(\sigma_{i^*}, a)(r^\sigma_{i^*})\; r^\sigma_{i^*}(s_{\sigma,i^*}, a, s'_{\sigma,i^*})\; Q^\eta_\sigma \\
= \;& \{\text{ Rearrange sums }\} \\
& \sum_{r_{i^*}, s_{i^*}, s'_{i^*}} \; \sum_{\{\sigma\in T \mid r^\sigma_{i^*}=r_{i^*} \,\wedge\, s_{\sigma,i^*}=s_{i^*} \,\wedge\, s'_{\sigma,i^*}=s'_{i^*}\}} \Upsilon_{i^*}(\sigma_{i^*}, a)(r_{i^*})\; r_{i^*}(s_{i^*}, a, s'_{i^*})\; Q^\eta_\sigma \\
= \;& \sum_{r_{i^*}} \Upsilon_{i^*}(\sigma_{i^*}, a)(r_{i^*}) \sum_{s_{i^*}, s'_{i^*}} \; \sum_{\{\sigma\in T \mid r^\sigma_{i^*}=r_{i^*} \,\wedge\, s_{\sigma,i^*}=s_{i^*} \,\wedge\, s'_{\sigma,i^*}=s'_{i^*}\}} r_{i^*}(s_{i^*}, a, s'_{i^*}) Q^\eta_\sigma
\end{aligned}$$

Consider the reactive transition r^* defined as:

$$r^* = \arg\min_{r_{i^*}} \sum_{s_{i^*}, s'_{i^*}} \; \sum_{\{\sigma\in T \mid r^\sigma_{i^*}=r_{i^*} \,\wedge\, s_{\sigma,i^*}=s_{i^*} \,\wedge\, s'_{\sigma,i^*}=s'_{i^*}\}} r_{i^*}(s_{i^*}, a, s'_{i^*}) Q^\eta_\sigma \,.$$

Next, we show that Eqn. (4.1) holds.

$$\begin{aligned}
& \sum_{\sigma\in T} \text{PR}^\eta((\sigma)^\uparrow) \\
=& \{\text{ Previous calculation }\} \\
& \sum_{r_{i^*}} \Upsilon_{i^*}(\sigma_{i^*}, a)(r_{i^*}) \sum_{s_{i^*}, s'_{i^*}} \; \sum_{\{\sigma\in T \mid r^\sigma_{i^*}=r_{i^*} \,\wedge\, s_{\sigma,i^*}=s_{i^*} \,\wedge\, s'_{\sigma,i^*}=s'_{i^*}\}} r_{i^*}(s_{i^*}, a, s'_{i^*}) Q^\eta_\sigma \\
\geqslant& \{\text{ Definition of } r^*,\ \textstyle\sum_{r_{i^*}} \Upsilon_{i^*}(\sigma^*, a)(r_{i^*}) = 1 \} \\
& \sum_{s_i, s'_i} \; \sum_{\{\sigma\in T \mid r^\sigma_{i^*}=r^* \,\wedge\, s_{\sigma,i^*}=s_{i^*} \,\wedge\, s'_{\sigma,i^*}=s'_{i^*}\}} r^*(s_{i^*}, a, s'_{i^*}) Q^\eta_\sigma \\
=& \{\text{ Equation (4.3) }\} \\
& \sum_{s_{i^*}, s'_{i^*}} \; \sum_{\{\sigma\in T \mid r^\sigma_{i^*}=r^* \,\wedge\, s_{\sigma,i^*}=s_{i^*} \,\wedge\, s'_{\sigma,i^*}=s'_{i^*}\}} r^*(s_{i^*}, a, s'_{i^*})\; Q^{\text{NR}(\eta,\sigma_{i^*},a,r^*)}_\sigma \\
=& \sum_{\{\sigma\in T \mid r^\sigma_{i^*}=r^*\}} r^*(s_{\sigma,i^*}, a, s'_{\sigma,i^*})\; Q^{\text{NR}(\eta,\sigma_{i^*},a,r^*)}_\sigma \\
=& \{\; \Upsilon'_{i^*}(\sigma_{i^*})(r^*) = 1 \text{ (by definition of } \text{NR}(\eta, \sigma_{i^*}, a, r^*) \text{ –Definition 4.3)} \;\} \\
& \sum_{\{\sigma\in T \mid r^\sigma_{i^*}=r^*\}} \Upsilon'_{i^*}(\sigma_{i^*}(r^*)\; r^*(s_{\sigma,i^*}, a, s'_{\sigma,i^*})\; Q^{\text{NR}(\eta,\sigma_{i^*},a,r^*)}_\sigma
\end{aligned}$$

$$
\begin{aligned}
&= \{\ \text{Equation (4.2)}\ \} \\
&\quad \sum_{\{\sigma \in T \mid r^{\sigma}_{i*} = r^*\}} \Pr^{\mathrm{NR}(\eta,\sigma_{i*},a,r^*)}((\sigma)^{\uparrow}) \\
&= \{\ \Pr^{\mathrm{NR}(\eta,\sigma_{i*},a,r^*)}((\sigma')^{\uparrow}) = 0 \text{ for all } \sigma' \text{ such that } r^{\sigma'}_{i*} \neq r^*\ \} \\
&\quad \sum_{\{\sigma \in T\}} \Pr^{\mathrm{NR}(\eta,\sigma_{i*},a,r^*)}((\sigma)^{\uparrow})
\end{aligned}
$$

□

By repeated application of Lemma 4.2, we can transform any η in such a way that the input scheduler for atom A_i is non-randomized, as stated in the following lemma.

LEMMA 4.3. *For all* A_i, $S = \biguplus_{m=1}^{M} (\sigma^m)^{\uparrow}$, $\eta = (\mathcal{I}, \{\Theta_1, \cdots, \Theta_N\}, \{\Upsilon_1, \cdots, \Upsilon_N\}) \in \mathrm{DIST}_P([\cdot])$, *there exists a non-randomized scheduler* Υ'_i *such that the scheduler*

$$\mathrm{NR}(\eta, \Upsilon'_i) = (\mathcal{I}, \{\Theta_1, \cdots, \Theta_N\}, \{\Upsilon_1, \cdots, \Upsilon'_i, \cdots, \Upsilon_N\})$$

complies with $\Pr^{\mathrm{NR}(\eta,\Upsilon'_i)}(S) \leqslant \Pr^{\eta}(S)$ *and* $\mathrm{NR}(\eta, \Upsilon'_i) \in \mathrm{DIST}_P([\cdot])$.

Proof. Suppose $\mathrm{ACTLAB}_i = \{a_1, \cdots, a_n\}$. Then, by Lemma 4.2, there exist $r_1, \cdots, r_n$ such that the scheduler $\mathrm{NR}(\eta, \sigma_i)$ defined as

$$\mathrm{NR}(\eta, \sigma_i) = \mathrm{NR}(\,\mathrm{NR}(\cdots \mathrm{NR}(\mathrm{NR}(\eta, \sigma_i, a_1, r_1), \sigma_i, a_2, r_2) \cdots), \sigma_i, a_n, r_n)$$

complies with

$$\Pr^{\mathrm{NR}(\eta,\sigma_i)}(S) \leqslant \Pr^{\eta}(S)\ . \tag{4.4}$$

Let $U = \{\sigma_i^1, \cdots, \sigma_i^N\}$ be the smallest set of local paths such that, if there exists $\sigma \in \mathrm{PATHS}(P)$ complying with $\mathrm{LEN}(\sigma) \leqslant \max_{1 \leqslant m \leqslant M} \mathrm{LEN}(\sigma^m)$ and $[\sigma]_i = \sigma_i$, then $\sigma_i \in \mathcal{U}$. Consider the scheduler

$$\eta^M = \mathrm{NR}(\,\mathrm{NR}(\cdots \mathrm{NR}(\mathrm{NR}(\eta, \sigma_i^1), \sigma_i^2) \cdots), \sigma_i^N)\ .$$

From Eqn. (4.4), we have $\Pr^{\eta^M}(S) \leqslant \Pr^{\eta}(S)$. Let Υ_i^M be the input scheduler for A_i in η^M. Now, consider any input scheduler Υ'_i such that $\Upsilon'_i(\sigma_i, a) = \Upsilon_i^M(\sigma_i, a)$ for all a, $\sigma_i \in U$, and $\Upsilon'_i(\sigma_i, a) = 1 : r_{\sigma_i, a}$, where $r_{\sigma_i,a}$ is any transition, for all a, $\sigma_i \notin U$. The probabilities of the paths in S are not affected by the nondeterministic resolutions for the paths having length greater than $\max_{1 \leqslant m \leqslant M} \mathrm{LEN}(\sigma^m)$. Then, we have

$$\Pr^{\mathrm{NR}(\eta,\Upsilon'_i)}(S) = \Pr^{\eta^M}(S) \leqslant \Pr^{\eta}(S)$$

as desired. □

The same transformation can be carried out on output schedulers, as formally stated below.

LEMMA 4.4. *For all* A_i, $S = \biguplus_{m=1}^{M} (\sigma^m)^{\uparrow}$, $\eta = (\mathcal{I}, \{\Theta_1, \cdots, \Theta_N\}, \{\Upsilon_1, \cdots, \Upsilon_N\}) \in \mathrm{DIST}_P([\cdot])$, *there exists a non-randomized scheduler* Θ'_i *such that the scheduler*

$$\mathrm{NR}(\eta, \Theta'_i) = (\mathcal{I}, \{\Theta_1, \cdots, \Theta'_i, \cdots, \Theta_N\}, \{\Upsilon_1, \cdots, \Upsilon_N\})$$

complies with $\Pr^{\mathrm{NR}(\eta,\Theta'_i)}(S) \leqslant \Pr^{\eta}(S)$.

Proof. A result similar to Lemma 4.2 can be proven for output schedulers. Then, we can apply the same argument as in Lemma 4.3. □

Now, it remains to show how to transform the interleaving scheduler into a non-randomized one. The proof for interleavings schedulers differs from the proofs of Lemma 4.3 and 4.4 in that we need to consider *global* paths, instead of local paths.

DEFINITION 4.4. Given $\eta = (\mathcal{I}, \{\Theta_1, \cdots, \Theta_N\}, \{\Upsilon_1, \cdots, \Upsilon_N\})$, the scheduler $\mathrm{NR}(\eta, \sigma^*, A_{i^*})$ is defined as

$$(\mathcal{I}', \{\Theta_1, \cdots, \Theta_N\}, \{\Upsilon_1, \cdots, \Upsilon_N\})$$

where $\mathcal{I}'(\sigma^*)(A_{i^*}) = 1$ and $\mathcal{I}'(\sigma) = \mathcal{I}(\sigma)$ for all $\sigma \neq \sigma^*$.

LEMMA 4.5. *For all* η, $\sigma^* \in \mathrm{PATHS}(P)$, $S = \biguplus_{m=1}^{M} (\sigma^m)^\uparrow$, *there exists* A_{i^*} *such that* $\mathrm{NR}(\eta, \sigma^*, A_{i^*})$ *complies with*

$$\mathrm{PR}^{\mathrm{NR}(\eta,\sigma^*,A_{i^*})}(S) \leqslant \mathrm{PR}^{\eta}(S) \,.$$

Proof. Let T be the set of all the global paths σ^m such that σ^* is a strict prefix of σ^m (that is, $\sigma^m{\downarrow_{\mathrm{LEN}(\sigma^*)}} = \sigma^*$ and $\mathrm{LEN}(\sigma^m) > \mathrm{LEN}(\sigma^*)$). Similarly as in Lemma 4.2, the core of the proof is to find A_{i^*} such that:

$$\mathrm{PR}^{\mathrm{NR}(\eta,\sigma^*,A_{i^*})}\Big(\biguplus_{\sigma\in T} (\sigma)^\uparrow\Big) \leqslant \mathrm{PR}^{\eta}\Big(\biguplus_{\sigma\in T} (\sigma)^\uparrow\Big) \,. \tag{4.5}$$

and then the lemma follows from

$$\begin{aligned}
&\mathrm{PR}^{\eta}(S)\\
&= \mathrm{PR}^{\eta}\Big(\biguplus_{\sigma\in T} (\sigma)^\uparrow\Big) + \mathrm{PR}^{\eta}\Big(\biguplus_{\sigma\in\{\sigma^m\}_{m=1}^{M}\setminus T} (\sigma)^\uparrow\Big)\\
&= \mathrm{PR}^{\eta}\Big(\biguplus_{\sigma\in T} (\sigma)^\uparrow\Big) + \mathrm{PR}^{\mathrm{NR}(\eta,\sigma^*,A_{i^*})}\Big(\biguplus_{\sigma\in\{\sigma^m\}_{m=1}^{M}\setminus T} (\sigma)^\uparrow\Big)\\
&\geqslant \mathrm{PR}^{\mathrm{NR}(\eta,\sigma^*,A_{i^*})}\Big(\biguplus_{\sigma\in T} (\sigma)^\uparrow\Big) + \mathrm{PR}^{\mathrm{NR}(\eta,\sigma^*,A_{i^*})}\Big(\biguplus_{\sigma\in\{\sigma^m\}_{m=1}^{M}\setminus T} (\sigma)^\uparrow\Big)\\
&= \mathrm{PR}^{\mathrm{NR}(\eta,\sigma^*,A_{i^*})}(S) \,.
\end{aligned}$$

In the following, we find A_{i^*} such that Eqn. (4.5) holds. For every $\sigma \in T$, let A_{g_σ} be the atom that performs an output in the k-th step (that is $A_{g_\sigma} = \mathrm{ACTIVE}(\sigma\langle k\rangle)$), and g^σ be the corresponding generative transition. Moreover, let a_σ be the label after the k-th step in σ and r_j be the reactive transition executed by A_j after the k-th step. Similarly as in the proof of Lemma 4.2, we define:

$$\begin{aligned}
Q^{\eta}_{\sigma} = \quad & \mathrm{PR}^{\eta}((\sigma{\downarrow_k})^\uparrow)\\
&\cdot \prod_{A_j\in\mathrm{REACTIVE}(\sigma\langle k_\sigma\rangle)} \Upsilon_j([\sigma{\downarrow_{k_\sigma}}]_j, a_\sigma)(r_j)\\
&\cdot \prod_{\substack{A_j\in\mathrm{REACTIVE}(\sigma\langle k_\sigma\rangle)\\ \mathrm{LEN}(\sigma)-1}} r_j(\pi_j(\sigma(k_\sigma)),\, a_\sigma,\, \pi_j(\sigma(k_\sigma+1)))\\
&\cdot \prod_{t=k+1} \eta(\sigma{\downarrow_t})(\sigma\langle t\rangle)\ \sigma\langle t\rangle(\sigma(t), \sigma(t+1))
\end{aligned}$$

and so we have

$$\forall \eta = (\mathcal{I}, \{\Theta_1, \cdots, \Theta_{\mathcal{N}}\}, \{\Upsilon_1, \cdots, \Upsilon_{\mathcal{N}}\}) : \forall \sigma \in T :$$
$$\mathrm{PR}^{\eta}((\sigma)^{\uparrow}) = \mathcal{I}(\sigma^*)(A_{g_\sigma})\ \Theta_{g_\sigma}([\sigma_{\downarrow k}]_{g_\sigma})(g^\sigma)\ \ g^\sigma(s_{\sigma,g}, a_\sigma, s'_{\sigma,g})\ Q^{\eta}_{\sigma}\ . \quad (4.6)$$

Note that the term $\mathcal{I}(\sigma^*)$ does not affect Q^{η}_{σ}: if a factor of the form $\mathcal{I}(\sigma')$ appears in $\mathrm{PR}^{\eta}((\sigma_{\downarrow k})^{\uparrow})$ or in $\prod_{t=k+1}^{\mathrm{LEN}(\sigma)-1} \eta(\sigma_{\downarrow t})(\sigma\langle t\rangle)\ \sigma\langle t\rangle(\sigma(t), \sigma(t+1))$ then we have $\sigma' \sqsubset \sigma^*$ (in the former case) or $\sigma^* \sqsubset \sigma'$ (in the latter). Hence,

$$\forall \sigma \in T : Q^{\eta}_{\sigma} = Q^{\mathrm{NR}(\eta, \sigma^*, A_{i^*})}_{\sigma}\ . \quad (4.7)$$

Let $s_{\sigma,g} = \pi_{g_\sigma}(\sigma(k))$ and $s'_{\sigma,g} = \pi_{g_\sigma}(\sigma(k+1))$. Then:

$$\begin{aligned}
&\sum_{\sigma \in T} \mathrm{PR}^{\eta}((\sigma)^{\uparrow}) \\
&= \sum_{\sigma \in T} \mathcal{I}(\sigma^*)(A_{g_\sigma})\ \Theta_{g_\sigma}([\sigma_{\downarrow k}]_{g_\sigma})(g^\sigma)\ \ g^\sigma(s_{\sigma,g}, a_\sigma, s'_{\sigma,g})\ Q^{\eta}_{\sigma} \\
&= \sum_{A_i, g_i, s_i, s'_i, a} \ \sum_{\{\sigma \in T \mid A_{g_\sigma} = A_i \wedge g^\sigma = g_i \wedge a_\sigma = a \wedge s_{\sigma,g} = s_i \wedge s'_{\sigma,g} = s'_i\}} \\
&\qquad \mathcal{I}(\sigma^*)(A_i)\ \ \Theta_{g_i}([\sigma_{\downarrow k}]_i)(g_i) \\
&\qquad g_i(s_i, a_\sigma, s'_i)\ Q^{\eta}_{\sigma} \\
&= \sum_{A_i} \mathcal{I}(\sigma^*)(A_i) \sum_{g_i, s_i, s'_i, a} \\
&\qquad \sum_{\{\sigma \in T \mid A_{g_\sigma} = A_i \wedge g^\sigma = g_i \wedge a_\sigma = a \wedge s_{\sigma,g} = s_i \wedge s'_{\sigma,g} = s'_i\}} \\
&\qquad \Theta_i([\sigma_{\downarrow k}]_i)(g_i)\ g_i(s_i, a, s'_i)\ Q^{\eta}_{\sigma}
\end{aligned}$$

Consider the atom A_{i^*} defined as follows:

$$A_{i^*} = \arg\min_{A_i} \ \sum_{g_i, s_i, s'_i, a} \ \sum_{\{\sigma \in T \mid g_\sigma = g_i \wedge a_\sigma = a \wedge s_{\sigma,g} = s_i \wedge s'_{\sigma,g} = s'_i\}} g_i(s_i, a, s'_i)\ \Theta_i([\sigma_{\downarrow k}]_i)(g_i)\ Q^{\eta}_{\sigma}$$

Next, we show that Eqn. (4.5) holds.

$$\begin{aligned}
&\sum_{\sigma \in T} \mathrm{PR}^{\eta}((\sigma)^{\uparrow}) \\
&= \{\ \text{Previous calculation}\ \} \\
&\sum_{A_i} \mathcal{I}(\sigma^*)(A_i) \sum_{g_i, s_i, s'_i, a} \\
&\qquad \sum_{\{\sigma \in T \mid A_{g_\sigma} = A_i \wedge g^\sigma = g_i \wedge a_\sigma = a \wedge s_{\sigma,g} = s_i \wedge s'_{\sigma,g} = s'_i\}} \\
&\qquad \Theta_i([\sigma_{\downarrow k}]_i)(g_i)\ g_i(s_i, a, s'_i)\ Q^{\eta}_{\sigma} \\
&\geqslant \{\ \text{Definition of } A_{i^*},\ \textstyle\sum_{A_i} \mathcal{I}(\sigma^*)(A_i) = 1\ \} \\
&\sum_{g_{i^*}, s_{i^*}, s'_{i^*}, a} \\
&\qquad \sum_{\{\sigma \in T \mid A_{g_\sigma} = A_{i^*} \wedge g^\sigma = g_{i^*} \wedge a_\sigma = a \wedge s_{\sigma,g} = s_{i^*} \wedge s'_{\sigma,g} = s'_{i^*}\}} \\
&\qquad \Theta_{i^*}([\sigma_{\downarrow k}]_{i^*})(g_{i^*})\ g_{i^*}(s_{i^*}, a, s'_{i^*})\ Q^{\eta}_{\sigma}
\end{aligned}$$

$$
\begin{aligned}
&= \{\text{ Equation (4.7) }\} \\
&\quad \sum_{g_{i^*}, s_{i^*}, s'_{i^*}, a} \\
&\quad \sum_{\{\sigma \in T \mid A_{g_\sigma} = A_{i^*} \wedge g^\sigma = g_{i^*} \wedge a_\sigma = a \wedge s_{\sigma,g} = s_{i^*} \wedge s'_{\sigma,g} = s'_{i^*}\}} \\
&\qquad \Theta_{i^*}([\sigma_{\downarrow k}]_{i^*})(g_{i^*})\, g_{i^*}(s_{i^*}, a, s'_{i^*})\, Q_\sigma^{NR(\eta,\sigma^*,A_{i^*})} \\
&= \sum_{\{\sigma \in T \mid A_{g_\sigma} = A_{i^*}\}} \Theta_{i^*}([\sigma_{\downarrow k}]_i)(g^\sigma)\, g^\sigma(s_{\sigma,g}, a_\sigma, s'_{\sigma,g})\, Q_\sigma^{NR(\eta,\sigma^*,A_{i^*})} \\
&= \{\, \mathcal{I}'(\sigma^*)(A_{i^*}) = 1 \text{ (by definition of } NR(\eta,\sigma^*,A_{i^*}) \text{ –Definition 4.4)} \,\} \\
&\quad \sum_{\{\sigma \in T \mid A_{g_\sigma} = A_{i^*}\}} \mathcal{I}'(A_{i^*})\, \Theta_{i^*}([\sigma_{\downarrow k}]_i)(g^\sigma)\, g^\sigma(s_{\sigma,g}, a_\sigma, s'_{\sigma,g})\, Q_\sigma^{NR(\eta,\sigma^*,A_{i^*})} \\
&= \{\text{ Equation (4.6) }\} \\
&= \sum_{\{\sigma \in T \mid A_{g_\sigma} = A_{i^*}\}} PR^{NR(\eta,\sigma^*,A_{i^*})}((\sigma)^\uparrow) \\
&= \{\, PR^{NR(\eta,\sigma^*,A_{i^*})}((\sigma')^\uparrow) = 0 \text{ for all } \sigma' \text{ such that } A_{g_{\sigma'}} \neq A_{i^*} \,\} \\
&\quad \sum_{\{\sigma \in T\}} PR^{NR(\eta,\sigma^*,A_{i^*})}((\sigma)^\uparrow)
\end{aligned}
$$

□

By repeated application of Lemma 4.5, we can construct a non-randomized interleaving scheduler, as we show in the proof of the following theorem.

LEMMA 4.6. *For all* $S = \biguplus_{m=1}^{M} (\sigma^m)^\uparrow$, $\eta = (\mathcal{I}, \{\Theta_1, \cdots, \Theta_N\}, \{\Upsilon_1, \cdots, \Upsilon_N\}) \in DIST_P([\cdot])$, *there exists a non-randomized scheduler* $\mathcal{I}'$ *such that the scheduler*

$$NR(\eta, \mathcal{I}') = (\mathcal{I}', \{\Theta_1, \cdots, \Theta_N\}, \{\Upsilon_1, \cdots, \Upsilon_i', \cdots, \Upsilon_N\})$$

complies with $PR^{NR(\eta,\mathcal{I}')}(S) \leqslant PR^\eta(S)$.

Proof. Let $\sigma^1, \cdots, \sigma^N$ be the set of all global paths having length less than or equal to $\max_{1 \leqslant m \leqslant M} LEN(\sigma^m)$. By Lemma 4.5, there exist atoms $A^1, \cdots, A^N$ such that the scheduler

$$\eta^M = NR(\, NR(\cdots NR(NR(\eta, \sigma^1, A^1), \sigma^2, A^2) \cdots), \sigma^N, A^N \,)$$

complies with $PR^{\eta^M}(S) \leqslant PR^\eta(S)$. Let $\mathcal{I}^M$ be the interleaving scheduler that defines η^M. Now, consider any interleaving scheduler $\mathcal{I}'$ such that $\mathcal{I}'(\sigma) = \mathcal{I}^M(\sigma)$ for all σ such that $LEN(\sigma) \leqslant \max_{1 \leqslant m \leqslant M} LEN(\sigma^m)$, and $\mathcal{I}(\sigma) = 1{:}A_\sigma$, where A_σ is any atom, for all σ such that $LEN(\sigma) > \max_{1 \leqslant m \leqslant M} LEN(\sigma^m)$. The probabilities of the paths in S are not affected by the nondeterministic resolutions for the paths having length greater than $\max_{1 \leqslant m \leqslant M} LEN(\sigma^m)$. Then, we have

$$PR^{NR(\eta,\mathcal{I}')}(S) = PR^{\eta^M}(S) \leqslant PR^\eta(S)$$

as desired. □

LEMMA 4.7. *For all* $\eta \in DIST_P([\cdot])$, *for all sets* $S = \biguplus_{m=1}^{M} \sigma^m$, *there exists* $\eta^{NR} \in NRDIST_P([\cdot])$ *such that*

$$PR^{\eta^{NR}}(S) \leqslant PR^\eta(S)\ .$$

Proof. Let $\eta = (\mathcal{I}, \{\Theta_1, \cdots, \Theta_N\}, \{\Upsilon_1, \cdots, \Upsilon_N\}) \in \textsc{dist}_P([\cdot])P$. By repeated application of Lemmata 4.3 and 4.4, we can find non-randomized input schedulers $\Upsilon'_1, \cdots, \Upsilon'_N$ and non-randomized output schedulers $\Theta'_1, \cdots, \Theta'_N$. By Lemma 4.6 we can find a non-randomized interleaving scheduler $\mathcal{I}'$ such that the scheduler

$$\eta^{NR} = (\mathcal{I}', \Theta'_1, \cdots, \Theta'_N, \Upsilon'_1, \cdots, \Upsilon'_N)$$

complies with $PR^{\eta^{NR}}(S) \leqslant PR^{\eta}(S)$.

Moreover, note that η^{NR} is non-randomized, that is, $\eta^{NR} \in \textsc{dist}_P([\cdot])$. □

The following lemma concerns "infinite-horizon" properties of the form $\biguplus_{m=1}^{\infty} (\sigma^m)^\uparrow$. It shows that an optimal scheduler can be constructed from schedulers η_N, whenever these schedulers are optimal for the "finite-horizon approximations" $\biguplus\{(\sigma^m)^\uparrow \mid \textsc{len}(\sigma^m) < N\}$. This optimal scheduler will be used in the proof of Theorem 4.1. Note that the lemma is not specific to non-randomized schedulers. We only require the set $\{\eta_N(\sigma) \mid 1 \leqslant N \leqslant \infty\}$ to be finite for all σ, in order to ensure the existence of a limit of the sequence $\{\eta_N\}_{N=1}^{\infty}$.

LEMMA 4.8. *Given an arbitrary scheduler $\eta \in \textsc{sched}_P$ and a set $S = \biguplus_{m=1}^{\infty} (\sigma^m)^\uparrow$, let S_N be the set $\biguplus_{m=1}^{N} (\sigma^m)^\uparrow$. If there is a sequence $\{\eta_N\}_{N=1}^{\infty}$ of schedulers such that for all $\sigma \in \textsc{paths}(P)$*

$$\{\eta_N(\sigma) \mid 1 \leqslant N \leqslant \infty\} \text{ is finite}$$

and for all N,

$$PR^{\eta_N}(S_N) \leqslant PR^{\eta}(S_N) \tag{4.8}$$

then there exists a scheduler η^{NR} such that

1. *η^{NR} is a limit of $\{\eta_N\}$ and*
2. *for all N exists $N' > N$ complying $\eta^{NR}(\sigma) = \eta_{N'}(\sigma)$ for all path σ such that $\textsc{len}(\sigma) \leqslant N$ and*
3. *$PR^{\eta^{NR}}(S) \leqslant PR^{\eta}(S)$.*

Proof.

1. By virtue of Theorem 3.1, the sequence $\{\eta_N\}$ has at least one limit. Let η^{NR} be a limit of $\{\eta_N\}$.
2. Given any N, we consider the set comprising all paths of length N. Since such set of paths is finite, and η^{NR} is a limit, the existence of N' is ensured by definition of limit.
3. Suppose, towards a contradiction, that $PR^{\eta^{NR}}(S) > PR^{\eta}(S)$. Since

$$PR^{\eta^{NR}}(S) = \sum_{m=1}^{\infty} PR^{\eta^{NR}}((\sigma^m)^\uparrow),$$

there exists N such that

$$PR^{\eta^{NR}}(\biguplus_{m}\{(\sigma^m)^\uparrow \mid \textsc{len}(\sigma^m) \leqslant N\}) > PR^{\eta}(S) \tag{4.9}$$

By the previous property, there exists $N' > N$ such that $\eta^{NR}(\sigma) = \eta_{N'}(\sigma)$ for all paths σ such that $\text{LEN}(\sigma) \leqslant N$. Now, we reason

$$\begin{aligned} & \Pr^{\eta^{NR}}(\biguplus_m \{(\sigma^m)^\uparrow \mid \text{LEN}(\sigma^m) \leqslant N\}) \\ = \; & \Pr^{\eta_{N'}}(\biguplus_m \{(\sigma^m)^\uparrow \mid \text{LEN}(\sigma^m) \leqslant N\}) \\ \leqslant \; & \Pr^{\eta_{N'}}(\biguplus_m \{(\sigma^m)^\uparrow \mid \text{LEN}(\sigma^m) \leqslant N'\}) \,. \end{aligned} \tag{4.10}$$

In addition,

$$\begin{aligned} & \Pr^{\eta^{NR}}(\biguplus_m \{(\sigma^m)^\uparrow \mid \text{LEN}(\sigma^m) \leqslant N\}) \\ > \; & \Pr^{\eta}(\biguplus_m (\sigma^m)^\uparrow) \\ \geqslant \; & \Pr^{\eta}(\biguplus_m \{(\sigma^m)^\uparrow \mid \text{LEN}(\sigma^m) \leqslant N'\}) \\ \geqslant \; & \{ \text{ Inequation (4.8) } \} \\ & \Pr^{\eta_{N'}}(\biguplus_m \{(\sigma^m)^\uparrow \mid \text{LEN}(\sigma^m) \leqslant N'\}) \,. \end{aligned}$$

This contradicts (4.10).

□

The following lemma simply combines Lemma 4.7 and Lemma 4.8 in order to show that non-randomized schedulers are sufficient to obtain the infimum probability of an "infinite-horizon" property as before.

LEMMA 4.9. *For all* $\eta \in \text{DIST}_P([\cdot])$, *for all sets* $S = \biguplus_{m=1}^{\infty} (\sigma^m)^\uparrow$ *there exists* $\eta^* \in \text{NRDIST}_P([\cdot])$ *such that* $\Pr^{\eta^*}(S) \leqslant \Pr^{\eta}(S)$.

Proof. Let S_N be as in the statement of Lemma 4.8. For each N, Lemma 4.7 ensures the existence of $\eta_N \in \text{NRDIST}_P([\cdot])$ such that $\Pr^{\eta_N}(\biguplus_{\sigma \in S_N} (\sigma)^\uparrow) \leqslant \Pr^{\eta}(\biguplus_{\sigma \in S_N} (\sigma)^\uparrow)$. Since the schedulers η_N are non-randomized and the number of transitions in the system is finite (by Definitions 1.2 and 1.12) we have that $\{\eta_N(\sigma)\}_{N=1}^{\infty}$ is finite for all σ. So, Lemma 4.8 ensures the existence of a scheduler η^{NR} such that $\Pr^{\eta^{NR}}(S) \leqslant \Pr^{\eta}(S)$.

The scheduler η^{NR} is indeed non-randomized, since it is a limit of non-randomized schedulers. By Theorem 3.1, we have $\eta^{NR} \in \text{NRDIST}_P([\cdot])$. □

Given two sets of schedulers $\mathbb{S}$, $\mathbb{S}'$, we give a sufficient conditions as to ensure $\sup_{\eta \in \mathbb{S}} \Pr^{\eta}(S) \leqslant \sup_{\eta' \in \mathbb{S}'} \Pr^{\eta'}(S)$ for all measurable S. In fact, we require the schedulers in $\mathbb{S}'$ to improve the probability of the schedulers in $\mathbb{S}$ for every set T of the form $\biguplus_{m=1}^{\infty} (\sigma^m)^\uparrow$, in the sense that

$$\forall \eta \in \mathbb{S} : \exists \eta' \in \mathbb{S}' : \; \Pr^{\eta'}(T) \leqslant \Pr^{\eta}(T)$$

holds. Note the apparent contradiction between the hypothesis and conclusion of the theorem: although we are dealing with the supremum, we require $\Pr^{\eta'}(T) \leqslant \Pr^{\eta}(T)$, instead of $\Pr^{\eta'}(T) \geqslant \Pr^{\eta}(T)$. This is due to the fact that we define $\Pr^{\eta}(S)$ as the *infimum* quantifying over all $\mathcal{R}$-covers (see Def. 4.2), and there is no symmetrical definition using suprema[†]. Later on, we will obtain Theorem 4.1 by taking $\mathbb{S}$, $\mathbb{S}'$ to be $\text{DIST}_P([\cdot])$ and $\text{NRDIST}_P([\cdot])$, respectively.

THEOREM 4.2. *Let* $\mathbb{S}$, $\mathbb{S}'$ *be sets of schedulers such that for all set* $T = \biguplus_{m=1}^{\infty} (\sigma^m)^\uparrow$,

$$\forall \eta \in \mathbb{S} : \exists \eta' \in \mathbb{S}' : \; \Pr^{\eta'}(T) \leqslant \Pr^{\eta}(T) \,.$$

[†]It takes no effort to find an equivalent definition using suprema and complements of $\mathcal{R}$-covers, but $\mathcal{R}$-covers are not closed under complement

Then, for all measurable S,

$$\sup_{\eta \in \mathcal{S}} \Pr^{\eta}(S) \leqslant \sup_{\eta' \in \mathcal{S}'} \Pr^{\eta'}(S) \, .$$

Proof. We prove the result by showing that, for all $\epsilon > 0$, there exists $\eta' \in \mathcal{S}'$ such that $\sup_{\eta \in \mathcal{S}} \Pr^{\eta}(S) - \Pr^{\eta'}(S) < \epsilon$.

Let $\eta^s \in \mathcal{S}$ be such that $\sup_{\eta \in \mathcal{S}} \Pr^{\eta}(S) - \Pr^{\eta^s}(S) < \epsilon/2$. By Lemma 4.1 (applied to the complement set $\overline{S}$), there exists a sequence $\{(\sigma^m)^\uparrow\}_{m=1}^{\infty}$ of disjoint cylinders such that $\overline{S} \subseteq \biguplus_m (\sigma^m)^\uparrow$ and

$$\Pr^{\eta^s}(\biguplus_m (\sigma^m)^\uparrow) - \Pr^{\eta^s}(\overline{S}) < \epsilon/2 \, . \tag{4.11}$$

By hypothesis, there exists $\eta' \in \mathcal{S}'$ such that

$$\Pr^{\eta'}(\biguplus_m (\sigma^m)^\uparrow) \leqslant \Pr^{\eta^s}(\biguplus_m (\sigma^m)^\uparrow) \, .$$

So, from (4.11) we have

$$\Pr^{\eta'}(\biguplus_m (\sigma^m)^\uparrow) - \Pr^{\eta^s}(\overline{S}) < \epsilon/2 \, .$$

From which we obtain

$$1 - \Pr^{\eta'}(\overline{\biguplus_m (\sigma^m)^\uparrow}) - (1 - \Pr^{\eta^s}(S)) < \epsilon/2 \, .$$

Simple arithmetic yields

$$\Pr^{\eta^s}(S) - \Pr^{\eta'}(\overline{\biguplus_m (\sigma^m)^\uparrow}) < \epsilon/2 \, . \tag{4.12}$$

Since $\overline{S} \subseteq \biguplus_m (\sigma^m)^\uparrow$ we have $\overline{\biguplus_m (\sigma^m)^\uparrow} \subseteq S$, and so $\Pr^{\eta'}(\overline{\biguplus_m (\sigma^m)^\uparrow}) \leqslant \Pr^{\eta'}(S)$. From (4.12) we obtain $\Pr^{\eta^s}(S) - \Pr^{\eta'}(S) < \epsilon/2$. Then,

$$\begin{aligned} \sup_{\eta \in \mathcal{S}} \Pr^{\eta}(S) - \Pr^{\eta'}(S) &= \sup_{\eta \in \mathcal{S}} \Pr^{\eta}(S) - \Pr^{\eta^s}(S) + \Pr^{\eta^s}(S) - \Pr^{\eta'}(S) \\ &= \epsilon/2 + \epsilon/2 = \epsilon \quad , \end{aligned}$$

as desired. □

Proof (of Theorem 4.1). By virtue of Lemma 4.9, we are under the hypotheses of Theorem 4.2 with $\mathcal{S} = \textsc{Dist}_{\mathrm{P}}([\cdot])$ and $\mathcal{S}' = \textsc{NRDist}_{\mathrm{P}}([\cdot])$, and so

$$\sup_{\eta \in \textsc{Dist}_{\mathrm{P}}([\cdot])} \Pr^{\eta}(S) \leqslant \sup_{\eta \in \textsc{NRDist}_{\mathrm{P}}([\cdot])} \Pr^{\eta}(S) \, .$$

Since $\textsc{NRDist}_{\mathrm{P}}([\cdot]) \subseteq \textsc{Dist}_{\mathrm{P}}([\cdot])$, we have

$$\sup_{\eta \in \textsc{NRDist}_{\mathrm{P}}([\cdot])} \Pr^{\eta}(S) \leqslant \sup_{\eta \in \textsc{Dist}_{\mathrm{P}}([\cdot])} \Pr^{\eta}(S)$$

and hence the result holds. □

4.2 NON-RANDOMIZED STRONGLY DISTRIBUTED SCHEDULERS

Corollary 4.1 establishes that, for the usual projection $[\![\cdot]\!]$, non-randomized *distributed schedulers* attain the same supremum probability as randomized ones. Unfortunately, if in the statement of Corollary 4.1 we consider *strongly distributed schedulers*, then the same claim is false.

4.2.1 *Randomization adds power to strongly distributed schedulers*

Consider the example in Fig. 4.1. Atoms A, B and C need to be "activated" by labels e_A, e_B and e_C, respectively. Atom E tosses a coin and activates A, B and C if the output of the coin is l, or B and C (and, later on, A) if the output of the coin is r. Atom R "remembers" the order in which the other atoms execute. The aim of the scheduler is to reach some state in R marked with ☺. It is clear that any non-randomized scheduler yields a probability of 0, 1/2 or 1. We show that there exists no $\eta \in \text{NRSDIST}_P([\![\cdot]\!])$ reaching ☺ with probability 1.

First, we note that, for non-randomized schedulers, the condition imposed to a strongly distributed interleaving scheduler reduces to the following one: for all atoms $A \neq B$ there cannot be two paths σ, σ' such that

1. $[\sigma]_A = [\sigma']_A \wedge [\sigma]_B = [\sigma']_B$ and
2. atom A is scheduled in σ and
3. atom B is scheduled in σ'.

Formally:

$$\forall A, B : A \neq B \implies \forall \sigma : \nexists \sigma' : [\sigma]_A = [\sigma']_A \wedge [\sigma]_B = [\sigma']_B \wedge \mathcal{I}(\sigma) = A \wedge \mathcal{I}(\sigma') = B \,. \quad (4.13)$$

In order to yield a probability of 1, any scheduler η must reach ☺ for both l and r. 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 B and C 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, Eqn. (4.13) implies that η must choose b before c in case the first output is r. Therefore, after b, atom R should output *w*, and E should output e_A. At this point, both A and C may become 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 ☺. Hence, there is no non-randomized strongly distributed scheduler yielding probability 1, and so the supremum quantifying over non-randomized strongly distributed schedulers is 1/2. Nevertheless, consider the scheduler in which

- If there is a transition enabled in E, then the transition in E is chosen
- 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 both in E and R)
- 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, if b and c are enabled, choose each one with probability 1/2, etc.

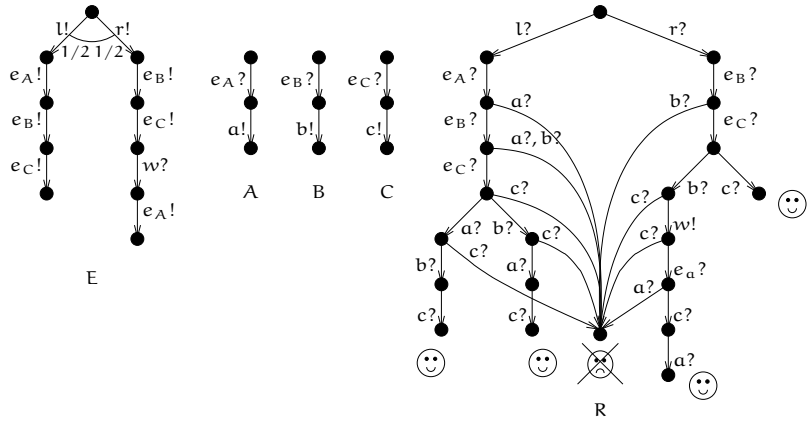


Figure 4.1: Example showing that randomization adds power to strongly distributed schedulers

This scheduler is in $\text{SDIST}_P(\llbracket\cdot\rrbracket)$, and yields a probability of $13/24 > 1/2$. Therefore, this example shows that *randomized choices add power to schedulers in* $\text{SDIST}_P(\llbracket\cdot\rrbracket)$.

From the same example, it is easy to see that the total order-based scheduler cannot emulate randomized strongly distributed schedulers in the realm of $\llbracket\cdot\rrbracket$. A total order scheduler assigns a fixed order for a, b and c, and any such order yields a probability of at most $1/2$.

4.2.2 *Expressive non-randomized strongly distributed schedulers*

Although for the projection $\llbracket\cdot\rrbracket$ non-randomized strongly distributed schedulers are not as powerful as the randomized ones, we present two sufficient conditions ensuring that non-randomized schedulers are sufficiently expressive.

One of the sufficient conditions we propose involves the notion of *traceable* projection. We explain traceable projections using a preliminary definition. Given a path $\sigma.c.s$ and an atom A_i such that $[\sigma.c.s]_i \neq [\sigma]_i$ and $\text{ACTIVE}(c) = A_j$, we say that the local path $[\sigma]_j$ leads to the local path $[\sigma.c.s]_i$ in $\sigma.c.s$. We define the traceable projections as those in which, given a local path σ_i, the local path σ_j that leads to σ_i is the same for all global paths σ.

Traceable projection

DEFINITION 4.5 (Traceable projection). We say that a projection $[\cdot]$ is *traceable* if, for all A_i, $\sigma.c.s$, $\sigma'.c'.s'$, such that

$$\begin{aligned}
&[\sigma.c.s]_i = \left[\sigma'.c'.s'\right]_i \\
&[\sigma.c.s]_i \neq [\sigma]_i \\
&\left[\sigma'.c'.s'\right]_i \neq \left[\sigma'\right]_i ,
\end{aligned}$$

we have $\text{ACTIVE}(c) = \text{ACTIVE}(c') = A_j$ and $[\sigma]_j = [\sigma']_j$ for some A_j. If $[\sigma.c.s]_i$ and $[\sigma]_j$ are as before, we write $[\sigma.c.s]_i \hookleftarrow [\sigma]_j$.

Moreover, for all σ, we require

$$[\sigma.c.s]_i \neq [\sigma]_i \implies [\sigma.c.s]_i \neq [\text{INIT}]_i \ . \tag{4.14}$$

Intuitively, $\sigma_i \hookleftarrow \sigma_j$ if each time the projection of a path becomes σ_i, the output is performed by A_j, and the projection to A_j before the output is σ_j.

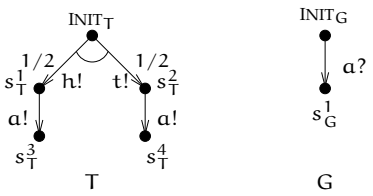


Figure 4.2: Projection $[\![\cdot]\!]$ is not traceable

Equation (4.14) is needed in order to ensure that, if $[\sigma]_i \leftharpoonup \sigma_j$, then $[\sigma]_i \neq [\text{INIT}]_i$. So, it holds that exactly one of these propositions is true: $[\sigma]_i = [\text{INIT}]_i$ or $[\sigma]_i \leftharpoonup \sigma_j$ for some σ_j.

EXAMPLE 4.1. The projection $[\![\cdot]\!]$ is *not* traceable. Consider the system depicted in Fig. 4.2. Since each of the generative transitions assigns positive probability to exactly one action label, and there is at most one reactive transition enabled in each state, we do not name the transitions, using thus action labels instead of transition names.

$$\begin{aligned} \sigma &= (\text{INIT}_T, \text{INIT}_G).(h!).(s_T^1, \text{INIT}_G) \\ \sigma' &= (\text{INIT}_T, \text{INIT}_G).(t!).(s_T^2, \text{INIT}_G) \\ c = c' &= (a!, a?) \\ s &= (s_T^3, s_G^1) \\ s' &= (s_T^3, s_G^1) \end{aligned}$$

We have

$$\begin{aligned} &[\![\sigma.c.s]\!]_G = [\![\sigma'.c'.s']\!]_G = \text{INIT}_G.a.s_G^1 \\ &[\![\sigma.c.s]\!]_G = \text{INIT}_G.a.s_G^1 \neq \text{INIT}_G = [\![\sigma]\!]_G \\ &[\![\sigma'.c'.s']\!]_G = \text{INIT}_G.a.s_G^1 \neq \text{INIT}_G = [\![\sigma']\!]_G \end{aligned}$$

However, we have

$$[\![\sigma]\!]_{\text{ACTIVE}(c)} = \sigma.h.s_T^1 \neq \sigma.t.s_T^2 = [\![\sigma']\!]_{\text{ACTIVE}(c')}$$

Intuitively, the projection is not traceable because, given the local path $\sigma_G = \sigma.a.s_G^1$, the scheduler cannot *trace* the local path that led to σ_G, it could be either $\sigma.h.s_T^1$ or $\sigma.t.s_T^2$.

The next example introduces a projection that *is* traceable, it is called *visible prefix* and denoted by $[\cdot]^{VP}$. We would meet this projection again in Sec. 6.2, while developing an algorithm that is proven to be correct under the assumption that the schedulers are in $\text{SDIST}_P([\cdot]^{VP}, \leqslant)$.

EXAMPLE 4.2. For each atom A_i, let $[\cdot]_i^{VP}$ be the projection defined as follows:

- $[\text{INIT}]_i^{VP} = \text{INIT}$,

- $[\sigma.c.s]_i^{VP} = \sigma.c.s$ if $\text{LABEL}(c) \in \text{ACTLAB}_i$ and
- $[\sigma.c.s]_i^{VP} = [\sigma]_i^{VP}$, otherwise.

Let σ, σ', c, c' be as in the previous example. Since $\text{LABEL}(c) = a \in \text{ACTLAB}_G$, we have

$$[\![\sigma.c.s]\!]_G = \sigma.c.s \neq \sigma'.c'.s' = [\![\sigma'.c'.s']\!]_G = \text{INIT}_G.a.s_G^1$$

Once G synchronizes with T, the schedulers is allowed to see all the history of the system and, particularly, the history of T.

In order to give another example, we note that

$$[\![\sigma.c.s]\!]_G = \sigma.c.s \neq \text{INIT} = [\![\sigma]\!]_G$$
$$[\![\sigma'.c'.s']\!]_G = \sigma'.c'.s' \neq \text{INIT} = [\![\sigma']\!]_G$$

since neither $h = \text{LABEL}(c)$ nor $t = \text{LABEL}(c')$ are in ACTLAB_G.

We show that $[\cdot]^{VP}$ is traceable. Let $\sigma.c.s$, $\sigma'.c'.s'$ be such that $[\sigma.c.s]_i^{VP} = [\sigma'.c'.s']_i^{VP}$, $[\sigma.c.s]_i^{VP} \neq [\sigma]_i^{VP}$ and $[\sigma'.c'.s']_i^{VP} \neq [\sigma']_i^{VP}$. From $[\sigma.c.s]_i^{VP} \neq [\sigma]_i^{VP}$, we get $[\sigma.c.s]_i^{VP} = \sigma.c.s$ and, from $[\sigma'.c'.s']_i^{VP} \neq [\sigma']_i^{VP}$, we get $[\sigma'.c'.s']_i^{VP} = \sigma'.c'.s'$. Hence, $[\sigma.c.s]_i^{VP} = [\sigma'.c'.s']_i^{VP}$ yields $\sigma.c.s = \sigma'.c'.s'$. Therefore, we have $\text{ACTIVE}(c) = \text{ACTIVE}(c')$ and $[\sigma]^{VP}_{\text{ACTIVE}(c)} = [\sigma']^{VP}_{\text{ACTIVE}(c)}$, as desired.

Moreover, if $[\sigma.c.s]_i^{VP} \neq [\sigma]_i^{VP}$, then $[\sigma.c.s]_{=}^{VP}\sigma.c.s \neq [\text{INIT}]_i^{VP}$, as required in the definition of traceable projections.

We introduced traceable projections in order to present one of the sufficient conditions to ensure that non-randomized schedulers are sufficiently expressive. The other sufficient condition we provide needs the notion of *projection equivalent for a set of atoms*.

Projection equivalent for a set of atoms

DEFINITION 4.6. Given a set of atoms $\mathcal{A} \in \text{ATOMS}(P)$, we say that projection $[\cdot]$ is *equivalent* for $\mathcal{A}$ if, for all σ, σ', $A_i, A_{i'} \in \mathcal{A}$, we have $[\sigma]_i = [\sigma']_i \implies [\sigma]_{i'} = [\sigma']_{i'}$.

Intuitively, $[\cdot]$ is equivalent for $\mathcal{A}$ if the projection of atom $A_i \in \mathcal{A}$ uniquely determines the projection of all other atoms in $\mathcal{A}$. The following example shows a system for which projection $[\![\cdot]\!]$ is equivalent for several sets of atoms.

EXAMPLE 4.3. Consider the system $P = A \parallel B_1 \parallel B_2 \parallel B_3 \parallel C_1 \parallel C_2$ in Fig. 4.3. Since each path σ is univocally identified by the labels output in σ, we use sequences of labels to refer to paths. The overall behaviour of P is as follows. In the initial state, the only atom that can generate an output is A. It outputs some sequence of the form $\{b, c\}^*$, until it outputs e_B. Then, the set of active atoms is $\mathcal{B} = \{B_1, B_2, B_3\}$ until B_3 generates e_C. Once this label has been output, the set of active atoms is $\mathcal{C} = \{C_1, C_2\}$.

The atoms in $\mathcal{B}$ are able to see the b's output by A, while the atoms in $\mathcal{C}$ are able to see the c's. In addition, each atom $B_i \in \mathcal{B}$ is able to see the outputs of the other atoms in $\mathcal{B}$. We show that the projection $[\![\cdot]\!]$ is equivalent for $\mathcal{B}$. Let σ be a path in B. Then σ has a sequence of labels of the form

$$\overbrace{b\cdots b}^{n_1}\overbrace{c\cdots c}^{m_1}\cdots\overbrace{b\cdots b}^{n_Q}\overbrace{c\cdots c}^{m_Q}\, e_B\, \overbrace{k_1\cdots k_1}^{r_1}\overbrace{k_2\cdots k_2}^{t_1}\cdots\overbrace{k_1\cdots k_1}^{r_Z}\overbrace{k_2\cdots k_2}^{t_Z}\, e_C\, p_x$$

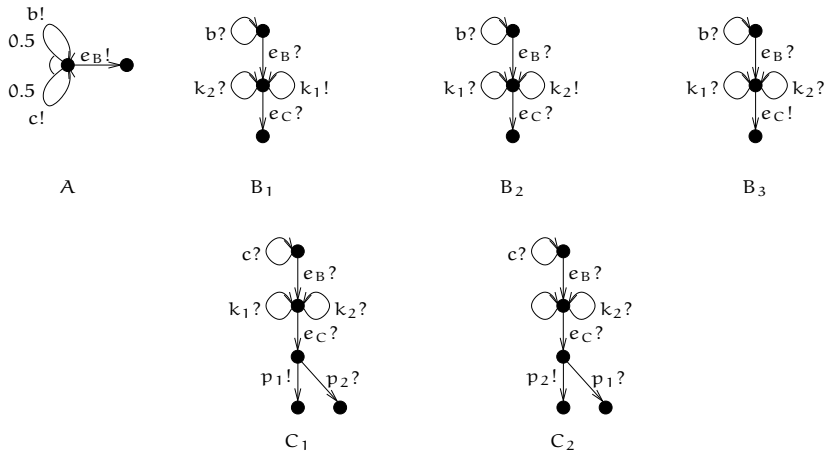


Figure 4.3: The projection $[\![\cdot]\!]$ is equivalent for the sets $\mathcal{B} = \{B_i\}_{i=1}^3$ and $\mathcal{C} = \{C_1, C_2\}$

where $n_1 \geqslant 0$, $n_q > 0$ for all $2 \leqslant q \leqslant Q$, $m_q > 0$ for all $1 \leqslant q < Q$, $m_Q \geqslant 0$, $r_1 \geqslant 0$, $r_z > 0$ for all $2 \leqslant z \leqslant Z$, $t_z > 0$ for all $1 \leqslant z < Z$, $t_Z \leqslant 0$ and $x \in \{1, 2\}$. The projection $[\![\sigma]\!]_{B_i}$ is the only path in the atom B_i whose labels are

$$\overbrace{b \cdots b}^{n_1+\cdots+n_Q}\, e_B\, \overbrace{k_1 \cdots k_1}^{r_1}\overbrace{k_2 \cdots k_2}^{t_1} \cdots \overbrace{k_1 \cdots k_1}^{r_Z}\overbrace{k_2 \cdots k_2}^{t_Z}\, e_C \ . \tag{4.15}$$

Let σ' be such that $[\![\sigma]\!]_{B_i} = [\![\sigma']\!]_{B_i}$. Note that the number of c's in σ' is not necessarily the same as in σ, since the projection over B_i eliminates the information concerning the number of c's. On the contrary, the sequence of k_1's and k_2's and the number of b's must coincide in both paths, since the projection preserves this sequence. Then, σ' is of the form

$$\overbrace{b \cdots b}^{n'_1}\overbrace{c \cdots c}^{m'_1} \cdots \overbrace{b \cdots b}^{n'_{Q'}}\overbrace{c \cdots c}^{m'_{Q'}}\, e_B\, \overbrace{k_1 \cdots k_1}^{r_1}\overbrace{k_2 \cdots k_2}^{t_1} \cdots \overbrace{k_1 \cdots k_1}^{r_Z}\overbrace{k_2 \cdots k_2}^{t_Z}\, e_C\, p_{x'}$$

with $n'_1 + \cdots + n'_{Q'} = n_1 + \cdots + n_Q$ (since the number of b's coincides in both σ and σ').

In order to prove that $[\![\cdot]\!]$ is equivalent for $\mathcal{B}$, we have to prove that, for all $B_j \in \mathcal{B}$, it holds $[\![\sigma]\!]_{B_j} = [\![\sigma']\!]_{B_j}$. Since all the atoms $B_j \in \mathcal{B}$ have the same alphabet, we have that $[\![\sigma]\!]_{B_j}$ is a path whose labels are as in (4.15). This determines only one path in B_j. For the same reason, the labels of $[\![\sigma']\!]_{B_j}$ are as in (4.15), and so $[\![\sigma]\!]_{B_j}$ and $[\![\sigma']\!]_{B_j}$ are the same path in B_j. Therefore, $[\![\sigma]\!]_{B_j} = [\![\sigma']\!]_{B_j}$.

A similar argument can be used to show that $[\![\cdot]\!]$ is equivalent for $\mathcal{C} = \{C_1, C_2\}$. In effect, given a path σ as before, if $[\![\sigma]\!]_{C_1} = [\![\sigma']\!]_{C_1}$, then σ and σ' differ only with respect to the number of b's. Since both C_1 and C_2 have the same alphabet, we conclude that $[\![\sigma]\!]_{C_2}$ and $[\![\sigma']\!]_{C_2}$ correspond to paths in C_2 having the same labels. Given a sequence of labels, there is at most one path in C_2 with such labels, thus implying $[\![\sigma]\!]_{C_2} = [\![\sigma']\!]_{C_2}$.

The following theorem gives sufficient conditions as to ensure that non-randomized strongly distributed schedulers are sufficiently expressive. Moreover, under these conditions, we have full expressiveness for the set of *total order-based schedulers* (Def. 2.3).

THEOREM 4.3. *Let* $\mathbb{S}$ *be a set of schedulers and* $[\cdot]$ *be a projection such that either:*

I. $\mathcal{S} = \text{SDIST}_P([\cdot], \text{RATE})$ *and* $[\cdot]$ *is traceable or*

II. $\mathcal{S} = \text{SDIST}_P([\cdot])$ *and there exists a partition* $\{\mathcal{A}_1, \cdots, \mathcal{A}_z\}$ *on* $\text{ATOMS}(P)$ *such that* $[\cdot]$ *is equivalent for* $\mathcal{A}_l$ *for all* $l = 1, \cdots, z$ *and*

$$\forall A_i, A_{i'} : \left(\exists s : \left|G_i(s)\right| > 0 \wedge \left|G_{i'}(s)\right| > 0\right) \implies \exists l : A_i, A_{i'} \in \mathcal{A}_l$$

Then, for any set $\mathcal{S}$ *of infinite traces,* $\mathcal{S}$ *being measurable, we have that*

$$\sup_{\eta \in \mathcal{S}} \text{PR}^\eta(\mathcal{S}) \leqslant \sup_{\eta \in \text{SDIST}_P([\cdot], \leqslant)} \text{PR}^\eta(\mathcal{S})$$

(The proof can be found in Subsection 4.2.4 below.)

COROLLARY 4.2. *Let* $[\cdot]$ *and* $\{\mathcal{A}_1, \cdots, \mathcal{A}_z\}$ *be as in case II in the previous theorem. Then, for all measurable* $\mathcal{S}$*,*

$$\sup_{\eta \in \text{SDIST}_P([\cdot])} \text{PR}^\eta(\mathcal{S}) = \sup_{\eta \in \text{SDIST}_P([\cdot], \leqslant)} \text{PR}^\eta(\mathcal{S})$$

Proof. By Theorem 4.3 (case *II*), we have

$$\sup_{\eta \in \text{SDIST}_P([\cdot])} \text{PR}^\eta(\mathcal{S}) \leqslant \sup_{\eta \in \text{SDIST}_P([\cdot], \leqslant)} \text{PR}^\eta(\mathcal{S}) \,.$$

The equality holds since $\text{SDIST}_P([\cdot], \leqslant) \subseteq \text{SDIST}_P([\cdot])$ (by Theorem 2.8). □

4.2.3 *Full-communication version of a projection*

The projection $[\![\cdot]\!]$ is not traceable (Example 4.1), and so case *(I)* in Theorem 4.3 does not apply to this projection. Hence, it would be useful to find some set $\mathbb{S}$ of non-randomized schedulers giving us a safe estimation of $\sup_{\eta \in \text{SDIST}_P([\![\cdot]\!], \text{RATE})} \text{PR}^\eta(\mathcal{S})$. Formally, the set $\mathbb{S}$ should comply:

$$\sup_{\eta \in \mathbb{S}} \text{PR}^\eta(\mathcal{S}) \geqslant \sup_{\eta \in \text{SDIST}_P([\![\cdot]\!], \text{RATE})} \text{PR}^\eta(\mathcal{S}) \,. \tag{4.16}$$

The existence of $\mathbb{S}$ is ensured, provided the existence of a traceable projection $[\cdot]'$ such that $[\![\cdot]\!] \sqsubseteq [\cdot]'$: in fact, by Theorem 2.7 we have

$$\sup_{\eta \in \text{SDIST}_P([\![\cdot]\!], \text{RATE})} \text{PR}^\eta(\mathcal{S}) \leqslant \sup_{\eta \in \text{SDIST}_P([\cdot]', \text{RATE})} \text{PR}^\eta(\mathcal{S}) \,,$$

and by Theorem 4.3 (applied to $[\cdot]'$) we have

$$\sup_{\eta \in \text{SDIST}_P([\![\cdot]\!], \text{RATE})} \text{PR}^\eta(\mathcal{S}) \leqslant \sup_{\eta \in \text{SDIST}_P([\cdot]', \leqslant)} \text{PR}^\eta(\mathcal{S}) \,.$$

Then, we can assume that the schedulers are non-randomized, provided that we relax the projection from $[\![\cdot]\!]$ to $[\cdot]'$.

Given a projection $[\cdot]$, we present a general mechanism to obtain a traceable projection $|[\cdot]|$ such that $[\cdot] \sqsubseteq |[\cdot]|$. In order to ensure $[\cdot] \sqsubseteq |[\cdot]|$, we define $|[\cdot]|$ in such a way that, if $[\sigma.c.s]_i \neq [\sigma]_i$, then $|[\sigma.c.s]|_i$ comprises all the information in $[\sigma.c.s]_i$. In addition, we want $|[\cdot]|$ to be traceable: we will prove traceability using the fact that $|[\sigma.c.s]|_i$ comprises all the information in the local path $|[\sigma.c.s]|_{\text{ACTIVE}(c)}$. Note that this local path is in $\text{LOCALPATHS}^{|[\cdot]|}_{\text{ACTIVE}(c)}$, while $[\sigma.c.s]_i$ is in $\text{LOCALPATHS}^{[\cdot]}_i$.

Except for the paths σ such that $[\sigma]_i = [\text{INIT}]_i$, the projection $|[\cdot]|$ yields a pair $(\sigma_i, \sigma_j^{|[\cdot]|})$, where $\sigma_i \in \text{LOCALPATHS}^{[\cdot]}_i$ and $\sigma_j^{|[\cdot]|} \in \text{LOCALPATHS}^{|[\cdot]|}_j$, where A_j is the active atom in the last transition c such that $[\sigma']_i \neq [\sigma'.c.s]_i$ and $[\sigma'.c.s]_i \sqsubseteq \sigma$.

DEFINITION 4.7. Given a projection $[\cdot]$, we inductively define the family of functions $|[\cdot]|_i$:

- $|[\text{INIT}]|_i = [\text{INIT}]_i$
- $|[\sigma.c.s]|_i = (\, [\sigma.c.s]_i \,,\, |[\sigma]|_{\text{ACTIVE}(c)} \,)$ if $[\sigma.c.s]_i \neq [\sigma]_i$
- $|[\sigma.c.s]|_i = |[\sigma]|_i$, otherwise.

The projection $|[\cdot]|$ is called the *full-communication version* of $[\cdot]$.

We prove that $[\cdot]$ does in fact comply with the properties that motivated its definition.

THEOREM 4.4. *Let* $[\cdot]$ *be a projection and* $|[\cdot]|$ *be the full communication version of* $[\cdot]$*, then* $[\cdot] \sqsubseteq |[\cdot]|$.

Proof. We have to prove

$$|[\sigma]|_i = \left|\left[\sigma'\right]\right|_i \implies [\sigma]_i = \left[\sigma'\right]_i \,.$$

We proceed by induction on $\text{LEN}(\sigma) + \text{LEN}(\sigma')$. If $\sigma = \sigma' = \text{INIT}$, we have $|[\sigma]|_i = |[\text{INIT}]|_i = |[\sigma']|_i$ and $[\sigma]_i = [\text{INIT}]_i = [\sigma']_i$.

If $\text{LEN}(\sigma) + \text{LEN}(\sigma') = n + 1$ assume, w.l.o.g., that $\text{LEN}(\sigma) > 1$ (the case $\text{LEN}(\sigma') > 1$ is symmetrical). Let $\sigma = \sigma''.c.s$ for some c, s. As in the definition of $|[\cdot]|$, we have two separate cases. In case $[\sigma'']_i = [\sigma]_i$, the definition of $|[\cdot]|$ gives $|[\sigma]|_i = |[\sigma'']|_i$, and it allows to prove the desired implication:

$$\begin{array}{ll} & |[\sigma]|_i = \left|\left[\sigma'\right]\right|_i \\ \implies & \{\ |[\sigma]|_i = |[\sigma'']|_i\ \} \\ & \left|\left[\sigma''\right]\right|_i = \left|\left[\sigma'\right]\right|_i \\ \implies & \{\ \text{Induction, } \text{LEN}(\sigma'') + \text{LEN}(\sigma') = \text{LEN}(\sigma) - 1 + \text{LEN}(\sigma') = n\ \} \\ & \left[\sigma''\right]_i = \left[\sigma'\right]_i \\ \implies & \{\ [\sigma'']_i = [\sigma]_i\ \} \\ & [\sigma]_i = \left[\sigma'\right]_i \end{array}$$

In case $[\sigma'']_i \neq [\sigma]_i$, we have $|[\sigma]|_i = (\, [\sigma]_i \,,\, |[\sigma]|_{\text{ACTIVE}(c)} \,)$. Then, $|[\sigma]|_i = |[\sigma']|_i$ implies

$$\left|\left[\sigma'\right]\right|_i = (\, [\sigma]_i \,,\, |[\sigma]|_{\text{ACTIVE}(c)} \,) \,. \tag{4.17}$$

Note that, for all σ^*, the definition of $|[\cdot]|$ ensures $|[\sigma^*]|_i = (\, \sigma_i \,,\, \sigma_j^{|[\cdot]|} \,) \implies [\sigma^*]_i = \sigma_i$. Applying this implication in Eqn. (4.17), we have $[\sigma']_i = [\sigma]_i$, as desired. □

THEOREM 4.5. *For all projections* $[\cdot]$*, the full-communication version* $|[\cdot]|$ *is traceable.*

Proof. Let $\sigma.c.s$, $\sigma'.c'.s'$ be such that $|[\sigma.c.s]|_i = |[\sigma'.c'.s']|_i$, $|[\sigma.c.s]|_i \neq |[\sigma]|_i$ and $|[\sigma'.c'.s']|_i \neq |[\sigma']|_i$. From $|[\sigma.c.s]|_i \neq |[\sigma]|_i$, we get

$$|[\sigma.c.s]|_i = (\, [\sigma.c.s]_i \,,\, |[\sigma]|_{\text{ACTIVE}(c)} \,)$$

(see Def. 4.7) and from $|[\sigma'.c'.s']|_i \neq |[\sigma']|_i$ we get

$$\left|\left[\sigma'.c'.s'\right]\right|_i = (\, \left[\sigma'.c'.s'\right]_i \,,\, \left|\left[\sigma'\right]\right|_{\text{ACTIVE}(c')} \,) \,.$$

Hence, $|[\sigma.c.s]|_i = |[\sigma'.c'.s']|_i$ yields $\text{ACTIVE}(c) = \text{ACTIVE}(c')$ and $|[\sigma]|_{\text{ACTIVE}(c)} = |[\sigma']|_{\text{ACTIVE}(c)}$, thus implying $|[\sigma]|_{\text{ACTIVE}(c)} = |[\sigma']|_{\text{ACTIVE}(c)}$, as desired.

Moreover, if $|[\sigma.c.s]|_i \neq |[\sigma]|_i$, we have

$$|[\sigma.c.s]|_i = ([\sigma.c.s]_i , |[\sigma]|_{\text{ACTIVE}(c)}) \neq \text{INIT}_i .$$

□

Because of the previous theorems, by quantifying over the set of non-randomized schedulers $\text{SDIST}_P(|[\cdot]|, \leqslant)$, we obtain a safe estimation of the value $\text{SDIST}_P([\cdot], \text{RATE})$. This is formalized in the following theorem.

THEOREM 4.6.

$$\sup_{\eta \in \text{SDIST}_P([\cdot],\text{RATE})} \text{PR}^\eta(S) \leqslant \sup_{\eta \in \text{SDIST}_P(|[\cdot]|,\leqslant)} \text{PR}^\eta(S)$$

for all measurable S.

Proof. Theorems 4.4 and 2.7 imply

$$\sup_{\eta \in \text{SDIST}_P([\cdot],\text{RATE})} \text{PR}^\eta(S) \leqslant \sup_{\eta \in \text{SDIST}_P(|[\cdot]|,\text{RATE})} \text{PR}^\eta(S) .$$

Projection $|[\cdot]|$ is traceable (Theorem 4.5), and so Theorem 4.3 (applied to $|[\cdot]|$) yields

$$\sup_{\eta \in \text{SDIST}_P([\cdot],\text{RATE})} \text{PR}^\eta(S) \leqslant \sup_{\eta \in \text{SDIST}_P(|[\cdot]|,\leqslant)} \text{PR}^\eta(S) .$$

□

4.2.4 *Proof of Theorem 4.3*

The overall structure of the proof resembles that of Theorem 4.1, in the sense that we first consider the "finite-horizon" sets of the form $\biguplus_i (\sigma_i)^\uparrow$ and then we extend the result to the supremum of arbitrary measurable sets using Lemma 4.8.

Let $\eta \in S$. We start by proving

$$\exists \eta' \in \text{SDIST}_P([\cdot], \leqslant) : \text{PR}^{\eta'}(\biguplus_{m=1}^{M} (\sigma^m)^\uparrow) \leqslant \text{PR}^\eta(\biguplus_{m=1}^{M} (\sigma^m)^\uparrow) . \tag{4.18}$$

The proof of Eqn. (4.18) resembles Lemma 4.7, since we give a construction that, starting from η, delivers a new scheduler in each step. However, the construction for interleaving schedulers in Lemma 4.7 cannot be used for total order-based schedulers, since such construction does not ensure that the resulting interleaving scheduler is total order-based. In order to produce a total order-based scheduler, we transform the interleaving scheduler using local paths, as explained in the following: in the first step, the scheduler is $\eta^0 = \eta$. At the q-th step in the construction we transform a scheduler

$$\eta^q = (\mathcal{J}^q, \{\Theta_i^q\}_i, \{\Upsilon_i^q\}_i)$$

into a scheduler

$$\eta^{q+1} = (\mathcal{J}^{q+1}, \{\Theta_i^q\}_i, \{\Upsilon_i^q\}_i) . \tag{4.19}$$

The transformation depends on a set S_G^q of global paths. The definition of S_G^q is different for cases *(I)* and *(II)*. First, we explain the rationale behind both cases, and then we prove the particularities in Claims 4.1 (for case *(I)*) and 4.2 (for case *(II)*) below.

In both cases, S_G^q complies with $(\sigma)^\uparrow \cap (\sigma')^\uparrow = \emptyset$ for all $\sigma, \sigma' \in S_G^q$, $\sigma \neq \sigma'$ and

$$\forall \sigma \in S_G^q, A_i, n < \text{LEN}(\sigma) : [\sigma_{\downarrow n}]_i = [\sigma]_i \implies \mathcal{J}(\sigma_{\downarrow n})(A_i) = 0 \,. \tag{4.20}$$

Using S_G^q, we define the set

$$S_L^q = \{[\sigma]_i \mid \sigma \in S_G^q \wedge \text{PR}^{\eta^q}((\sigma)^\uparrow) > 0 \wedge \left| G_i(\text{LAST}(\sigma)) \right| > 0\}$$

comprising the local paths in S_G^q in which there are generative transitions enabled. At each step q, we choose a local path $\sigma_{i^*}^q \in S_L^q$.

The interleaving scheduler $\mathcal{J}^{q+1}$ in Eqn. (4.19) is defined as

$$\mathcal{J}^{q+1} = \text{NR}(\mathcal{J}^q, S_G^q, \sigma_{i^*}^q) \,, \tag{4.21}$$

where $\text{NR}(\cdot)$ is defined as follows:

I. $\text{NR}(\mathcal{J}, S_G^q, \sigma_{i^*})(\sigma)(A_{i^*}) = 1$ if $[\sigma]_{i^*} = \sigma_{i^*}$ and $\sigma \in S_G^q$ and

II. $\text{NR}(\mathcal{J}, S_G^q, \sigma_{i^*})(\sigma)(A_i) = \mathcal{J}(\sigma)(A_i)$ for all σ, $A_i \in \text{ATOMS}(P)$, such that either $\sigma \notin S_G^q$ or $[\sigma]_{i^*} \neq \sigma^*$ and

III. $\text{NR}(\mathcal{J}, S_G^q, \sigma_{i^*})(\sigma)(A_i) = 0$ if $[\sigma]_{i^*} = \sigma_{i^*}$, $\sigma \in S_G^q$ and $A_i \neq A_{i^*}$.

(Note that the definition does not depend on the particular q, and so we dropped the superscript.)

Next, we explain how a total order can be constructed using the paths $\sigma_{i^*}^q$. In $\mathcal{J}^{q+1}$, the atom A_{i^*} is chosen in all paths $\sigma \in S_G^q$ such that $[\sigma]_{i^*} = \sigma_{i^*}^q$, thus "winning" over all other paths in S_L^q. It suggests that the total order must comply with:

$$\forall \sigma'_j \in S_L^q : \sigma'_j \neq \sigma_{i^*}^q \implies \sigma_{i^*}^q < \sigma'_j \,, \tag{4.22}$$

that is, $\sigma_{i^*}^q$ is smaller than all the paths in S_L^q. Local paths in S_L^q might appear in S_L^{q+n} for some $n > 1$ and, in fact, it might be the case that $\sigma_{j^*}^{q+n} \in S_L^q$. In this case, by Eqn. (4.22),

$$\sigma_{i^*}^q < \sigma_{j^*}^{q+n}$$

This suggests that we obtain a scheduler η' based on a total order in which

$$\sigma_{i_1^*}^1 < \cdots < \sigma_{i_q^*}^q \tag{4.23}$$

for all steps q (later on, we show that the number of steps is finite).

We need to ensure the existence of a scheduler η' such that η' is based on an order as in (4.23) and η' complies with (4.18). The existence of such an η' is ensured by properties *(i)*, *(ii)*, *(iii)* below. The proof of these properties depends on the set S_G^q, which in turn depends on whether we are under the hypothesis *(I)* or *(II)*. Later on, we prove these properties in Claims 4.1 and 4.2.

i. for all q, a proper $\sigma^q_{i^*} \in S^q_L$ exists so that the resulting scheduler yields less probability, that is,

$$\mathrm{PR}^{\eta^{q+1}}\Big(\biguplus_{m=1}^{M} (\sigma^m)^\uparrow\Big) \leqslant \mathrm{PR}^{\eta^q}\Big(\biguplus_{m=1}^{M} (\sigma^m)^\uparrow\Big)\,,$$

where is η^{q+1} is as in Eqn. (4.19) and $\mathcal{I}^{q+1}$ is as in Eqn. (4.21).

ii. after a finite number $\mathcal{Q}$ of steps, the interleaving scheduler is non-randomized for all paths σ such that $\mathrm{LEN}(\sigma) \leqslant \max_m \mathrm{LEN}(\sigma^m)$.

iii.

$$\forall n \geqslant 1 : \sigma^q_{i^*} \notin S^{q+n}_L\,. \tag{4.24}$$

Suppose that $\sigma^q_{i^*} \in S^{q+n}_L$. Then, it might be the case that $\sigma^{q+n}_{j^*} \neq \sigma^q_{i^*}$. Since the total order complies with (4.22) and (4.23) then

$$\sigma^q_{i^*} < \sigma^{q+n}_{j^*} < \sigma^q_{i^*}$$

Intuitively, we must ensure that the "winning" path $\sigma^q_{i^*}$ does not compete anymore in the future, since it might be defeated by another path $\sigma^{q+n}_{j^*}$, and so $\leqslant$ would not be a total order.

In the proof of claims 4.1 and 4.2, property *(ii)* is proved by showing the following properties:

$$\forall \sigma_i \in S^q_L : \exists \sigma \in S^q_G : [\sigma]_i = \sigma_i \wedge 0 < \mathcal{I}^q(\sigma)(A_i) < 1 \tag{4.25}$$

and

$$\forall \sigma, A_i : \mathcal{I}^q(\sigma)(A_i) = 1 \implies \mathcal{I}^{q+1}(\sigma)(A_i) = 1 \quad , \tag{4.26}$$

In fact, Eqn. (4.25) ensures that at least one randomized choice in $\mathcal{I}^q$ is non-randomized in $\mathcal{I}^{q+1}$ (namely, the one corresponding to $\sigma^q_{i^*}$) and Eqn. (4.26) ensures that the number of non-randomized choices cannot decrease during the construction. Hence, property *(ii)* follows from the fact that there is a finite number of paths having length at most $\max_m \mathrm{LEN}(\sigma^m)$.

Properties *(i)*, *(ii)* and *(iii)* above make it possible to transform η into a scheduler η' such that η' complies with (4.18). Moreover, we can find a total order $\leqslant$ complying with (4.23) in such a way that η' is based on $\leqslant$. Note that (4.23) says nothing about the local paths σ_j such that $\sigma_j \neq \sigma^q_{i^*_q}$ for all q. For such paths σ_j, the order can be defined arbitrarily, since these paths do not occur in global paths σ such that $\mathrm{LEN}(\sigma) \leqslant \max_m \mathrm{LEN}(\sigma^m)$, that is,

$$\forall \sigma : \mathrm{LEN}(\sigma) \leqslant \max_m \mathrm{LEN}(\sigma^m) \implies [\sigma]_j \neq \sigma_j\,.$$

Next, we generalize (4.18) to infinite families $\{\sigma^m\}_{m=1}^\infty$:

$$\forall \eta \in \mathcal{S} : \exists \eta' \in \mathrm{SDIST}_P([\cdot], \leqslant) : \mathrm{PR}^{\eta'}(S) \leqslant \mathrm{PR}^{\eta}(S) \tag{4.27}$$

for all $S = \biguplus_{m=1}^\infty (\sigma)^\uparrow$.

In order to prove (4.27), let $S_M = \biguplus_{\{\sigma \in S \,|\, \mathrm{LEN}(\sigma) < M\}} (\sigma)^\uparrow$. Then, by (4.18),

$$\forall \eta \in \mathcal{S} : \exists \eta^M \in \mathrm{SDIST}_P([\cdot], \leqslant) : \mathrm{PR}^{\eta'}(S_M) \leqslant \mathrm{PR}^{\eta}(S_M)\,.$$

Since the η^M are non-randomized, Lemma 4.8 ensures that there exists η^d such that η^d is a limit of $\{\eta^M\}_M$ and

$$\mathrm{PR}^{\eta^d}(S) \leqslant \mathrm{PR}^{\eta}(S) \,.$$

Since, for all M, $\eta^M \in \mathrm{SDIST}_P([\cdot], \leqslant)$, Theorem 3.1 implies $\eta^d \in \mathrm{SDIST}_P([\cdot], \leqslant)$, and so Eqn. (4.27) holds.

By virtue of Eqn. (4.27), we can apply Theorem 4.2, and so the result holds provided that properties *(i)*, *(ii)* and *(iii)* hold for both cases *(I)* and *(II)*. Next, we tackle each case separately.

Claim 4.1 (Case *(I)*). *Properties (i), (ii) and (iii) in p. 83 hold under the hypothesis (I) of Theorem 4.3.*

Proof. First, we show how to construct the sets S_G^q. Such sets are obtained from *fringes*.

Fringe

Definition 4.8. A *fringe* $\mathcal{F}$ is a finite set $\{\sigma^m\}_m$ such that $\forall_{m \neq m'} (\sigma)^\uparrow \cap (\sigma')^\uparrow = \emptyset$ and $\biguplus_m (\sigma^m)^\uparrow = (\textsc{init})^\uparrow$. A subfringe is a finite set that only complies $(\sigma)^\uparrow \cap (\sigma')^\uparrow = \emptyset$. The spawn of a fringe $\mathcal{F}$ generated by $F \subseteq \mathcal{F}$ is the fringe

$$\textsc{spawn}(\mathcal{F})(F) = \{\sigma \in \mathcal{F} \mid \sigma \notin F\} \biguplus \{\sigma.c.s \mid \sigma \in F\} \,.$$

We also say that $\mathcal{F}$ is spawned for all elements in F.

In addition, $\mathcal{F} \sqsubseteq \mathcal{F}'$ iff for all $\sigma^m \in \mathcal{F}$ there exists $\sigma'_{m'} \in \mathcal{F}'$ such that $\sigma^m \sqsubseteq \sigma'_{m'}$.

In each step of the construction, we consider a fringe $\mathcal{F}^q$ such that

$$\exists A_i \in \textsc{atoms}(P), \sigma \in \mathcal{F}^q \,:\, \mathrm{PR}^{\eta^q}((\sigma)^\uparrow) > 0 \wedge 0 < \mathcal{I}^q(\sigma)(A_i) < 1 \,. \tag{4.28}$$

In order to construct $\mathcal{F}^1$, we start with the fringe $\mathcal{F}^{0+} = \{\textsc{init}\}$. In general, we will obtain $\mathcal{F}^{q+1}$ from a fringe $\mathcal{F}^{q+}$ as follows. If

$$\forall \sigma \in \mathcal{F}^{q+} \,:\, \big(\textsc{len}(\sigma) \leqslant \max_m \textsc{len}(\sigma^m) \implies \exists A_{i_\sigma} \,:\, \mathcal{I}^q(\sigma)(A_{i_\sigma}) = 1\big) \,,$$

we consider some local path $\sigma_{i'}$ such that

$$\exists \sigma \in \mathcal{F}^{q+} \,:\, \textsc{len}(\sigma) \leqslant \max_m \textsc{len}(\sigma^m) \wedge \mathrm{PR}^{\eta}((\sigma)^\uparrow) > 0 \\ \wedge\, [\sigma]_{i'} = \sigma_{i'} \wedge \mathcal{I}^q(\sigma)(A_{i'}) = 1 \,.$$

Note that, since η is rate-based, $\mathcal{I}^q(\sigma)(A_{i'}) > 0$ implies $\mathcal{I}^q(\sigma')(A_{i'}) > 0$ for all σ' such that $[\sigma']_{i'} = \sigma_{i'}$, and so $\mathcal{I}^q(\sigma')(A_{i'}) = 1$ for all such σ' in $\mathcal{F}^q$.

We spawn $\mathcal{F}^{q+}$ for all the global paths σ such that $[\sigma]_{i'} = \sigma_{i'}$, obtaining $\mathcal{F}^{q++}$. This spawning can be repeated until either all paths in the fringe $\mathcal{F}^{q+\cdots+}$ have length greater than $\max_m \textsc{len}(\sigma^m)$ (and so we can finish the construction by letting $S_G^Q = \mathcal{F}^{q+\cdots+}$ —see property *(ii)*) or Eqn. (4.28) holds. In order to obtain $\mathcal{F}^{q+}$ from $\mathcal{F}^q$, we consider the local path σ_{i^*} obtained in the q-th step and we spawn $\mathcal{F}^q$ for all paths σ such that $[\sigma]_{i^*} = \sigma_{i^*}$.

In order to define S_G^q from $\mathcal{F}^q$, we start by defining

$$S_G^{q-} = \{\sigma \in \mathcal{F}^q \mid \exists A_i \,:\, 0 < \mathcal{I}(\sigma)(A_i) < 1 \wedge \textsc{len}(\sigma) \leqslant \max_m \textsc{len}(\sigma^m) \\ \wedge\, \mathrm{PR}^{\eta^q}((\sigma)^\uparrow) > 0\} \,.$$

Equation (4.28) ensures that S_G^{q-} is nonempty. Then, let

$$S_G^q = S_G^{q-} \cup \{\sigma \in \mathcal{F}^q \mid \text{LEN}(\sigma) \leqslant \max_m \text{LEN}(\sigma^m) \wedge \text{PR}^{\eta^q}((\sigma)^\uparrow) > 0 \wedge \exists \sigma' \in S_G^{q-}, A_j \in \text{ATOMS}(P) : [\sigma]_j = [\sigma']_j\} .$$

Note that, by construction, the sets S_G^q comply with Eqn. (4.25).

Now we show that requirement *(i)* is fulfilled. To this end, we resort to the following theorem, whose proof is in Appendix A.

THEOREM 4.7. *Let S_G be a set of finite global paths such that, for all $\sigma^m, \sigma^n \in S_G$, $\sigma^m \neq \sigma^n$ it holds $(\sigma^m)^\uparrow \cap (\sigma^n)^\uparrow = \emptyset$. Let $\eta \in \text{DIST}_P([\cdot])$ having interleaving scheduler $\mathcal{I}$, and let*

$$S_L = \{[\sigma]_i \mid \sigma \in S_G \wedge \text{PR}^{\eta^q}((\sigma)^\uparrow) > 0 \wedge \left| G_i(\text{LAST}(\sigma)) \right| > 0\} .$$

If there exist functions $R_i : S_L \cap \text{LOCALPATHS}_i \to \mathbb{R}_{\geqslant 0}$ such that

$$\forall \sigma \in S_G : \mathcal{I}(\sigma)(A_i) = \frac{R_i([\sigma]_i)}{\sum_{i'} R_i([\sigma]_{i'})} ,$$

then there exists $\sigma_{i^*} \in S_L$ such that $R_{i^*}(\sigma_{i^*}) > 0$ and the scheduler η' in which $\mathcal{I}$ is replaced by $\text{NR}(\mathcal{I}, S_G, \sigma_{i^*})$ complies with

$$\text{PR}^{\eta'}(\biguplus_{\sigma^m \in S_G} (\sigma^m)^\uparrow) \leqslant \text{PR}^{\eta}(\biguplus_{\sigma^m \in S_G} (\sigma^m)^\uparrow) .$$

Since the schedulers we are dealing with are rate based, we can take $R_i = \text{RATE}_i$ for all i. Hence, the $\sigma_{i^*}^{q+1}$ we need is the one provided by the theorem.

In order to fulfill the requirement *(ii)*, we resort to Equation (4.25) and Equation (4.26). We have already shown that the former holds. The latter holds since the scheduler $\mathcal{I}^{q+1}$ is obtained using Theorem 4.7 (recall that such theorem ensures $R_i(\sigma_{i^*}^q) > 0$).

With respect to the requirement *(iii)*, we resort to the following lemma (for the proof, see Appendix A).

LEMMA 4.10. *Let $[\cdot]$ be traceable. Let $\mathcal{F}^1, \cdots, \mathcal{F}^K$ be a set of fringes and $\eta^1, \cdots, \eta^K$ be a set of schedulers such that, $\mathcal{F}^1 = \{\text{INIT}_P\}$ and for all k, there exists $\sigma_{i^*}^k$ such that*

$$\mathcal{F}^{k+1} = \text{SPAWN}(\mathcal{F}^k)(\{\sigma \mid [\sigma]_{i^*} = \sigma_{i^*}^k\}) \neq \mathcal{F}^k ,$$

and for all $k' \geqslant k$

$$\forall \sigma \in \mathcal{F}^k : [\sigma]_{i^*} = \sigma_{i^*}^k \implies \mathcal{I}^{k'}(\sigma)(A_{i^*}) = 1 \tag{4.29}$$

and

$$\forall \sigma : \text{PR}^{\eta^{k'}}((\sigma)^\uparrow) > 0 \implies \text{PR}^{\eta^k}((\sigma)^\uparrow) > 0 . \tag{4.30}$$

Then, for all k, $0 < n < K - k$,

$$\forall \sigma \in \mathcal{F}^{k+n} : \text{PR}^{\eta^{k+n}}((\sigma)^\uparrow) > 0 \implies \sigma_{i^*}^k \neq [\sigma]_{i^*} .$$

Because of the definition of S_G^q for this case, this lemma implies Eqn. (4.24). The fringes $\mathcal{F}^k$ in the hypothesis are obtained exactly as in our construction, and Eqn. (4.29) holds since, in each step, we change the value of $\mathcal{I}$ for the values beyond the current fringe. Equation (4.30) holds since, in each step, we take an atom A_i such that $0 < \mathcal{I}(\sigma)(A_i) < 1$ and define $\mathcal{I}(\sigma)(A_i) = 1$. So, in each new scheduler there are no new paths with positive probability. □

CLAIM 4.2 (Case *(II)*). *Properties (i), (ii) and (iii) in p. 83 hold under the hypothesis (II) of Theorem 4.3.*

Proof. Recall that we are dealing with sets of the form $\biguplus_{m=1}^{M} (\sigma^m)^{\uparrow}$. Let σ be such that $\text{LEN}(\sigma) \leqslant \max_m \text{LEN}(\sigma^m)$ and $0 < \mathcal{J}^q(\sigma)(A_i) < 1$ for some A_i. Consider any other path σ' such that $[\sigma]_j = [\sigma']_j$ for some A_j. If both A_i, A_j have enabled transitions in σ, then $[\cdot]$ must be equivalent for these atoms, and so $[\sigma]_i = [\sigma']_i$. Then (by (1.9)), A_i has also transitions enabled in σ'. In general, for all σ, σ', A_j such that $\left|\, G_j(\text{LAST}(\sigma)) \,\right| > 0$ and $[\sigma]_j = [\sigma']_j$, we have

$$\{A_i \mid \left|\, G_i(\text{LAST}(\sigma)) \,\right| > 0\} = \{A_i \mid \left|\, G_i(\text{LAST}(\sigma')) \,\right| > 0\}$$

and

$$\forall A_i : \left|\, G_i(\text{LAST}(\sigma)) \,\right| > 0 \implies [\sigma]_i = [\sigma']_i \,.$$

Hence, since we are dealing with strongly distributed schedulers,

$$\forall A_i : \mathcal{J}^q(\sigma)(A_i) = \mathcal{J}^q(\sigma')(A_i) \tag{4.31}$$

for all σ, σ', such that $[\sigma]_j = [\sigma']_j$ for some A_j.

In each step, we start to construct S_G^q by selecting a path σ^s such that $\text{LEN}(\sigma^s) \leqslant \max_m \text{LEN}(\sigma^m)$, σ^s is minimal with respect to the prefix order and $0 < \mathcal{J}^q(\sigma^s)(A_i) < 1$. Then,

$$S_G^q = \{\sigma \mid \exists A_j : \left|\, G_j(\text{LAST}(\sigma^s)) \,\right| > 0 \wedge [\sigma^s]_j = [\sigma]_j\} \,.$$

So, the set S_L^q comprising all the local paths in S_G^q has at most one local path for each atom.

We obtain the requirement *(i)* by using Theorem 4.7. For all $\sigma_i \in S_L^q$ we define $R_i(\sigma_i) = \mathcal{J}^q(\sigma^*)(A_i)$, where σ^* is any path in S_G^q. We prove that these functions are suitable for Theorem 4.7. In fact, if $\sigma \in \sigma_G^q$, then

$$\begin{aligned}
&\frac{R_i([\sigma]_i)}{\sum_j R_j([\sigma]_j)} \\
&= \{ \text{ Definition of } R_j \} \\
&\quad \frac{\mathcal{J}^q(\sigma^*)(A_i)}{\sum_j \mathcal{J}^q(\sigma^*)(A_j)} \\
&= \{ \textstyle\sum_j \mathcal{J}^q(\sigma^*)(A_j) = 1 \} \\
&\quad \mathcal{J}^q(\sigma^*)(A_i) \\
&= \{ \text{ Equation (4.31)} \} \\
&\quad \mathcal{J}^q(\sigma)(A_i)
\end{aligned}$$

as required.

Requirement *(ii)* is proven via Eqn. (4.25) and Eqn. (4.26). Equation (4.25) holds by construction of S_G^q and Eqn. (4.26) holds since the resulting scheduler is obtained using Theorem 4.7.

Next, we prove the requirement *(iii)*. Suppose, towards a contradiction, that $\sigma_{i^*}^q \in S_L^{q+k}$ for some $k > 0$. Then, let σ^{q+k} be the path in S_L^{q+k} such that $\text{PR}^{\eta^{q+k}}((\sigma^{q+k})^{\uparrow}) > 0$ and $[\sigma^{q+k}]_{i^*} = \sigma_{i^*}^q$. Since $0 < \mathcal{J}^{q+k}(\sigma^{q+k})(A_{i^*}) < 1$, we have $\sigma^{q+k} \notin S_G^q$ (otherwise, it would be $\mathcal{J}^{q+k}(\sigma^{q+k})(A_z) = 1$ for some A_z). Since in the construction of S_G^q we considered all the minimal paths whose

projection over A_{i^*} is $\sigma^q_{i^*}$, there must be σ' such that $\sigma' \sqsubset \sigma$ and $\sigma' \in S^q_G$. Since $\Pr^{\eta^{q+k}}((\sigma^{q+k})^\uparrow) > 0$ and $\mathrm{NR}(\mathcal{I}^q, S^q_G, \sigma^q_{i^*})$ chooses A_{i^*} in σ', we have that A_{i^*} generates the transition after σ', i. e. $\mathrm{ACTIVE}(\sigma^{q+k}\langle \mathrm{LEN}(\sigma')\rangle) = A_{i^*}$. Then, property (1.11) implies $\sigma_{i^*} = [\sigma^{q+k}]_{i^*} \neq [\sigma']_i = \sigma_{i^*}$, □

4.3 ON THE (NON)EXISTENCE OF A SCHEDULER YIELDING THE SUPREMUM PROBABILITY

In this section, as in the rest of this chapter (and the next chapter), we consider the projection $[\![\cdot]\!]$ since, as we explained explained in the introduction, our aim is to show that some of the well-known properties for full-information schedulers do not hold anymore if we restrict to the set of distributed schedulers. In this section we stick to the usual projection $[\![\cdot]\!]$, in order to show that the loss of these properties is not a consequence of our general approach to projections.

In the realm of full-information schedulers, for every reachability property there exists a Markovian non-randomized scheduler attaining the supremum probability [65]. This is not the case when restricting to the set of distributed schedulers. In fact, such maximizing distributed scheduler may not exist at all.

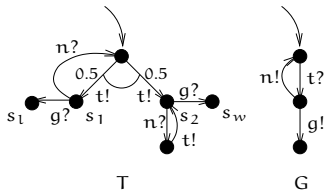


Figure 4.4: G has to guess that the coin has landed tails at least once

Consider the system consisting of atoms T and G in Fig. 4.4 (initial states are indicated by an incoming arrow). We show that, in this particular system, 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 (where n stands for "next"), 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 (which stands for "guess"). If T is in state s_2 and G outputs g, then the goal 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 the occurrence of only 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 yielding 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 distributed scheduler yielding probability one. So, although the supremum is 1, there is no scheduler yielding such probability.

4.4 FINITE-MEMORY (AND MARKOVIAN) SCHEDULERS

In the setting of full-information schedulers, the supremum probability of a reachability property can be calculated by considering *only* the set of Markovian schedulers [65]. We show that, under partial information, this is not true anymore.

In our setting, one may think of two types of Markovian schedulers: *globally Markovian* distributed schedulers, which choose the same distribution on compound transitions whenever the global states coincide, and *locally Markovian* distributed schedulers, which choose the same local transitions whenever the local states coincide.

DEFINITION 4.9. A scheduler is *globally Markovian* iff $\eta(\sigma)(c) = \eta(\sigma')(c)$ for all σ, σ' such that $\text{LAST}(\sigma) = \text{LAST}(\sigma')$.

DEFINITION 4.10. An input scheduler Υ_i is *Markovian* iff, for all a, r, it holds $\Upsilon_i(\sigma_i, a)(r) = \Upsilon_i(\sigma_i', a)(r)$ whenever $\text{LAST}(\sigma_i) = \text{LAST}(\sigma_i')$. The definition of Markovian output schedulers is similar. An interleaving scheduler $\mathcal{I}$ is *Markovian* iff $\mathcal{I}(\sigma)(A) = \mathcal{I}(\sigma')(A)$ whenever $\text{LAST}(\sigma) = \text{LAST}(\sigma')$ for all atom A. A scheduler η is locally Markovian iff the input, output and interleaving schedulers defining η are Markovian.

Note that $\text{LAST}(\sigma) = \text{LAST}(\sigma')$ iff $\forall A_i : [\![\text{LAST}(\sigma)]\!]_i = [\![\text{LAST}(\sigma')]\!]_i$ which, in turn, is equivalent to

$$\forall A_i : [\![\sigma]\!]_i = \sigma_i \cdot s_i \wedge [\![\sigma']\!]_i = \sigma_i' \cdot s_i \,.$$

Hence, an interleaving scheduler $\mathcal{I}$ is Markovian iff

$$\forall \sigma, \sigma' : \forall A_i : [\![\sigma]\!]_i = \sigma_i \cdot s_i \wedge [\![\sigma']\!]_i = \sigma_i' \cdot s_i \implies \mathcal{I}(\sigma) = \mathcal{I}(\sigma') \,. \quad (4.32)$$

We profit from this equivalence in the definition of locally N-Markovian schedulers.

DEFINITION 4.11. A scheduler is globally N-Markovian if $\eta(\sigma \cdot \sigma') = \eta(\sigma')$ for all σ' of length N.

Note that the set of globally Markovian schedulers equals the set of globally 1-Markovian schedulers.

DEFINITION 4.12. We say that an input scheduler Υ_i is N-Markovian iff $\Upsilon_i(\sigma_i \cdot \sigma_i', a) = \Upsilon_i(\sigma_i', a)$ for all a, σ_i and σ_i' such that $\text{LEN}(\sigma_i') = N$. The definition of N-Markovian output schedulers is similar. An interleaving scheduler is N-Markovian iff

$$\forall \sigma, \sigma' : \forall A_i : [\![\sigma]\!]_i = \sigma_i^1 \cdot \sigma_i^2 \wedge [\![\sigma']\!]_i = \sigma_i^{1'} \cdot \sigma_i^2 \wedge \text{LEN}(\sigma_i^2) = N \implies \mathcal{I}(\sigma) = \mathcal{I}(\sigma') \,.$$

A scheduler η is locally N-Markovian iff the input, output and interleaving schedulers defining η are N-Markovian.

Note that, by Eqn. (4.32), we have that an interleaving scheduler is 1-Markovian iff it is Markovian according to Def. 4.10.

The following example show that the supremum probability obtained among all possible locally Markovian schedulers does not agree with the supremum probability obtained among all distributed schedulers.

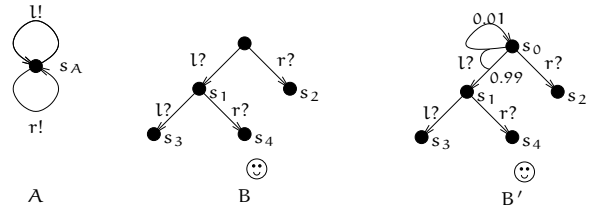


Figure 4.5: Atom A must lead B to the smiling state

EXAMPLE 4.4. Consider the system comprising atoms A and B in Fig. 4.5. First, we consider non-randomized schedulers. A non-randomized locally Markovian scheduler must output the same label in every path. So, if we quantify over *non-randomized* locally Markovian schedulers, the supremum probability of reaching ☺ is 0. The supremum under 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: once the first l has been output, the global state is different and so the scheduler can choose the transition that outputs r.

In the following example we show that globally Markovian schedulers are less expressive than distributed schedulers for reachability properties.

EXAMPLE 4.5. Consider the system comprising atoms A and B′ in Fig. 4.5.

Again, the aim of the scheduler is to reach ☺. Consider a scheduler η such that η is non-randomized and globally Markovian. In the initial state (s_A, s_0), atom A must output l. The label l must also be output in the path $(s_A, s_0).l!.(s_A, s_0)$, since the scheduler is globally Markovian. So, we have $\Theta_A((s_A.l!.s_A)) = l!$. Since the scheduler is distributed, this implies that l is also output in the path $(s_A, s_0).l!.(s_A, s_1)$. 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. Hence, the only globally Markovian distributed scheduler η reaches ☺ with probability 0. The distributed scheduler that chooses l and then r reaches ☺ with probability 0.99.

Although, for the sake of completeness, we have presented both globally and locally Markovian schedulers, handling distributed globally Markovian schedulers is quite difficult: we use Example 4.5 to illustrate that it is not possible to characterize globally Markovian schedulers in terms of the input/output schedulers. Consider any distributed scheduler η^d that chooses l and then r (regardless of the state of B): it is the loop in s_0 what causes the scheduler to be non-Markovian for $A \parallel B'$. Recall that this scheduler is indeed globally Markovian in the system $A \parallel B$. This shows that the set of globally Markovian schedulers is very sensitive to the structure of the system under consideration.

In addition, there is no condition in terms of the local schedulers to check that a globally Markovian scheduler is distributed: consider the globally

Markovian scheduler η^m choosing l! if B is in state s_0, and r! if B is in state s_1. This scheduler is not distributed because the loop implies the existence of the paths $\sigma^1 = (s_A, s_0).l.(s_A.s_0)$ and $\sigma^2 = (s_A, s_0).l.(s_A.s_1)$ complying $[\![\sigma^1]\!]_A = [\![\sigma^2]\!]_A$, while η^m chooses l! for one of the paths and r! for the other: the fact that the scheduler is not distributed is a consequence of the loop.

These issues make it difficult to deal with globally Markovian schedulers, and so in the rest of this Chapter we restrict to locally Markovian schedulers.

We say that a scheduler has (local) finite memory if it is (locally) N-Markovian for some N. We denote the finite-memory distributed schedulers of a system P by $\text{LFINMEM}(P)$ and the non-randomized finite-memory schedulers by $\text{NRLFINMEM}(P)$. We illustrate the limitations of finite-memory schedulers using atom A in Fig. 4.5. 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 number of r's is exactly the previous number 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 number of r's grows arbitrarily.

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 are strictly less expressive than (infinite-memory) distributed schedulers. However, if we consider only reachability properties, we obtain the following theorem.

THEOREM 4.8.

$$\forall U \subseteq \text{STATES}(P) : \sup_{\eta \in \text{DIST}_P([\![\cdot]\!])} \text{PR}^\eta(\text{REACH}(U)) = \sup_{\eta \in \text{NRLFINMEM}(P)} \text{PR}^\eta(\text{REACH}(U)) . \quad (4.33)$$

Proof. We prove the result by showing that, for all ϵ, there exists $\eta^m \in \text{NRLFINMEM}(P)$ such that $\sup_{\eta \in \text{DIST}_P([\![\cdot]\!])} \text{PR}^\eta(\text{REACH}(U)) - \text{PR}^{\eta^m}(\text{REACH}(U)) \leqslant \epsilon$.

Let $\epsilon > 0$, and let $\eta^s \in \text{DIST}_P([\![\cdot]\!])$ be a scheduler such that

$$\sup_{\eta \in \text{DIST}_P([\![\cdot]\!])} \text{PR}^\eta(\text{REACH}(U)) - \text{PR}^{\eta^s}(\text{REACH}(U)) < \epsilon/2 .$$

We denote by $\text{REACH}_N(U)$ the set of paths that reach some element in U before the N steps. Let N^* be such that $\text{PR}^{\eta^s}(\text{REACH}(U)) - \text{PR}^{\eta^s}(\text{REACH}_{N^*}(U)) < \epsilon/2$. Such an N^* is ensured to exist since

$$\text{PR}^{\eta^s}(\text{REACH}(U)) = \lim_{N \to \infty} \text{PR}^{\eta^S}(\text{REACH}_N(U)) .$$

The set REACH_{N^*} can be written as a disjoint finite union of sets of cylinders: in fact,

$$\text{REACH}_{N^*} = \biguplus \{ (\varpi)^\uparrow \mid \text{LEN}(\sigma) \leqslant N^* \wedge \text{LAST}(\sigma) \in U \wedge \forall k < \text{LEN}(\sigma) : \sigma(k) \notin U \} .$$

By Corollary 4.1, we know that

$$\sup_{\eta \in \text{NRDIST}_P([\![\cdot]\!])} \text{PR}^\eta(\text{REACH}_{N^*}(U)) = \sup_{\eta \in \text{DIST}_P([\![\cdot]\!])} \text{PR}^\eta(\text{REACH}_{N^*}(U)) .$$

Moreover, since for the set REACH_{N^*} the only relevant resolutions of the non-determinism are those before the N^*-th step (and since the schedulers in $\text{NRDIST}_P([\![\cdot]\!])$ are non-randomized), we have that the set $\{\text{PR}^\eta(\text{REACH}_{N^*}(U)) \mid \eta \in \text{NRDIST}_P([\![\cdot]\!])\}$ is finite, and so there exists $\eta^{NR} \in \text{NRDIST}_P([\![\cdot]\!])$ such that

$$\text{PR}^{\eta^{NR}}(\text{REACH}_{N^*}(U)) = \sup_{\eta \in \text{DIST}_P([\![\cdot]\!])} \text{PR}^\eta(\text{REACH}_{N^*}(U)) \geqslant \text{PR}^{\eta^s}(\text{REACH}_{N^*}(U)) \,.$$

Let Θ_i^{NR}, Υ_i^{NR} and $\mathcal{I}^{NR}$ be the schedulers that define η^{NR}. 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^{NR} up to the N^*-th step. The scheduler η^m obtained by composing Θ_i^m, Υ_i^m and $\mathcal{I}^m$ is N^*-Markovian, and we have:

$$\begin{aligned}
&\sup_{\eta \in \text{DIST}_P([\![\cdot]\!])} \text{PR}^\eta(\text{REACH}(U)) - \text{PR}^{\eta^m}(\text{REACH}(U)) \\
=\ &\sup_{\eta \in \text{DIST}_P([\![\cdot]\!])} \text{PR}^\eta(\text{REACH}(U)) - \text{PR}^{\eta^s}(\text{REACH}(U)) \\
&\qquad + \text{PR}^{\eta^s}(\text{REACH}(U)) - \text{PR}^{\eta^m}(\text{REACH}(U)) \\
<\ &\epsilon/2 + \text{PR}^{\eta^s}(\text{REACH}(U)) - \text{PR}^{\eta^m}(\text{REACH}(U)) \\
=\ &\epsilon/2 + \text{PR}^{\eta^s}(\text{REACH}(U)) - \text{PR}^{\eta^s}(\text{REACH}_{N^*}(U)) \\
&\qquad + \text{PR}^{\eta^s}(\text{REACH}_{N^*}(U)) - \text{PR}^{\eta^m}(\text{REACH}(U)) \\
<\ &\epsilon/2 + \epsilon/2 + \text{PR}^{\eta^s}(\text{REACH}_{N^*}(U)) - \text{PR}^{\eta^m}(\text{REACH}(U)) \\
\leqslant\ &\epsilon + \text{PR}^{\eta^{NR}}(\text{REACH}_{N^*}(U)) - \text{PR}^{\eta^m}(\text{REACH}(U)) \\
=\ &\epsilon + \text{PR}^{\eta^m}(\text{REACH}_{N^*}(U)) - \text{PR}^{\eta^m}(\text{REACH}(U)) \\
\leqslant\ &\epsilon + \text{PR}^{\eta^m}(\text{REACH}(U)) - \text{PR}^{\eta^m}(\text{REACH}(U)) \\
=\ &\epsilon \,.
\end{aligned}$$

□

Theorem 4.8 is false in case strongly distributed schedulers are considered: the example in Fig. 4.1 is also a counterexample for this theorem. This theorem can also be contrasted with the fact that, given a fixed amount of memory, randomized schedulers are needed.

4.5 DISCUSSION

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 *non-randomized* strongly distributed schedulers are strictly less expressive than strongly distributed schedulers, that is, randomization adds power to strongly distributed schedulers. Hence, randomization leads to higher (or lower) values in the worst-case probabilities than the non-randomized version.

However, as an interesting result, we proved that non-randomized *distributed* schedulers are equally expressive as distributed schedulers *for any measurable property*. Since traditional schedulers for MDPs are a particular case of distributed schedulers (just consider a distributed system having only one component) we conclude that non-randomized 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. [25, 133].

As pointed out in [38], 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 [122]. 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 [78]. 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 randomized schedulers are more powerful than non-randomized schedulers given a fixed amount of memory.

SUMMARY Table 1 summarizes some of the results in this chapter. It indicates whether or not a given subset is as expressive as the set of all distributed schedulers (strongly distributed schedulers, resp.). For example, the $\surd$ corresponding to "Distributed", "Infinite Memory" and "Non-randomized" indicates that non-randomized distributed schedulers are as powerful as distributed schedulers. The projection considered is $[\![\cdot]\!]$.

		Non-randomized	Possibly randomized
Distributed	N-Markovian	$\times$	$\times$
	Finite memory	$\times/\surd^{*}$	$\times/\surd^{*}$
	Infinite memory	$\surd$	$\surd^{\dagger}$
Strongly distributed	N-Markovian	$\times$	$\times$
	Finite memory	$\times$	$\times$
	Infinite memory	$\times$	$\surd^{\dagger}$

*: $\surd$ for reachability properties, $\times$ for general properties.
†: trivially true. This subset *is* the set of all distributed (strongly distributed, resp.) schedulers.

Table 1: Expressive subsets of schedulers

5

UNDECIDABILITY

In this chapter, we address the automatic calculation of the worst-case probability. In particular, we investigate to which extent such calculation is possible in case the information available to each of the components is partial, and we find several negative results.

We start with a quantitative problem, namely, to calculate the supremum probability that a state in a given set U is reached. Then, we consider the qualitative problem of deciding whether there exists a scheduler reaching some state in U with probability 1.

As in Sections 4.3 and 4.4, we restrict ourselves to projection $[\![\cdot]\!]$, in order to show that undecidability is not a consequence of our general approach to projections. Hence, our results also hold in other frameworks in which the projections are similar to $[\![\cdot]\!]$ [45, 64].

5.1 QUANTITATIVE CASE

Let $\text{REACH}(U)$ denote all infinite paths ω such that $\omega(i) \in U$ for some i. The aim of this section is to prove the following theorem:

THEOREM 5.1 (Approximation of the maximum reachability problem is undecidable). *Given an IPIOA P, a set U of states and $\delta > 0$, there is no algorithm that computes r such that $|r - \sup_{\eta \in \text{DIST}_P} \text{PR}^\eta(\text{REACH}(U))| < \delta$.*

In this theorem, the number δ plays the role of an error threshold. The theorem states that there is no algorithm to compute $\sup_{\eta \in \text{DIST}_P} \text{PR}^\eta(\text{REACH}(U))$ within a given threshold δ. Then, the value $\sup_{\eta \in \text{DIST}_P}$ cannot even be approximated.

The proof of Theorem 5.1 is based on the reduction of the maximum acceptance problem on probabilistic finite-state automata (PFA) [118] to the maximum reachability problem on IPIOA. Since the PFA maximum acceptance problem is undecidable, this reduction implies the undecidability of the maximum reachability problem on IPIOA.

DEFINITION 5.1. A *PFA* is a quintuple (Q, Σ, l, q_i, q_f) where Q is a finite set of *states* with $q_i, q_f \in Q$ being the *initial* and *accepting state* respectively, Σ is the *input alphabet*, and $l : \Sigma \times Q \to (Q \to [0, 1])$ is the *transition function* s.t. $l(a, q) \in \text{PROB}(Q)$ for all $a \in \Sigma$ and $q \in Q$. As in [118], we assume that q_f is absorbing, i.e. $l(a, q_f)(q_f) = 1$ for all $a \in \Sigma$.

Notice that l is a total function, thus implying that all labels are enabled in all states.

Before introducing the PFA maximum acceptance problem, we present the translation of PFA into IPIOA. By doing so, we can define the probability of accepting a word on a PFA using the translated model, thus avoiding a definition of probabilistic semantics specific to PFA.

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 Q. Moreover, A

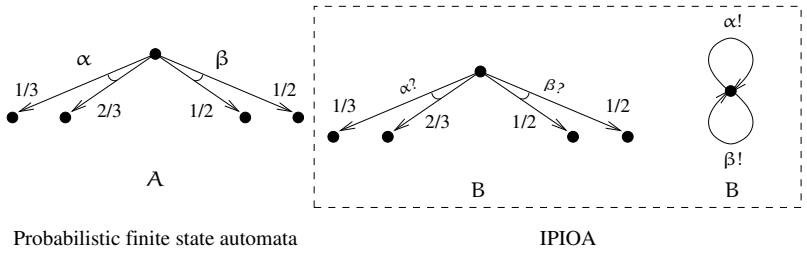


Figure 5.1: From PFA to IPIOA

encodes the transition function l 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 non-randomized scheduler for B that chooses the symbols in w.

The definition of $P_{\mathcal{A}}$ for each PFA $\mathcal{A}$ is formalized in the following.

DEFINITION 5.2 ($P_{\mathcal{A}}$ and the probability of accepting a word). Atom $A_{\mathcal{A}}$ is defined as the tuple $(Q, \Sigma, G_A, R_A, q_i)$. Both G_A and R_A have local enabledness conditions, and so we define them in terms of the local states (recall Def. 1.2, and contrast with Def. 1.12): $G_A(q) = \emptyset$ for all q, $R_A(s, \alpha) = \{l(\alpha, s)\}$ for all $\alpha \in \Sigma$.

Atom $B_{\mathcal{A}}$ is defined as the tuple $(\{\text{INIT}_B\}, \Sigma\}, G_B, R_B, \text{INIT}_B)$. Atom $B_{\mathcal{A}}$ has also local enabledness conditions:

$$G_B(\text{INIT}_B) = \{1 : (\alpha, \text{INIT}_B)\}_{\alpha \in \Sigma}$$

and $R_B(\text{INIT}_B, \alpha) = \{1 : \text{INIT}_B\}$. (This definition of R_B is needed by requirement Eqn. (1.1).)

The IPIOA $P_{\mathcal{A}}$ has $\text{ATOMS}(P_{\mathcal{A}}) = \{A_{\mathcal{A}}, B_{\mathcal{A}}\}$. Let $U = \{(q_f, \text{INIT}_B)\}$ be the set of accepting states. Then, the probability $\text{PR}(\text{accept } w)$ of accepting an infinite word $w = \alpha_1 \alpha_2 \cdots$ of symbols from Σ is $\text{PR}^{\eta}(\text{REACH}(U))$, where η is the only possible scheduler defined by the output scheduler Θ_B such that

$$\Theta_B(\overbrace{\text{INIT}_B \cdots \text{INIT}_B}^{n \text{ times}}) = 1 : (\alpha_n, \text{INIT}_B) \ .$$

Atom A is input deterministic, since it has exactly one enabled transition at every pair state/label. Hence there exists only one possible input scheduler for A (the scheduler choosing the only possible transition). In addition, it is also worth noting that, although we are dealing with infinite words, our criterion for acceptance is to *pass through* the accepting state using the word (i.e., a word is accepted iff a finite prefix reaches the accepting state). Note that, since q_f is absorbent (by Def. 5.1), this is equivalent to the criterion for Büchi automata.

Figure 5.1 shows a simple PFA and its corresponding IPIOA module.

Stated in terms of Def. 5.2, Corollary 3.4 in [118] becomes:

THEOREM 5.2 (Corollary 3.4 in [118]). *For any fixed $0 < \epsilon < 1$, the following problem is undecidable: Given a module $P_{\mathcal{A}}$ as in Def. 5.2 such that either*

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

decide whether case 1 holds.

In [118] it is pointed out that, as a consequence of Theorem 5.2, the approximation of the maximum acceptance probability is also undecidable. This statement is formalized in the following corollary.

COROLLARY 5.1 (Approximation of the maximum acceptance probability is undecidable). *Given* $P_{\mathcal{A}}$ *as in Def. 5.2 and* $\delta > 0$*, the following problem is undecidable: find* r *such that* $|r - \sup_w \Pr(\textit{accept } w)| < \delta$.

Proof. Note that, for an instance in Theorem 5.2 either $\sup_w \Pr(\text{accept } w) > 1 - \epsilon$ (iff case 1 holds) or $\sup_w \Pr(\text{accept } w) \leqslant \epsilon$ (iff case 2 holds).

We will assume, w.l.o.g., that $\epsilon < 1/4$. Suppose, towards a contradiction, that we can solve the problem for $\delta = 1/8$. We prove that, if the answer r is less than $1/2$, then case 2 holds. Otherwise, case 1 holds.

Let $S = \sup_w \Pr(\text{accept } w)$. If $r < 1/2$ then

$$
\begin{array}{ll}
 & S \\
< & \{ \text{ Condition imposed to } r \text{ by the specification of the algorithm } \} \\
 & r + 1/8 \\
< & \{ \text{ We assumed that } r < 1/2 \} \\
 & 1/2 + 1/8 = 5/8 < 3/4 < 1 - \epsilon.
\end{array}
$$

This is the opposite of case 1, and so case 2 holds.

If $r \geqslant 1/2$ then $S > \epsilon$, this is the opposite of case 2, and so case 1 holds. □

So far, we outlined the results in [118]. In order to prove Theorem 5.1, we show that infinite words can be seen as non-randomized schedulers (Lemma 5.1). Hence, the problem of finding the supremum over the set of words is equivalent to the problem of finding the supremum over the set of *non-randomized* distributed schedulers. By Corollary 4.1, this problem is equivalent to that of finding the supremum over the set of *all* distributed schedulers.

We note that, given the result in [118], the proof for non-randomized schedulers is not quite a relevant contribution. Our main contribution is to show that the undecidability result also holds for randomized schedulers, as a consequence of Corollary 4.1. (In [78], we prove this result using a restricted version of Corollary 4.1.)

The following lemma states that each word in the PFA $\mathcal{A}$ can be seen as a non-randomized scheduler in $P_{\mathcal{A}}$ and vice versa.

LEMMA 5.1 (Words and schedulers). *Given* $P_{\mathcal{A}}$ *as in Def. 5.2, each word* w *corresponds to a non-randomized scheduler* η *and vice versa, in the sense that* $\Pr(\textit{accept } w) = \Pr^{\eta}(\textsc{Reach}(U))$.

Proof. By definition, $\Pr(\text{accept } w) = \Pr^{\eta}(\textsc{Reach}(U))$, where η is defined by Θ_B as in Def. 5.2.

Conversely, let η be a non-randomized scheduler for $P_{\mathcal{A}}$. Let Θ_B be the scheduler that defines η. For any $n > 0$, there is exactly one local path σ_B having probability greater than 0 in η and $\textsc{Len}(\sigma_B) = n$. This is due to the fact that $B_{\mathcal{A}}$ has no probabilistic transitions. Then, take $w = w_1 \cdot w_2 \cdots$ to be the word defined by $w_n = \textsc{Label}(\sigma_B\langle n \rangle)$. □

Now we are ready to prove Theorem 5.1.

Proof (of Theorem 5.1). As a consequence of Lemma 5.1 and Corollary 5.1 the computation of the maximum reachability probability restricted to non-randomized schedulers is an undecidable problem in general, since it is undecidable for the particular case of IPIOA obtained from PFA as in Def. 5.2.

By Corollary 4.1, we have

$$\sup_{\eta \in \text{NRDIST}(P)} \text{PR}^{\eta}(\text{REACH}(U)) = \sum_{\eta \in \text{DIST}_P} \text{PR}^{\eta}(\text{REACH}(U)) \, .$$

Hence, if we were able to compute $\sum_{\eta \in \text{DIST}_P} \text{PR}^{\eta}(\text{REACH}(U))$ we would able to compute $\sup_w \text{PR}(\text{accept } w)$, thus contradicting Corollary 5.1. □

EXTENSION TO STRONGLY DISTRIBUTED SCHEDULERS Note that, in the systems $P_{\mathcal{A}}$ obtained in the reduction, only one of the atoms has generative transitions. So, for all $\mathcal{A}$ we have

$$\text{DIST}_{P_{\mathcal{A}}} = \text{SDIST}_{P_{\mathcal{A}}} = \text{SDIST}_{P_{\mathcal{A}}}([\![\cdot]\!], \text{RATE}) = \text{SDIST}_{P_{\mathcal{A}}}([\![\cdot]\!], \leqslant) \, .$$

This equality yields the following theorem.

THEOREM 5.3. *Let* S *be one of* SDIST_P, $\text{SDIST}_P([\![\cdot]\!], \text{RATE})$, *or* $\text{SDIST}_P([\![\cdot]\!], \leqslant)$. *There is no algorithm that computes* r *such that* $|r - \sup_{\eta \in S} \text{PR}^{\eta}(\text{REACH}(U))| < \delta$ *for all IPIOA* P, *set* U *and* $\delta > 0$

5.2 FINITE MEMORY SCHEDULERS

Theorem 4.8 states that, for reachability properties, the supremum under finite memory non-randomized schedulers is the same as under general distributed schedulers. Together with Theorem 5.1 we get that the supremum reachability problem is still undecidable if we restrict to finite-memory schedulers.

The problem for non-randomized schedulers having at most N memory is decidable, since this set of schedulers is finite, and so the supremum can be found by exhaustive exploration. However, if we want to use such schedulers to approximate the value under general distributed schedulers, the amount of memory N needed in order to get an accurate approximation of the probability under cannot be calculated. Formally, let $\text{NRLFINMEM}_N(P)$ be the set of non-randomized locally N-markovian schedulers for P. Then:

THEOREM 5.4. *Given* $\epsilon > 0$, *there is no algorithm computing* N *such that* $\sup_{\eta \in \text{DIST}_P([\cdot])} \text{PR}^{\eta}(\text{REACH}(U)) - \sup_{\eta \in \text{NRLFINMEM}_N(P)} \text{PR}^{\eta}(\text{REACH}(U)) < \epsilon$.

Proof. Suppose, towards a contradiction, that the problem is decidable. Since $\text{NRLFINMEM}_N(P)$ is finite, then there exists an algorithm to find a value r such that $\sup_{\eta \in \text{DIST}_P([\cdot])} \text{PR}^{\eta}(\text{REACH}(U)) - r < \epsilon$. Such algorithm proceeds in two steps: first, it computes N. Then it performs an exhaustive search on $\text{NRLFINMEM}_N(P)$ in order to look for the scheduler yielding the maximum probability (note that $\text{NRLFINMEM}_N(P)$ is a finite set). The existence of such algorithm contradicts Theorem 5.1. Since Theorem 5.1 holds also if we restrict to systems having only two atoms, we cannot compute N even for systems having only two atoms. □

Given that this theorem holds also if we restrict the problem to systems having only two atoms, it holds also in case we consider strongly distributed schedulers.

Even if we can obtain a suitable N (for instance, the choices after step N are irrelevant to the problem, or we are able to prove the probability of reaching U after N steps are negligible), then the problem of calculating $\text{NRLFINMEM}_N(P)$ is still complex, as shown by the following theorem.

THEOREM 5.5. *Let* S *be one of* $\text{LFINMEM}_1(P)$, $\text{NRLFINMEM}_1(P)$ $\text{GFINMEM}_1(P)$, $\text{NRGFINMEM}_1(P)$. *The problem of computing* $\sup_{\eta \in S} \text{PR}^\eta(\text{REACH}(U))$ *is NP-hard.*

Proof. We reduce the 3SAT problem to the supremum reachability problem. The following reduction was suggested by Peter Niebert [123]. 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

$$\text{FALSE}_k = 1:(b, (v_k, \text{FALSE})) \quad \text{and} \quad \text{TRUE}_k = 1:(b, (v_k, \text{TRUE})).$$

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. Then, the problem is NP-hard. Note that, since the system has no cycles, the set of Markovian distributed schedulers coincides with the set of comprising all schedulers. □

5.3 QUALITATIVE CASE

Section 4.3 presents an example in which the supremum of a reachability property is 1, while there is no scheduler yielding such probability. Hence, the problem of calculating whether the supremum is 1 is different to the problem of calculating whether there is a scheduler yielding probability 1. Unfortunately, both problems are undecidable.

The proofs in this section reduce the Post Correspondence Problem (PCP) to the qualitative reachability problem for IPIOA.

The PCP problem can be stated as follows: given words $u_1, \ldots, u_n$ and $v_1, \ldots, v_n$ over an alphabet $\mathcal{S}$. Is there a finite non-empty sequence of indices $k = k_1 \cdots k_m$ such that $u_{k_1} \cdots u_{k_m} = v_{k_1} \cdots v_{k_m}$?

Intuitively, we can think that we are given n blocks with two words, as shown in the following example:

aba	c
a	bacab
1	2

In this example, $u_1 = \text{aba}$, $u_2 = \text{c}$, $v_1 = \text{a}$ and $v_2 = \text{bacab}$. The sequence of indices 1, 2, 1 is a solution, since $u_1 \cdot u_2 \cdot u_1 = \text{abacaba} = v_1 \cdot v_2 \cdot v_1$.

We say that (w, k) is an upper pair iff $w = u_{k_1} \cdots u_{k_n}$. We say that (w, k) is a lower pair iff $w = v_{k_1} \cdots v_{k_n}$. Note that a word w can appear in an upper pair (in this case, we say that the word is an *upper word*) iff w is in the regular language $(u_1 + \ldots + u_n)^*$ (which we call the *upper language*), and similarly for the words that can appear in a lower pair. We denote by $\text{WL}(w)$ the length of a word w.

$\text{WL}(w)$

Then, an instance of the PCP problem has a solution iff there exists an upper pair (w, k) such that (w, k) is also a lower pair.

5.3.1 *Distributed schedulers*

In this subsection, we prove undecidability under distributed schedulers. Subsection 5.3.2 deals with strongly distributed schedulers.

THEOREM 5.6. *The following problems are undecidable: Given an IPIOA* P *and a set of states* U*,*

1. *decide whether or not* $\sup_{\eta \in \text{DIST}_P} \text{PR}^\eta(\text{REACH}(U)) = 1$.
2. *decide whether or not there exists* $\eta \in \text{DIST}_P$ *such that* $\text{PR}^\eta(\text{REACH}(U)) = 1$.

Proof. Our proof strategy is similar to the one in [144]: we reduce the Post correspondence problem (PCP), which is known to be undecidable.

Given a PCP instance $u_1, \ldots, u_n$, $v_1, \ldots, v_n$, we construct three atoms *W*, *S*, I. Roughly speaking, *W* probabilistically chooses either "upper" or "lower". If *W* chooses "upper", then *W* probabilistically chooses an upper word w , communicating the symbols in w to *S* and the indices k_i to I (and similarly if *W* chooses "lower"). Once w ends (the end of w is also decided probabilistically), then *W* outputs *stop*. After *stop*, I is able to output any sequence of indices to *S* (some of the behaviours we are interested in are those in which I communicates the indices it has received from *W*). Then, *S* has to guess whether *W* has chosen either "upper" or "lower". The set of states U is the set in which *S* has guessed correctly.

The set ACTLAB_W is $S \cup \{1, \cdots, n\} \cup \{stop, \tau_W\}$.[†] The behaviour of *W* is as follows: *W* has no nondeterministic choices. In the initial state there is a probabilistic output transition:

$$\frac{1}{2} : (\tau_W, initUp) + \frac{1}{2} : (\tau_W, initLo) \ .$$

The states *initUp* and *initLo* represent the fact that *W* has chosen "upper" or "lower" respectively. In *initUp* there is a probabilistic output transition

$$\frac{1}{n} : (1, startU_1) + \cdots + \frac{1}{n} : (n, startU_n) \ .$$

Note that the number k in $\frac{1}{n} : (k, startU_k)$ is a label in ACTLAB_W (recall that $\{1, \cdots, n\} \subseteq \text{ACTLAB}_W$). The states $startU_i$ represent the fact that the word w will start with u_i. Similarly, the states $startL_i$ represent the fact that word w will start with v_i. In each state $startU_i$ there is a transition $1 : (u_{i_1}, U_{i_1})$, where u_{i_1} is the first symbol in u_i [‡] and U_{i_1} represents the fact that the first symbol in U_i has been output. From each state U_{i_j} with $j < \text{WL}(u_i) - 1$ there is a transition $1 : (u_{i_{j+1}}, U_{i_{j+1}})$. In the state $U_{i_{\text{WL}(u_i)-1}}$, there is a transition $\frac{1}{2}(u_{i_{\text{WL}(w)}}, initUp) + \frac{1}{2}(u_{i_{\text{WL}(w)}}, outputStopWU)$. In state *outputStopWU*, the atom just goes to state *endWU* by outputting *stop*, that is, it has only one transition: $1 : (stop, endWU)$. The state *endWU* indicates that the upper word has ended. We omit the symmetric definitions for the case in which *W* chooses "lower" (in this case the respective states are $startL_i$, L_{i_j}, *outputStopWL* and *endWL*). Since *W* must be input-enabled, each state has input transitions for each $l \in \text{ACTLAB}_W$. However, because of the definition of the atoms, the paths in which the labels are output by other atoms have probability 0 for all schedulers, and so the definitions of the input transitions are irrelevant.

For atom I, we have

$$\text{ACTLAB}_I = S \cup \{1, \cdots, n\} \cup \{1', \cdots, n'\} \cup \{stop, stop'\} \ .$$

[†] As usual, the label τ_W is a placeholder for the output of internal transitions, that is, $\tau_i \notin \text{ACTLAB}_j$ if $i \neq j$.

[‡] For simplicity, we omitted the case in which some of the words u_k (v_k, resp.) are empty. In this case, when the index k is output in the state *initUp*, *W* returns to *initUp* instead of moving to $startU_k$.

The labels $\{1', \cdots n'\}$ are indices to be communicated to S. However, such labels must be different from the labels $\{1, \cdots, n\}$ output by *W*, since S is not allowed to observe such labels. We need the label *stop'* for the same reason: I needs to indicate the other atoms when it has stopped, and the construction is easier if the label used to indicate the stop is not the same as in *W*. In the initial state INIT_I, atom I reacts to all $1 \leqslant i \leqslant n$ using the input transition $1{:}\text{INIT}_I$, and it reacts to *stop* with the transition $1{:}\mathit{outputI}$. Other input transitions are irrelevant. In the state *outputI* there are output transitions $1{:}(i', \mathit{outputI})$ for each $1 \leqslant i \leqslant n$, and also a transition $1{:}(\mathit{stop'}, \mathit{endI})$.

The set ACTLAB_S is $\mathcal{S} \cup \{1', \cdots, n'\} \cup \{\mathit{stop'}, \tau_S\}$. In the initial state S reacts to the labels in $\mathcal{S} \cup \{1', \cdots, n'\}$ using the input transition $1{:}\text{INIT}_S$, and it reacts to *stop'* with the transition $1{:}\mathit{guessS}$. In *guessS* there are two output transitions: $1{:}(\tau_S, \mathit{tryUp})$ and $1{:}(\tau_S, \mathit{tryLo})$.

So, the set U to be reached is

$$U = \{(\mathit{endWU}, \mathit{endI}, \mathit{tryUp}) \,,\, (\mathit{endWL}, \mathit{endI}, \mathit{tryLo})\} \,.$$

We prove the following: the instance of the PCP problem does not have a solution iff there exists a distributed scheduler such that $\text{PR}^\eta(\text{REACH}(U)) = 1$. In addition, the problem has no solution iff $\sup_{\eta \in \text{DIST}_P} \text{PR}^\eta(\text{REACH}(U)) = 1$.

Suppose that the problem has no solution. Then every pair (w, k) can be an upper or a lower pair, but it cannot be both. We can construct the following distributed scheduler for P: input and output schedulers for *W* are uniquely defined (there are no nondeterministic choices). The output scheduler for I chooses the transitions that output the indices in the same order as they were output by *W*. The output scheduler for S has to decide only between going to *tryUp* or going to *tryLo*. The only paths with probability greater than 0 in which this choice is performed have a sequence of action labels of the form $a_1 \cdots a_q k'_1 \cdots k'_r \mathit{stop'}$. If $(a_1 \cdots a_q \,,\, k'_1 \cdots k'_r)$ is an upper pair, then the output scheduler chooses *tryUp*, otherwise it chooses *tryLo*. If the path has positive probability, and $a_1 \cdots a_r k'_1 \cdots k;_q$ is an upper pair then, by construction of *W*, *W* is in state *endWU*. Conversely, if $a_1 \cdots a_r k'_1 \cdots k'_q$ is a lower pair, then *W* is in state *endWL*, and so the scheduler we constructed reaches U with probability 1.

Now assume that the PCP problem has a solution. Suppose (towards a contradiction) that $\sup_{\eta \in \text{DIST}_P} \text{PR}^\eta(\text{REACH}(U)) = 1$ (to get case 2 in the theorem statement, suppose that there exists $\eta \in \text{DIST}_P$ such that $\text{PR}^\eta(\text{REACH}(U)) = 1$). Then, by Theorem 4.1 for every $\epsilon > 0$ there exists a non-randomized distributed scheduler η^ϵ such that

$$\text{PR}^{\eta^\epsilon}(\text{REACH}(U)) > 1 - \epsilon \,. \tag{5.1}$$

Since the PCP problem has a solution, let $(w, k = k_1 \cdots k_r)$ be an upper pair that is also a lower pair. Then, there exist two paths σ, σ' whose projection to I is of the form $k_1 \cdots k_r \mathit{stop}$ and, in one of them *W* has chosen "upper" while in the other one it has chosen "lower". By construction, we have

$$\text{PR}^{\eta^\epsilon}((\sigma)^\uparrow) = \text{PR}^{\eta^\epsilon}((\sigma')^\uparrow) = \frac{1}{2}\left(\frac{1}{n}\frac{1}{2}\right)^{r+1} \tag{5.2}$$

for all ϵ. We denote the probabilities in Eqn. (5.2) by δ. Note that these probabilities do not depend on the scheduler, since the nondeterministic

choice occurs after the word (as well as the indices output by I) is completed. Then, δ depends only on the PCP instance.

We show that $\Pr^{\eta^\epsilon}(\textsc{Reach}(U)) \leqslant 1-\delta$. After W has completed a word, the output scheduler for I starts to choose indices. It can either choose infinitely many indices, or finally output *stop'*. If *stop'* is never output after $[\![\sigma]\!]_I$, then a state in U cannot be reached after σ, and so the scheduler reaches U with probability less than or equal to $1-\delta$ as desired. If *stop'* is finally output, let $l_1 \cdots l_r$ *stop'* be the sequence of labels output by I after it has observed $[\![\sigma]\!]_I$ (note that $l_k \in \{1', \cdots, n'\}$ for all k). In both σ, σ' the projection to S is $w l_1 \cdots l_r$ *stop'*. Hence, if the output scheduler for W chooses "upper" in σ, then it also chooses "upper" in σ'. By construction, the set U cannot be reached after choosing "upper" in σ', and so η^ϵ reaches U with probability less than or equal than $1-\delta$. In symbols:

$$\Pr^{\eta^\epsilon}(\textsc{Reach}(U)) \leqslant 1-\delta\ .$$

The same happens in case the scheduler for W chooses "lower".

By taking any $\epsilon < \delta$, we obtain $\Pr^{\eta^\epsilon}(\textsc{Reach}(U)) \leqslant 1-\epsilon$, thus reaching a contradiction of (5.1). □

5.3.2 *Strongly distributed schedulers*

This subsection is devoted to prove the following analogous of Theorem 5.6.

THEOREM 5.7. *The following problems are undecidable: Given an IPIOA* P *and a set of states* U*,*

1. *decide whether or not* $\sup_{\eta \in \textsc{SDist}_P} \Pr^\eta(\textsc{Reach}(U)) = 1$.
2. *decide whether or not there exists* $\eta \in \textsc{SDist}_P$ *such that* $\Pr^\eta(\textsc{Reach}(U)) = 1$.
3. *decide whether or not* $\sup_{\eta \in \textsc{SDist}_P([\![\cdot]\!], \leqslant)} \Pr^\eta(\textsc{Reach}(U)) = 1$.
4. *decide whether or not there exists a scheduler* $\eta \in \textsc{SDist}_P([\![\cdot]\!], \leqslant)$ *such that* $\Pr^\eta(\textsc{Reach}(U)) = 1$.

Proof. We use the same idea as in the case of distributed schedulers. When proving such result, we defined three atoms W, S and I. Here, we reuse the atom W, except for a little modification explained later. The atom S is replaced by two atoms S_{Up} and S_{Lo}. Atom I is replaced by a set of atoms $\{I_i\}_{i=1}^n \cup \{I_{stop}\}$. The intended meaning is that S_{Up} and S_{Lo} are a team that must guess whether the word is an upper or a lower one, according to the same information that S receives in the other reduction (namely, the sequence of symbols output by W and the sequence of indices output by I, such sequence being now output by the team comprising atoms $\{I_i\}_{i=1}^n$). Atoms S_{Up} and S_{Lo} take the guess in the following fashion: if S_{Up} believes that it is an upper word, then it outputs u. Conversely, if S_{Lo} believes that it is a lower word, then it outputs l. So, both S_{Up} and S_{Lo} behave as S, until the point in which S decides, i.e. at the state *guessS*. In this state, S_{Up} has enabled the output transition $1{:}(u, tryUp)$, and S_{Lo} has enabled the transition $1{:}(l, tryLo)$. In W, the state *endWU* reacts to u with the input transition $1{:}good$ and reacts to l with $1{:}bad$. Conversely, the state *endWL* reacts to u with the input transition $1{:}bad$ and reacts to l with $1{:}good$. Then, the state *good* is

reached in the cases in which "the set of atoms $\{S_{Up}, S_{Lo}\}$" guesses correctly "upper" or "lower".

Each atom I_i has all the input transitions in I. In addition, in the initial state there is an input transition $1{:}endI_i$ reacting to *stop'*. The atom I_{stop} has all the input transitions in I, and only one output transition $1{:}(stop', endI_{stop})$. So, once I_{stop} decides to stop, all the I_i reach the state $endI_i$. Each atom I_i has enabled the output transition $1{:}(i', \text{INIT}_I)$.

Note that the set of atoms in this new reduction can be partitioned into the subsets $\mathcal{A} = \{W\}$, $\mathcal{B} = \{I_i\}_{i=1}^{n} \cup \{I_{stop}\}$ and $\mathcal{C} = \{S_{Up}, S_{Lo}\}$. At every state, all the atoms in exactly one of these sets have an output transition enabled. Moreover, $[\![\cdot]\!]$ is a projection equivalent for $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{B}$. The equivalence for $\mathcal{A}$ is trivial since $\mathcal{A}$ has only one element. The equivalence for $\mathcal{B}$ holds since, at every state, the projection for all the I_i and I_{stop} gives the symbols output by W. The equivalence for $\mathcal{C}$ holds since, at every state, the projection for both S_{Up} and S_{Lo} gives the symbols output by W and the indices output by the I_i. Then, Corollary 4.2 gives that $\text{SDIST}_P([\![\cdot]\!])$ and $\text{SDIST}_P([\![\cdot]\!], \leqslant)$ have the same expressive power.

Note that the IPIOA under consideration resembles that of Example 4.3, p 77

Therefore, we can repeat the argument in the proof for distributed schedulers in order to prove that the supremum probability of reaching *good* is 1 (there exists a scheduler reaching *good* with probability 1, resp.) iff the PCP problem has no solution. □

5.4 COMPARISON WITH EXISTING RESULTS

Theorem 5.1 also holds for the probabilistic modules in [64] (indeed, this is the framework we used in [78]), for the Switched PIOA in [46] and for the schema of partial information presented in [63] (we already discussed these frameworks in Sec. 1.4). The proof for Switched PIOA is essentially the same as ours, since the difference between both frameworks is related to the interleaving and interleaving is not an issue in our reduction.

In [63] the lack of information is modelled using a equivalence relation $\sim$ on states. Two paths σ, σ' are equivalent iff they have the same length and $\sigma(i) \sim \sigma'(i)$ for all i. The maximum acceptance problem for PFA can be reduced to this framework by taking $\sim$ to be the relation such that $s \sim s'$ for all s. Using this relation, the scheduler must decide the next action to perform based solely on the number of actions executed before, and so each scheduler in [63] is equivalent to a scheduler for the atom B as Def. 5.2. Hence, the problem of finding the maximum reachability probability is equivalent to the problem of finding the maximum reachability probability for an IPIOA as in Def. 5.2.

We remark that [63] defines a model checking algorithm, but it calculates the supremum corresponding to *Markovian* partial-information policies, i.e., to the subset of partial-information policies restricted to choose (distributions on) actions by reading only the (corresponding portion of the) current state rather than the full past history.

Probabilistic Büchi Automata (PBA [20]) are similar to PFA, but they have a different acceptance criterion. A path is accepting iff infinitely many states are accepting. A word is accepted iff the probability of the accepting paths is positive. In [16, 86], it is proven that both the emptiness problem (that is, whether the language accepted by an automaton is empty or not) and the

specification problem (that is, whether a finite transition system satisfies a PBA-specification) are undecidable.

Part II

TECHNIQUES AND ALGORITHMS

6 ALGORITHMS

Existing algorithms for verification of probabilistic systems are based on the compose-and-schedule approach explained in Subsection 0.1.3. In this approach, the system is verified under total information schedulers for the compound system.

In this chapter, we explore how model checking techniques can be adapted to deal with partial information. In Chapter 5, we have shown that there is no algorithmic solution to the general problem of verification under partial information. Despite of this negative result, we present two algorithms. One of them calculates an overestimation of the maximum probability that the system fails. The other one exhaustively explores the set of non-randomized distributed Markovian schedulers, in order to look for schedulers in which the probability of a failure is not acceptable. We present a branch-and-bound technique to elide some subsets of schedulers during the exploration.

The algorithms we present translate an IPIOA P into a Markov Decision Process (MDP) M. The aim of the translations in these algorithms is that the analysis of M under total information is useful to analyse P under partial information. Then, our algorithms for IPIOA profit from the well-known algorithms for MDPs under total information.

In this chapter, we make extensive use of the following definitions.

DEFINITION 6.1 (MDP). An *MDP* is a tuple $M = (S, \text{ACTIONS}, P, \text{INIT})$, where S is a finite set of states, ACTIONS is a finite set of actions identifiers, $P : (S \times \text{ACTIONS} \times S) \to [0,1]$ is the (three-dimensional) probability matrix, $\text{INIT} \in S$ is the initial state. $\text{ACTIONS}(s)$ denotes the set of actions enabled in state s, i.e. the set of actions α such that $P(s,\alpha,t) > 0$ for some $t \in S$. For every state $s \in S$, we require that $\text{ACTIONS}(s) \neq \emptyset$ and $\sum_{s' \in S} P(s,\alpha,s') = 1$ for every action $\alpha \in \text{ACTIONS}(s)$. (In particular, we assume that M does not have terminal states.)

To shorten notation, we often write $\alpha(s,s')$ instead of $P(s,\alpha,s')$.

A path in an MDP is a sequence $s_1.\alpha_1.\cdots.\alpha_{n-1}.s_n$. Schedulers for MDPs map paths to probability distributions on ACTIONS. The probability $\text{PR}^\eta((\sigma)^\uparrow)$ of a path $\sigma = s_0.\alpha_1.s_1\cdots\alpha_n.s_n$ is defined inductively: the probability of the initial state is 1. The probability of a path $\sigma.\alpha_n.s_n$ is $\text{PR}^\eta(\sigma)\cdot\eta(\sigma)(\alpha_n)\cdot P(\text{LAST}(\sigma),\alpha_n,s_n)$. SCHED_M denotes the set of schedulers for the MDP M.

ASSUMPTION 6.1. In this chapter, we restrict to simple IPIOA, that is, IPIOA as defined in Sec. 1.1. Simple IPIOA can be seen as general IPIOA complying with certain restrictions (see Subsection 1.2.3).

6.1 FROM IPIOA TO MDPS

Before introducing our algorithms, we present a straightforward translation from IPIOA to MDP. This translation is useful to follow that compose-and-schedule approach: the resulting MDP is the composition of the atoms in

the IPIOA, and the schedulers for the MDP correspond to full-information schedulers for the IPIOA.

Given an Interleaved PIOA P, we construct an MDP $\text{MDP}(P)$. The systems P and $\text{MDP}(P)$ are equivalent, in the sense that

$$\sup_{\eta \in \text{SCHED}_P} \text{PR}^\eta(\mathcal{S}) = \sup_{\eta \in \text{SCHED}_{\text{MDP}(P)}} \text{PR}^\eta(\mathcal{S}) \,.$$

Since there is no concept of action labels in the MDP setting, the states of the MDP contain the action label that led the MDP to such a state. A fictitious label $a_{\text{INIT}} \notin \text{ACTLAB}_P$ needs to be introduced, since the initial state has no previous label.

DEFINITION 6.2. The set of states of $\text{MDP}(P)$ is $S = (\text{ACTLAB} \cup \{a_{\text{INIT}}\}) \times \prod_i S_i$, where a_{INIT} is a fictitious label introduced because the initial state has no previous label. Each action in the MDP specifies a generative transition and, in addition, it specifies how the other atoms react to this generative transition. So, each element in ACTIONS has the form $(g_i, f_1, \cdots, f_{i-1}, f_{i+1}, \cdots, f_N)$, where g_i is a generative transition of the atom i and

$$f_j : \text{ACTLAB}_j \cap \text{ACTLAB}(g_i) \to T_{R_j}$$

$\text{ACTLAB}(g_i)$

where $\text{ACTLAB}(g_i) = \{a \mid \exists s, s' : g_i(s, a, s') > 0\}$ and T_{R_j} is the set of reactive transitions in A_j. Each action α 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 $c_{\alpha,a}$ for each label $a \in \text{ACTLAB}_i$. Such compound transition is defined as $(g_i, a, f_{r_1}(a), \cdots, f_{r_k}(a))$, where $r_1, \cdots, r_k$ are the atoms that react to a. An action α in the MDP is enabled in s iff the corresponding compound transitions are enabled in the Interleaved PIOA. The probability matrix is defined as

$$P(\,(a,s)\,,\,(g_i, f_1, \cdots, f_{i-1}, f_{i+1}, \cdots, f_N)\,,\,(a', (s'_1, \cdots, s'_N))\,) = g_i(s, a', s'_i) \cdot \prod_{k=1}^{m} f_{j_k}(a')(s, a', s'_{j_k}) \,. \quad (6.1)$$

The initial state is $(a_{\text{INIT}}, \text{INIT})$, where INIT is the initial state of the IPIOA. In order to comply with the restriction $\text{ACTIONS}(s) \neq \emptyset$, if s has no transitions enabled, we define $\text{ACTIONS}(s) = \{\alpha_\varsigma\}$ where $P(s, \alpha_\varsigma, s) = 1$ and $P(s, \alpha_\varsigma, s') = 0$ for all $s' \neq s$.

In case the MDP arises from an IPIOA, each state is of the form (a, s). In this case, we write

$$s_1.\alpha_1.a_2.s_2.\cdots.\alpha_{n_1}.a_n.s_n$$

instead of $s_1.\alpha_1.(a_2, s_2).\cdots.\alpha_{n_1}.(a_n, s_n)$. In addition, we write $\alpha(s, a, s')$ instead of $\alpha(s, (a, s'))$.

Note that, while IPIOA paths involve compound transitions, MDP paths involve actions. So, in order to draw a link between both formalisms, we have to restrict to properties that are sensitive only to states.

DEFINITION 6.3. We say that set of infinite paths $\mathcal{S}$ is state-based iff

$$\forall \omega, \omega' : (\forall k{:}\omega(k) = \omega'(k)) \implies \omega \in \mathcal{S} \iff \omega' \in \mathcal{S} \,.$$

The following theorem allows us to calculate extremal probabilities for a IPIOA under total information by calculating extremal probabilities for the corresponding MDP.

THEOREM 6.1. *Let $\mathcal{S}$ be a measurable, state-based set of infinite paths. Then,*

$$\sup_{\eta \in \text{SCHED}_P} \Pr^{\eta}(\mathcal{S}) = \sup_{\eta \in \text{SCHED}_{\text{MDP}(P)}} \Pr^{\eta}(\mathcal{S}) .$$

6.2 AN OVERESTIMATION FOR TOTAL ORDER-BASED SCHEDULERS

In this section, we present an algorithm for reachability properties on I/O deterministic IPIOA, that is, on IPIOA such that $\left| G_i(s) \right| \leqslant 1$ and $\left| R_i(s, a) \right| = 1$ for all i, s, a.

The correctness of our algorithm involves the total order-based schedulers under the projection $[\cdot]^{\text{VP}} \sqsupseteq [\![\cdot]\!]$. We have seen such projection already in Example 4.2, as an example of a traceable projection. The definition is restated below.

DEFINITION 6.4. For each atom A_i, the visible prefix projection $[\cdot]^{\text{VP}}$ is defined as follows:

- $[\text{INIT}]^{\text{VP}}_i = \text{INIT}$,
- $[\sigma.c.s]^{\text{VP}}_i = \sigma.c.s$ if $\text{LABEL}(c) \in \text{ACTLAB}_i$ and
- $[\sigma.c.s]^{\text{VP}}_i = [\sigma]^{\text{VP}}_i$, otherwise.

Given an IPIOA P and a set of states U, the algorithm constructs an Markov decision process M and a set of states $M(U)$ and performs a reachability analysis on M assuming total information. We prove that the results of the reachability analysis for M overestimate the result for P assuming schedulers in $\text{SDIST}_P([\cdot]^{\text{VP}}, \leqslant)$, that is:

$$\sup_{\eta \in \text{SDIST}_P([\cdot]^{\text{VP}}, \leqslant)} \Pr^{\eta}_P(\text{REACH}(U)) \leqslant \sup_{\eta \in \text{SCHED}_M} \Pr^{\eta}_M(\text{REACH}(M(U))) .$$

Note that, by Theorem 4.3 (case *(I)*), we have

$$\sup_{\eta \in \text{SDIST}_P([\cdot]^{\text{VP}}, \text{RATE})} \Pr^{\eta}(\text{REACH}(U)) \leqslant \sup_{\eta \in \text{SDIST}_P([\cdot]^{\text{VP}}, \leqslant)} \Pr^{\eta}(\text{REACH}(U)) .$$

In addition, by Theorem 2.7 we have

$$\sup_{\eta \in \text{SDIST}_P([\![\cdot]\!], \text{RATE})} \Pr^{\eta}(\text{REACH}(\mathcal{S})) \leqslant \sup_{\eta \in \text{SDIST}_P([\cdot]^{\text{VP}}, \text{RATE})} \Pr^{\eta}(\text{REACH}(\mathcal{S})) ,$$

for all measurable $\mathcal{S}$. Hence, the value calculated by the algorithm is also an overestimation of $\sup_{\eta \in \text{SDIST}_P([\![\cdot]\!], \text{RATE})} \Pr^{\eta}(\text{REACH}(U))$, that is,

$$\sup_{\eta \in \text{SDIST}_P([\![\cdot]\!], \text{RATE})} \Pr^{\eta}(\text{REACH}(U)) \leqslant \sup_{\eta \in \text{SCHED}_M} \Pr^{\eta}_M(\text{REACH}(M(U))) \quad (6.2)$$

We use the following example in order to illustrate how the MDP constructed by our algorithm simulates the original IPIOA P. Suppose that ABC Corp. is planning two meetings. Each member of ABC Corp. must assist to

exactly one of these meetings. ABC Corp. has a business and a technical division. The aim of ABC Corp. is to foster collaboration among people belonging to different divisions. They plan to assign a meeting to each member using a computer program. Such program randomly selects one of the meetings (with probability 1/2 each) without showing the selection. Each member of ABC Corp. is required to use the program during the course of the week, in order to know the meeting which he/she is assigned to. When a member asks the program, the program assigns to this member the meeting previously selected. Then, the program randomly selects another meeting to be assigned to the next member.

We would like the members in each meeting to be well-balanced with respect to the division they belong to. Suppose that a member of ABC Corp. decides to check the probability that all members of the technical division are assigned to the same meeting by analysing all possible orderings for the members of ABC Corp. Then, he discovers that, if the members ask the program in order $m_1, \cdots, m_N$ (where N is the total number of members) then the member m_i is assigned with probability 1/2 to each meeting, for all i. So, the probability that all members of the technical division are assigned to the same meeting is $1/2^{N_T}$ for all cases, where N_T is the number of members of the technical division. Roughly speaking, the order is selected beforehand and the system is verified by assuming that the atoms execute in the selected order. This analysis must be carried out for every possible order.

The scenario is a bit more complex if we allow the members of ABC corp. to ask the computer for a second time in case they do not like the meeting they were assigned to. Suppose that we fix an order as before. Once the first member asks the computer, we must consider the cases in which the member asks for a second time immediately, the case in which it waits until another member has asked, etc. So, once the member asks the computer, we can obtain several new orders by "inserting" the member at different places in the previous order.

In this way, the MDP constructed starts by choosing among all the possible orders on atoms. The selected order is kept as part of the state. Then, each time a transition is executed, the MDP updates the order of all the atoms participating in the transition. Each possible order corresponds to a different nondeterministic option. The next atom to execute a transition is determined by the current order.

In the following, we show how to construct the MDP M from the IPIOA P. The MDP M starts with a nondeterministic choice. This choice selects a total order on the atoms having enabled generative transitions in the their initial state. (In general, for any state s, we call these atoms the *enabled* atoms –denoted by $\text{ENATOMS}(s)$). The interpretation of such order is that the atoms will perform outputs according to it. After this first choice has been performed, M has several available transitions. All of these transitions execute the generative transition corresponding to the atom on the bottom of the order, but, in addition, each transition determines how the atoms are ordered after the transition has been executed. If the order prior to the execution is o_1, then the new order o_2 must comply $A_i <_{o_1} A_j \implies A_i <_{o_2} A_j$ for all A_i, A_j such that the label a output by the generative transition is neither in ACTLAB_i nor in ACTLAB_j. That is, the order of atoms that are not aware of a does not change. By repeating this mechanism, M picks the maximum atom (according to the current order) and reorders the atoms. Note that the

ENATOMS(s)

Algorithm 1 Overestimation of supremum probabilities under total order-based schedulers

1: OVERESTIMATE$(P, U) : \mathbb{R}_{\geqslant 0}$
2: Using P, construct an MDP M according to Def. 6.6
3: Calculate $S \leftarrow \sup_{\eta \in \text{SCHED}_M} \text{PR}^\eta(\text{REACH}(\{(a, s, o) \mid s \in U\}))$
4: `Return` S

restriction on the reordering ensures that, if at some point of the execution it holds that $A_i <_o A_j$, then A_j will not execute after A_i unless A_i or A_j get new information.

In general, the decision of whether A_i executes before A_j is taken right after the last step in which A_i or A_j receives information. So, this choice cannot depend on subsequent probabilistic choices (thus reflecting the motivation we used to define total order-based schedulers).

Let $\mathcal{O}_P$ be the set comprising all total orders on subsets of ATOMS(P). Moreover, given $S \subseteq \text{ATOMS}(P)$, let $\mathcal{O}(S)$ be the set comprising all total orders on S.

DEFINITION 6.5. Given an order o, a *reordering function* for o is a function $r_o : \text{ACTLAB} \times \prod_i S_i \to \mathcal{O}_P$ such that:
(1) $r_o(a, s) \in \mathcal{O}(\text{ENATOMS}(s))$ and
(2) $(A_i <_o A_j \wedge a \notin \text{ACTLAB}_i \wedge a \notin \text{ACTLAB}_j) \implies \forall s : A_i <_{r_o(a,s)} A_j$.
Let $\mathcal{R}_o$ denote all the reordering functions for o.

DEFINITION 6.6. The set of states of M is $S = (\,(\text{ACTLAB} \cup \{a_{\text{INIT}}\}) \times (\prod_i S_i) \times \mathcal{O}_P\,) \cup \{\text{INIT}_M\}$, where a_{INIT} is a fictitious label ($a_{\text{INIT}} \neq \text{ACTLAB}$), and $\text{INIT}_M \notin S_P$ is the initial state of M. The set of actions in INIT_M is defined by:

$$\text{ACTIONS}(\text{INIT}_M) = \{\alpha_{\text{INIT}_M, o} \mid o \in \mathcal{O}(\text{ENATOMS}(\text{INIT}_P))\},$$

with $P(\text{INIT}_M, \alpha_{\text{INIT}_M,o}, (a_{\text{INIT}}, \text{INIT}_P, o)) = 1$. For the remaining states, we have

$$\text{ACTIONS}((a, s, o)) = \{\alpha_{s,o,r_o} \mid r_o \in \mathcal{R}_o\}\,.$$

where α_{s,o,r_o} is defined as follows. Given a state $s = (s_1, \cdots, s_N)$, an enabled atom i and a label a', let $c_{s,i,a'}$ be the compound transition

$$c_{s,i,a'} = (G_i(s_i), a', R_{j_1}(s_j, a'), \cdots, R_{j_m}(s_{j_m}, a'))\,.$$

Then,

$$P((a, s, o), \alpha_{s,o,r_o}, (a', s', o')) = c_{s,\min_i^{<_o} A_i, a'}(s, s')$$

if $r_o(a', s') = o'$. Otherwise, $P((a, s, o), \alpha_{s,o',r}, (a', s', o'')) = 0$. For the states in which $\text{ENATOMS}(s) = \emptyset$, we define $\text{ACTIONS}(s) = \{\alpha_\varsigma\}$ with $P(s, \alpha_\varsigma, s) = 1$.

Our algorithm simply translates the IPIOA P to an MDP M and applies the algorithm in [25] to calculate $\sup_{\eta \in \text{SCHED}_M} \text{PR}^\eta(\text{REACH}(M(U)))$, where

$$M(U) = \{(a, s, o) \mid s \in U\}\,.$$

This is summarized in Algorithm 6.2 (for simplicity, the figure shows only the supremum, but it also applies to the infimum).

In Sec. 8.2 we analyse the model of a protocol using our algorithm, and obtain results more realistic than those obtained by analysing MDP(P) as obtained by Def. 6.2.

The correctness of Algorithm 6.2 is stated in the following theorem.

THEOREM 6.2. *Given a PIOA* P*, let* M *be the MDP constructed as in Def. 6.6. Then,*

$$\sup_{\eta \in \mathrm{SDIST}_P([\cdot]^{\mathrm{VP}}, \leqslant)} \mathrm{PR}_P^{\eta}(\mathrm{REACH}(U)) \leqslant \sup_{\eta \in \mathrm{SCHED}_M} \mathrm{PR}_M^{\eta}(\mathrm{REACH}(M(U)))$$

and

$$\inf_{\eta \in \mathrm{SCHED}_M} \mathrm{PR}_M^{\eta}(\mathrm{REACH}(M(U))) \leqslant \inf_{\eta \in \mathrm{SDIST}_P([\cdot]^{\mathrm{VP}}, \leqslant)} \mathrm{PR}_P^{\eta}(\mathrm{REACH}(U)) \,.$$

The following corollary is deduced from (6.2).

COROLLARY 6.1.

$$\sup_{\eta \in \mathrm{SDIST}_P([\![\cdot]\!], \mathrm{RATE})} \mathrm{PR}_P^{\eta}(\mathrm{REACH}(U)) \leqslant \sup_{\eta \in \mathrm{SCHED}_M} \mathrm{PR}_M^{\eta}(\mathrm{REACH}(M(U)))$$

and

$$\inf_{\eta \in \mathrm{SCHED}_M} \mathrm{PR}_M^{\eta}(\mathrm{REACH}(M(U))) \leqslant \inf_{\eta \in \mathrm{SDIST}_P([\![\cdot]\!], \mathrm{RATE})} \mathrm{PR}_P^{\eta}(\mathrm{REACH}(U)) \,.$$

For the proof of Theorem 6.2, we need an ancillary lemma.

LEMMA 6.1. *Given a PIOA* P*, let* M *be the MDP constructed as in Def. 6.6. For all schedulers* η *in* $\mathrm{SDIST}_P([\cdot]^{\mathrm{VP}}, \leqslant)$*, there exists a scheduler* $\eta' \in \mathrm{SCHED}_M$ *and a bijection* h_η *mapping the finite paths with positive probability in* η *to the finite paths with positive probability in* η' *having length greater than* 1*. Moreover,* h_η *complies with the following properties:*

1. $\mathrm{LEN}(h_\eta(\sigma)) = \mathrm{LEN}(\sigma) + 1$ *for all* σ*,*
2. $\mathrm{PR}^{\eta}((\sigma)^{\uparrow}) = \mathrm{PR}^{\eta'}((h_\eta(\sigma))^{\uparrow})$*,*
3. *Let* $\leqslant^{\eta}$ *be the total order that defines the interleaving scheduler in* η*. Then,* $\mathrm{LAST}(h_\eta(\sigma)) = (a, s, o)$*, where* a *is the last action label in* σ *(or* a_{INIT} *in case* $\sigma = \mathrm{INIT}_P$*),* $s = \mathrm{LAST}(\sigma)$*, and* o *is the order such that*

$$A_i <_o A_j \quad \textit{iff} \quad [\sigma]_i^{\mathrm{VP}} <^{\eta} [\sigma]_j^{\mathrm{VP}} \quad . \tag{6.3}$$

In particular, we have $\mathrm{LAST}(\sigma) \in U$ *iff* $\mathrm{LAST}(h_\eta(\sigma)) \in M(U)$*.*

Proof. Given η, we define η' and h_η by induction on the path length. The properties for h_η are proved inductively along the definition. The base case of the induction concerns all the paths of length 1 (namely INIT_P). First, let o_{INIT} be the order $A_i <_{o_{\mathrm{INIT}}} A_j$ iff $[\mathrm{INIT}_P]_i^{\mathrm{VP}} <^{\eta} [\mathrm{INIT}_P]_j^{\mathrm{VP}}$. Then, we define

$$\eta'(\mathrm{INIT}_M) = \alpha_{\mathrm{INIT}_M, o_{\mathrm{INIT}}}$$
$$h_\eta(\mathrm{INIT}_P) = \mathrm{INIT}_M.\alpha_{\mathrm{INIT}_M, o_{\mathrm{INIT}}}.(a_{\mathrm{INIT}}, \mathrm{INIT}_P, o_{\mathrm{INIT}}) \,.$$

Note that, so far, h_η is mapping all paths of length 1 in P to all paths of length 2 having positive probability in η'. Property *(1)* holds by definition of h_η. With respect to property *(2)*:

$$\mathrm{PR}^{\eta'}((h_\eta(\mathrm{INIT}_P))^{\uparrow}) = \mathrm{PR}^{\eta'}((\mathrm{INIT}_M.\alpha_{\mathrm{INIT}_M, o_{\mathrm{INIT}}}.(a_{\mathrm{INIT}}, \mathrm{INIT}_P, o_{\mathrm{INIT}}))^{\uparrow})$$
$$= 1 = \mathrm{PR}^{\eta}((\mathrm{INIT}_P)^{\uparrow}) \,.$$

Property *(3)* holds by definition of o_{INIT}.

For the inductive case, assume that we have defined η' for all paths of length $N + 1$ (that is, h_η maps all the paths of length N in P). Let $\sigma.c.s$ be a path of length $N + 1$ in P. By inductive hypothesis, we have $h_\eta(\sigma.c.s) =$

$\sigma'.\alpha.(a, s, o)$, for some a, s, and o complying Eqn. (6.3) for $\sigma.c.s$. Recalling the definition of M, we find that the transitions enabled in $h_\eta(\sigma)$ are of the form α_{s,o,r_o}. Let A_{i^*} be the atom such that $\mathcal{I}(\sigma.c.s) = 1{:}A_{i^*}$ and, for all a', let $c^*_{a'}$ be the compound transition $(\Theta_{i^*}(\sigma.c.s), a', \Upsilon_{j_1}(\sigma.c.s, a'), \cdots, \Upsilon_{j_m}(\sigma.c.s, a'))$. Moreover, for all a', s', let

$$\sigma_{a',s'} = \sigma.c.s.c^*_{a'}.s' \ .$$

We define the function r'_o as $r'_o(a', s') = o'$, where $A_i <_{o'} A_j$ iff $[\sigma_{a',s'}]^{VP}_i <^\eta [\sigma_{a',s'}]^{VP}_j$ for all $A_i, A_j \in \text{ENATOMS}(s)$.

We show that r'_o is a reordering function. By definition,

$$r'_o(a', s') \in \mathcal{O}(\text{ENATOMS}(s)) \ .$$

In addition, if $A_i <_o A_j$ and $a' \notin \text{ACTLAB}_i$ and $a' \notin \text{ACTLAB}_j$, then we have $[\sigma.c.s]^{VP}_i <^\eta [\sigma.c.s]^{VP}_j$ (since o complies with Eqn. (6.3)). Moreover, since $a' \notin \text{ACTLAB}_i$ (since $a' \notin \text{ACTLAB}_j$, resp.), we conclude $[\sigma_{a',s'}]^{VP}_i = [\sigma.c.s]^{VP}_i$ for all s' ($[\sigma_{a',s'}]^{VP}_j = [\sigma.c.s]^{VP}_j$ for all s', resp.) and hence

$$[\sigma_{a',s'}]^{VP}_i = [\sigma.c.s]^{VP}_i <^\eta [\sigma.c.s]^{VP}_j = [\sigma_{a',s'}]^{VP}_j \ .$$

Therefore, $A_i <_{o'} A_j$.

Since r'_o is a reordering function, the following definitions are legitimate:

$$\eta'(h_\eta(\sigma.c.s)) = \alpha_{s,o,r'_o}$$
$$h_\eta(\sigma_{a',s'}) = h_\eta(\sigma.c.s).\alpha_{s,o,r'_o}.(a', s', r'_o(a', s'))$$

Property *(1)* is easily verified:

$$\text{LEN}(\sigma_{a',s'}) = 1 + \text{LEN}(\sigma.c.s) = 1 + (1 + \text{LEN}(h_\eta(\sigma.c.s))) =$$
$$1 + \text{LEN}(h_\eta(\sigma.c.s).\alpha_{s,o,r'_o}.(a', s', r'_o(a', s'))) = 1 + h_\eta(\sigma_{a',s'}) \ .$$

Property *(3)* holds by definition of r'_o.

Next, we prove property *(2)*. Since $\leqslant^\eta$ is the total order for interleaving scheduler $\mathcal{I}$ that defines η and Eqn. (6.3) holds for $\sigma.c.s$, we have

$$A_{i^*} = \min^{<_o} A_i \ . \tag{6.4}$$

Then,

$$\begin{aligned}
&\text{PR}^{\eta'}((h_\eta(\sigma_{a',s'}))^\uparrow) \\
=&\{\ \text{Definition of } h_\eta(\sigma_{a',s'})\ \} \\
&\text{PR}^{\eta'}((h_\eta(\sigma.c.s)^\uparrow) \cdot P((a, s, o), \alpha_{s,o,r'_o}, (a', s', r'_o(a', s'))) \\
=&\{\ \text{Inductive hypothesis}\ \} \\
=&\,\text{PR}^{\eta}((\sigma.c.s)^\uparrow) \cdot P((a, s, o), \alpha_{s,o,r'_o}, (a', s', r'_o(a', s'))) \\
=&\{\ \text{Definition of } P(\cdot, \alpha_{s,o,r_o}, \cdot)\ \} \\
&\text{PR}^{\eta}((\sigma.c.s)^\uparrow) \cdot c_{s,\min^{<_o} A_i, a'}(s, s') \\
=&\{\ \text{Equation (6.4), definition of } c^*_{a'}\ \} \\
&\text{PR}^{\eta}((\sigma.c.s)^\uparrow) \cdot c^*_{a'}(s, s') \\
=&\{\ \mathcal{I}(\sigma.c.s) = A_{i^*}\ \} \\
=&\,\text{PR}^{\eta}((\sigma_{a',s'})^\uparrow)
\end{aligned}$$

□

Proof of Theorem 6.2. First, we consider the case of the supremum. Let $\eta \in \textsc{sdist}_P([\cdot]^{VP}, \leqslant)$ and $\epsilon > 0$. We show that there exists $\eta' \in \textsc{sched}_M$ such that $\Pr^{\eta'}(\textsc{reach}(M(U))) > \Pr^{\eta}(\textsc{reach}(U)) - \epsilon$. This η' is precisely the scheduler whose existence is ensured by Lemma 6.1. Let

$$\mathcal{R} = \{\sigma \mid \textsc{last}(\sigma) \in U \wedge \forall k < \textsc{len}(\sigma) : \sigma(k) \notin U\}$$

and $\mathcal{R}_k = \{\sigma \in \mathcal{R} \mid \textsc{len}(\sigma) = k\}$. Since $\textsc{reach}(U) = \biguplus_{\sigma_m \in \mathcal{R}} (\sigma_m)^{\uparrow}$, we have

$$\Pr^{\eta}(\textsc{reach}(U)) = \sum_{\sigma_m \in \mathcal{R}} \Pr^{\eta}((\sigma_m)^{\uparrow}) = \lim_{N \to \infty} \sum_{k=1}^{N} \sum_{\sigma \in \mathcal{R}_k} \Pr^{\eta}((\sigma)^{\uparrow}) .$$

Hence, there exists N_ϵ such that

$$\Pr^{\eta}(\textsc{reach}(U)) - \sum_{k=1}^{N_\epsilon} \sum_{\sigma \in \mathcal{R}_k} \Pr^{\eta}((\sigma)^{\uparrow}) < \epsilon .$$

To prove the result, we show that

$$\Pr^{\eta'}(\textsc{reach}(U)) \geqslant \sum_{k=1}^{N_\epsilon} \sum_{\sigma \in \mathcal{R}_k} \Pr^{\eta}((\sigma)^{\uparrow}) .$$

By Lemma 6.1 (property *(2)*), we have

$$\sum_{k=1}^{N_\epsilon} \sum_{\sigma \in \mathcal{R}_k} \Pr^{\eta}(\sigma)^{\uparrow} = \sum_{k=1}^{N_\epsilon} \sum_{\sigma \in \mathcal{R}_k} \Pr^{\eta'}(h_\eta(\sigma)^{\uparrow})$$

From property *(3)* and the fact that h_η is a bijection, we conclude that $(h_\eta(\sigma))^{\uparrow} \cap (h_\eta(\sigma'))^{\uparrow} = \emptyset$ whenever $\sigma, \sigma' \in \mathcal{R}$ and $\sigma \neq \sigma'$: otherwise, it should be $h_\eta(\sigma) \sqsubset h_\eta(\sigma')$, implying $h_\eta(\sigma')(\textsc{len}(\sigma)) = \textsc{last}(h_\eta(\sigma)) \in M(U)$ and in turn $\sigma'(\textsc{len}(\sigma)) \in U$, which is clearly false since $\textsc{len}(\sigma) < \textsc{len}(\sigma')$ and $\sigma \in \mathcal{R}$ (recall the paths σ in $\mathcal{R}$ are required to *not* have states in U, except for $\textsc{last}(\sigma)$).

Then,

$$\sum_{k=1}^{N_\epsilon} \sum_{\sigma \in \mathcal{R}_k} \Pr^{\eta'}(h_\eta(\sigma)^{\uparrow}) = \Pr^{\eta'}(\biguplus_{k=1}^{N_\epsilon} \biguplus_{\sigma \in \mathcal{R}_k} (h_\eta(\sigma))^{\uparrow}) .$$

Property *(3)* in Lemma 6.1 implies

$$\biguplus_{k=1}^{N_\epsilon} \biguplus_{\sigma \in \mathcal{R}_k} (h_\eta(\sigma))^{\uparrow} \subseteq M(U)$$

and so

$$\Pr^{\eta'}(\textsc{reach}(M(U))) \geqslant \Pr^{\eta'}(\biguplus_{k=1}^{N_\epsilon} \biguplus_{\sigma \in \mathcal{R}_k} (h_\eta(\sigma))^{\uparrow}) = \sum_{k=1}^{N_\epsilon} \Pr^{\eta}(\textsc{reach}(\mathcal{R}_k))$$

as desired.

With respect to the infimum, we again consider $\eta \in \textsc{sdist}_P([\cdot]^{VP}, \leqslant)$, and the corresponding scheduler η' obtained in the previous lemma. Now, we show that $\Pr^{\eta'}(\textsc{reach}(U)) \leqslant \Pr^{\eta}(\textsc{reach}(U))$. Let

$$\mathcal{R} = \{\sigma \mid \textsc{last}(\sigma) \in \ \wedge \forall k < \textsc{len}(\sigma) : \sigma(k) \notin\}$$

and $\mathcal{R}_k = \{\sigma \in \mathcal{R} \mid \textsc{Len}(\sigma) = k\}$. Next we show that, for all N,

$$\sum_{k=1}^{N} \sum_{\sigma \in \mathcal{R}_k} \Pr^{\eta'}((\sigma)^{\uparrow}) \leqslant \Pr^{\eta}(\textsc{Reach}(U)) \,,$$

and hence the result is implied by the equality

$$\Pr^{\eta'}(\textsc{Reach}(U)) = \lim_{N \to \infty} \sum_{k=1}^{N} \sum_{\sigma \in \mathcal{R}_k} \Pr^{\eta'}((\sigma)^{\uparrow}) \,.$$

The justifications of the following calculation are the same as for the supremum.

$$\begin{aligned}
&\sum_{k=1}^{N} \sum_{\sigma \in \mathcal{R}_k} \Pr^{\eta'}((\sigma)^{\uparrow}) \\
=&\sum_{k=1}^{N} \sum_{\sigma \in \mathcal{R}_k} \Pr^{\eta}((h_{\eta}^{-1}(\sigma))^{\uparrow}) \\
=&\Pr^{\eta}(\biguplus_{k=1}^{N} \biguplus_{\sigma \in \mathcal{R}_k} (h_{\eta}^{-1}(\sigma))^{\uparrow}) \\
\leqslant&\Pr^{\eta}(\textsc{Reach}(U))
\end{aligned}$$

□

6.3 UNDERESTIMATION OF SUPREMUM REACHABILITY PROBABILITIES UNDER DISTRIBUTED SCHEDULERS

Section 6.2 presents an algorithm that is useful to prove that a given model is *correct* in the sense that: if the supremum value V computed by the algorithm is less than or equal to the greatest admissible value S, then the model can be deemed as correct. However, in case $V > S$ we cannot ensure that the model is incorrect, since the computed values are overestimations of the supremum quantifying over $\textsc{sdist}_P([\![\cdot]\!], \leqslant)$ (Sec. 6.2), and so it might be the case that all the schedulers yielding probabilities greater than V are not in $\textsc{sdist}_P([\![\cdot]\!], \leqslant)$.

In this section, we devise an algorithm that can be used to ensure that a model is *incorrect* with respect to a reachability property under distributed schedulers. Specifically, given the greatest admissible value S, the algorithm can be used to show that $\sup_{\eta \in \textsc{dist}_P} \Pr^{\eta}(\textsc{Reach}(U)) > S$. To this aim, the algorithm successively calculates the probability of reaching U for each scheduler in a representative subset of Markovian schedulers. If some Markovian scheduler yielding probability greater than S is found, then the system is incorrect. On the contrary, if no scheduler is found, it might be the case that there exists a non-Markovian scheduler (or a randomized Markovian scheduler) yielding probability greater than S, and so we are not assured that the system is correct.

In order to ease explanations, we start by presenting a basic algorithm to calculate the value for all Markovian schedulers. Then, we present an improvement that allows to safely discard some sets of schedulers, and so not all values need to be calculated. This improvement uses the branch-and-bound technique.

For the basic algorithm, we consider a transformation $P[C \leftarrow O]$ on IPIOA. This transformation has two parameters (aside from the IPIOA P on which the transformation is carried out). The first parameter is a nondeterministic choice C. Formally, C is a state s^* such that $\left|G_i(s^*)\right| > 0$ (a pair (s^*, a^*) such that $\left|R_i(s^*, a^*)\right| > 0$, resp.) The second parameter is one of the available options O for such a choice. Formally, O is a generative transition $g^* \in G_i(s^*)$ (a reactive transition $r^* \in R_i(s^*, a^*)$, resp.). The transformation fixes the transition O to be always chosen in C. Formally, $P[C \leftarrow O]$ is an IPIOA that coincides with P except for the generative structure $G_i^{P[C\leftarrow O]}$ for the atom A_i (the reactive structure $R_i^{P[C\leftarrow O]}$, resp.). This generative structure is defined as $G_i^{P[C\leftarrow O]}(s) = G(s)$ for all $s \neq s^*$ and $G_i^{P[C\leftarrow O]}(s^*) = \{g^*\}$ ($R_i^{P[C\leftarrow O]}(s, a) = R(s, a)$ for all $(s, a) \neq (s^*, a^*)$ and $R_i^{P[C\leftarrow O]}(s^*, a^*) = \{r^*\}, resp.$). Note that, if P' is obtained by resolving all I/O nondeterministic choices in P, the only nondeterminism remaining in P' is due to interleaving. Since we aim to check the model under distributed schedulers (in which the interleaving scheduler has total information), we can calculate the supremum for P' assuming total information. We assume that a procedure SUPTOTALINFO exists to calculate $\sup_{\eta \in \text{SCHED}_P} \text{PR}^\eta(\text{REACH}(U))$, by considering the translation to MDPs in Sec. 6.1. This procedure can be found in e. g. [25]. Moreover, from [25], we know that the scheduler η^* yielding the supremum probability for P' is *globally* Markovian and non-randomized, and so η^* corresponds to a *locally* Markovian non-randomized scheduler η_P for P: the interleaving [†] scheduler $\mathcal{I}_P$ resolves the remaining nondeterminism just like η^* does, and the output and input schedulers choose the only transition that remains enabled in P'.

Algorithm 2 Explore all non-randomized distributed Markovian schedulers

```
1:  Global MAXSUP ← −1
2:  SUPMARKOVIAN(P) : {INCORRECT, UNDETERMINED}
3:    If P has an I/O nondeterministic choice C Then /*Expandable node*/
4:      For All options O for C Do /*Backtrack*/
5:        P' ← P[C ← O]
6:        If SUPMARKOVIAN(P') = INCORRECT Then
7:          Return INCORRECT
8:        End If
9:      End For
10:   Else /*P is I/O deterministic*/
11:     s_P ← SUPTOTALINFO(P)
12:     MAXSUP ← max(s_P, MAXSUP)
13:     If MAXSUP > S Then
14:       Return INCORRECT
15:     End If
16:   End If
17:   Return UNDETERMINED
```

The backtracking algorithm SUPMARKOVIAN performs a depth-first search on the space of all distributed Markovian non-randomized schedulers (for an introduction to backtracking algorithms, see [27]). If the algorithm finds

[†] Note that, with respect to the interleaving scheduler, the restrictions for both locally and globally Markovian schedulers coincide

a distributed Markovian scheduler yielding probability greater than S, then it returns INCORRECT. Otherwise, it returns UNDETERMINED. In the search, each node expansion corresponds to a nondeterministic choice (line 3), and each branch corresponds to an option in this choice. If the algorithm finds a value greater than S, then it returns INCORRECT (line 7). Otherwise, the algorithm backtracks by trying another option for the choice (line 4). If none of the choices can find a value greater than S, then UNDETERMINED is returned (line 17). In case P is I/O deterministic, the algorithm calculates the supremum value quantifying over all possible interleavings (line 11) and this value is stored in s_P. The greatest value obtained so far is stored in the variable MAXSUP, and so this variable is updated in case $s_P >$ MAXSUP (line 12). In addition, if MAXSUP $>$ S, then the algorithm returns INCORRECT (line 14).

In order to avoid considering all the possible schedulers, we improve this algorithm by pruning some branches using the *branch-and-bound* technique. In this technique, the algorithm calculates an upper bound for the values on the branch to be explored. If this bound is less than or equal to the maximum value achieved so far, the algorithm does not explore branch. In our particular case, each branch corresponds to an option in a nondeterministic choice. The bound obtained in our algorithm corresponds to the supremum probability quantifying over all schedulers. We justify the bound using the following lemma. In the following, let MARKOVIAN(P) denote the set of all distributed non-randomized locally Markovian schedulers for P.

MARKOVIAN(P)

LEMMA 6.2.

$$\sup_{\eta \in \text{MARKOVIAN}(P')} \Pr^{\eta}(\text{REACH}(U)) \leqslant \sup_{\eta \in \text{SCHED}_{P'}} \Pr^{\eta}(\text{REACH}(U)) \ .$$

Moreover, if P' *is obtained from* P *by successive applications of* $P[\cdot \leftarrow \cdot]$, *we have*

$$\sup_{\eta \in \text{SCHED}_{P'}} \Pr^{\eta}(\text{REACH}(U)) \leqslant \sup_{\eta \in \text{SCHED}_{P}} \Pr^{\eta}(\text{REACH}(U)) \ .$$

Proof. The first inequation holds by the set inclusion

$$\text{MARKOVIAN}(P') \subseteq \text{SCHED}_{P'} \ .$$

For the second inequation, we note that each scheduler η' for P' is also a scheduler for P. Then,

$$\text{SCHED}_{P} \subseteq \text{SCHED}_{P'}$$

and the lemma follows. □

Algorithm 3 uses the procedure PRUNE in order to decide whether a given branch is pruned. This procedure returns True iff the supremum quantifying over all schedulers for P is less than or equal to MAXSUP. By Lemma 6.2, for every P' obtained by applying $P[\cdot \leftarrow \cdot]$, we have

$$\sup_{\eta \in \text{MARKOVIAN}(P')} \Pr^{\eta}(\text{REACH}(U)) \leqslant \sup_{\eta \in \text{SCHED}_{P}} \Pr^{\eta}(\text{REACH}(U)) \ .$$

So, in case PRUNE returns True we have

$$\sup_{\eta \in \text{MARKOVIAN}(P')} \Pr^{\eta}(\text{REACH}(U)) \leqslant \text{MAXSUP} \ . \tag{6.5}$$

Algorithm 3 Discard some fruitless sets of schedulers

```
1:  Global MAXSUP ← −1
2:  PRUNE(P) : Boolean
3:    t_P ← SUPTOTALINFO(P)
4:    If t_P ⩽ MAXSUP Then
5:      Return True
6:    Else
7:      Return False
8:    End If
9:  SUPMARKOVIAN(P) : {INCORRECT, UNDETERMINED}
10:   If P has an I/O nondeterministic choice C Then /*Expandable node*/
11:     For All options O for C Do /*Backtrack*/
12:       P′ ← P[C ← O]
13:       If PRUNE(P′) Then
14:         Try next O
15:       Else If SUPMARKOVIAN(P′) = INCORRECT Then
16:         Return INCORRECT
17:       End If
18:     End For
19:   Else /*P is I/O deterministic*/
20:     s_P ← SUPTOTALINFO(P)
21:     MAXSUP ← max(s_P, MAXSUP)
22:     If MAXSUP > S Then
23:       Return INCORRECT
24:     End If
25:   End If
26:   Return UNDETERMINED
```

Note that MAXSUP is either -1 or it is the value yielded by a scheduler in MARKOVIAN(P). In the former case, we have

$$\sup_{\eta \in \text{MARKOVIAN}(P)} \Pr^{\eta}(\text{REACH}(U)) > \text{MAXSUP}\,,$$

and so there is no pruning and this algorithm behaves as the previous one. For the latter case, we note that MAXSUP is updated only in case a scheduler yields a value greater than the current MAXSUP (line 21). Hence, Eqn. (6.5) ensures that the values for the systems P' will not alter the value of MAXSUP. Therefore, the return value of SUPMARKOVIAN is not affected by the values arising from such P'. The modified procedure SUPMARKOVIAN profits from this fact and, as soon as it creates a new system P', it checks whether PRUNE(P') returns True. In this case, this system P' is disregarded and the algorithm tries another nondeterministic option (line 14).

6.4 FURTHER WORK

One of the drawbacks of the algorithm in Sec. 6.2 is that the results obtained are still quite pessimistic. In fact, we have proven that the results obtained are safe bounds with respect to the projection $[\cdot]^{\text{VP}}$. Intuitively, the algorithm assumes that, once an atom executes, it is able to see the full history of the system. We would like to generalize this algorithm to consider projections

other than $[\cdot]^{\text{VP}}$, in such a way that, given a projection $[\cdot]$, the results obtained are safe estimations for $[\cdot]$ (but not necessarily for other projections $[\cdot]'$ such that $[\cdot] \sqsubseteq [\cdot]'$).

The generalization to other projections is also a pending task for the algorithm in Sec. 6.3. Contrarily to the algorithm in Sec. 6.2 (for which a practical application is shown in Chapter 8) we have not used Algorithm 3 in practice. Some questions will remain open until we have an implementation of the algorithm. For instance, the real impact of the branch-and-bound optimization cannot be measured until we analyse a significant set of existing models using this technique.

7

PARTIAL ORDER REDUCTION

Partial order reduction (POR) [83, 56] is a well-known technique to cope with the state explosion problem. Given a system and a property, the technique of partial order reduction yields another system with less transitions. This decrement in the amount of transitions results in faster analyses, since the amount of reachable states is thus also decreased.

The reduction exploits the structure of the product model naturally introduced when interleaving the components. The idea is to eliminate redundant states and transitions but keeping some *representative* ones. Such states and transitions are representative in the sense that, if a given *path* of the original system is relevant to the property being checked, then a corresponding *path* should be found in the reduced system. To ensure that representative states and transitions remain in the reduced model, the reduced model must comply with certain conditions [56].

POR for probabilistic model checking of $\text{LTL}_{\setminus\{\text{NEXT}\}}$ properties was simultaneously introduced in [12] and [57]. In the probabilistic case, states and transitions in the reduced system must be representative in the sense that, given a *scheduler* in the original system, a corresponding *scheduler* in the reduced system must exist. The probabilities of the paths relevant to the property being checked must coincide for both schedulers. The works [12, 57] show that, in the setting of total information schedulers, the reduced model should meet one extra condition apart from those of the non-probabilistic case. This condition (call it **A5**) is quite technical and non-intuitive, and it has been introduced precisely to *not* eliminate the behaviour introduced by non-distributed schedulers.

We take advantage of the fact that, under distributedness assumptions, not all schedulers need to be preserved in the original system. As a consequence, condition **A5** can be relaxed for distributed schedulers and eliminated for strongly distributed schedulers.

7.1 PARTIAL ORDER REDUCTION AND RESTRICTED SCHEDULERS

The reduced system is constructed by traversing the state space. When expanding a given state, not all the transitions enabled are considered. A set of transitions, called the *ample set*, is calculated for each state s, and only transitions in the ample set are considered during the traversal. 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.

We focus on the case of LTL properties not containing the next operator. Given a set AP of atomic propositions and a labelling function $\mathsf{L} : S \to \mathcal{P}(\mathsf{AP})$, the set of $\text{LTL}_{\setminus\{\text{NEXT}\}}$ 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 $l \in \mathsf{AP}$. For all ϕ and for all infinite path ρ, we define $\rho \models \phi$ (read ρ satisfies ϕ) The definition proceeds by induction on ϕ.

- $\rho \models \text{TRUE}$
- $\rho \models l \iff l \in L(\rho(1))$
- $\rho \models \neg\phi \iff \rho \not\models \phi$
- $\rho \models \phi_1 \wedge \phi_2 \iff \rho \models \phi_1 \wedge \rho \models \phi_2$
- $\rho \models \phi_1 U \phi_2 \iff \exists M : \rho\uparrow^m \models \phi_2 \wedge \forall m < M : \rho(m) \models \phi_1$

Intuitively, an infinite path ρ satisfies $\phi_1 U \phi_2$ (denoted by $\rho \models \phi_1 U \phi_2$) iff there is a 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.

Note that the verity of $\rho \models \phi$ depends only on the *trace* of ρ, that is, the sequence $\text{TRACE}(\rho) = L(\rho(1))\,L(\rho(2))\cdots$. TRACE

The following abbreviations are usual: $F\phi \equiv \text{TRUE}U\phi$, $G\phi \equiv \neg F\neg\phi$. The intuitive meaning of F is that there exists a state in ρ such that ϕ holds (Finally, ϕ holds), while the intuitive meaning of G is that ϕ holds for all states in ρ (ϕ holds Globally). We write $\text{PR}^\eta(\phi)$ for $\text{PR}^\eta(\{\rho \mid \rho \models \phi\})$.

Since existing model checkers use MDPs, the partial order reduction may be applied on the MDP underlying a given IPIOA (recall Sec. 6.1). So, we explain the technique from the point of view of MDPs. Hence, it is convenient to consider the schedulers for P as if they were schedulers for $\text{MDP}(P)$. To this end, we first explain how to transform a path σ in $\text{MDP}(P)$ into a path $\text{IPIOA}(\sigma)$ in P in the obvious way:

- $\text{IPIOA}((a_{\text{INIT}}, \text{INIT})) = \text{INIT}$
- $\text{IPIOA}(\sigma.(g_i, f_1, \cdots, f_N).a.s) = \text{IPIOA}(\sigma).(g_i, a, f_{j_1}(a), \cdots, f_{j_m}(a)).s$

Because of Theorem 4.1, in the remaining of this chapter we deal only with I/O non-randomized schedulers, and so we define $\eta_{\text{MDP}(P)}$ only for I/O non-randomized η (that is, neither Θ nor Υ have randomized resolutions).

DEFINITION 7.1. Given an I/O non-randomized scheduler η for P, we define the corresponding scheduler $\eta_{\text{MDP}(P)}(\sigma)$ for $\text{MDP}(P)$ as follows:

$$\eta_{\text{MDP}(P)}(\sigma)(\overbrace{(g_i, f_1, \cdots, f_N)}^{\alpha}) = \mathcal{I}(\text{IPIOA}(\sigma))(A_i)$$

if $\Theta_i(\text{IPIOA}(\sigma)) = g_i$ and $\Upsilon_j(\text{IPIOA}(\sigma), a) = f_j(a)$ for all j. Otherwise,

$$\eta_{\text{MDP}(P)}(\sigma)(\alpha) = 0\,.$$

NOTATION 7.1. Given $\eta \in \text{SCHED}_P$ and $\alpha \in \text{ACTIONS}(\text{MDP}(P))$, we write $\eta(\sigma)(\alpha)$ for $\eta_{\text{MDP}(P)}(\text{IPIOA}(\sigma))(\alpha)$.

Restrictions on the ample set are based on the notion of *independence*. Given an action $\alpha = (g_i, \cdots)$, let $\text{INV}(\alpha)$ be the set of all atoms involved in the execution of α. Formally,

$$\text{INV}(\alpha) = \{A_j \mid \exists a \in \text{ACTLAB}_j : \exists s, s'_i : g_i(s, a, s') > 0\}\,.$$

DEFINITION 7.2. We say that two actions α, β are *independent* iff $(\exists s : \{\alpha, \beta\} \in \text{ACTIONS}(s)) \implies \text{INV}(\alpha) \cap \text{INV}(\beta) = \emptyset$.

So, two actions are independent only if the execution of one of them does not interfere with the execution of the other one. Note that the order of execution is irrelevant and that neither of them can disable the other. Notice also that this definition is of a more structural nature than the one in [12]. For variable-based IPIOA, we can use a more permissive notion of independence that allows better reductions, see Sec. 9.1.

We need some additional definitions before presenting the restrictions for POR. A compound transition α is *stutter* iff $\alpha(s, a, s') = 0$ for all s such that $\alpha \in \text{ACTIONS}(s)$ and s' such that $L(s) \neq L(s')$, for all a. An *end component* is a pair (T, A) where $A : T \to \mathcal{P}(\text{ACTIONS})$ and T is a set of states such that:

1. $\emptyset \neq A(s) \subseteq \text{ACTIONS}(s)$ for all $s \in T$,
2. $\alpha(s, a, t) > 0$ implies $t \in T$, for all $s \in T$, $\alpha \in A(s)$, $a \in \text{ACTLAB}$
3. for every $s, t \in T$ there exists a path from s to t.

The restrictions for the ample sets of [12] to preserve $\text{LTL}_{\setminus\{\text{NEXT}\}}$ properties under unrestricted full-history dependent schedulers are listed below. $\widehat{S}$ denotes the set of reachable states in the reduced system $\widehat{P}$ [†], which is constructed by taking $\text{AMPLE}(s)$ to be the set of enabled actions in $s \in \widehat{S}$.

(**A1**) *For all states* $s \in S$, $\emptyset \neq \text{AMPLE}(s) \subseteq \text{ACTIONS}(s)$,

(**A2**) *If* $s \in \widehat{S}$ *and* $\text{AMPLE}(s) \neq \text{ACTIONS}(s)$ *then each action* $\alpha \in \text{AMPLE}(s)$ *is stutter,*

(**A3**) *For each path* $\sigma = s.\alpha_1.a_1.s_1.\alpha_2.\cdots.\alpha_n.a_n.s_n.\gamma\cdots$ *in* $\text{MDP}(P)$ *where* $s \in \widehat{S}$ *and* γ *is dependent on* $\text{AMPLE}(s)$ *there exists an index* $1 \leqslant i \leqslant n$ *such that* $\alpha_i \in \text{AMPLE}(s)$,

(**A4**) *If* (T, A) *is an end component in* $\widehat{P}$*, then*

$$\alpha \in \bigcap_{t \in T} A(t) \implies \alpha \in \bigcup_{t \in T} \text{AMPLE}(t) .$$

(**A5**) *If* $s.\alpha_1.a_1.s_1.\alpha_2.a_2.s_2.\cdots.\alpha_n.a_n.s_n.\gamma.a_{n+1}.s_{n+1}$ *is a path in* $\text{MDP}(P)$ *where* $s \in \widehat{S}$, $\alpha_1, \cdots, \alpha_n, \gamma \notin \text{AMPLE}(s)$ *and* γ *is probabilistic (that is, there exist* s', a', t' *such that* $0 < \gamma(s', a', t') < 1$*) then* $\left|\, \text{AMPLE}(s) \,\right| = 1$.

Recall Example 1.4. In order to ease reading, we repeat some of the graphics here. Let's see to which extent we can reduce the system in Fig. 7.3. For the sake of simplicity, we identify an action with its generative transition in case no ambiguity arises. By condition **A1**, $\text{AMPLE}(\text{INIT}_{||})$ cannot be empty. Condition **A3** prevents the set comprising only the action $\alpha_T = 1/2h! + 1/2t!$, since this action is dependent of $g_h!$ and $g_t!$, and these actions cannot be executed without executing α_T. Condition **A3** also requires $c_h!$ to be in the ample whenever $c_t!$ is (and vice versa). According to the restrictions taken into account so far, we have only two candidates: $\text{ACTIONS}(\text{INIT}_{||})$ (i. e. no reduction) and $A_c = \{c_h!, c_t!\}$. Now, we note that A_c is prevented by condition **A5**, since in the initial state we can execute the probabilistic action $1/2h! + 1/2t!$ and A_c has two actions. So, according the restrictions in [12, 57], it must be $\text{AMPLE}(s) = \text{ACTIONS}(s)$.

If we reduce the system using $\{c_h!, c_t!\}$, we obtain the system in Fig. 7.4. Note that the total information scheduler in Fig. 7.2 cannot be simulated in

[†]Since we reduce $\text{MDP}(P)$, it would be correct to write $\widehat{\text{MDP}(P)}$, but is looks quite ugly.

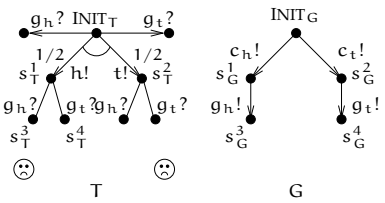


Figure 7.1: T tosses a coin, G guesses heads or tails

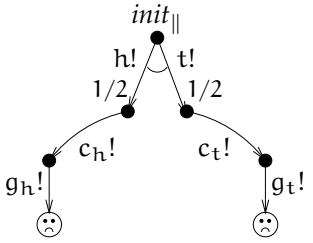


Figure 7.2: A total information scheduler

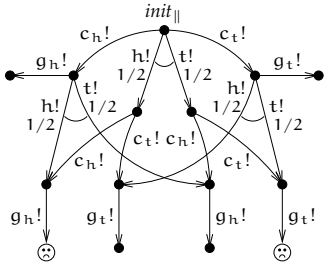


Figure 7.3: System $P = T \parallel G$

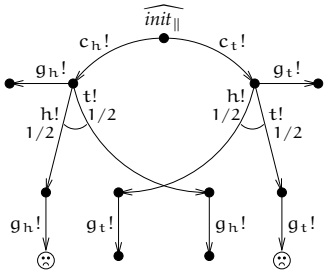


Figure 7.4: A POR based reduction

Figure 7.5: POR and distributed schedulers

the reduced system. In fact, there is no scheduler leading to ☹ in the reduced system. However, the supremum probability considering distributed schedulers is the same in both systems.

Hence, one may wonder to which extent **A5** is needed in case the schedulers are assumed to be distributed, and/or strongly distributed. We show that **A5** can be weakened in case distributed schedulers are assumed, and it can be disregarded under strongly distributed schedulers. In particular, the ample $\text{AMPLE}(\text{INIT}_{\|}) = A_c$ is valid under both settings. In some examples, this weakening results in better reductions, as shown in Chapter 9.

7.2 AN IMPROVEMENT FOR RESTRICTED SCHEDULERS

In this section, we present two theorems (one for distributed schedulers, and the other for strongly distributed ones) to ensure the correctness of our improved partial order reduction. We also show how these theorems for distributed schedulers articulate with existing model checking approaches based on total information. The theorems are stated for simple IPIOA (as defined in Sec. 1.1). The theorem for strongly distributed schedulers admits a more complicated formulation that allows us to deal with generalized IPIOA (as in Sec. 1.2). In Sec. 9.1, we show how to use such general formulation in order to cope with (generalized) IPIOA arising from PRISM models.

Our first theorem concerns distributed schedulers. In this setting of these schedulers, we can replace **A5** by

(**A5**′) *If* $s.\alpha_1.a_1.s_1.\alpha_2.a_2.s_2 \cdots \alpha_n.a_n.s_n.\gamma.a_{n+1}.s_{n+1}$ *is a path in* $\text{MDP}(P)$ *such that* $s \in \widehat{S}$, $\alpha_1,\cdots,\alpha_n,\gamma \notin \text{AMPLE}(s)$ *and* γ *is probabilistic then* $\text{ACTIVE}(\beta) = \text{ACTIVE}(\beta')$, *for all* $\beta, \beta' \in \text{AMPLE}(s)$

as formalized in the following theorem.

THEOREM 7.1. *Let* ϕ *be an* $\mathrm{LTL}_{\setminus\{\mathrm{NEXT}\}}$ *formula and* P *be a simple IPIOA. Let* $\widehat{P}$ *be a reduction of* $\mathrm{MDP}(P)$ *complying with conditions* **A1**–**A4**, **A5**′. *Then,*

$$\sup_{\eta \in \mathrm{DIST}_P} \Pr^{\eta}(\phi) \leqslant \sup_{\eta \in \mathrm{SCHED}_{\widehat{P}}} \Pr^{\eta}(\phi) .$$

In case we assume *strongly distributed schedulers*, **A5** can be disregarded.

THEOREM 7.2. *Let* ϕ, P *be as in Theorem 7.1. Let* $\widehat{P}$ *be a reduction of* $\mathrm{MDP}(P)$ *complying with conditions* **A1**–**A4**. *Then,*

$$\sup_{\eta \in \mathrm{SDIST}_P} \Pr^{\eta}(\phi) \leqslant \sup_{\eta \in \mathrm{SCHED}_{\widehat{P}}} \Pr^{\eta}(\phi) .$$

Let's examine the example in Fig. 7.5 in the light of Theorems 7.1 and 7.2. Let η^w be the scheduler in Fig. 7.2. According to Theorem 7.1, the reduction in Fig. 7.2 is correct in case distributed schedulers are assumed. However, in the original system P we have $\Pr^{\eta^w}(F\,☹) = 1$, while in $\widehat{P}$ we have $\Pr^{\eta}(F\,☹) \leqslant \frac{1}{2}$ for all η. This is due to the fact η^w is not distributed. In fact, the supremum over all distributed schedulers in P is $\frac{1}{2}$, which coincides with $\sup_{\eta \in \mathrm{SCHED}_{\widehat{P}}} \Pr^{\eta}(F\,☹)$. Recall now the example in Fig. 2.1 with atoms T, A, B and Z. We mentioned that the scheduler of Fig. 7.2 *is* distributed in this setting. Call this scheduler η^d. If we assume strongly distributed schedulers, the reduction in Fig. 7.4 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. 7.4, since $c_h!$ and $c_t!$ are generated by atoms A and B, resp. This is exactly what we want, since the scheduler η^d is a valid distributed scheduler for $T \parallel A \parallel B \parallel Z$, and so a corresponding scheduler yielding probability 1 must exist in the reduced system.

7.3 CORRECTNESS OF OUR TECHNIQUES

The proofs of 7.1 and 7.2 are quite technical and several details are involved. However, these proofs rely on the same principle as in the non-probabilistic case. We profit from this similarity to give the reader an overall description of the proof strategy, so that the reader has a previous knowledge of the general picture before facing the technical details. In this description, we assume that the schedulers are non-randomized. In the detailed proof, we show how to overcome the issues introduced by randomized schedulers.

7.3.1 *Overview of the proof*

In the non-probabilistic case, the standard argument is as follows. For every property ϕ, we need to prove that ϕ is satisfied in all paths in P if and only if ϕ is satisfied in all paths in $\widehat{P}$. Since $\widehat{P}$ is a subgraph of P, one implication is trivial. For the other implication, the conditions on the reduction are used to prove that, if some path ρ in P does not satisfy ϕ, then $\widehat{\rho} \not\models_{\widehat{P}} \phi$ for some $\widehat{\rho}$.

Similarly, in our case it is sufficient to prove that, for each scheduler η in the original system, there exists a corresponding $\widehat{\eta}$ in the reduced system. The probability values for η and $\widehat{\eta}$ must coincide for all paths relevant to ϕ. We prove that, for each *distributed* (*strongly distributed*, resp.) scheduler, there

is a corresponding scheduler in the reduced system that yields the same probability value. As a consequence, it may be the case that, for some non-distributed schedulers, there are no corresponding schedulers in the reduced system. However, this causes no harm since schedulers are assumed to be distributed.

Given a non-probabilistic system P, let $\rho = s_1.\alpha_1.a_1.s_2.\alpha_2.a_2.\cdots$ and ϕ such that $\rho \not\models_{\widehat{P}} \phi$. We sketch how the corresponding path $\widehat{\rho}$ is constructed in the standard approach. If $\alpha_1 \in \text{AMPLE}(s_1)$, then $\widehat{\rho}$ starts with $s_1.\alpha_1.a_1$ and the construction continues from $s_2.\alpha_2.a_2.\cdots$. On the contrary side, if $\alpha_1 \notin \text{AMPLE}(s_1)$, then $\widehat{\rho}$ cannot start with $s_1.\alpha_1.a_1$, since α_1 is not enabled for s_1 in $\widehat{P}$. However, condition **A1** ensures that $\text{AMPLE}(s) \neq \emptyset$. For simplicity, let's consider the case in which some $\beta \in \text{AMPLE}(s_1)$ is eventually executed in ρ. W.l.o.g., we can take such a β to be the first transition α_n in ρ such that $\alpha_n \in \text{AMPLE}(s_1)$. Then, by condition **A3** and definition of independence, we have that $\rho' = s_1.\alpha_n.a_n.s'_2.\alpha_1.a_1.\cdots.s'_{n-1}.\alpha_{n-1}.a_{n-1}.s_n.\alpha_{n+1}.\cdots$ is a path in P. (Here, s'_i denotes the state such that $\alpha_{i-1}(s'_i, a_i, s'_{i+1}) = 1$, since the system is non-probabilistic.) Let $\ell_i = L(s_i)$ for all i. Then, since **A2** requires the transitions in $\text{AMPLE}(s)$ to be stuttering, the sequence $L(\rho)$ of labels in ρ has the form $\ell_1 \cdots \ell_n\ell_n\ell_{n+2}\cdots$. Condition **A2** can be used to prove that $L(\rho') = \ell_1\ell_1\ell_2\cdots$. So, since $L(\rho)$ and $L(\rho')$ differ only in the amount of times that each ℓ_i appears, and $\text{LTL}_{\setminus\{\text{NEXT}\}}$ formulae are stuttering-invariant, $\phi \not\models_P \rho'$. Having found ρ', we let $\widehat{\rho}$ start with $s_1.\alpha_n.a_n$ and continue the construction using $s'_2.\alpha_1.a_1.\cdots.s'_{n-1}.\alpha_{n-1}.a_{n-1}.s_n.\alpha_{n+1}.\cdots$. The case in which no transition β is executed in ρ is similar (see [56]).

In summary, the key step of the construction is to "move" β across the α_i's so that it executes immediately after s_1. In the probabilistic case, we must deal with schedulers (which have a tree-like structure) instead of mere paths, and so it is not clear how a transition can be moved. Consider the scheduler η in Fig. 7.6 and the reduction in Fig. 7.4. The corresponding scheduler in $\widehat{T \parallel G}$ cannot start with the probabilistic transition $\frac{1}{2}h! + \frac{1}{2}t!$, since it is not enabled in $\widehat{\text{INIT}_{\parallel}}$. However, the same probabilistic effect is obtained by the scheduler $\widehat{\eta}$ that executes $c_h!$ in the first place, as illustrated in Fig. 7.7. In this figure, $c_h!$ is moved across both $h!$ and $t!$. In the general case, the transition in the ample set is moved across the transitions in all branches. Note that, in order to move $c_h!$ after INIT, we rely on the fact that $c_h!$ is executed after both $h!$ and $t!$. In fact, there is no way to transform the non-distributed scheduler in Fig. 7.2 into a scheduler for the reduced system in Fig. 7.4, since $c_h!$ and $c_t!$ are transitions in $\text{AMPLE}(\text{INIT}_{\parallel})$, but $c_h!$ is chosen in one of the branches, while $c_t!$ is chosen in the other.

Groesser *et al.* [12] showed how schedulers for the original system can be mapped to schedulers in the reduced system. They require condition **A5** because the transformation is not possible for some schedulers and some reductions, even if such reductions comply with **A1**–**A4**. However, we show that a similar transformation can be carried out for all schedulers η complying with the following condition:

$$\eta(\sigma)(\alpha) \in \text{AMPLE}(s_1) \ \wedge\ \eta(\sigma') \in \text{AMPLE}(s_1) \implies \eta(\sigma) = \eta(\sigma') \qquad (7.1)$$

for all $\sigma = s_1.\alpha_1.\cdots.\alpha_{n-1}.s_n$, $\sigma' = s_1.\alpha'_1.\cdots.\alpha'_{n'-1}.s'_{n'}$ such that the α_k's and the α'_k's are independent from $\text{AMPLE}(s_1)$. Roughly speaking, the first ample transition must be the same in all branches in which an ample transition appears.

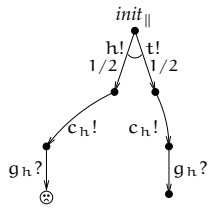


Figure 7.6: A distributed scheduler

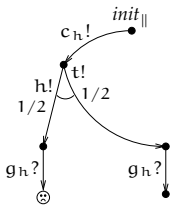


Figure 7.7: The corresponding scheduler in the reduced system

Next we show how to ensure Eqn. (7.1) in the cases of non-randomized distributed and strongly distributed schedulers. First, we show that Eqn. (7.1) holds whenever **A5**′ does and η is distributed. Let I be $\cup_{\beta \in \text{AMPLE}(s_1)} \text{INV}(\beta)$ and let σ, σ' be as in (7.1). Since the actions α_k (and the actions α'_k) are independent from all the transitions in $\text{AMPLE}(s_1)$, we have $I \cap \text{INV}(\alpha_k) = I \cap \text{INV}(\alpha_{k'}) = \emptyset$ for all k. Then, $[\![\sigma]\!]_i = [\![\sigma']\!]_i = \pi_i(s_1)$ for all $A_i \in I$. By **A5**′, we have $\text{ACTIVE}(\eta(\sigma)) = \text{ACTIVE}(\eta(\sigma'))$. Let $A_i = \text{ACTIVE}(\eta(\sigma))$ and let Θ_i be the output scheduler that defines η. Then, $\Theta_i([\![\sigma]\!]_i) = \Theta_i([\![\sigma']\!]_i)$, and so the generative transition is the same in both $\eta(\sigma)$ and $\eta(\sigma')$. The same argument can be used to show that the reactive transitions are the same in both $\eta(\sigma)$ and $\eta(\sigma')$, and so $\eta(\sigma) = \eta(\sigma')$. In conclusion, (7.1) holds whenever the schedulers are assumed to be distributed and **A5**′ holds.

In case η is strongly distributed, let

$$A_i = \text{ACTIVE}(\eta(\sigma)) \qquad \text{and} \qquad A_{i'} = \text{ACTIVE}(\eta(\sigma')) \,.$$

Then, if σ, σ' are as in (7.1), we have $[\![\sigma]\!]_i = [\![\sigma']\!]_i$ and $[\![\sigma]\!]_{i'} = [\![\sigma']\!]_{i'}$. Since η is strongly distributed, we have $A_i = \text{ACTIVE}(\eta(\sigma')) = A_{i'}$. Following the same reasoning as in the case of distributed schedulers, we have $\Theta_i([\![\sigma]\!]_i) = \Theta_i([\![\sigma']\!]_i)$ and the similarly for input schedulers. So, $\eta(\sigma) = \eta(\sigma')$ as required.

The bottom line is that the restrictions imposed to schedulers (together with **A5**′, in case distributed schedulers are assumed) allow to transform every scheduler in P into a scheduler in $\widehat{P}$ without requiring **A5**.

7.3.2 *Proof of the correctness theorems*

We prove Theorem 7.2 using a generalized notion of independence. This generalization allows us to consider independence relations that are specific for some classes of IPIOA. Particularly, in Chapter 9, we consider an independence relation for IPIOA in which the state of the system is given by a set of variables.

In our proof, the notion of independence is very close to the notion of projection, and our generalization concerns not only the independence relation but also the projection under which the schedulers are assumed to be distributed.

REMARK 7.1. We prove a variation on Theorem 7.2 that holds for projections other than $[\![\cdot]\!]$ and independence relations other than the one in Def. 7.2. On the contrary, we prove Theorem 7.1 without further generalization, that is, it holds for the independence relation in Def. 7.2 and the projection $[\![\cdot]\!]$.

In addition to the projections for atoms used throughout the thesis, we need to introduce projections for each action α.

DEFINITION 7.3. Given an action α, a pair $(\pi^{\alpha}, \pi^{\neg\alpha})$ with $\pi^{\alpha} : S_P \to S_P^{\alpha}$ and $\pi^{\neg\alpha} : S_P \to S_P^{\neg\alpha}$ is called an *α-projection* if there exists a bijection $h : S_P \to S_P^{\alpha} \times S_P^{\neg\alpha}$ such that $h(s) = (\pi^{\alpha}(s), \pi^{\neg\alpha}(s))$. Given a state s, we write $(s^{\alpha}, s^{\neg\alpha})$ for $(\pi^{\alpha}(s), \pi^{\neg\alpha}(s)) = h(s)$. Moreover, we identify s and $h(s)$, thus yielding $s = (s^{\alpha}, s^{\neg\alpha})$.

Given a state s, s^{α} represents the portion of the state affected by the execution of α, and $s^{\neg\alpha}$ represent the portion that remains unaffected. Recall that $\text{INV}(\alpha)$ is the set of atoms A_i such that α outputs a label in ACTLAB_i. Then, a very natural α-projection can be obtained using $\text{INV}(\alpha)$. Later on, we use these α-projections, and we find them closely related to $[\![\cdot]\!]$. Hence, we denote the functions as $\pi^{\alpha,[\![\cdot]\!]}$.

DEFINITION 7.4. For all α, let $\pi^{\alpha,[\![\cdot]\!]}((s_1, \cdots, s^{\mathcal{N}})) = (s_{i_1}, \cdots, s_{i_m})$ with $\{A_{i_1}, \cdots, A_{i_m}\} = \text{INV}(\alpha)$ and $\pi^{\neg\alpha,[\![\cdot]\!]}((s_1, \cdots, s^{\mathcal{N}})) = (s_{j_1}, \cdots, s_{j_{m'}})$ with $\{A_{j_1}, \cdots, A_{j_{m'}}\} = \overline{\text{INV}(\alpha)}$.

Note that $S^{\alpha,[\![\cdot]\!]} = \prod_{A_i \in \text{INV}(\alpha)} S_i$ and $S^{\neg\alpha,[\![\cdot]\!]} = \prod_{A_i \notin \text{INV}(\alpha)} S_i$. Hence, the bijection from $S_P = \prod_{A_i} S_i$ to $\prod_{A_i \in \text{INV}(\alpha)} S_i \times \prod_{A_i \notin \text{INV}(\alpha)}$ simply reorders a tuple.

The generalization to projections other than $[\![\cdot]\!]$ and independence relations other than the one in Def. 7.2 is possible only if the projections and the relations comply with certain conditions. Such conditions also involve a set of α-projections being suitable for the projection and the relation under consideration. If the independence relation, the α-projections and the projection comply with these conditions, they form an independence structure.

DEFINITION 7.5. An *independence structure* for a system P is a triple

$$(\mathcal{R}_I, \{(\pi^{\alpha}, \pi^{\neg\alpha})\}_{\alpha \in \text{ACTIONS}}, [\cdot])$$

where $\mathcal{R}_I$ is a symmetric relation (called the *independence relation*) and, for each α, $(\pi^{\alpha}, \pi^{\neg\alpha})$ is an α-projection. If $\alpha \mathcal{R}_I \beta$, we require:

1. $\alpha(s, a, s') > 0 \implies (\beta \in \text{ACTIONS}(s') \iff \beta \in \text{ACTIONS}(s))$
2. $\alpha(\,(s^{\alpha}, s^{\neg\alpha})\,,\, a\,,\, (s'^{\alpha}, s'^{\neg\alpha})\,) > 0 \implies s^{\neg\alpha} = s'^{\neg\alpha}$. Intuitively, α can only change the portion of the state modelled by S^{α}.
3. $\beta(\,(s^{\alpha}, s^{\neg\alpha})\,,\, a\,,\, (s'^{\alpha}, s'^{\neg\alpha})\,) > 0 \implies s^{\alpha} = s'^{\alpha}$ Intuitively, since β is independent from α, it cannot change the part of the state changed by α. Together with the previous property, we obtain the following result: let $\alpha_k \mathcal{R}_I \beta$ for all $k = 1, \cdots, n$, then

$$\begin{aligned} & \sigma = s^0.\alpha^1.a^1.s^1.\cdots.\alpha^n.a^n.s^n.\beta.a.s^{n+1} \\ \implies\ & \exists s^{\beta}, s'^{\beta}, s^{0^{\neg\beta}}, \cdots, s^{n^{\neg\beta}} : \\ & \qquad \sigma = (s^{\beta}, s^{0^{\neg\beta}}).\alpha^1.a^1.(s^{\beta}, s^{1^{\neg\beta}}) \\ & \qquad\qquad \cdots.\alpha^n.a^n.(s^{\beta}, s^{n^{\neg\beta}}).\beta.a.(s'^{\beta}, s^{n^{\neg\beta}}) \end{aligned} \tag{7.2}$$

This implication reflects the fact that none of the α^k can change the part of the state changed by β, and that β can change only the part

of the state corresponding to s^{k^β}. We make explicit another similar implication:

$$\begin{aligned} & \sigma = s^0.\beta.a.s^1.\alpha^1.a^1.s^2.\cdots.\alpha^n.a^n.s^{n+1} \\ \Longrightarrow\ & \exists s^\beta, s'^\beta, s^{0^{\neg\beta}}, \cdots, s^{n^{\neg\beta}} : \\ & \qquad \sigma = (s^\beta, s^{0^{\neg\beta}}).\beta.a.(s'^\beta, s^{0^{\neg\beta}}).\alpha^1.a^1.(s'^\beta, s^{1^{\neg\beta}}) \\ & \qquad\qquad \cdots.\alpha^n.a^n.(s'^\beta, s^{n^{\neg\beta}}) \end{aligned} \tag{7.3}$$

4.

$$\begin{aligned} & \alpha(\,(s^\alpha, s^{\neg\alpha}),\, a,\, (s'^\alpha, s^{\neg\alpha})\,) \cdot \beta(\,(s'^\alpha, s^{\neg\alpha}),\, b,\, (s'^\alpha, s''^{\neg\alpha})\,) \\ = \ & \beta(\,(s^\alpha, s^{\neg\alpha}),\, b,\, (s^\alpha, s''^{\neg\alpha})\,) \cdot \alpha(\,(s^\alpha, s''^{\neg\alpha}),\, a,\, (s'^\alpha, s''^{\neg\alpha})\,). \end{aligned}$$

That is, the order of α and β can be exchanged yielding the same probabilities.

5. If $\alpha \mathcal{R}_I \beta$, then $\text{ACTIVE}(\alpha) \neq \text{ACTIVE}(\beta)$.

6. If $\alpha^k \mathcal{R}_I \beta$ for all $1 \leqslant k \leqslant n$, then $\left[\sigma.\alpha^1.a^1.s^1.\cdots.\alpha^n.a^n.s^n\right]_i = [\sigma]_i$ for all A_i such that $A_i \in \text{AFFECT}(\beta)$, where

$$\text{AFFECT}(\beta) = \{A_j \mid \exists s, a \in \text{ACTLAB}(\beta) : \left|\, R_j(s,a) \,\right| > 1\} \cup \{\text{ACTIVE}(\beta)\}.$$

This set comprises all the atoms whose schedulers might change the probability that β is scheduled. Note that, if $\left|\, R_j(s,a) \,\right| = 1$ and β outputs a, then A_j does not choose how to react to a (since there is only one option), and so A_j does not affect the probability that β is executed.

By imposing this restriction, the schedulers of the atoms in $\text{AFFECT}(\beta)$ cannot change the probability for β by looking to the actions α^k independent from β.

7. If $\text{LAST}(\sigma) = (s^\beta, s^{\neg\beta})$, then

$$\begin{aligned} & \left[\sigma.\alpha^1.a^1.(s^\beta, s^{1^{\neg\beta}}) \cdots \alpha^n.a^n.(s^\beta, s^{n^{\neg\beta}})\right]_i \\ \neq\ & \left[\sigma.\alpha'^1.a'^1.(s^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^{n'}.a'^{n'}.(s^\beta, s'^{n'^{\neg\beta}})\right]_i \\ \Longrightarrow & \\ & \left[\sigma.\beta.a.(t^\beta, s^{\neg\beta}).\alpha^1.a^1.(t^\beta, s^{1^{\neg\beta}}) \cdots \alpha^n.a^n.(t^\beta, s^{n^{\neg\beta}})\right]_i \\ \neq\ & \left[\sigma.\beta.a.(t'^\beta, s^{\neg\beta}).\alpha'^1.a'^1.(t'^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^{n'}.a'^{n'}.(t'^\beta, s'^{n'^{\neg\beta}})\right]_i \end{aligned} \tag{7.4}$$

Any reasonable projection should comply with this condition: we need to require it explicitly since our definition of projection is very general. It says that, if a transition is inserted in two distinguishable paths (that is, in two paths having different projections), then the resulting paths are still distinguishable. In usual projections, the information available to an atom comprises the information (if any) that each of the transitions allows the atom to observer: the more the number of transitions, the more the information available, and so the insertion of a transition cannot turn two distinguishable paths to be undistinguishable.

8. Let $\text{LAST}(\sigma) = (s^{\beta}, s^{\neg\beta})$. For all A_i, for all

$$\sigma^1 = \sigma.\alpha^1.a^1.(s^{\beta}, s^{1^{\neg\beta}}) \cdots \alpha^n.a^n.(s^{\beta}, s^{n^{\neg\beta}}).\beta.a.(t^{\beta}, s^{n^{\neg\beta}}) \\ .\gamma^1.b^1.u^1. \cdots .\gamma^m.b^m.u^m$$

$$\sigma^2 = \sigma.\alpha'^1.a'^1.(s^{\beta}, s'^{1^{\neg\beta}}) \cdots \alpha'^{n'}.a'^{n'}.(s^{\beta}, s'^{n'^{\neg\beta}}).\beta.a.(t'^{\beta}, s'^{n'^{\neg\beta}}) \\ .\gamma'^1.b'^1.u'^1. \cdots .\gamma'^{m'}.b'^{m'}.u'^{m'}$$

(with possibly $m = 0$ and/or $m' = 0$) such that $[\sigma^1]_i \neq [\sigma^2]_i$, it must be

$$[\sigma.\beta.a.(t^{\beta}, s^{\neg\beta}).\alpha^1.a^1.(t^{\beta}, s^{1^{\neg\beta}}) \cdots \alpha^n.a^n.(t^{\beta}, s^{n^{\neg\beta}}) \\ .\gamma^1.b^1.u^1. \cdots .\gamma^m.b^m.u^m]_i$$

$$\neq [\sigma.\beta.a.(t'^{\beta}, s^{\neg\beta}).\alpha'^1.a'^1.(t'^{\beta}, s'^{1^{\neg\beta}}) \cdots \alpha'^{n'}.a'^{n'}.(t'^{\beta}, s'^{n'^{\neg\beta}}) \\ .\gamma'^1.b'^1.u'^1. \cdots .\gamma'^{m'}.b'^{m'}.u'^{m'}]_i$$

Intuitively, if two paths can be distinguished, then they remain distinguishable if β is moved after σ in both paths.

Let $\mathcal{R}_I^{\text{INV}}$ denote the independence relation in Def. 7.2. Then:

The proofs of all the remaining lemmata are in Appendix C

LEMMA 7.1. *For all simple IPIOA* P, *the triple* $(\mathcal{R}_I^{\text{INV}}, \{(\pi^{\alpha,[\![\cdot]\!]}, \pi^{\neg\alpha,[\![\cdot]\!]})\}_\alpha, [\![\cdot]\!])$ *is an independence structure.*

We prove Theorem 7.2 by proving the variation below. Note that it uses a general independence structure, and it requires the projection to be traceable.

THEOREM 7.3. *Let* ϕ *be an* $\text{LTL}_{\setminus\{\text{NEXT}\}}$ *formula,* P *be an (extended) IPIOA,* $[\cdot]$ *be traceable, and* $(\mathcal{R}_I, \{(\pi^{\alpha}, \pi^{\neg\alpha})\}_\alpha, [\cdot])$ *be an independence structure. Let* $\widehat{P}$ *be a reduction of* $\text{MDP}(P)$ *complying with conditions* **A1**–**A4**, *by taking the independence relation to be* $\mathcal{R}_I$. *Then,*

$$\sup_{\eta \in \text{SDIST}_P([\cdot])} \text{PR}^{\eta}(\phi) \leqslant \sup_{\eta \in \text{SCHED}_{\widehat{P}}} \text{PR}^{\eta}(\phi) \,.$$

Since the projection $[\![\cdot]\!]$ is not traceable, Theorem 7.2 is not a particular case of Theorem 7.3. First, we show that Theorem 7.3 implies Theorem 7.2, and then we focus on the proofs of Theorems 7.1 and 7.3. Recall that the full-communication version of a projection (Def. 4.7) is traceable (Theorem 4.5). The first step in the proof of Theorem 7.2 is to show that $|[\![\cdot]\!]|$ also yields an independence structure.

LEMMA 7.2. *For all simple IPIOA* P, *the triple* $(\mathcal{R}_I^{\text{INV}}, \{(\pi^{\alpha,[\![\cdot]\!]}, \pi^{\neg\alpha,[\![\cdot]\!]})\}_\alpha, |[\![\cdot]\!]|)$ *is an independence structure.*

Note that, according to Lemma 7.2, the functions $\pi^{\alpha,[\![\cdot]\!]}$ for $[\![\cdot]\!]$ are also suitable for the full-communication version $|[\![\cdot]\!]|$.

Proof of Theorem 7.2 (Assuming Theorem 7.3). Let ϕ, P and $\widehat{P}$ be as in Theorem 7.2. By considering the independence structure in Lemma 7.2 we are under the hypothesis of Theorem 7.3, since Theorem 4.5 ensures that $|[\![\cdot]\!]|$ is

a traceable projection, and simple IPIOA are a particular case of generalized IPIOA Subsection 1.2.3. Hence, Theorem 7.3 (applied to $|[\![\cdot]\!]|$) yields

$$\sup_{\eta \in \text{SDIST}_P(|[\![\cdot]\!]|)} \text{PR}^\eta(\phi) \leqslant \sup_{\eta \in \text{SCHED}_{\tilde{P}}} \text{PR}^\eta(\phi) \,.$$

Theorems 4.4 and 2.3 give

$$\sup_{\eta \in \text{SDIST}_P} \text{PR}^\eta(\phi) \leqslant \sup_{\eta \in \text{SDIST}_P(|[\![\cdot]\!]|)} \text{PR}^\eta(\phi) \,,$$

and so

$$\sup_{\eta \in \text{SDIST}_P} \text{PR}^\eta(\phi) \leqslant \sup_{\eta \in \text{SCHED}_{\tilde{P}}} \text{PR}^\eta(\phi) \,,$$

which is what we want to prove. □

The proof of Theorems 7.1 and 7.3 starts by formally defining the transformation on schedulers discussed in Subsection 7.3.1. In the general case, the transformation takes a global path σ^s and a scheduler η, and yields a scheduler η' such that $\eta'(\sigma^s)(\beta) = 1$ for some $\beta \in \text{AMPLE}(\text{LAST}(\sigma^s))$. This scheduler is denoted by $\eta[\sigma^s \leftarrow \beta]$. Intuitively, the transformation obtains $\eta[\sigma^s \leftarrow \beta]$ by moving β across the actions that η schedules before β. In Subsection 7.3.1, we discussed the case of non-randomized schedulers and, in order to carry the transformation out, we imposed a restriction on the schedulers (namely, Eqn. (7.1)). Now, we impose restrictions on randomized schedulers.

In Sec. 4.1 we have seen that input and output schedulers can be transformed into non-randomized schedulers without increasing/decreasing the probability of the property being checked (Lemma 4.3 and Lemma 4.4). Hence, we can assume that the schedulers are I/O non-randomized (that is, the input and output schedulers are non-randomized). In addition to this assumption, we show that we can restrict to schedulers complying with a generalization of Eqn. (7.1). In this restriction, we require the action in the ample set to be chosen with probability 1.

$$\begin{aligned} &\eta(\overbrace{\sigma^s.\alpha^1.s^1.\cdots.\alpha^n.s^n}^{\sigma})(\beta) > 0 \wedge \beta \in \text{AMPLE}(\text{LAST}(\sigma^s)) \\ \wedge\ &\eta(\overbrace{\sigma^s.\alpha'^1.s'^1.\cdots.\alpha'^{n'}.s'^{n'}}^{\sigma'})(\beta') > 0 \wedge \beta' \in \text{AMPLE}(\text{LAST}(\sigma^s)) \\ &\qquad \Longrightarrow \eta(\sigma) = \eta(\sigma') = 1 : \beta = 1 : \beta' \end{aligned} \tag{7.5}$$

where $\alpha^k, \alpha^{k'} \notin \text{AMPLE}(\text{LAST}(\sigma^s))$ for all k, k', and σ^s is the path to which we apply the transformation, that is, we want $\eta[\sigma^s \leftarrow \beta](\sigma^s)(\beta'') = 1$ for some $\beta'' \in \text{AMPLE}(\text{LAST}(\sigma^s))$.

Since the schedulers we are dealing with are I/O non-randomized, in order to get $\eta(\sigma)(\beta) = 1$ it suffices to ensure $\mathcal{I}(\sigma)(\text{ACTIVE}(\beta)) = 1$. For the proof of Theorem 7.1 (concerning distributed schedulers), we can assume that the interleaving scheduler is non-randomized (Theorem 4.1). By property *(6)* of independence structures (Def. 7.5), we have $[\sigma]_i = [\sigma']_i = [\sigma^s]_i$ for all $A_i \in \text{AFFECT}(\beta)$. Moreover by **A5′**, we have $\text{ACTIVE}(\beta) = \text{ACTIVE}(\beta')$. Using these equalities, and the fact that η is distributed, we deduce $\beta = \beta'$ in the same way as we did for Eqn. (7.1). Note that, in order to make Eqn. (7.5) hold, the restriction $\eta \in \text{DIST}_P([\cdot])$ is unnecessarily strong: in fact, Eqn. (7.5) holds if $\Theta_i(\sigma) = \Theta_i(\sigma')$ for all σ, σ' such that $[\sigma]_i = [\sigma']_i$ and $\sigma^s \sqsubseteq \sigma$, $\sigma^s \sqsubseteq \sigma'$ (and similarly for input schedulers). This motivates the following definition.

DEFINITION 7.6. We say that a scheduler $\eta = (\mathcal{I}, \{\Theta_i\}_i, \{\Upsilon_i\}_i) \in \text{SCHED}_P$ is *distributed after* σ^s iff for all A_i, $\Theta_i(\sigma) = \Theta_i(\sigma')$, $\Upsilon_i(\sigma, a) = \Upsilon_i(\sigma', a)$ whenever σ, σ' comply with $[\sigma]_i = [\sigma']_i$ and $\sigma^s \sqsubseteq \sigma$, $\sigma^s \sqsubseteq \sigma'$.

We say that a scheduler $\eta \in \text{SCHED}_P$ is strongly distributed after σ^s iff η is distributed after σ^s and for all A_i, A_j,

$$\frac{\mathcal{I}(\sigma)(A_i)}{\mathcal{I}(\sigma)(A_i) + \mathcal{I}(\sigma)(A_j)} = \frac{\mathcal{I}(\sigma')(A_i)}{\mathcal{I}(\sigma')(A_i) + \mathcal{I}(\sigma')(A_j)}$$

whenever σ, σ' comply with $\text{PR}^\eta((\sigma)^\uparrow) > 0$, $\text{PR}^\eta((\sigma')^\uparrow) > 0$ $[\sigma]_i = [\sigma']_i$, $[\sigma]_j = [\sigma']_j$ and $\sigma^s \sqsubseteq \sigma$, $\sigma^s \sqsubseteq \sigma'$.

In the setting of strongly distributed schedulers we have seen that randomized resolutions of nondeterminism add expressive power to interleaving schedulers (Subsection 4.2.1), and so we cannot ensure that the action β in the ample set is scheduled with probability 1 in η, as required by (7.5). Next, we present a lemma allowing us to assume that strongly distributed schedulers after σ^s comply with Eqn. (7.5).

LEMMA 7.3. *Let $\mathcal{S}$ be a measurable set of infinite paths, let $[\cdot]$ be a traceable projection and*

$$\mathcal{A} = \{A \in \text{ATOMS}(P) \mid \exists \alpha \in \text{AMPLE}(\text{LAST}(\sigma^s)) : A = \text{ACTIVE}(\alpha)\} .$$

If $\eta = (\mathcal{I}, \{\Theta_i\}_i, \{\Upsilon_i\}_i)$ is strongly distributed after σ^s under $[\cdot]$, then there exists $A_{i^*} \in \mathcal{A}$, $\eta^* = (\mathcal{I}^*, \{\Theta_i\}_i, \{\Upsilon_i\}_i)$ such that all the following properties hold:

I. $\text{PR}^{\eta^*}(\mathcal{S}) \geqslant \text{PR}^\eta(\mathcal{S})$

II. η^* *is strongly distributed after* σ^s

III. for all $A_j \in \mathcal{A}$, $\sigma' \sqsupseteq \sigma^s$, *we have*

$$\text{PR}^{\eta^*}((\sigma')^\uparrow) > 0 \wedge \mathcal{I}^*(\sigma')(A_j) > 0 \implies \exists \sigma'' : \sigma^s \sqsubseteq \sigma'' \sqsubseteq \sigma' \wedge \mathcal{I}^*(\sigma'')(A_{i^*}) = 1$$

IV. for all σ, *either* $\mathcal{I}^*(\sigma) = \mathcal{I}(\sigma)$ *or* $\mathcal{I}^*(\sigma)(A_j) = 1$ *for some* A_j. *Moreover* $\eta^*(\sigma) = \eta(\sigma)$ *for all* σ *such that* $\sigma^s \not\sqsubseteq \sigma$.

We show how that the scheduler η^* complies with Eqn. (7.5). Let η^* be as in the previous lemma, and let σ, σ', β, β' be as in Eqn. (7.5). Since β (β', resp.) is in the ample set, property *(5)* of independence structures implies that β is the first action in σ generated by $\text{ACTIVE}(\beta)$ after σ^s (otherwise, if $\text{ACTIVE}(\alpha^k) = \text{ACTIVE}(\beta)$ for some k then, by *(5)* implies $\alpha^k \not\mathcal{R}_I \beta$, thus contradicting **A3**). The same holds for β', σ', $\text{ACTIVE}(\beta')$. Moreover, β (β', resp.) is the first action in the ample executed after σ^s (recall that, in Eqn. (7.5), we have $\alpha^k, \alpha^{k'} \notin \text{AMPLE}(\text{LAST}(\sigma^s))$). Since $\text{ACTIVE}(\beta), \text{ACTIVE}(\beta') \in \mathcal{A}$, property *(III)* ensures that $\text{ACTIVE}(\beta) = \text{ACTIVE}(\beta') = A_{i^*}$ and so

$$\mathcal{I}^*(\sigma)(\text{ACTIVE}(\beta)) = \mathcal{I}^*(\sigma')(\text{ACTIVE}(\beta')) = 1 .$$

The choice of the interleaving scheduler coincides for both paths. The choices of the output and input schedulers coincide, since η^* is distributed after σ^s (*(II)*), and so the argument for distributed schedulers applies here as well. Then, $\eta^*(\sigma) = 1{:}\beta = 1{:}\beta'$.

So far we have shown that, under the hypotheses of Theorems 7.1 and 7.3, we can assume that Eqn. (7.5) holds. Note that Eqn. (7.5) does not require an action in $\textsc{Ample}(\textsc{Last}(\sigma^s))$ to be chosen by η (it requires that, if an action in the ample set is chosen, then the same action must be chosen in σ, σ'). Given η and σ^s, if there exists β as in Eqn. (7.5), we define $\eta[\sigma^s \leftarrow \beta]$ in such a way that $\eta[\sigma^s \leftarrow \beta](\sigma^s)(\beta) = 1$. If no such β exists, we define $\eta[\sigma^s \leftarrow \beta'']$ in such a way that $\eta[\sigma^s \leftarrow \beta](\sigma^s)(\beta'') = 1$, for some $\beta'' \in \textsc{Ample}(\textsc{Last}(\sigma^s))$.

Hence, it suffices to define $\eta[\sigma^s \leftarrow \beta]$ for all η, σ^s, $\beta \in \textsc{Ample}(\textsc{Last}(\sigma^s))$ such that Eqn. (7.5) holds and

$$\eta(\sigma.\alpha^1.s^1.\cdots.\alpha^n.s^n)(\beta') = 1 \wedge \beta' \in \textsc{Ample}(\textsc{Last}(\sigma^s)) \implies \beta' = \beta$$

whenever $\alpha^k \notin \textsc{Ample}(\textsc{Last}(\sigma^s))$, and $\beta' \in \textsc{Ample}(\textsc{Last}(\sigma^s))$.

The scheduler $\eta[\sigma^s \leftarrow \beta]$ is defined using a correspondence among paths. To each path in η we assign a set of paths, as illustrated in Fig. 7.8. First, we explain the figure, and then we give a formal definition of the correspondence $\mathcal{C}$. If β occurs in a path σ, then $\mathcal{C}(\sigma)$ is a singleton set $\{\sigma'\}$, where σ' coincides with σ except for the fact that β is chosen after σ^s. Consider the path $\sigma = \sigma^s.\alpha^1.a^1.s^1.\cdots.\alpha^n.a^n.s^n.\beta.s^{n+1}$ in Fig. 7.8. From the implication (7.2), we have

$$\sigma = \sigma^s.\alpha^1.a^1.(s^\beta, s^{1^{\neg\beta}}).\cdots.\alpha^n.a^n.(s^\beta, s^{n^{\neg\beta}}).\beta.a.(s'^\beta, s^{n^{\neg\beta}}) \ .$$

In the figure, the x-lines represent changes arising from the execution of the actions α^k: in other words, these lines represent the sequence $s^{1^{\neg\beta}} \cdots s^{n^{\neg\beta}}$. The dotted lines (as well as the dashed lines) represents changes arising from the execution of β: it represents the transition from s^β to s'^β. The dashed line represents another possible outcome of β. A different outcome of α^1 may lead to a different extension of σ^s: this extension is represented as an o-line. Note that the arrow comprising the x-line and the dotted line in η maps to the path σ' in which the outcome of β are the dots, and σ' continues with x. The same happens for all possible combinations of x, o, dots and dashes. In the lowermost path σ^l, the action β does not occur. For such paths, we define the set $\mathcal{C}(\sigma^l)$ as

$$\begin{array}{ll}\{ & \sigma^s.\beta.a_1.(s^{1^\beta}, s^{1^{\neg\beta}}).\cdots.(s^{1^\beta}, s^{n^{\neg\beta}}) \\ & \sigma^s.\beta.a_2.(s^{2^\beta}, s^{1^{\neg\beta}}).\cdots.(s^{2^\beta}, s^{n^{\neg\beta}}) \\ & \cdots \\ & \sigma^s.\beta.a_Q.(s^{Q^\beta}, s^{1^{\neg\beta}}).\cdots.(s^{Q^\beta}, s^{n^{\neg\beta}}) \quad \}\end{array}$$

where all the a_q, s^{q^β} are possible outcomes of β. In the figure, the path made up of + maps to a set comprising

- a path comprising a +-line (representing the sequence $s^{1^{\neg\beta}} \cdots s^{n^{\neg\beta}}$) and a dotted line (representing the outcome a_1, s^{1^β}) and
- a path comprising a + line and a dashed line (representing another outcome of β).

DEFINITION 7.7. Let $\textsc{Last}(\sigma^s) = s = (s^\beta, s^{\neg\beta})$ for some $\beta \in \textsc{Ample}(s)$, and let $\alpha^1 \cdots \alpha^n \notin \textsc{Ample}(s)$. Then,

1. If σ is of the form

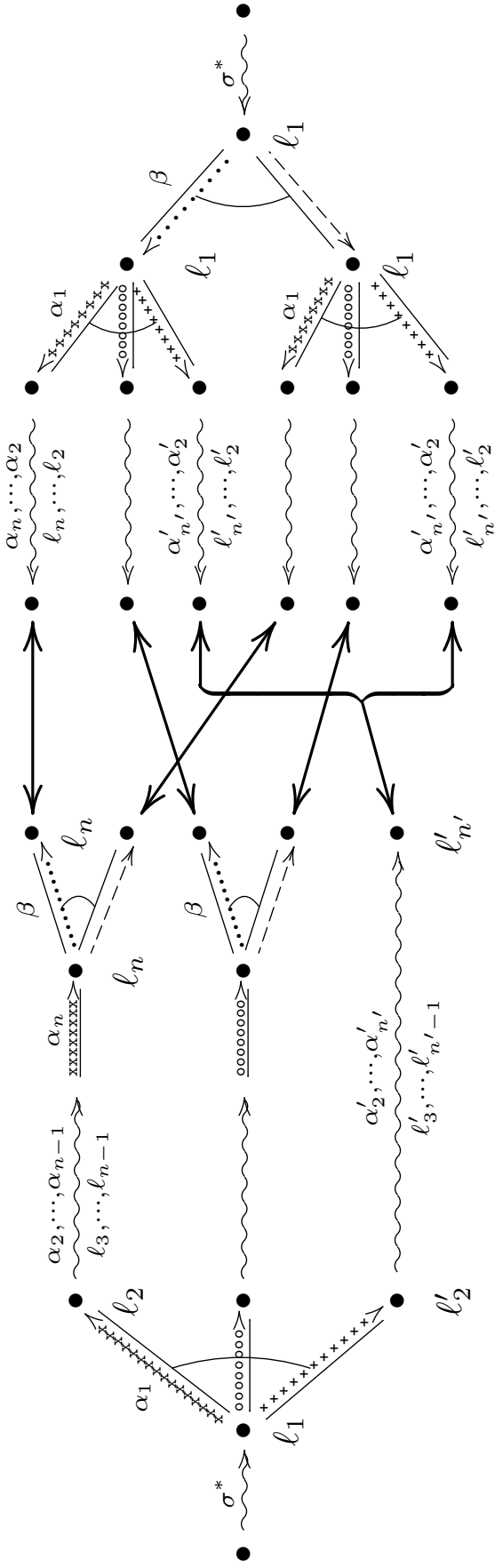


Figure 7.8: Mapping paths in η to paths starting with β

$$\sigma^s.\alpha^1.a^1.(s^\beta, s^{1^{\neg\beta}}).\cdots.\alpha^n.a^n.(s^\beta, s^{n^{\neg\beta}}) \qquad \triangleleft\alpha\triangleright$$

we define

$$\mathcal{C}(\sigma) = \{\sigma^s.\beta.a.(s'^\beta, s^{\neg\beta}).\alpha^1.a^1.(s'^\beta, s^{1^{\neg\beta}}).\cdots.\alpha^n.a^n.(s'^\beta, s^{n^{\neg\beta}}) \mid \beta(s, a, (s'^\beta, s^{\neg\beta})) > 0\}\,.$$

We consider the case in which $\sigma = \sigma^s$ as a particular case of this one. So,

$$\mathcal{C}(\sigma^s) = \{\sigma^s.\beta.a.(s'^\beta, s^{\neg\beta}) \mid \beta(s, a, (s'^\beta, s^{\neg\beta})) > 0\}\,.$$

In the following, we write $\sigma \sim \triangleleft\alpha\triangleright$ to indicate that σ has the form of the path marked with $\triangleleft\alpha\triangleright$ above. $\sigma \sim \triangleleft\alpha\triangleright$

2. If σ is of the form

$$\sigma^s.\alpha^1.a^1.(s^\beta, s^{1^{\neg\beta}}).\cdots.\alpha^n.a^n.(s^\beta, s^{n^{\neg\beta}}).\beta.a.(s'^\beta, s^{n^{\neg\beta}}) \qquad \triangleleft\alpha\,\beta\triangleright$$

then

$$\mathcal{C}(\sigma) = \{\sigma^s.\beta.a.(s'^\beta, s^{\neg\beta}).\alpha^1.a^1.(s'^\beta, s^{1^{\neg\beta}}).\cdots.\alpha^n.a^n.(s'^\beta, s^{n^{\neg\beta}})\}\,.$$

(Note that, in this case, $\mathcal{C}(\sigma)$ is a singleton set.)

In the following, we write $\sigma \sim \triangleleft\alpha\,\beta\triangleright$ to denote that σ has the form of the path marked with $\triangleleft\alpha\,\beta\triangleright$ above.

3. If σ is of the form

$$\sigma^s.\alpha^1.a^1.(s^\beta, s^{1^{\neg\beta}}).\cdots.\alpha^n.a^n.(s^\beta, s^{n^{\neg\beta}}).\beta.a.(s'^\beta, s^{n^{\neg\beta}}).\gamma^1.b^1.t^1.\cdots.\gamma^m.b^m.t^m \qquad \triangleleft\alpha\,\beta\,\gamma\triangleright$$

then

$$\mathcal{C}(\sigma) = \{\sigma^s.\beta.a.(s'^\beta, s^{\neg\beta}).\alpha^1.a^1.(s'^\beta, s^{1^{\neg\beta}}).\cdots.\alpha^n.a^n.(s'^\beta, s^{n^{\neg\beta}}).\gamma^1.b^1.t^1.\cdots.\gamma^m.b^m.t^m\}.$$

(Again, $\mathcal{C}(\sigma)$ is a singleton set.)

In the following, we write $\sigma \sim \triangleleft\alpha\,\beta\,\gamma\triangleright$ to denote that σ has the form of the path marked with $\triangleleft\alpha\,\beta\,\gamma\triangleright$ above.

4. If $\sigma^s \not\sqsubseteq \sigma$, we define $\mathcal{C}(\sigma) = \{\sigma\}$.

   In the following, we write $\sigma \sim \triangleleft\neg\sigma^s\triangleright$ to indicate that $\sigma^s \not\sqsubseteq \sigma$.

We write $\sigma \not\sim \triangleleft\alpha\triangleright$ to denote $\sigma \sim \triangleleft\alpha\,\beta\triangleright$ or $\sigma \sim \triangleleft\alpha\,\beta\,\gamma\triangleright$ or $\sigma \sim \triangleleft\neg\sigma^s\triangleright$. $\sigma \not\sim \triangleleft\alpha\triangleright$

Prior to the definition of $\eta[\sigma^s \leftarrow \beta]$ in terms of $\mathcal{C}$, we need to discuss some properties of $\mathcal{C}$. If neither σ' nor σ'' are of the form $\triangleleft\alpha\,\beta\triangleright$, we have $\mathcal{C}(\sigma') \cap \mathcal{C}(\sigma'') = \emptyset$ whenever $\sigma' \neq \sigma''$. In fact, the following lemma holds by definition of $\mathcal{C}$.

LEMMA 7.4. *Let σ', σ'' be such that, $\sigma' \neq \sigma''$, $\mathrm{PR}^\eta((\sigma')^\dagger) > 0$, $\mathrm{PR}^\eta((\sigma'')^\dagger) > 0$.*

Then, $\mathcal{C}(\sigma') \cap \mathcal{C}(\sigma'') \neq \emptyset$ implies $\sigma' \sim \triangleleft\alpha\triangleright$ (or, symmetrically, $\sigma'' \sim \triangleleft\alpha\triangleright$) and $\sigma'' = \sigma'.\beta.a.(s'^\beta, s^{n^{\neg\beta}})$ (resp., $\sigma' = \sigma''.\beta.a.(s'^\beta.s^{n^{\neg\beta}})$) for some s'^β. Note that $\sigma'' \sim \triangleleft\alpha\,\beta\triangleright$ (resp., $\sigma' \sim \triangleleft\alpha\,\beta\triangleright$). For such σ', σ'', we have $\mathcal{C}(\sigma') \cap \mathcal{C}(\sigma'') = \mathcal{C}(\sigma'')$ (resp. $\mathcal{C}(\sigma') \cap \mathcal{C}(\sigma'') = \mathcal{C}(\sigma')$).

Then, the definition of $\eta[\sigma^s \leftarrow \beta]$ goes as follows.

DEFINITION 7.8.

$$\eta[\sigma^s \leftarrow \beta](\sigma) = \begin{cases} 1{:}\beta & \text{if } \sigma = \sigma^s \\ \eta(\sigma') & \text{if there exists } \sigma' \not\sim \triangleleft\alpha\triangleright \\ & \text{complying } \mathrm{PR}^\eta((\sigma')^\uparrow) > 0 \text{ and } \mathcal{C}(\sigma') = \{\sigma\} \\ \eta(\sigma'') & \text{if the previous case does not hold,} \\ & \text{and there exists } \sigma'' \sim \triangleleft\alpha\triangleright \text{ complying} \\ & \mathrm{PR}^\eta((\sigma'')^\uparrow) > 0 \text{ and } \sigma \in \mathcal{C}(\sigma'') \end{cases}$$

In order to explore the subtleties of the definition, consider the case in which both σ' and σ'' exist. In this case, according to Lemma 7.4 we have $\sigma' \sim \triangleleft\alpha\,\beta\triangleright$ and $\sigma'' \sim \triangleleft\alpha\triangleright$. Then, the restriction "if the previous case does not hold" gives the preference to σ' (instead of σ'') Note that σ is of the form $\sigma^s.\beta.a.(s'^\beta, s^{\neg\beta}).\alpha^1 \cdots (s'^\beta, s^{n^{\neg\beta}})$, and so the preference for σ' results in

$$\eta[\sigma^s \leftarrow \beta](\sigma^s.\beta.a.(s'^\beta, s^{\neg\beta}).\alpha^1 \cdots (s'^\beta, s^{n^{\neg\beta}})) = \eta(\sigma^s.\alpha^1 \cdots (s^\beta, s^{n^{\neg\beta}}).\beta.a.(s'^\beta, s^{n^{\neg\beta}}) ,$$

as desired. On the contrary, a preference for σ'' (which is of the form $\triangleleft\alpha\triangleright$) would define $\eta[\sigma^s \leftarrow \beta](\sigma^s.\beta.a.(s'^\beta, s^{\neg\beta}).\alpha^1 \cdots (s'^\beta, s^{n^{\neg\beta}}))$ as $1{:}\beta$, which is clearly incorrect since this occurrence of β has been already executed in σ.

In rigor, Def. 7.8 does not define the scheduler completely since, for certain paths, there might be the case that neither σ' nor σ'' exists. However, given two schedulers η', η'' as in Def. 7.8, the following lemma implies that η' and η'' differ only for paths with zero probability.

LEMMA 7.5. *Let $\eta[\sigma^s \leftarrow \beta]$ be as in Def. 7.8. If $\mathrm{PR}^{\eta[\sigma^s \leftarrow \beta]}((\sigma)^\uparrow) > 0$ then either $\sigma = \sigma^s$ or there exists σ' such that $\sigma \in \mathcal{C}(\sigma')$ and $\mathrm{PR}^\eta((\sigma')^\uparrow) > 0$.*

The definition of $\eta[\sigma^s \leftarrow \beta]$ allows to define a scheduler such that $\eta[\sigma^s \leftarrow \beta](\sigma^s) = 1{:}\beta$ for some $\beta \in \mathrm{AMPLE}(\mathrm{LAST}(\sigma^s))$. In order to obtain a scheduler in the reduced system, the transformation from η to $\eta[\sigma^s \leftarrow \beta]$ must be carried out until only actions in the reduced system are chosen in all paths. Suppose that $\eta(\mathrm{INIT})(\alpha) > 0$ for some $\alpha \notin \mathrm{AMPLE}(\mathrm{INIT})$. Then, we obtain the scheduler $\eta' = \eta[\mathrm{INIT} \leftarrow \beta]$ for some $\beta \in \mathrm{AMPLE}(\mathrm{INIT})$. Now, it may be the case that $\eta'(\mathrm{INIT}.\beta.a.s')(\alpha) > 0$ for some $\alpha \notin \mathrm{AMPLE}(s')$, and so the transformation must be carried out again for the path $\mathrm{INIT}.\beta.a.s'$. Recall that the transformation can be carried out only if condition (7.5) holds. This is ensured by the following lemma.

LEMMA 7.6. *Let P be a IPIOA, and $(\mathcal{R}_I, \{(\pi^\alpha, \pi^{\neg\alpha})\}_\alpha, [\cdot])$ be an independence structure such that either:*

1. *P is a simple IPIOA, η is distributed after σ^s and $(\mathcal{R}_I, \{(\pi^\alpha, \pi^{\neg\alpha})\}_\alpha, [\cdot]) = (\mathcal{R}_I^{\mathrm{INV}}, \{(\pi^{\alpha,[\![\cdot]\!]}, \pi^{\neg\alpha,[\![\cdot]\!]})\}_\alpha, [\![\cdot]\!])$ or*
2. *η is strongly distributed after σ^s and $[\cdot]$ is traceable.*

For all $\sigma' = \sigma^s.\beta.a.s$ *with* $\Pr^{\eta[\sigma^s \leftarrow \beta]}((\sigma')^\uparrow) > 0$, *the scheduler* $\eta[\sigma^s \leftarrow \beta]$ *is distributed (in case (1)) or strongly distributed (in case (2)) after* σ'.

In order to prove that two schedulers (in our case, η and $\eta[\sigma^s \leftarrow \beta]$) yield the same probability for an $\mathrm{LTL}_{\setminus\{\mathrm{NEXT}\}}$ property, we use the concept of stuttering-invariant *cylinder*. Given n sets of labels $\ell_1 \cdots \ell_n \in \mathcal{P}(AP)$ such that $\ell_k \neq \ell_{k+1}$, the stuttering-invariant cylinder $\mathrm{CYL}(\ell_1^+, \cdots, \ell_n^+)$ is defined as the set of infinite paths ρ such that ρ has a finite prefix σ complying

$$\exists k_1 > 0, \cdots, k_n > 0 : \mathrm{TRACE}(\sigma) = \overbrace{\ell_1 \dots \ell_1}^{k_1 \text{ times}} \cdots \overbrace{\ell_n \dots \ell_n}^{k_n \text{ times}} .$$

For such a σ, we write $\sigma \sim \ell_1^+, \cdots, \ell_n^+$. $\sigma \sim \ell_1^+, \cdots, \ell_n^+$

The key of the correctness proof is the following lemma.

LEMMA 7.7. *For all cylinders* $\mathrm{CYL}(\ell_1^+, \cdots, \ell_n^+)$,

$$\Pr^{\eta}(\mathrm{CYL}(\ell_1^+, \cdots, \ell_n^+)) = \Pr^{\eta[\sigma^s \leftarrow \beta]}(\mathrm{CYL}(\ell_1^+, \cdots, \ell_n^+)) .$$

Using Lemma 7.7, we can prove Theorems 7.1 and 7.3 by resorting to the same argument as in [12].

Proof of Theorems 7.1 and 7.3. We prove the theorems by proving *Proof of Theorems 7.1 and 7.3*

$$\forall \eta \in \mathrm{DIST}_P([\cdot]) : \exists \eta' \in \mathrm{SCHED}_{\widehat{P}} : \Pr^{\eta'}(\phi) \geqslant \Pr^{\eta}(\phi)$$

(for Theorem 7.1) and

$$\forall \eta \in \mathrm{SDIST}_P([\cdot]) : \exists \eta' \in \mathrm{SCHED}_{\widehat{P}} : \Pr^{\eta'}(\phi) \geqslant \Pr^{\eta}(\phi)$$

(for Theorem 7.3).

From standars arguments in measure theory, it suffices to prove that, for all schedulers $\eta \in \mathrm{DIST}_P([\cdot])$ ($\eta \in \mathrm{SDIST}_P([\cdot])$, resp.) there exists $\eta' \in \mathrm{SCHED}_{\widehat{P}}$ such that $\Pr^{\eta'}(\mathrm{CYL}(\ell_1^+, \cdots, \ell_n^+)) \geqslant \Pr^{\eta}(\mathrm{CYL}(\ell_1^+, \cdots, \ell_n^+))$ for all cylinder $\mathrm{CYL}(\ell_1^+, \cdots, \ell_n^+)$.

Given η, we construct a sequence of schedulers η_n with $\eta_0 = \eta$. Then, we construct η' using the schedulers η_n. Such schedulers comply with the following properties:

$$\forall \sigma : \mathrm{LEN}(\sigma) \leqslant n \implies \exists \beta \in \mathrm{AMPLE}(\mathrm{LAST}(\sigma)) : \eta_n(\sigma) = 1 : \beta \tag{7.6}$$

$$\Pr^{\eta_n}(\mathrm{CYL}(\ell_1^+, \cdots, \ell_n^+)) \geqslant \Pr^{\eta}(\mathrm{CYL}(\ell_1^+, \cdots, \ell_n^+)) \tag{7.7}$$

$$\eta_n \text{ is distributed (strongly distributed, resp.) after } \sigma^s \tag{7.8}$$

for all σ^s of length $n+1$ such that $\Pr^{\eta_n}(\sigma^s) > 0$.

In order to obtain the scheduler η_{n+1} from η_n, we construct schedulers $\eta_{n,S}$, where S is a finite set of finite paths of length $n+1$. The scheduler corresponding to the empty set is defined as $\eta_{n,\emptyset} = \eta_n$. Given $\eta_{n,S}$ and a path σ^s such that $\sigma^s \notin S$, $\mathrm{LEN}(\sigma^s) = n+1$ and $\Pr^{\eta_{n,S}}(\sigma^s) > 0$, we obtain the scheduler $\eta_{n,S\cup\{\sigma^s\}}$.

The following explanations, concerning the construction of $\eta_{n,S\cup\{\sigma^s\}}$ from $\eta_{n,S}$, are intended to be a summary of the arguments motivating the definitions and lemmata before this proof. In the case of strongly distributed schedulers, we apply Lemma 7.3 to $\eta_{n,S}$, in order to obtain a scheduler η^*. In the case of distributed schedulers, we simply let $\eta^* = \eta_{n,S}$. As explained right after Lemma 7.3, condition (7.5) holds for η^*, and so we can define $\eta_{n,S\cup\{\sigma^s\}} = \eta^*[\sigma^s \leftarrow \beta]$.

Starting from $\eta_{n,\emptyset}$, this construction can be repeated until we reach a set S^n such that S^n comprises all paths of length $n+1$ with positive probability in η_{n,S^n} (the construction finishes, since there are finitely many paths of length $n+1$). Then, we define $\eta_{n+1} = \eta_{n,S^n}$.

The properties (7.6), (7.7) required for η_{n+1} hold by the following properties for the schedulers $\eta_{n,S}$: by definition,

$$\eta_{n,S}(\sigma) = 1 : \beta$$

for some $\beta \in \textsc{ample}(\textsc{last}(\sigma))$ for all $\sigma \in S$. By Lemma 7.7 and property *(I)* in Lemma 7.3

$$\begin{aligned}\Pr^{\eta_{n,S\cup\{\sigma^s\}}}(\textsc{cyl}(\ell_1^+,\cdots,\ell_n^+)) &= \Pr^{\eta^*[\sigma^s\leftarrow\beta]}(\textsc{cyl}(\ell_1^+,\cdots,\ell_n^+)) \\ &= \Pr^{\eta^*}(\textsc{cyl}(\ell_1^+,\cdots,\ell_n^+)) \geqslant \Pr^{\eta_{n,S}}(\textsc{cyl}(\ell_1^+,\cdots,\ell_n^+))\,.\end{aligned}$$

It remains to check property (7.8). In the case of distributed schedulers $\eta^* = \eta_{n,S}$ and so, by inductive hypothesis, η^* is distributed after σ for all $\sigma \in S$. From Lemma 7.6, we get that $\eta_{n,S}$ is distributed after σ'', for every path σ'' of length $n+2$ such that $\sigma' \sqsubset \sigma''$ for some σ' in S. In the case of strongly distributed schedulers, by Lemma 7.3, property *(II)*, η^* is strongly after σ^s. Since the scheduler obtained using Lemma 7.3 only changes suffixes of σ^s (property *(IV)*), η^* is strongly distributed after σ for all $\sigma \in S \cup \{\sigma^s\}$. Again, from Lemma 7.6, we get that $\eta_{n,S}$ is strongly distributed after σ'', for every path σ'' of length $n+2$ such that $\sigma' \sqsubset \sigma''$ for some σ' in S. Then, property (7.8) holds for η_{n+1} since, for all paths of length $n+2$ having positive probability in η_{n+1}, there exists a prefix of length $n+1$ with positive probability in $\eta_{n,S^n} = \eta_{n+1}$.

The schedulers η_n are used to construct a scheduler for the reduced system. Let η' be the scheduler defined as $\eta'(\sigma) = \eta_{\textsc{len}(\sigma)}(\sigma)$. From property (7.6), we get $\eta'(\sigma) = 1:\beta$ for some $\beta \in \textsc{ample}(\textsc{last}(\sigma))$ for all σ. Then $\eta' \in \textsc{sched}_{\widehat{P}}$.

It remains to prove $\Pr^{\eta'}(\textsc{cyl}(\ell_1^+,\cdots,\ell_n^+)) \geqslant \Pr^{\eta}(\textsc{cyl}(\ell_1^+,\cdots,\ell_n^+))$. As explained in [12, 86], property (7.7) is not sufficient to conclude

$$\Pr^{\eta'}(\textsc{cyl}(\ell_1^+,\cdots,\ell_n^+)) \geqslant \Pr^{\eta}(\textsc{cyl}(\ell_1^+,\cdots,\ell_n^+))\,. \tag{7.9}$$

Here, we simply recall the arguments in [12, 86]. Consider the system $A \parallel B$ in Fig. 7.9 and the formula $\phi = \mathsf{F}\text{☺}$. If $\eta(\textsc{init}) = \alpha$ and $\textsc{ample}(\textsc{init}) = \{\beta\}$, then the scheduler η' chooses β for all the paths, and so $\Pr^{\eta}(\phi) = 1$, while $\Pr^{\eta'}(\phi) = 0$. However, the ample set $\{\beta\}$ violates condition (**A4**), since $\textsc{init}$ and α form an end component and β is enabled in $\textsc{init}$. Note that, in this case, the path $\textsc{init}.\beta.\textsc{init}.\beta.\cdots$ has positive probability in η' (namely 1) while it has no corresponding path in η.

We explain the case in which (**A4**) holds using Fig. 7.10. For the atoms in the figure, $\{\beta\}$ is a valid ample set for $\textsc{init}$. Let s' be state in which A is in

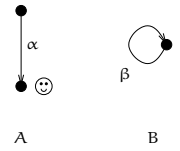


Figure 7.9: Example showing the need for (**A4**).

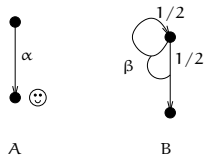


Figure 7.10: Another example showing the need for (**A4**)

its initial state, while B is in its final state. Then, it must be $\alpha \in \text{AMPLE}(s')$. Again, consider a scheduler such that $\eta(\text{INIT}) = \alpha$. Then, the path σ such that $\text{PR}^{\eta}(\sigma) = 1$ has the following corresponding paths in η':

$$\sigma'_1 = \text{INIT}.\beta.s'.\alpha.\textcircled{\scriptsize ☺}$$
$$\sigma'_2 = \text{INIT}.\beta.\text{INIT}.\beta.s'.\alpha.\textcircled{\scriptsize ☺}$$
$$\dots$$

As expected, we have

$$\sum_{i=1}^{\infty} \text{PR}^{\eta'}((\sigma'_i)^{\uparrow}) = 1 = \text{PR}^{\eta}(\sigma)\ .$$

In addition, note that the path $\sigma^{\beta^{\infty}} = \text{INIT}.\beta.\text{INIT}.\beta.\cdots$, in which the action α is delayed indefinitely, has probability 0.

In general, (**A4**) ensures that all the infinite paths in η' "ending" in an end component have corresponding paths in η. It is a well-known result that the set of paths that do not "end" in an end component (such as $\sigma^{\beta^{\infty}}$) has probability 0 (see [65, Thm. 3.2]). Then, all the paths with positive probability in η' have corresponding paths in η. This allows to conclude (7.9) for all cylinders $\text{CYL}(\ell_1^+, \cdots, \ell_n^+)$. This inequality, in turn, implies

$$\text{PR}^{\eta'}(\phi) \geqslant \text{PR}^{\eta}(\phi)$$

for any $\text{LTL}_{\setminus\{\text{NEXT}\}}$ formula ϕ. □

7.4 USING OUR TECHNIQUE WITH EXISTING MODEL CHECKING ALGORITHMS

We emphasize that, although the correctness of the reduction relies on the assumption that the schedulers are distributed (strongly distributed, resp.), the reduced system is analysed assuming *total information* (because of the undecidability results in Chapter 5, the verification under partial information cannot be carried out in a fully automated fashion). The result of the verification thus corresponds to a *pessimistic* analysis of the reduced system. As a consequence, the bounds obtained are still safe, but they are not so tight as for distributed (strongly distributed, resp.) schedulers.

As an example, suppose that we are interested in finding the supremum probability that a system P fails under distributed schedulers. Suppose that 0.1 is the highest probability of failure allowed by the specification. Moreover, suppose that, by using the standard model checking algorithm for MDPs (e.g. [25]), we calculate that the supremum probability of a failure quantifying over all schedulers is 0.15. According to this analysis, the system would not meet the specification. However, the schedulers yielding probabilities greater than 0.1 might be "unrealistic" schedulers as the one in Fig. 7.2. Suppose that we construct $\widehat{P}$ as described above. Then, we can use the algorithm in [25] to calculate $S = \sup_{\eta \in \text{SCHED}_{\widehat{P}}} \text{PR}^{\eta}(\mathsf{F}\, \mathit{Fail})$. If $S = 0.05$, then Theorem 7.1 above ensures that $\sup_{\eta \in \text{DIST}_P} \text{PR}^{\eta}(\mathsf{F}\, \mathit{Fail}) \leqslant 0.05$, and so the system meets the specification. In this sense, the bounds are *safe* with respect to $\text{DIST}_P([\cdot])$. Note that, in this case, the reduction has prevented some schedulers that are not distributed and so the verification on $\widehat{P}$ is more accurate than the verification on P.

7.5 RELATED WORK

Partial Order Reduction for probabilistic systems was introduced in [12] and [57]. These works consider $\text{LTL}_{\setminus\{\text{NEXT}\}}$ formulae. A similar technique, with different conditions on the reduction, can be used for CTL formulae [19]. Yet other conditions are needed for properties involving rewards [87].

The conditions used for CTL in probabilistic systems are more restrictive than the ones for non-probabilistic systems. Hence, an interesting question is to which extent the conditions for probabilistic CTL properties and reward models can be relaxed if the schedulers are assumed to be distributed.

Our general definition of independence (Def. 7.5) enforces that independent actions affect disjoint parts of the system (in our particular definition of independence, they affect different atoms). This is in contrast to definitions of independence as in [12], which are stated in terms of the compound system and, in consequence, are not based on the affected parts of the system. This induces a clear connection to partial information since, if the state of an atom A is not affected by an action α, the actions in A cannot be scheduled using information provided by α. This connection is at the core of our improvements for distributed systems.

For non-probabilistic systems, independence is also defined in terms of the compound system and an issue similar to that of Fig. 7.5 arises for CTL formulae. This issue introduces an additional POR restriction with respect to $\text{LTL}_{\setminus\{\text{NEXT}\}}$. Hence, it would be interesting to study whether the techniques of partial order reduction for non-probabilistic systems can profit from a definition of independence like the one in this thesis.

Part III

APPLICATIONS AND CONCLUSIONS

8

ANONYMOUS FAIR SERVICE

"The only thing worse than generalizing from one example is generalizing from no examples at all"
Bob Scheifler and Jim Gettys. Principles of the X Window System

In this chapter, we use the algorithm in Sec. 6.2 to analyse two variants of a protocol for *anonymous fair service*. Given a server S and two clients A, B, the goal of the protocol is that, regardless of the rates at which A and B ask for service, S replies to their requests in a fair fashion (in the sense that, in the long run, they receive the same number of replies). By anonymous, we mean that the clients cannot be identified, and so the server cannot simply count how many times it has replied to each of the clients.

We give models for two variants of a protocol. These models are aimed to check fairness (and *not* anonymity). For the verification of these models, we assume rate schedulers. In this particular system, such assumption means that the clients send requests at a certain rate, which might change over time. Hence, a possible behaviour of the system is the one in which, at the beginning, client A sends 1 request every 3 seconds and, after 7 requests, A starts sending 1 request every 8 seconds. Client B may send $1 + \log_{11}(n+1)$ requests every 9 seconds, where n is the amount of messages already sent. So, while sending the first requests, B sends approximately 1 request every 9 seconds. The rate increases as each message is sent and, by the time of the twelfth message, B sends 2 messages every 9 seconds.

We conclude that the algorithm to calculate overestimations discussed in Sec. 6.2 yields results more realistic than the verification under total information schedulers.

8.1 THE SPECIFICATION OF THE PROTOCOL

A rough sketch of our protocol is the following: the server keeps track of the order in which requests are received. At most two requests may be pending, since we assume that clients cannot perform requests while waiting [†]. The first step in the execution of the server is to toss a coin. The outcome of the coin determines the order in which the requests are replied: in case the coin lands heads, the server replies the oldest request. Otherwise, the server replies the second request. Then, the coin is tossed again. We name this protocol AFS_1, since later on we propose a variation named AFS_2.

The code for the clients is very simple. We show the code for one of them in Fig. 8.1. The code for the other one is symmetric.

Figure 8.2 shows the PRISM code for the server. The model uses variables `first` and `second` to store the order in which the request arrived. The intuitive interpretation of these variables is that they represent a queue: `first` stores the first element in the queue (that is, the oldest request), and `second` stores the last element in the queue (that is, the newest request). The coin

[†]Otherwise, it is impossible to guarantee fairness, since one of the entities may perform requests at an arbitrarily high rate.

```
module one

status1 : [0..1] init 0; // 0 means free
                         // 1 means waiting

// 1 asks for service (so, it changes to the ''waiting'' state)
[ask1] (status1=0) -> (status1'=1);
// 1 gets served (so, it changes to the ''free'' state)
[serve1] (status1=1) -> (status1'=0);
endmodule
```

Figure 8.1: PRISM code for an AFS client

decides whether the next request to reply is the first one in the queue, or the last one. A value of x for `first` means that the first request came from client x. Note that this model is inappropriate to check anonymity, since the server knows which of the clients has sent each of the requests. The important point is that this protocol can be carried out even if the server does not know the identities of the senders (and the replies are routed to the right client using, for instance, a proxy trusted by the clients). Since our results do not apply to long-run properties, the state of our model keeps track of the numbers of replies to each client. Hence, we can focus on the probabilities p_m defined below.

DEFINITION 8.1 (p_m). For all $\eta \in \mathrm{SDIST}_P([\![\cdot]\!], \mathrm{RATE})$, $m \in \mathbb{N}$, let p^η_m be the probability that, at some point of the execution under η, the number n_1 of replies to client 1 is greater than or equal to $n_2 + m$, where n_2 is the amount of replies to client 2. In terms of our model,

$$p^\eta_m \;=\; \mathrm{PR}^\eta(\,\mathrm{REACH}(\{s \mid s(\texttt{served1}) \geqslant s(\texttt{served2}) + m\})\,)\,,$$

where $s(\texttt{var})$ denotes the value of `var` in state s.

p_m Let $p_m = \sup_{\eta \in \mathrm{SDIST}_P([\![\cdot]\!], \mathrm{RATE})} p^\eta_m$.

Note that p_m concerns a reachability property, and so we can calculate it using the algorithm in Sec. 6.2. This calculation relies on the fact the state of the system keeps track of the number of replies. Since such number is not bounded, the state space of the system should be, in principle, infinite. In order to overcome this problem, we model the system in such a way that it stops after one of the clients is served `maxServed` times.

The server comprises a variable `fail`, whose intended meaning is that, if a state in which `fail=1` is reached, then the model does not reflect the protocol we are trying to analyse. For instance, `fail` is set to 1 when a client sends a message while it is waiting for a reply. (Recall the assumption stating that clients wait until the reply arrives.) One of the consistency checks we performed using PRISM was to show that $P_{max}(\texttt{fail} = 1) = 0$ holds, that is, the maximum probability of reaching a state with $\texttt{fail} = 1$ is 0 (in other words such probability is 0 for all schedulers).

A variation on AFS_1

We explore a variation on this protocol, which we call AFS_2. In this variation, the behaviour of the server is different at a certain point of the execution,

```
module server
first : [0..2] init 0;  //The first one that asked for service
                        // (0 means none asked yet)
second : [0..2] init 0; //The second one that asked for service
coin : [0..2] init 0;   //The outcome of the coin.
                        //0 Indicates that the coin will be tossed
                        //1 Indicates that the first one will be served
                        //2 Indicates that the second one will be served
served1 : [0..maxServed] init 0; //How many times 1 has been served
served2 : [0..maxServed] init 0; // How many times 2 has been served

fail : [0..1] init 0; // If fail is 1, something undesirable happened

//Toss the coin if needed
[] (coin=0) -> 0.5 : (coin'=1) + 0.5 : (coin'=2);

//The first one to ask was 1 and the coin indicated that the
//first one must be served. So, we serve 1
[serve1] (coin=1) & (first=1)
            & (served1 < maxServed) & (served2 < maxServed)
                -> (served1'=served1+1) & (first' = second)
                                        & (second'=0) & (coin'=0);
//The first one to ask was 2
[serve2] (coin=1) & (first=2)
            & (served2 < maxServed) & (served1 < maxServed)
                -> (served2'=served2+1) & (first'=second)
                                        & (second'=0) & (coin'=0);

//The coin indicates that the second one must be served (these two cases
//      are symmetric wrt the previous ones)
[serve1] (coin=2) & (second=1)
            & (served1 < maxServed) & (served2 < maxServed)
                -> (served1'=served1+1) & (second'=0) & (coin'=0);
[serve2] (coin=2) & (second=2)
            & (served2 < maxServed) & (served1 < maxServed)
                -> (served2'=served2+1) & (second'=0) & (coin'=0);

// 1 asks for service and 2 did not ask yet
[ask1] (first=0) -> (first'=1);
// 1 asks for service but it is already in the queue
// (that should not happen)
[ask1] (first=1) -> (fail'=1);
// 1 asks for service but 2 asked before
[ask1] (first=2) -> (second'=1);
// 2 asks for service and 1 did not ask yet
[ask2] (first=0) -> (first'=2);
// That should not happen (same as above)
[ask2] (first=2) -> (fail'=1);
// 2 asks for service but 1 asked before
[ask2] (first=1) -> (second'=2);

[] (served1 = maxServed) -> (served1' = maxServed);
[] (served2 = maxServed) -> (served2' = maxServed);
endmodule
```

Figure 8.2: PRISM code for the AFS_1 server

namely, at the point following a reply to a client, say A. More precisely, it differs from AFS_1 when, at this point of the execution, client B is waiting for a reply. In AFS_1, the server tosses the coin: if the outcome indicates that the next component to serve is the one that arrived later, then the server must wait until A sends a new request. We define AFS_2 so that the server replies to B. Hence, AFS_2 seems to be more appealing since, by replying to B immediately, the server increases its throughput. However, as we will see later, our analyses show that the fairness of the service may result affected by the variation introduced. Figure 8.3 shows the code for the modified server. The code for clients is exactly the same as for AFS_1 (see Fig. 8.1).

8.2 ANALYSIS

We apply the algorithm in Sec. 6.2, which obtains a safe estimation of $p_m = \sup_{\eta \in \mathrm{SDIST}_P([\![\cdot]\!],\mathrm{RATE})} p^{\eta}_m$, as ensured by Corollary 6.1.

The algorithm in Sec. 6.2 proceeds by constructing an MDP whose states comprise not only the local state of each atom, but also an ordering on atoms.

Given the PRISM model for AFS_1, AFS_2, we constructed a PRISM model in which a module is added. The state of this module is an ordering on the modules of the original system (namely, the two clients and the server). The transitions in this *ordering* module O take care of updating the ordering after each global transition. The restrictions on reordering functions (Def. 6.5) hold because of the way in which the transitions in O are defined. The modules of the clients and the server are modified so that a module M can execute a transition only if M is the first module in the ordering. Because of the restrains imposed by the PRISM language, the code of O is large and redundant, and thus we do not show it here. A software bundle containing all the PRISM code can be found at: `cs.famaf.unc.edu.ar/~sgiro/thesis/afs.tar.gz`.

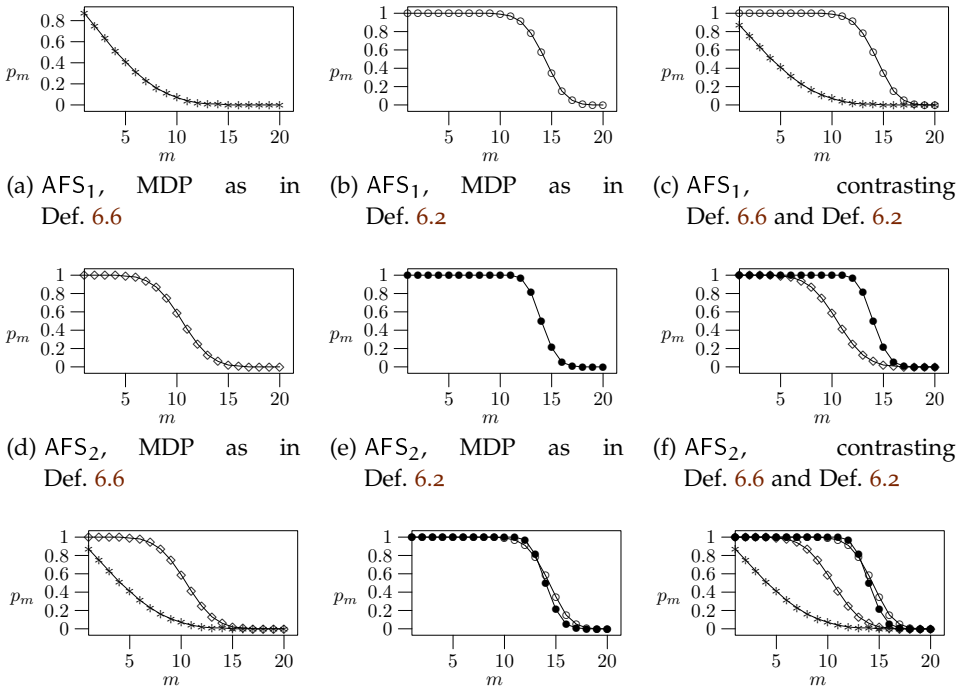
Figure 8.4 shows the results of our analysis. The plots in this figure give information about two different aspects. On the one hand, they compare the results according to the algorithm in Sec. 6.2 (whichs construct an MDP encoding total order schedulers) against the results using the straightforward MDP construction in Sec. 6.1. On the other hand they give a clue about the fairness of AFS_1 with respect to AFS_2. The first two rows in Fig. 8.4 are useful to compare the results using the machinery in Sec. 6.2 against the one in Sec. 6.1. The first two columns compare AFS_1 against AFS_2.

From Fig. 8.4g, it is clear that, according to the algorithm in Sec. 6.2, the protocol AFS_1 ensures a fair service with greater probability than that of AFS_2: for instance, in AFS_1 it is quite unlikely that client 1 is served 10 times more than client 2 ($p_{10} = 0.07$), while in AFS_2 this happens quite often ($p_{10} = 0.59$) in the worst case. Note that the MDP in Def. 6.2 does not provide a clear verdict with respect to which of the protocols is more likely to result in fair behaviours: in Fig. 8.4h, we see that the probabilities p_m for both protocols are very similar. Moreover, for some values of m (namely $11 \leqslant m \leqslant 13$) the probabilities p_m are greater for AFS_2 while, for some other values (namely $14 \leqslant m \leqslant 18$), they are greater for AFS_1.

An encouraging conclusion is that the results using the straightforward MDP construction in Def. 6.2 are excessively pessimistic: this indicates that

```
module server
first : [0..2] init 0;  //The first one that asked for service
                        // (0 means none asked yet)
second : [0..2] init 0; //The second one that asked for service
coin : [0..2] init 0;   //The outcome of the coin.
                        //0 Indicates that it needs to be tossed
                        //1 Indicates that the first one will be served
                        //2 Indicates that the second one will be served
served1 : [0..maxServed] init 0; //How many times 1 has been served
served2 : [0..maxServed] init 0; // How many times 2 has been served
fail : [0..1] init 0; // If fail is 1, something undesirable happened
server_state : [0..3] init 0;
               //0 Tossing the coin (iff coin = 0), or moving to an
               //   intermediate state before replying the request
               //1 Serving 1
               //2 Serving 2
               //3 Serving the one that was waiting

//Toss the coin if needed
[] (coin=0) & (server_state=0) -> 0.5 : (coin'=1) + 0.5 : (coin'=2);

//The first one to ask was 1 and the coin indicated that the
//first one must be served. So, we start to serve 1
[] (coin=1) & (first=1) & (served1 < maxServed)
    -> (server_state'=1) & (first' = second) & (second'=0) & (coin'=0);
// Serve 1. If a client is waiting, we serve it next (set server_state
    =3)
[serve1] (server_state=1) & (first!=0) & (served1 < maxServed)
    -> (served1'=served1+1) & (server_state'=3);
// Serve 1. If nobody waits, toss the coin (set server_state=0)
[serve1] (server_state=1) & (first=0) & (served1 < maxServed)
    -> (served1'=served1+1) & (server_state'=0);
//The first one to ask was 2, and the coin was tossed
//This case comprises three commands symmetrical to the previous ones,
//which are thus omitted
...
//The coin indicates that the second one must be served
[] (coin=2) & (second=1) & (served1 < maxServed)
     -> (server_state'=1) & (second'=0) & (coin'=0);
[] (coin=2) & (second=2) & (served2 < maxServed)
     -> (server_state'=2) & (second'=0) & (coin'=0);

// Someone is waiting, serve him
[] (server_state=3) & (first=1) & (served1 < maxServed)
     -> (server_state'=1) & (first' = second) & (second'=0) & (coin'=0);
[] (server_state=3) & (first=2) & (served2 < maxServed)
     -> (server_state'=2) & (first' = second) & (second'=0) & (coin'=0);

// 1 asks for service and 2 did not ask yet
[ask1] (first=0) -> (first'=1);
// That should not happen
[ask1] (first=1) -> (fail'=1);
// 1 asks for service but 2 asked before
[ask1] (first=2) -> (second'=1);

// 2 asks for service and 1 did not ask yet
//This case is symmetrical wrt to the previous one
...
endmodule
```

Figure 8.3: PRISM code for the AFS_2 server



(a) AFS_1, MDP as in Def. 6.6

(b) AFS_1, MDP as in Def. 6.2

(c) AFS_1, contrasting Def. 6.6 and Def. 6.2

(d) AFS_2, MDP as in Def. 6.6

(e) AFS_2, MDP as in Def. 6.2

(f) AFS_2, contrasting Def. 6.6 and Def. 6.2

(g) contrasting AFS_1 and AFS_2. MDP as in Def. 6.6

(h) contrasting AFS_1 and AFS_2. MDP as in Def. 6.2

(i) contrasting all the results

Figure 8.4: Analysis of AFS_1 and AFS_2.

our algorithm in Sec. 6.2 makes a significant difference, as we can see in Figures 8.4c and 8.4f. Note that, for the particular value $m = 10$, the worst-case for AFS_1 using our algorithm yields a probability of 0.07, while using Def. 6.2 model the probability is 0.99 (see Fig. 8.4c). With respect to AFS_2, the probabilities for $m = 10$ are 0.59 and 1 (see Fig. 8.4f).

8.3 FURTHER WORK

We applied the algorithm in Sec. 6.2 by manually adding a reordering module, and thus our analysis is not completely automated. We plan to construct a program to automatically generate the reordering module, so that we can perform analysis in a completely automated fashion.

As we discussed in Sec. 6.4, one of the drawbacks of the algorithm for total order-based schedulers in Sec. 6.2 is the inability to deal with projections other than $[\cdot]^{\mathrm{VP}}$. We hope that, by finding algorithms for projections finer than $[\cdot]^{\mathrm{VP}}$, we can obtain tighter estimations of extremal probabilities.

One obvious pending task is to use our algorithm to check models other than AFS. Since our algorithm is devised for systems in which the interleaving nondeterminism is resolved according to local information, we plan to look at case studies that arose in the setting of rate-based PIOA, in which the rates depends on the local state [137].

9 PARTIAL ORDER REDUCTION IN PRACTICE

In this chapter, we show how the results in Chapter 7 ease the verification of two particular models. Our version of Partial Order Reduction, implemented on the PRISM model checker [97], was able to verify these models and outperform existing implementations of the technique. The details of the implementation can be found in [73].

The language used by PRISM is not based on input/output automata, and so the first step is to find a suitable encoding of the PRISM into IPIOA. Section 9.1 presents this encoding, together with a definition of independence that exploits the structure of the IPIOA yielded by the encoding. Section 9.2 presents the results of the analysis of a model for the *dining cryptographers* [44], a well-known protocol to ensure anonymity. Section 9.3 studies the binary exponential back-off protocol (as described in [102]) used in the IEEE 802.3 standards.

9.1 PARTIAL ORDER REDUCTION FOR PRISM MODULES

In the semantics used by PRISM [97], each model is interpreted as an MDP $M = (S_M, \text{ACTIONS}_M, P_M, \text{INIT}_M)$. A document on PRISM semantics can be found in [1]. In the following, we show how to interpret M as an IPIOA P, in order to present an analogous of Theorem 7.2.

Throughout this section, we use the model in Figure 9.1 as a running example. In particular, we use it to establish nomenclature. Module `m1` has 3 *commands*, while module `m2` has 4 of them. The command C_2 in line 2 has $\text{LABEL}(C_2) = a$ The command C_{10} in line 10 is unlabelled. We let $\text{LABEL}(C_{10}) = \tau_{C_{10}}$. Whenever C has a subscript i, we write τ_i instead of τ_{C_i}. — LABEL(C)

Each state $s \in S_M$ is a valuation on the variables of the system. Valuations are heavily used throughout this chapter, and hence we define notations for them. Given a set of variables $\mathcal{V}$, we write $\mathbb{V}(\mathcal{V})$ for the set of valuations over $\mathcal{V}$. Given $s \in S_M$, we write $s(v)$ for the value of variable v at s. Given $\mathcal{V}' \subseteq \mathcal{V}$, the restriction of valuation V to $\mathcal{V}'$ (denoted by $(V)\big|_{\mathcal{V}'}$) is the valuation $V' \in \mathbb{V}(\mathcal{V}')$ such that $V'(v) = V(v)$ for all $v \in \mathcal{V}'$. The set of variables in the system is denoted by $\mathcal{V}_M$, and so the set of states of the system is $\mathbb{V}(\mathcal{V}_M)$. — $\mathbb{V}(\mathcal{V})$, $V(v)$, $\mathcal{V}_M$

The MDP constructed from the model in Fig. 9.1 has 8 actions:

1. $\alpha_{2,8}$ corresponds to the synchronization of commands in lines 2, 8. We have $\text{LABEL}(\alpha_{2,8}) = a$. — LABEL(α)
2. $\alpha_{2,9}$ corresponds to the synchronization of commands in lines 2, 9
3. $\alpha_{3,8}$ corresponds to the synchronization of commands in lines 3, 8
4. $\alpha_{3,9}$ corresponds to the synchronization of commands in lines 3, 9
5. α_4 corresponds to the command in line 4
6. α_{10} corresponds to the command in line 10. We have $\text{LABEL}(\alpha_{10}) = \tau_{10}$
7. α_{11} corresponds to the command in line 11

The set of labels in the system is $\mathcal{L}_M = \{\text{LABEL}(\alpha) \mid \alpha \in \text{ACTIONS}_M\}$. — $\mathcal{L}_M$

Our partial order technique profits from the limited information available to the adversaries: the less information is shared, the more states are eliminated. The encoding we present in this section allows to specify, for each

```
1  module m1
2  [a] (p=1|p=2) -> 0.5:(v'=1)&(x'=1) + 0.5:(v'=2)&(x'=1);
3  [a] (q=1) -> 0.5:(x'=1) + 0.5:(x'=x+1)
4  [c] (q=2) -> (x'=1);
5  endmodule
6
7  module m2
8  [a] (r=1) -> (y'=1);
9  [a] (p=1) -> 0.2:(z'=1) + 0.8:(z'=2);
10 [] (q=1) -> (p'=1);
11 [b] (p=1) -> (y'=2)&(z'=1);
12 endmodule
```

Figure 9.1: PRISM code for an AFS client

READVAR(α) READLAB(α)

action α in the MDP M, a set of variables (denoted by READVAR(α)) and a set of labels (denoted by READLAB(α)) that α "reads", in the sense that the probability that α is scheduled depends only on such variables and labels. For instance, we can specify

$$\text{READVAR}(\alpha_{2,8}) = \{p, r\}$$
$$\text{READLAB}(\alpha_{2,8}) = \{a\}$$

Note that a scheduler needs to see variables p and r in order to know whether $\alpha_{2,8}$ is enabled or not. In fact, we require:

$$\forall s, s' : \big(\forall v \in \text{READVAR}(\alpha) : s(v) = s'(v)\big) \implies (\ \alpha \in \text{ACTIONS}(s) \iff \alpha \in \text{ACTIONS}(s')\ ) \quad (9.1)$$

That is, the read variables in READVAR(α) must be sufficient to determine whether α is enabled or not.

NOTATION 9.1. Let READVAR(α) $\subseteq \mathcal{V}' \subseteq \mathcal{V}_M$ and $t \in \mathbb{V}(\mathcal{V}')$. We write $\alpha \in \text{ACTIONS}(t)$ to denote

$$\exists s \in \mathbb{V}(\mathcal{V}_M) : \alpha \in \text{ACTIONS}(s) \wedge \forall v \in \mathcal{V}' : s(v) = t(v)\ .$$

In addition to (9.1), we require

$$\text{LABEL}(\alpha) \in \text{READLAB}(\alpha) \quad (9.2)$$

Note that READLAB($\alpha_{2,8}$) complies with this condition, as $a \in$ READLAB($\alpha_{2,8}$). Although PRISM models do not provide the means to specify that $\alpha_{2,8}$ depends on labels other than a (and so in our implementation we simply define READLAB(α) = {LABEL(α)}), we leave the door open for the extension to an arbitrary set of labels.

We write READ(α) for READVAR(α) $\cup$ READLAB(α).

WRITEVAR(α)

By inspection of the model, we can obtain the set of variables that an action α *writes*. For instance, $\alpha_{2,9}$ writes v, x and y. The set WRITEVAR(α) contains all such variables. We define

$$\text{WRITE}(\alpha) = \text{WRITEVAR}(\alpha) \cup \{\text{LABEL}(\alpha)\}$$
$$\text{VAR}(\alpha) = \text{READVAR}(\alpha) \cup \text{WRITEVAR}(\alpha)$$

Note that, for all α, $s \in \mathbb{V}(\mathcal{V}_M)$ such that $\alpha \in \text{ACTIONS}(s)$, the function $\alpha(s,\cdot)$ can be seen as a probability distribution on $\mathbb{V}(\text{WRITEVAR}(\alpha))$. Considering the actions in our running example, for all s we have:

$$\begin{aligned}
\alpha_{2,8}(s,\cdot) &= 0.5{:}\{v \to 1, x \to 1, y \to 1\} + 0.5{:}\{v \to 2, x \to 1, y \to 1\} \\
\alpha_{2,9}(s,\cdot) &= \quad (0.5 \cdot 0.2){:}\{v \to 1, x \to 1, z \to 1\} \\
&\quad + (0.5 \cdot 0.8){:}\{v \to 1, x \to 1, z \to 2\} \\
&\quad + (0.5 \cdot 0.2){:}\{v \to 2, x \to 1, z \to 1\} \\
&\quad + (0.5 \cdot 0.8){:}\{v \to 2, x \to 1, z \to 2\}
\end{aligned}$$

For all s such that $s(x) = 27$, we have:

$$\begin{aligned}
\alpha_{3,9}(s,\cdot) &= \quad (0.5 \cdot 0.2){:}\{x \to 1, z \to 1\} \\
&\quad + (0.5 \cdot 0.8){:}\{x \to 1, z \to 2\} \\
&\quad + (0.5 \cdot 0.2){:}\{x \to 28, z \to 1\} \\
&\quad + (0.5 \cdot 0.8){:}\{x \to 28, z \to 2\}
\end{aligned}$$

In this action, the distribution on $\text{WRITEVAR}(\alpha_{3,9})$ depends on the state s. For technical reasons, we need that read variables are sufficient to determine $\alpha(s,\cdot)$, that is:

$$\forall \alpha, s, s' : \big(\forall v \in \text{READVAR}(\alpha) : s(v) = s'(v)\big) \implies \alpha(s,\cdot) = \alpha(s',\cdot) \quad (9.3)$$

This restriction implies:

$$\forall \alpha, s_\alpha \in \mathbb{V}(\text{READVAR}(\alpha)) : \alpha(s_\alpha,\cdot) = \sum_{q=1}^{Q} p_q^{s_\alpha} : V_q^{s_\alpha} \quad (9.4)$$

Note that Eqn. (9.3) forces us include the variable x in $\text{READVAR}(\alpha_{3,9})$.

The probability that α is scheduled depends only on the sequence of elements in $\text{READ}(\alpha)$ that appear along the execution history. This sequence is inductively defined below. The function $(\cdot)\big|_\alpha$ plays the same role as projections for IPIOA, but it applies to our setting of MDPs arising from PRISM models.

$$\begin{aligned}
(\text{INIT}_M)\big|_\alpha &= (\text{INIT})\big|_{\text{READVAR}(\alpha)} \\
(\sigma.\beta.s)\big|_\alpha &= (\sigma)\big|_\alpha . \text{LABEL}(\alpha) . (s)\big|_{\text{READVAR}(\alpha)} \\
&\qquad \text{if } \text{LABEL}(\beta) \in \text{READLAB}(\alpha) \\
(\sigma.\beta.s)\big|_\alpha &= (\sigma)\big|_\alpha . \kappa . (s)\big|_{\text{READVAR}(\alpha)} \\
&\qquad \text{if } \text{LABEL}(\beta) \notin \text{READLAB}(\alpha) \\
&\qquad \text{and } \text{WRITEVAR}(\beta) \cap \text{READVAR}(\alpha) \neq \emptyset \\
(\sigma.\beta.s)\big|_\alpha &= (\sigma)\big|_\alpha \\
&\qquad \text{if } \text{LABEL}(\beta) \notin \text{READLAB}(\alpha) \\
&\qquad \text{and } \text{WRITEVAR}(\beta) \cap \text{READVAR}(\alpha) = \emptyset
\end{aligned}$$

Note that the information observable in $\sigma.\beta.s$ differs from the information observable in σ only in case the label of β is observed by α, or in case β writes a variable read by α. In the former case, the information observable in $\sigma.\beta.s$ comprises $\text{LABEL}(\beta)$ and the new values for the variables in $\text{READLAB}(\alpha)$.

In the latter case, the information observable comprises only the new values for the variables. The label κ is a mere padding indicating that the label in β is not observable. Of course, we could go without this padding, but we would lose that elegance conferred by the hypnotic alternation of valuations and labels.

κ

We defined $(\cdot)\big|$ with the aim of defining a restriction on the information used to schedule the actions. Now, we can express this restriction formally.

DEFINITION 9.1. The set of schedulers with partial information (denoted by PARINFO(M)) comprises all the schedulers η such that

$$\frac{\eta(\sigma)(\alpha)}{\eta(\sigma)(\alpha)+\eta(\sigma)(\beta)} = \frac{\eta(\sigma')(\alpha)}{\eta(\sigma')(\alpha)+\eta(\sigma')(\beta)} \,. \tag{9.5}$$

for all paths σ, σ' such that

$$(\sigma)\big|_\alpha = (\sigma')\big|_\alpha \;\wedge\; (\sigma)\big|_\beta = (\sigma')\big|_\beta$$

and $\Pr^\eta((\sigma)^\uparrow) > 0$, $\Pr^\eta((\sigma')^\uparrow) > 0$, $\eta(\sigma)(\alpha)+\eta(\sigma)(\beta) > 0$, $\eta(\sigma')(\alpha)+\eta(\sigma')(\beta) > 0$.

The motivation for this restriction is the same as the one for strongly distributed schedulers in Sec. 2.1: the probability that η schedules α (β, resp.) depends only on the sequence σ_α (σ_β, resp.) of events visible to α (β, resp.). Hence, the probability that η schedules α instead of β coincides for all paths σ, σ' having $(\sigma)\big|_\alpha = (\sigma')\big|_\alpha = \sigma_\alpha$ and $(\sigma)\big|_\beta = (\sigma')\big|_\beta = \sigma_\beta$.

We show how, using the sets READ(α), WRITE(α), the MDP M arising from a PRISM model can be interpreted as an IPIOA. This IPIOA has one atom for each action in the MDP. This interpretation of *actions* as atoms might seem strange at first sight. In fact, one can think of a more intuitive interpretation in which *modules* are mapped to atoms. The latter interpretation has, however, an important drawback: in the IPIOA formalism the availability of information is specified at the atom level and so, in this interpretation, we cannot specify that different transitions in the same module are scheduled according to different information.

Next, we give the definition of the atom A_α corresponding to action α in the MDP. After the formal definition, we give intuitive explanations.

DEFINITION 9.2. Given an MDP M such that

$$S_M = \mathbb{V}(\mathcal{V}_M) \,,$$

for all action α defined by $\alpha(s,\cdot) = \sum_{q=1}^{Q} p_q^s : V_q^s$ (here, $s \in \mathbb{V}(\text{READVAR}(\alpha))$) we define the atom A_α as follows:

- $S_\alpha = \mathbb{V}(\text{VAR}(\alpha))$
- $\text{ACTLAB}_\alpha =$
$$\{a_V \mid a \in \text{READLAB}(\alpha) \wedge V \in \mathbb{V}(\mathcal{V}') \wedge \mathcal{V}' \subseteq \mathcal{V}_M\}$$
$$\cup\, \{a_V \mid a \in \mathcal{L}_M \wedge V \in \mathbb{V}(\mathcal{V}') \wedge \mathcal{V}' \subseteq \mathcal{V}_M \wedge \mathcal{V}' \cap \text{VAR}(\alpha) \neq \emptyset\}$$
- for all $s \in S_\alpha$, if $\alpha \in \text{ACTIONS}(s)$ let $G_\alpha(s) = \{g_\alpha\}$, where

$$g_\alpha(s, a_V, s') = \begin{cases} p_q^s & \text{if } a = \text{LABEL}(\alpha) \text{ and there exists } q \text{ such that:} \\ & \forall v \in \text{WRITEVAR}(\alpha) : s'(v) = V_q^s(v) \text{ and} \\ & \forall v \in \text{VAR}(\alpha) \setminus \text{WRITEVAR}(\alpha) : s'(v) = s(v) \\ 0 & \text{otherwise} \end{cases}$$

if $\alpha \notin \text{ACTIONS}(s)$ let $G_\alpha(s) = \emptyset$.

- for all $s \in S_\alpha$, $a \in \mathcal{L}_M$, $V \in \mathbb{V}(\mathcal{V}')$, where $\mathcal{V}' \subseteq \mathcal{V}_M$ and $\mathcal{V}' \cap \text{VAR}(\alpha) \neq \emptyset$, let

$$R_\alpha(s, a_V) = \{r_{s,a_V}\},$$

where

$$r_{s,a_V}(s, a_V, s') = \begin{cases} 1 & \text{if } \forall v \in \text{VAR}(\alpha) \cap \mathcal{V}' : s'(v) = V(v) \text{ and} \\ & \forall v \in \text{VAR}(\alpha) \setminus \mathcal{V}' : s'(v) = s(v) \\ 0 & \text{otherwise} \end{cases}$$

- $\text{INIT}_\alpha = (\text{INIT}(v))\big|_{\text{VAR}(\alpha)}$

Note that, since $S_\alpha = \mathbb{V}(\text{VAR}(\alpha))$, the state of the IPIOA $P = \|_\alpha A_\alpha$ is $\prod_\alpha \mathbb{V}(\text{VAR}(\alpha))$. This implies that, for each variable v in the MDP, each state might have several copies of v, say $v_{\alpha_1}, \cdots, v_{\alpha_N}$: each of these copies corresponds to an action α_n such that $v \in \text{VAR}(\alpha_n)$. The IPIOA works in such a way that

$$s(v_{\alpha_j}) = s(v_{\alpha_k}) \tag{9.6}$$

for all j, k, for all reachable state s in the model. This property holds since we define g_α in such a way that, when A_α executes and changes its write variables to V_q, it does not output $\text{LABEL}(\alpha)$, but $\text{LABEL}(\alpha)_{V_q}$. By definition of ACTLAB, the label $\text{LABEL}(\alpha)_{V_q}$ is in ACTLAB_β for all atoms A_β such that $\text{VAR}(\beta) \cap \text{WRITEVAR}(\alpha) \neq \emptyset$. Hence, such atoms A_β react to this label using $r_{s,\text{LABEL}(\alpha)_{V_q}}$, thus reaching a state s' such that $\forall v \in \text{VAR}(\alpha) \cap \mathcal{V}' : s'(v) = V(v)$.

In the following, we establish several properties relating the MDP M arising from a PRISM model to the IPIOA $P = \|_\alpha A_\alpha$.

By Eqn. (9.6) there exists a bijection $h^S(\cdot)$ from the reachable states of P to S_M such that:

$$h^S(s)(v) = \pi_{\alpha_j}(s)(v_{\alpha_j})$$

where α_j is any action such that $v \in \text{VAR}(\alpha_j)$, and π_{α_j} is the projection that extracts the portion corresponding to α_j from an element in $\prod_\alpha \text{VAR}(\alpha)$. Equation (9.6) ensures that the particular α_j chosen is irrelevant.

In addition to the bijection, we consider the function h^A that maps each compound transition in P to an action in M:

$$h^A(g_\alpha, a_V, r_{s_1,a_V}, \cdots, r_{s_m,a_V}) = \alpha .$$

Note that this function is surjective but not injective, since the information about the particular V is disregarded. However, the product function $h^S \times h^A$ is injective in the following sense:

$$\forall (\alpha, s') \in \text{ACTIONS}_M \times S_M : \big(\exists s : \alpha(s, s') > 0\big) \implies \exists!(c, t) : (h^A(c), h^S(t)) = (\alpha, s') . \tag{9.7}$$

Intuitively, since the state s' comprises the information about the valuation V (in fact, $(s')\big|_{\text{WRITEVAR}(\alpha)} = V$) the ambiguity concerning the valuation in c is resolved by s'.

Property (9.7) implies that the functions:

$$h^*(s^0.c^1.\cdots.c^n.s^n) = h^S(s^0).h^A(c^1).\cdots.h^A(c^n).h^S(s^n) \tag{9.8}$$

$$h^\omega(s^0.c^1.\cdots.c^n.s^n.\cdots) = h^S(s^0).h^A(c^1).\cdots.h^A(c^n).h^S(s^n).\cdots \tag{9.9}$$

are bijections from the finite paths (infinite paths, resp.) of P to the finite paths (infinite paths, resp.) of M. Moreover, if $c = (g_\alpha, r_{s_1,a_V}, \cdots, r_{s_m,a_V})$ and $\alpha(s,\cdot) = \sum_{q=1}^{Q} p^s_q : V^s_q$, then

$$c(s,s') = g_\alpha(s,a_V,s') = p^s_q = \alpha(s,s')\ , \tag{9.10}$$

where q is the index such that $(s')\big|_{\text{WRITEVAR}(\alpha)} = V^s_q$.

Each scheduler for P defines a scheduler for M, and vice versa. In fact, we can define a bijection i from the schedulers of P to those of M. If $\eta = (\mathcal{I}, \{\Theta_i\}_i, \{\Upsilon_i\}_i)$, then $i(\eta)$ is defined as:

$$i(\eta)\,(\sigma)\,(\alpha) = \mathcal{I}\,(h^{*^{-1}}(\sigma))\,(A_\alpha)\ . \tag{9.11}$$

That is, the probability that α is scheduled after σ in $i(\eta)$ is the probability that A_α is scheduled after $h^{*^{-1}}(\sigma)$ in η. The fact that i is a bijection follows from the fact that $h^{*^{-1}}$ is.

By (9.10), we have

$$\forall \sigma\ :\ \mathrm{PR}^\eta_P((\sigma)^\uparrow) = \mathrm{PR}^{i(\eta)}_M((\!|h^*(\sigma)|\!)^\uparrow)\ . \tag{9.12}$$

From (9.12) (together with Carathéodory extension theorem) and the fact that i is a bijection, we obtain:

$$\sup_{\eta\in\text{SCHED}_M} \mathrm{PR}^\eta_M(\mathcal{S}) = \sup_{\eta\in\text{SCHED}_P} \mathrm{PR}^{\eta_P}(h^{\omega^{-1}}(\mathcal{S}))\ . \tag{9.13}$$

In this equation, we have naturally extended $h^{\omega^{-1}}$ from infinite paths to sets of infinite paths:

$$h^{\omega^{-1}}(\mathcal{S}) = \{h^{\omega^{-1}}(\rho)\ \mid\ \rho\in\mathcal{S}\}\ .$$

The bijection h^* also allows us to define a projection for P:

$$(\!|\sigma|\!)_\alpha = (h^*(\sigma))\big|_\alpha\ .$$

From this definition, we deduce:

$$(\!|\sigma|\!)_\alpha = (\!|\sigma'|\!)_\alpha \iff (h^*(\sigma))\big|_\alpha = (h^*(\sigma'))\big|_\alpha \tag{9.14}$$

$$(\sigma)\big|_\alpha = (\sigma')\big|_\alpha \iff (\!|h^{*^{-1}}(\sigma)|\!)_\alpha = (\!|h^{*^{-1}}(\sigma')|\!)_\alpha\ . \tag{9.15}$$

Now, we can link the schedulers in $\text{PARINFO}(M)$ to those in $\text{SDIST}_P((\!|\cdot|\!))$.

LEMMA 9.1. *Given an MDP M, let $P = \|_{\alpha\in\text{ACTIONS}_M} A_\alpha$ (where A_α is the atom in Def. 9.2). Then,*

$$i(\eta)\in\text{PARINFO}(M) \iff \eta\in\text{SDIST}_P((\!|\cdot|\!))\ .$$

Proof. Let $\eta = (\mathcal{I}, \{\Theta_i\}_i, \{\Upsilon_i\}_i)$ be a scheduler for P. By (9.11), we have $i(\eta)\in\text{PARINFO}(M)$ iff

$$\frac{\mathcal{I}\,(h^{*^{-1}}(\sigma))\,(\alpha)}{\mathcal{I}\,(h^{*^{-1}}(\sigma))\,(\alpha) + \mathcal{I}\,(h^{*^{-1}}(\sigma))\,(\beta)} = \frac{\mathcal{I}\,(h^{*^{-1}}(\sigma'))\,(\alpha)}{\mathcal{I}\,(h^{*^{-1}}(\sigma'))\,(\alpha) + \mathcal{I}\,(h^{*^{-1}}(\sigma'))\,(\beta)}$$

for all paths σ, σ' such that

$$\begin{aligned}
&(\sigma)\big|_\alpha = (\sigma')\big|_\alpha \;\wedge\; (\sigma)\big|_\beta = (\sigma')\big|_\beta \\
\wedge\;& \mathrm{PR}^{i(\eta)}((\sigma)^\uparrow) > 0 \;\wedge\; \mathrm{PR}^{i(\eta)}((\sigma')^\uparrow) > 0 \\
\wedge\;& i(\eta)(\sigma)(\alpha) + i(\eta)(\sigma)(\beta) > 0 \;\wedge\; i(\eta)(\sigma')(\alpha) + i(\eta)(\sigma')(\beta) > 0 \,.
\end{aligned}$$

From (9.15), we get

$$\begin{aligned}
&(\sigma)\big|_\alpha = (\sigma')\big|_\alpha \wedge (\sigma)\big|_\beta = (\sigma')\big|_\beta \\
&\qquad \iff (\!|h^{*^{-1}}(\sigma)|\!)_\alpha = (\!|h^{*^{-1}}(\sigma')|\!)_\alpha \wedge (\!|h^{*^{-1}}(\sigma)|\!)_\beta = (\!|h^{*^{-1}}(\sigma')|\!)_\beta \,.
\end{aligned}$$

By (9.12) we have

$$\mathrm{PR}^{i(\eta)}((\sigma)^\uparrow) > 0 \iff \mathrm{PR}^{\eta}((h^{*^{-1}}(\sigma))^\uparrow) > 0 \,.$$

In addition, (9.11) yields

$$i(\eta)(\sigma)(\alpha) + i(\eta)(\sigma)(\beta) > 0 \iff \mathcal{I}(h^{*^{-1}}(\sigma))(\alpha) + i(\eta)(h^{*^{-1}}(\sigma))(\beta) > 0 \,.$$

Therefore, $i(\eta) \in \textsc{ParInfo}(M)$ iff

$$\forall \sigma, \sigma' \in \textsc{Paths}(M), \alpha, \beta \in \textsc{Actions}_M \,:\, F(\sigma, \sigma', \alpha, \beta) \,, \tag{9.16}$$

where F is the following predicate:

$$\begin{aligned}
F(\sigma,\sigma',\alpha,\beta) = \big(\;& (\!|h^{*^{-1}}(\sigma)|\!)_\alpha = (\!|h^{*^{-1}}(\sigma')|\!)_\alpha \\
& \wedge (\!|h^{*^{-1}}(\sigma)|\!)_\beta = (\!|h^{*^{-1}}(\sigma')|\!)_\beta \\
& \wedge \mathrm{PR}^{\eta}((h^{*^{-1}}(\sigma))^\uparrow) > 0 \;\wedge\; \mathrm{PR}^{\eta}((h^{*^{-1}}(\sigma'))^\uparrow) > 0 \\
& \wedge \mathcal{I}(h^{*^{-1}}(\sigma))(A_\alpha) + \mathcal{I}(h^{*^{-1}}(\sigma))(A_\beta) > 0 \\
& \wedge \mathcal{I}(h^{*^{-1}}(\sigma'))(A_\alpha) + \mathcal{I}(h^{*^{-1}}(\sigma'))(A_\beta) > 0 \;\big) \\
& \implies \frac{\mathcal{I}(h^{*^{-1}}(\sigma))(A_\alpha)}{\mathcal{I}(h^{*^{-1}}(\sigma))(A_\alpha) + \mathcal{I}(h^{*^{-1}}(\sigma))(A_\beta)} \\
& \qquad = \frac{\mathcal{I}(h^{*^{-1}}(\sigma'))(A_\alpha)}{\mathcal{I}(h^{*^{-1}}(\sigma'))(A_\alpha) + \mathcal{I}(h^{*^{-1}}(\sigma'))(A_\beta)}
\end{aligned}$$

We show that (9.16) is equivalent to:

$$\forall \sigma^1, \sigma^2 \in \textsc{Paths}(P), \alpha, \beta \in \textsc{Actions}_M \,:\, F(h^*(\sigma^1), h^*(\sigma^2), \alpha, \beta) \,. \tag{9.17}$$

The implication (9.16) $\implies$ (9.17) holds since, assuming that F holds for all $\sigma, \sigma' \in \textsc{Paths}(M)$, it holds in particular for the paths $h^*(\sigma^1)$, $h^*(\sigma^2)$, where $\sigma^1, \sigma^2 \in \textsc{Paths}(P)$. In order to prove the implication (9.17) $\implies$ (9.16) we recall that h^* is a bijection and hence it is surjective. Then, for all σ, σ', we can find σ^1, σ^2 such that $h^*(\sigma^1) = \sigma$ and $h^*(\sigma^2) = \sigma'$. If we assume (9.17), we have that the predicate $F(h^*(\sigma^1), h^*(\sigma^2), \alpha, \beta) = F(\sigma, \sigma', \alpha, \beta)$ holds.

Therefore, $i(\eta) \in \text{PARINFO}(M)$ iff (9.17) holds. By specializing F for $h^*(\sigma)$, $h^*(\sigma')$, and using the fact that ${h^*}^{-1}(h^*(\sigma)) = \sigma$, we get $i(\eta) \in \text{PARINFO}(M)$ iff for all $\sigma, \sigma' \in \text{PATHS}(P)$, α, β, we have

$$\begin{aligned}
&\big(\quad (\!|\sigma|\!)_\alpha = (\!|\sigma'|\!)_\alpha \ \wedge\ (\!|\sigma|\!)_\beta = (\!|\sigma'|\!)_\beta \\
&\ \wedge\ \text{PR}^\eta((\sigma)^\uparrow) > 0 \ \wedge\ \text{PR}^\eta((\sigma')^\uparrow) > 0 \\
&\ \wedge\ \mathcal{I}(\sigma)(A_\alpha) + \mathcal{I}(\sigma)(A_\beta) > 0 \\
&\ \wedge\ \mathcal{I}(\sigma')(A_\alpha) + \mathcal{I}(\sigma')(A_\beta) > 0 \quad \big) \\
&\qquad \Longrightarrow \quad \frac{\mathcal{I}(\sigma)(A_\alpha)}{\mathcal{I}(\sigma)(A_\alpha) + \mathcal{I}(\sigma)(A_\beta)} \\
&\qquad\qquad = \frac{\mathcal{I}(\sigma')(A_\alpha)}{\mathcal{I}(\sigma')(A_\alpha) + \mathcal{I}(\sigma')(A_\beta)}
\end{aligned}$$

This predicate is equivalent to the definition of strongly distributed scheduler (Def. 2.1). □

We have used (9.12) and Carathéodory extension theorem to obtain Equation (9.13). Applying the same reasoning and Lemma 9.1, we obtain:

$$\sup_{\eta \in \text{PARINFO}(M)} \text{PR}^\eta_M(\mathcal{S}) = \sup_{\eta \in \text{SDIST}_P((\!|\cdot|\!))} \text{PR}^\eta_P(h^{\omega^{-1}}(\mathcal{S})) \,.$$

Hence, from now on we can focus on the problem of calculating

$$\sup_{\eta \in \text{SDIST}_P((\!|\cdot|\!))} \text{PR}^\eta_P(\mathcal{S}) \,.$$

Since the projection in this expression is $(\!|\cdot|\!)$ (and not $[\![\cdot]\!]$) we cannot apply Theorem 7.2, but a variation on it. Such a variation uses a different independence relation defined as:

$$\begin{aligned}
\alpha \,\mathbb{I}\, \beta \iff\ & \text{WRITE}(\alpha) \cap (\text{READ}(\beta) \cup \text{WRITE}(\beta)) = \emptyset \\
& \wedge\ \text{WRITE}(\beta) \cap (\text{READ}(\alpha) \cup \text{WRITE}(\alpha)) = \emptyset \,.
\end{aligned}$$

Intuitively, if $\alpha \,\mathbb{I}\, \beta$, then the events generated by α do not overlap with the events related to β, and vice versa (by event we mean a label outputting or a change to a variable).

The proof of the following theorem can be found in Appendix D.

The definition of MDP(P) *is given in Def. 6.2, p. 108*

THEOREM 9.1. *Given an MDP* M, *let* $P = \|_{\alpha \in \text{ACTIONS}_M} A_\alpha$. *Moreover, let* $\widehat{P}$ *be a reduction of* MDP(P) *complying with conditions* **A1**–**A4**, *by taking the independence relation to be* $\mathbb{I}$. *Then,*

$$\sup_{\eta \in \text{SDIST}_P((\!|\cdot|\!))} \text{PR}^\eta(\phi) \leqslant \sup_{\eta \in \text{SCHED}_{\widehat{P}}} \text{PR}^\eta(\phi) \,.$$

By definition of P, the systems M and MDP(P) are isomorphic in the sense that there are two bijections $i^A : \text{ACTIONS}_M \to \text{ACTIONS}_{\text{MDP}(P)}$ and $i^S : S_M \to S_{\text{MDP}(P)}$ such that

$$i^S(\text{INIT}_M) = i^S(\text{INIT}_P) \,,$$

for all $s \in S_M$:

$$\text{ACTIONS}(i^S(s)) = \{i^A(\alpha) \mid \alpha \in \text{ACTIONS}(s)\}$$

and for all $\alpha \in \text{ACTIONS}_M$, $s, s' \in S_M$:

$$\alpha(s, s') = i^A(\alpha)(i^S(s), i^S(s')) .$$

Hence, it is irrelevant whether the reductions are performed on M or on $\text{MDP}(P)$.

We extended PRISM with algorithms to perform reductions complying with **A1–A4**. For each of the states in the system, our algorithms explore subsets of $\text{ACTIONS}(s)$ in order to find a valid ample set. The time spent in this processing is counted as part of the *construction time* in the case studies presented in the following sections.

Since the PRISM language does not provide the means to specify the sets $\text{READVAR}(\alpha)$, we define $\text{READVAR}(\alpha)$ as the set of variables that occur unprimed in a command C such that $\text{LABEL}(\alpha) = \text{LABEL}(C)$. For instance, we have:

Note that x occurs unprimed in the command in line 3

$$\begin{aligned} \text{READVAR}(\alpha_{2,8}) &= \text{READVAR}(\alpha_{2,9}) \\ = \text{READVAR}(\alpha_{3,8}) &= \text{READVAR}(\alpha_{3,9}) = \{p, q, r, x\} \\ \text{READVAR}(\alpha_4) &= \{q\} \\ \text{READVAR}(\alpha_{10}) &= \{q\} \\ \text{READVAR}(\alpha_{11}) &= \{p\} \end{aligned}$$

Our definition of $\text{READVAR}(\cdot)$ corresponds to the assumption that the information used to schedule an action α such that $\text{LABEL}(\alpha) = a$ is the information observable by all the actions β such that $\text{LABEL}(\beta) = a$. In other words, information is not defined for each action, but for each label. Although some other assumptions might allow for better reductions (for instance, we could have defined $\text{READVAR}(\alpha_{2,8}) = \{p, r\}$), we find this assumption appealing and, since the internal representation of PRISM models is based on the labels, it lessened our effort to implement the calculation of $\text{READVAR}(\cdot)$.

Since PRISM provides no means to specify that a label affects the probability that a command is scheduled, we define:

$$\text{READLAB}(\alpha) = \{\text{LABEL}(\alpha)\} .$$

We emphasize that, in practice, we do not perform translations among formalisms. We provided the interpretation of MDPs as IPIOA in order to apply the theoretical machinery in previous chapters but, given that Theorem 9.1 is about reductions on $\text{MDP}(P)$, and taking into account that $\text{MDP}(P)$ is isomorphic to the original MDP M, there is no need for translations.

9.2 ANALYSING THE DINING CRYPTOGRAPHERS

The dining cryptographers [44] is a well-known protocol in which a group communicates a message in such a way that the sender is untraceable. Here, we give a brief explanation of the properties of the protocol. The interested reader is referred to [68].

In its metaphorical explanation, the protocol is carried out during a dinner in the context of a cryptography conference: the conference chair has made arrangements with the restaurant so that, in case a cryptographer wants to

pay for the dinner, he/she is able to do it. Once the dinner has ended, the cryptographers want to know whether some of the cryptographers has paid for the dinner, or the dinner has been funded by the conference committee. However, they do not want to force the payer to disclose himself. This is the point at which the cryptographers follow the protocol. It proceeds in such a way that, when it has finished, each cryptographer i displays a bit B_i. The protocol ensures that there is cryptographer j that paid for the dinner if and only if the sum of all bits has the same parity as N (where N is the number of cryptographers). If this is the case, then the protocol guarantees that, regardless of the bits B_i, the probability that cryptographer j is the payer is 1/N (where N is the number of cryptographers). Then, there is no way for the cryptographers (nor for the conference chair) to guess who the payer is by looking at the bits B_i. In conclusion, the bits B_i communicate the message but, after seeing the B_i, the maximum probability of guessing who is the payer is the same probability that guessing with no information at all (namely 1/N).

Table 2 reports results checking anonymity. Column "%" indicates in percentage how small is the reduced model with respect of the full system. Thus, for instance, the size of the state space of the reduced model is 23.58% of the size of the state space of the full model for 11 cryptographers (i.e., more than 4 times smaller). Note that, in general, the construction time of the system is significantly more expensive for POR when compared to the construction time of the full system. Nonetheless, the calculation time of the probability values is significantly larger in the full model. Thus, the total processing time on large systems is better under POR (see the 11 cryptographers). We remark that the old POR reduction (including **A5**) achieves the same results in this case study. However, these results show that our implementation of POR for symbolic model checking can be very effective.

9.3 ANALYSING THE BINARY EXPONENTIAL BACKOFF PROTOCOL

n

The protocol analysed in this section is carried out by a set of n stations that share a communication channel. These states use the protocol in order to coordinate the access to the channel, since different stations cannot access the channel at the same time. Here, we give a brief explanation of the protocol. The interested reader is referred to [142].

	Full			**A1–A4** reduct.			
N	size	constr.	total	size	%	constr.	total
7*	287666	0m00.19	0m03.53	115578	40.18	0m13.01	0m16.59
8*	1499657	0m00.30	0m16.18	526329	35.10	0m36.69	0m52.96
9*	7695856	0m00.44	1m24.84	2363896	30.72	1m46.16	2m29.15
10	39005612	0m00.70	4m41.10	10495991	26.91	4m48.19	6m40.37
11	195718768	0m01.11	29m43.34	46159864	23.58	13m12.84	21m02.46

* Entries marked with * run on a Pentium 4 630, 3.0Ghz with 2Gb memory, while all the others run on an Opteron 8212 (dual core) with 32Gb memory.

Table 2: Summary of Experimental Results

(a) Size comparison

Model	Full	**A1–A5** reduct.		**A1–A4** reduct.		
n / N / 2^K	size	size	% full	size	% full	% **A5**
4 / 3 / 4	532326	191987	36.07	126629	23.79	65.96
5 / 3 / 4	13866186	2752750	19.85	1690227	12.19	61.40
6 / 3 / 4	357387872	36974560	10.35	21771724	6.09	58.88
4 / 3 / 8	3020342	913379	30.24	604457	20.01	66.18
5 / 3 / 8	115442928	18569442	16.09	11585347	10.04	62.39
6 / 3 / 8	4318481408	353075296	8.18	212917856	4.93	60.30

(b) Time comparison

Model	Full		**A1–A5** reduct.		**A1–A4** reduct.	
n / N / 2^K	constr.	total	constr.	total	constr.	total
4 / 3 / 4	0m01.39	1m04.27	0m18.96	1m22.98	0m18.02	1m13.36
5 / 3 / 4	0m03.49	11m32.99	1m16.82	8m15.60	1m14.53	6m50.12
6 / 3 / 4	0m07.55	5h03m39.81	4m00.95	1h13m11.39	5m15.06	53m35.43
4 / 3 / 8	0m02.05	3m33.62	0m23.85	3m01.88	0m22.78	2m28.50
5 / 3 / 8	0m05.41	1h21m13.54	1m36.41	30m42.82	1m41.18	22m09.23
6 / 3 / 8	0m13.30	—	5m14.95	12h31m57.39	6m44.82	7h45m46.75

These experiments ran on an Opteron 8212 (dual core) with 32Gb memory.

Table 3: Experimental results for the binary exponential backoff protocol

When several stations access the channel at the same time, they detect the collision. After the first collision, the station S_i probabilistically chooses among two options: it either retries immediately (that is, it retries after 0 time units) or it retries after 1 time unit. It might be the case that, in the retrial, S_i faces a new collision. In this case, S_i probabilistically chooses a delay time between 0 and 3. In general, after the j-th collision, S_i probabilistically chooses a delay time ranging from 0 to $2^j - 1$. Hence, the more they collide, the less likely that two stations delay for the same time. The protocol can be parameterized by specifying bounds for the chosen values, as well as for the number of retrials. First, it depends on a number K: after the j-th collision, the choice of S_i ranges from 0 to $\min\{2^j - 1, 2^K - 1\}$. In addition, the parameter $N \geqslant K$ specifies the maximum number of unsuccessful retrials: if after N collisions the station S_i was not able to access the channel, then the station gives up.

K

N

Table 3 shows the model size (that is, the number of states in the model) and the memory consumption required to check the probability that a station gives up in the worst-case. Subtable (a) shows reductions yielding sizes up to 5% of the full state space. More interesting is that reduction using only **A1–A4** is significantly more efficient than the old reduction with **A1–A5** (up to 58.88% in case 6/3/4). As shown in subtable (b), we obtained similar satisfactory results on time comparison, notably (again) in case 6/3/4. We note that the 6/3/8 full model could not be analysed because the state space was too large to fit in the hybrid engine of PRISM.

9.4 DISCUSSION AND FURTHER WORK

Of course, not all examples we ran yielded impressive results as the ones in Tables 2 and 3. We have experienced very little reduction in cases in which components depend very much from each other. This is nonetheless reasonable as our technique is precisely devised for distributed system with little sharing. In particular, both case studies have few communication points and significant local processing.

It is in our plans to report soon on the details of the implementation of the tool under development. In addition, we plan to measure the effectiveness of our technique in other existing protocols. Our aim was to find protocols in which the theoretical improvement results in improved performance, and this aim was accomplished. However, this is just the first step in the evaluation of the practical impact of our technique. One of the pending tasks in this direction is to study how the reduction in the setting of *symbolic* model checking (as in PRISM) compares to the reduction in the *explicit* setting (used in LiQuor [50]).

10 CONCLUDING REMARKS

"No sabe (nadie puede saber) mi innumerable
contrición y cansancio"
Jorge Luis Borges. Ficciones, *El jardín de senderos que se bifurcan*

The first section of this chapter discusses our contributions with respect to the aims and motivations in Chapter 0. The second section provides ideas for further research.

10.1 CONTRIBUTIONS

We started this thesis by arguing that full-history dependent schedulers are unrealistic, and thus restricted schedulers should be considered. Based on these considerations, Chapter 1 resorts to the approach based on local schedulers [64, 45], which resolve local nondeterminism based on local information. Thus, the composition of several local schedulers forms a scheduler for the whole system. *Chapter 1*

By the time we began this research, our goal was to develop model checking algorithms for distributed schedulers. Given the undecidability results in Chapter 5, now we know that this goal cannot be achieved in its full generality. However, our algorithm in Section 6.2 introduces a new approach to overcome undecidability results: instead of considering the usual projection (mapping each path of the system into a path of an atom) we prove the algorithm sound for an alternative projection. If this alternative projection does not disclose too much information, the results obtained by the algorithm are not as pessimistic as in the case of total information schedulers. This approach to algorithms is not the only main contribution in this thesis: we have also seen that the notion of distributed schedulers, besides being useful to obtain tight bounds on maximal probabilities, is also useful to obtain better reductions based on the POR technique (Chapter 7).

The paragraph above outlines the more important results in the thesis, but in the meanwhile we also found several intermediate (but fundamental) notions and results. As usual, some of these results are interesting in their own, and they have spawned new questions. In the beginning, we developed the mechanism to compose local schedulers presented in Subsection 1.1.4. The motivation for these mechanisms was to prove our results in a framework for asynchronous systems with parallel composition *alla* CSP (in which the composition allows all possible sequences of actions, up to synchronization). Existing frameworks are not suitable to work with such models: the systems in [64] are completely synchronous (thus resulting in a straightforward notion of composition), while the composition in [45] is based on a token structure, and the sequences of actions in the composed system are constrained by the token-passing transitions. Our approach using an interleaving scheduler considers asynchronous behaviours, and parallel composition works in the usual way. The introduction of an interleaving scheduler is also useful

to generalize the interleaving mechanism in [147] (where nondeterminism is not considered), thus leading to rate-based schedulers.

Chapter 2 After introducing the interleaving scheduler, we noticed that the partial order reduction technique could not be improved substantially, the obstacle being the unrealistic power of interleaving schedulers. This observation motivated the definition of *strongly distributed schedulers*, in which a natural restriction is imposed on the interleaving scheduler. This restriction reflects the fact that the interleaving of a set of entities depends only on the information available to such entities.

While developing the results in this thesis, we noticed that several proofs shared a certain structure, in which an infinite sequence of schedulers is used to show the existence of a scheduler complying with a certain property.

Chapter 3 Our notion of *limit schedulers* presents a general way to construct a scheduler using the schedulers in a sequence. Theorem 3.2 gives a condition that suffices to ensure that properties from the schedulers in the sequence map to limit schedulers.

Chapter 4 Having in mind several classes of schedulers, we were interested on how these classes are related. A substantial part of Chapter 4 is devoted to find conditions on sets of schedulers. These conditions ensure that the subset of non-randomized schedulers is as expressive as the whole set (in the sense that the maximal probabilities for the subset are the same as for the whole set). This chapter also discusses finite memory schedulers, showing that Markovian distributed schedulers do not attain maximal probabilities even for reachability properties. However, for these properties, we showed that the set of *finite memory schedulers* is fully expressive. This set comprises all N-Markovian schedulers (that is, all schedulers that remember the last N-steps) for all N. Although the results in Chapter 4 do not have direct practical implications, the results in this chapter are extensively used throughout the rest of the thesis. For instance, the conditions to ensure total-order based schedulers are as expressive as randomized strongly distributed schedulers (Theorem 4.3) are used to prove undecidability of qualitative reachability (Theorem 5.7), and to show the correctness of the algorithm in Sec. 6.2.

Chapter 5 We were not deterred by the first undecidability result we found: after proving that the maximum reachability probability cannot be calculated, we wondered whether algorithms for other properties exist. Then, we found that qualitative properties are also impossible to check automatically. Moreover, it is expensive to check a system even under Markovian and non-randomized schedulers.

Chapter 6 Despite the undecidability results, Section 6.2 presents an algorithm to perform *sound* analyses of the system (in the sense that, if the system is deemed correct by the algorithm, then it is correct). This algorithm results useful to check that a system is correct. Section 6.3 introduces an algorithm useful to prove that a system is incorrect.

Chapter 7 For the sake of simplicity, the presentation of the POR technique uses the usual projection and a naive independence relation. However, our proof of the results uses general projections and relations that comply with certain properties. This generalization allows us to profit from the structure of variable-based formalisms (such as PRISM), and to obtain finer independence relations resulting in better reductions.

We were able to use our techniques and algorithms in practical cases:

Chapter 8 Chapter 8 shows how one of our algorithms can be applied to analyse a pro-

tocol for fair service. Chapter 9 discusses partial order reduction for PRISM models.

Chapter 9

10.2 FUTURE RESEARCH DIRECTIONS

Compositionality

Distributed schedulers were introduced to obtain techniques for compositional reasoning†. This goal is achieved for synchronous systems in [64]. In the realm of asynchronous systems, the framework in [45] can be used for compositional reasoning only for special subsets of systems and schedulers (see [45, Chapter 11]). Given that the goal is accomplished in the synchronous framework in [64], but it is not in the asynchronous framework in [45], we tend to think that the noise is introduced by the asynchronous behaviour. Then, we would like to study whether our novel approach to asynchrony (introducing restricted interleaving schedulers) eases the development of compositionality results for asynchronous systems.

Scheduling assumptions in distributed computing

Another interesting direction arises in the field of distributed computing. Several results in this field rely on scheduling assumptions such as "the scheduler cannot use information that has not been read by any of the processes" [9, 7, 8, 37, 48, 49]. At first sight, these assumptions can be put in terms of our projections. For instance, two paths are to be considered equivalents iff they differ only wrt. a value that has not been read. Then, an interesting question is to which extent these projections comply with the properties for projections introduced in this thesis (that is, whether they are or not traceable, whether they are equivalent for some subset of the components, etc.). In particular, this would allow to check whether randomization does or does not add power to the schedulers complying with these assumptions. Since several works restrict to non-randomized schedulers [9, 7, 8, 37], in both cases we would learn new lessons: if randomization adds no power to the scheduler, then the results are more general than we currently know. If randomization does add power, then randomized schedulers might make a difference in concrete protocols, and this is a strong motivation to study whether the protocols are still correct in presence of randomized schedulers.

Distributed schedulers and model checking techniques

One of the main contributions of this thesis is the idea that, although distributed schedulers harden the model checking algorithms (even to the point of undecidability), the assumption that the schedulers are distributed may be used to improve reduction techniques. We plan to study how the distributed schedulers impact on other techniques such as symmetry reduction [113] or abstraction [95].

Are strongly distributed sufficiently general?

Recalling the argument in [147], we have shown how to construct an interleaving scheduler using exponential distributions. These schedulers correspond to the assumption that, given a state s, the time elapsed in s until the next output is distributed exponentially, while the mean of this distribution depends on the whole local history. These schedulers are strongly distributed, and this is no surprise: the interleaving between two atoms relies on a parameter that depends only on their local histories. In fact, our definition of strongly distributed schedulers would be suspicious if rate schedulers were not strongly distributed. This can be seen as a (weak) additional

†In the sense that the set of trace distributions is a precongruence for parallel composition. There are some compositionality results for full-history dependent schedulers, but refinement is defined in terms of simulation, not trace containment [134, 116].

argument to support that strongly distributed schedulers do not rule out realistic schedulers. What makes this argument pretty weak is the fact that it considers only exponential distributions, and this is quite a strong assumption. In order to get a stronger argument, we should consider the schedulers in which the elapsed time is distributed according to an arbitrary distribution, which is selected by the adversary on the basis of the local history. An interesting question is, thus, whether these schedulers are strongly distributed or not. No matter what the answer is, the results are relevant: if all these schedulers are strongly distributed, then we get a nice argument to support strongly distributed schedulers. If some of these schedulers are not strongly distributed, then the definition of strongly distributed schedulers is ruling out realistic behaviours: it is perfectly possible that each entity looks at his local history and then selects a distribution for the time to delay the next output, and so this behaviour must be considered.

(Un)decidability of the infimum reachability problem

A final pending question concerns decidability. So far, works on undecidability for probabilistic systems have focused on the problem of calculating the supremum reachability probability. A possible cause of this interest is that a word is accepted by a probabilistic automata iff the probability of reaching an accepting state is *greater* than a given threshold (and stochastic languages are not closed by complement in general). Hence, from the point of view of language acceptance, the supremum probability results in a much more useful measure than the infimum probability. However, from the point of view of model checking, the infimum probability is relevant: although we cannot check safety properties such as "the system fails with probability at most 0.01" (because of the results in Chapter 5), the infimum probability is useful for liveness properties such as "the system replies with probability at least 0.99". Then, the study of whether the infimum can be automatically calculated worths our consideration.

10.3 A CONCLUSION'S CONCLUSION

We faced several difficulties along the way. The first one (and one of the most disappointing) was undecidability. Another milestone with which we clashed is the fact that distributed schedulers are not sufficient to eliminate the extra restriction for partial order reduction. This motivated the introduction of strongly distributed schedulers. We faced yet another difficulty when we found that randomization adds power to strongly distributed schedulers (as explained in Subsection 4.2.1). Of course, I just mentioned those issues that can be summarized in this short conclusion.

While seeing all the trip in a retrospective view, it feels quite satisfying to deliver the techniques and algorithms in this thesis (it feels even better when taking into account all the milestones we have had to jump across the trip).

Part IV

APPENDIX

"Era come un grandioso piano segreto, e come in ogni piano segreto nessuno era mai a conoscenza di tutti i dettagli"

Riccardo Raccis. Il paradosso di plazzi, *3*

A

PROOFS OF CHAPTER 4

THEOREM 4.7

DEFINITION A.1. Let $\mathcal{I}$ be an interleaving scheduler and S_G be a set of finite global paths, and let σ_{i^*} be a local path such that $[\sigma^m]_{i^*} = \sigma_{i^*}$ for some $\sigma^m \in S_G$.

The interleaving scheduler $\mathrm{NR}(\mathcal{I}, S_G, \sigma_{i^*})$ is defined as

I. $\mathrm{NR}(\mathcal{I}, S_G, \sigma_{i^*})(\sigma)(A_{i^*}) = 1$ if $[\sigma]_{i^*} = \sigma_{i^*}$ and $\sigma \in S_G$ and

II. $\mathrm{NR}(\mathcal{I}, S_G, \sigma_{i^*})(\sigma)(A_i) = \mathcal{I}(\sigma)(A_i)$ for all σ, $A_i \in \mathrm{ATOMS}(P)$, such that either $\sigma \notin S_G$ or $[\sigma]_{i^*} \neq \sigma^*$ and

III. $\mathrm{NR}(\mathcal{I}, S_G, \sigma_{i^*})(\sigma)(A_i) = 0$ if $[\sigma]_{i^*} = \sigma_{i^*}$, $\sigma \in S_G$ and $A_i \neq A_{i^*}$.

Given $\eta = (\mathcal{I}, \{\Theta_i\}_i, \{\Upsilon_i\}_i)$, we define

$$\mathrm{NR}(\eta, S_G, \sigma_{i^*}) = \left(\, \mathrm{NR}(\mathcal{I}, S_G, \sigma_{i^*}) \,, \{\Theta_i\}_i \,, \{\Upsilon_i\} \right).$$

THEOREM. *Let S_G be a set of finite global paths such that, for all $\sigma^m, \sigma^n \in S_G$, $\sigma^m \neq \sigma^n$ it holds $(\sigma^m)^\uparrow \cap (\sigma^n)^\uparrow = \emptyset$. Let $\eta \in \mathrm{DIST}_P([\cdot])$ having interleaving scheduler $\mathcal{I}$, and let*

$$S_L = \{[\sigma]_i \mid \sigma \in S_G \wedge \mathrm{PR}^{\eta^q}((\sigma)^\uparrow) > 0 \wedge \left| G_i(\mathrm{LAST}(\sigma)) \right| > 0\}.$$

If there exist functions $R_i : S_L \cap \mathrm{LOCALPATHS}_i \to \mathbb{R}_{\geqslant 0}$ such that

$$\forall \sigma \in S_G \,:\, \mathcal{I}(\sigma)(A_i) = \frac{R_i([\sigma]_i)}{\sum_{i'} R_i([\sigma]_{i'})},$$

then there exists $\sigma_{i^*} \in S_L$ such that $R_{i^*}(\sigma_{i^*}) > 0$ and

$$\mathrm{PR}^{\mathrm{NR}(\eta, S_G, \sigma_{i^*})}\Big(\biguplus_{\sigma^m \in S_G} (\sigma^m)^\uparrow\Big) \leqslant \mathrm{PR}^{\eta}\Big(\biguplus_{\sigma^m \in S_G} (\sigma^m)^\uparrow\Big).$$

Proof. For simplicity, we write η' instead of $\mathrm{NR}(\eta, S_G, \sigma_{i^*})$. The probabilities changed in η' are only those of the paths σ_m such that $\sigma_g \sqsubset \sigma_m$ for some $\sigma_g \in S_G$. Let

$$p = \{\sigma \in \{\sigma^m\} \mid \exists \sigma_g \,:\, \sigma_g \in S_G \wedge \sigma_g \sqsubset \sigma_m\}. \tag{A.1}$$

We find σ_{i^*} such that $\mathrm{PR}^{\eta'}(\biguplus_{\sigma \in p} (\sigma)^\uparrow) \leqslant \mathrm{PR}^{\eta}(\biguplus_{\sigma \in p} (\sigma)^\uparrow)$. For all $\sigma \in p$, let $H(\sigma)$ be the (uniquely defined) path such that $H(\sigma) \in S_G$ and $H(\sigma) \sqsubseteq \sigma$ (since $\sigma \in p$, we can take $H(\sigma) = \sigma_g$ in Eqn. (A.1)). Let $I(\sigma)$ be $\mathrm{ACTIVE}(\sigma\langle \mathrm{LEN}(H(\sigma))\rangle)$. Moreover, for all σ such that $\mathcal{I}(H(\sigma))(I(\sigma)) > 0$, let

$$Q^\eta_\sigma = \frac{\mathrm{PR}^\eta((\sigma)^\uparrow)}{\mathcal{I}(H(\sigma))(I(\sigma))}.$$

Note that, since the factor $\mathcal{I}(H(\sigma))(I(\sigma))$ appears only once in $\mathrm{PR}^\eta((\sigma)^\uparrow)$, we have

$$Q^\eta_\sigma = Q^{\eta''}_\sigma \tag{A.2}$$

for all σ such that $\mathcal{I}(H(\sigma))(I(\sigma)) > 0$, for all η'' such that η'' differs from η only for $\mathcal{I}(H(\sigma))$. If $\mathcal{I}(H(\sigma))(I(\sigma)) = 0$, we define $Q^{\eta}_{\sigma} = 0$. Then,

$$PR^{\eta}((\sigma)^{\uparrow}) = \mathcal{I}(H(\sigma))(I(\sigma)) \cdot Q^{\eta}_{\sigma} = \frac{R(H(\sigma, I(\sigma)))}{\sum_j R(H(\sigma, j))} \cdot Q^{\eta}_{\sigma}$$

for all σ, where $H(\sigma, i) = [H(\sigma)]_i$. As a consequence

$$PR^{\eta}((\sigma)^{\uparrow}) \cdot \sum_j R(H(\sigma, j)) = R(H(\sigma, I(\sigma))) \cdot Q^{\eta}_{\sigma} \,. \tag{A.3}$$

We show that

$$\sigma_{i^*} = \arg \min_{\{\sigma_i \mid R_i(\sigma_i) > 0\}} \sum_{\{\sigma \in p \mid I(\sigma) = A_i \wedge H(\sigma, i) = \sigma_i\}} Q^{\eta}_{\sigma} + \sum_{\{\sigma \in p \mid H(\sigma, i) \neq \sigma_i\}} PR^{\eta}((\sigma)^{\uparrow})$$

yields

$$PR^{NR(\eta, S_G, \sigma^*)}(\biguplus_m (\sigma_m)^{\uparrow}) \leqslant PR^{\eta}(\biguplus_m (\sigma_m)^{\uparrow}) \,.$$

In the following calculation, let $J = \sum_{\sigma_i \in S_L} R_i(\sigma_i)$.

$$\begin{aligned}
&PR^{\eta}(\biguplus_{\sigma \in p} (\sigma)^{\uparrow}) \\
=&\frac{1}{J} \sum_{\sigma \in p} PR^{\eta}((\sigma)^{\uparrow}) \cdot J \\
=&\frac{1}{J} \Big(\sum_{\sigma \in p} PR^{\eta}((\sigma)^{\uparrow}) \cdot \Big(\sum_{\{\sigma_j \mid H(\sigma, j) = \sigma_j\}} R_j(\sigma_j) \Big) \\
&\quad + \sum_{\sigma \in p} PR^{\eta}((\sigma)^{\uparrow}) \cdot \Big(\sum_{\{\sigma_j \mid H(\sigma, j) \neq \sigma_j\}} R_j(\sigma_j) \Big) \Big) \\
=&\frac{1}{J} \Big(\overbrace{\sum_{A_i} \sum_{\sigma_i} \sum_{\{\sigma \in p \mid I(\sigma) = A_i \wedge H(\sigma, i) = \sigma_i\}} PR^{\eta}((\sigma)^{\uparrow}) \cdot \Big(\sum_{\{\sigma_j \mid H(\sigma, j) = \sigma_j\}} R_j(\sigma_j) \Big)}^{(\star)} \\
&\quad + \sum_{\sigma \in p} PR^{\eta}((\sigma)^{\uparrow}) \cdot \Big(\sum_{\{\sigma_j \mid H(\sigma, j) \neq \sigma_j\}} R_j(\sigma_j) \Big) \Big) \\
=&\frac{1}{J} \Big((\star) + \sum_{\sigma \in p} \sum_{A_j} \sum_{\{\sigma_j \mid H(\sigma, j) \neq \sigma_j\}} PR^{\eta}((\sigma)^{\uparrow}) \cdot R_j(\sigma_j) \Big) \\
=&\frac{1}{J} \Big((\star) + \sum_{\sigma \in p} \sum_{A_i} \sum_{\{\sigma_i \mid H(\sigma, i) \neq \sigma_i\}} PR^{\eta}((\sigma)^{\uparrow}) \cdot R_i(\sigma_i) \Big) \\
=&\frac{1}{J} \Big((\star) + \sum_{A_i} \sum_{\sigma_i} \sum_{\{\sigma \in p \mid H(\sigma, i) \neq \sigma_i\}} PR^{\eta}((\sigma)^{\uparrow}) \cdot R_i(\sigma_i) \Big) \\
=&\{ \text{ Apply Equation (A.3) in (1) } \} \\
&\frac{1}{J} \Big(\sum_{A_i} \sum_{\sigma_i} \Big(\sum_{\{\sigma \in p \mid I(\sigma) = A_i \wedge H(\sigma, i) = \sigma_i\}} R_i(\sigma_i) \cdot Q^{\eta}_{\sigma} \\
&\qquad + \sum_{\{\sigma \in p \mid H(\sigma, i) \neq \sigma_i\}} PR^{\eta}((\sigma)^{\uparrow}) \cdot R_i(\sigma_i) \Big) \Big) \\
=&\frac{1}{J} \Big(\sum_{A_i} \sum_{\sigma_i} R(\sigma_i) \Big(\sum_{\{\sigma \in p \mid I(\sigma) = A_i \wedge H(\sigma, i) = \sigma_i\}} Q^{\eta}_{\sigma}
\end{aligned}$$

$$
\begin{aligned}
&\qquad\qquad + \sum_{\{\sigma\in p \mid H(\sigma,i)\neq\sigma_i\}} PR^{\eta}((\sigma)^{\uparrow})\Big)\Big)\\
&\geqslant\\
&\frac{1}{J}\Big(\sum_{A_i}\sum_{\sigma_i} R(\sigma_i)\Big(\sum_{\{\sigma\in p \mid I(\sigma)=\wedge\, H(\sigma,i^*)=\sigma_{i^*}\}} Q^{\eta}_{\sigma}\\
&\qquad\qquad + \sum_{\{\sigma\in p \mid H(\sigma,i^*)\neq\sigma_{i^*}\}} PR^{\eta}((\sigma)^{\uparrow})\Big)\Big)\\
&=\frac{1}{J}\Big(\sum_{A_i}\sum_{\sigma_i} R(\sigma_i)\Big)\cdot\Big(\sum_{\{\sigma\in p \mid I(\sigma)=A_{i^*}\,\wedge\, H(\sigma,i^*)=\sigma_{i^*}\}} Q^{\eta}_{\sigma}\\
&\qquad\qquad + \sum_{\{\sigma\in p \mid H(\sigma,i^*)\neq\sigma_{i^*}\}} PR^{\eta}((\sigma)^{\uparrow})\Big)\\
&= \sum_{\{\sigma\in p \mid I(\sigma)=A_{i^*}\,\wedge\, H(\sigma,i^*)=\sigma_{i^*}\}} Q^{\eta}_{\sigma}\\
&\qquad\qquad + \sum_{\{\sigma\in p \mid H(\sigma,i^*)\neq\sigma_{i^*}\}} PR^{\eta}((\sigma)^{\uparrow})\\
&=\{\ \sigma_{i^*}=\arg\min_{\{\sigma_i \mid\}}\cdots \quad \text{and so Eqn. (A.2) applies}\ \}\\
&\qquad \sum_{\{\sigma\in p \mid I(\sigma)=A_{i^*}\,\wedge\, H(\sigma,i^*)=\sigma_{i^*}\}} Q^{NR(\eta,S_G,\sigma_{i^*})}_{\sigma}\\
&\qquad\qquad + \sum_{\{\sigma\in p \mid H(\sigma,i^*)\neq\sigma_{i^*}\}} PR^{\eta}((\sigma)^{\uparrow})
\end{aligned}
$$

Note that this is the probability of p under $\mathrm{NR}(\eta, S_G, \sigma_{i^*})$. In fact, the summand

$$\sum_{\{\sigma\in p \mid I(\sigma)=A_{i^*}\,\wedge\, H(\sigma,i^*)=\sigma_{i^*}\}} Q^{NR(\eta,S_G,\sigma_{i^*})}_{\sigma}$$

reflects item *(I)* in the definition of $\mathrm{NR}(\eta, S_G, \sigma_{i^*})$, the summand

$$\sum_{\{\sigma\in p \mid H(\sigma,i^*)\neq\sigma_{i^*}\}} PR^{\eta}((\sigma)^{\uparrow})$$

reflects item *(II)*. In the step of the calculation that introduces the inequality $\geqslant$, the summands

$$\sum_{\sigma_i\neq\sigma_{i^*}} R(\sigma_i)\Big(\sum_{\{\sigma\in p \mid I(\sigma)=A_i\,\wedge\, H(\sigma,i)=\sigma_i\}} Q^{\eta}_{\sigma} + \sum_{\{\sigma\in p \mid H(\sigma,i)\neq\sigma_i\}} PR^{\eta}((\sigma)^{\uparrow})\Big)$$

are discarded, reflecting item *(III)*. □

LEMMA 4.10

Let $[\cdot]$ be traceable. Let $\mathcal{F}^1,\cdots,\mathcal{F}^K$ be a set of fringes and $\eta^1,\cdots,\eta^K$ be a set of schedulers such that, $\mathcal{F}^1=\{\mathrm{INIT}_P\}$ and for all k, there exists $\sigma^k_{i^*}$ such that

$$\mathcal{F}^{k+1}=\mathrm{SPAWN}(\mathcal{F}^k)(\{\sigma \mid [\sigma]_{i^*}=\sigma^k_{i^*}\})\neq\mathcal{F}^k\ ,$$

and for all $k'\geqslant k$

$$\forall\sigma\in\mathcal{F}^k : [\sigma]_{i^*}=\sigma^k_{i^*} \implies \mathcal{I}^{k'}(\sigma)(A_{i^*})=1 \tag{A.4}$$

and

$$\forall \sigma : \mathrm{PR}^{\eta^{k'}}((\sigma)^{\uparrow}) > 0 \implies \mathrm{PR}^{\eta^{k}}((\sigma)^{\uparrow}) > 0 \,. \tag{A.5}$$

Then, for all k, $0 < n < K - k$,

$$\forall \sigma \in \mathcal{F}^{k+n} : \mathrm{PR}^{\eta^{k+n}}((\sigma)^{\uparrow}) > 0 \implies \sigma^{k}_{i^*} \neq [\sigma]_{i^*} \,.$$

Proof. Suppose towards a contradiction, that there exists $\sigma^* \in \mathcal{F}^{k+n}$ such that

$$\sigma^{k}_{i^*} = [\sigma^*]_{i^*} \tag{A.6}$$

and $\mathrm{PR}^{\eta^{k+n}}((\sigma^*)^{\uparrow}) > 0$ for some $n > 0$, k. Then, since

$$\mathcal{F}^{j+1} = \textsc{Spawn}(\mathcal{F}^{j})(\{\sigma \mid [\sigma]_{i^*} = \sigma^{j}_{i^*}\})$$

for all j, we have that there exists σ^k such that $\sigma^k \in \mathcal{F}^k$, $\sigma^k \sqsubset \sigma^*$. Moreover, since

$$\mathcal{F}^{k+1} = \textsc{Spawn}(\mathcal{F}^{k})(\{\sigma \mid [\sigma]_{i^*} = \sigma^{k}_{i^*}\}) \,,$$

it must be

$$[\sigma^k]_{i^*} \neq \sigma^{k}_{i^*} \,.$$

Otherwise, it would be $\mathcal{I}(\sigma^k) = A_{i^*}$ and $\sigma^k \sqsubset \sigma^*$: in this setting, property (1.11) implies $[\sigma^*]_{i^*} \neq \sigma^{k}_{i^*}$, thus contradicting Eqn. (A.6). Note also that, by Eqn. (A.5), we have $\mathrm{PR}^{\eta^{k}}((\sigma^*)^{\uparrow}) > 0$ and hence $\mathrm{PR}^{\eta^{k}}((\sigma^k)^{\uparrow}) > 0$. Let σ be a path in $\mathcal{F}^k$ such that $[\sigma]_{i^*} = \sigma^{k}_{i^*}$ (the existence of such σ is ensured by $\mathcal{F}^k \neq \mathcal{F}^{k+1}$). Then, by renaming σ^k as σ', we have that the existence of σ^* implies the following statement: there exists $\sigma, \sigma' \in \mathcal{F}^k$, σ_i such that $[\sigma]_i = \sigma_i$, $[\sigma']_i \neq \sigma_i$ and there is a path σ'' (namely, σ^*) such that $\sigma' \sqsubset \sigma''$, $[\sigma'']_i = \sigma_i$ and $\mathrm{PR}^{\eta^{k}}((\sigma'')^{\uparrow}) > 0$. We prove that this statement cannot hold under the hypotheses of the theorem, thus reaching a contradiction. Note that the statement does not depend on n. In order to write the statement more succintly, we say that $(\sigma, \sigma', \sigma'', \sigma_i)$ is a fail for $\mathcal{F}^k$. W.l.o.g. we take assume that σ'' is as short as possible (i.e., it is minimal with respect to the prefix relation).

We prove that all the $\mathcal{F}^k$ have no fails by induction on K. If $K = 1$ there is only one fringe $\mathcal{F}^1 = \{\textsc{Init}\}$. In a fail $(\sigma, \sigma', \sigma'', \sigma_i)$ it must be $[\sigma]_i \neq [\sigma']_i$, and so $\sigma \neq \sigma'$. Since $\mathcal{F}^1$ has no two different paths, it cannot have fails.

For the inductive step, assume that $\mathcal{F}^{K-1}$ has no fails. Let σ_{i^*} be the local path used to spawn $\mathcal{F}^K$ from $\mathcal{F}^{K-1}$. For all $\sigma \in \mathcal{F}^K$, we say that σ is *old* (is *new*, resp.) if $\sigma \in \mathcal{F}^{K-1}$ (if $\sigma \notin \mathcal{F}^{K-1}$, resp.) Suppose, towards a contradiction, that $\mathcal{F}^K$ has a fail $(\sigma, \sigma', \sigma'', \sigma_i)$. We consider four possible cases:

- Both σ and σ' are old.

  In this case, $(\sigma, \sigma', \sigma'', \sigma_i)$ are a fail in $\mathcal{F}^{K-1}$, contradicting the inductive hypothesis.

- The path σ is old and σ' is new.

  If $[\sigma']_i = [\sigma'{\downarrow_{-1}}]_i$, then $(\sigma, \sigma'{\downarrow_{-1}}, \sigma'', \sigma_i)$ is a fail in $\mathcal{F}^{K-1}$.

If $[\sigma']_i \neq [\sigma'_{\downarrow -1}]_i$ we note that, since A_{i^*} produces the output in $\sigma'_{\downarrow -1}$, property (1.11) implies

$$[\sigma''_{\downarrow -1}]_{i^*} \neq [\sigma'_{\downarrow -m}]_{i^*} \tag{A.7}$$

for all $m > 1$ [†]. Let σ_s be the smallest prefix of $\sigma'_{\downarrow -1}$ such that $[\sigma_s]_i = [\sigma'_{\downarrow -1}]_i$. If $\sigma_s = \text{INIT}_P$, then $[\sigma'_{\downarrow -1}]_i = [\text{INIT}]_i$ while σ'' has changed its projection over A_i at least once (namely, in the step from $\sigma'_{\downarrow -1}$ to σ'), and hence by Eqn. (4.14) we have $[\sigma'_{\downarrow -1}]_i \neq [\sigma'']_i$, thus implying that $(\sigma, \sigma'_{\downarrow -1}, \sigma'', \sigma_i)$ is a fail. If $\sigma_s \neq \text{INIT}_P$, by Eqn. (A.7) we have $[\sigma''_{\downarrow -1}]_{i^*} \neq [\sigma_{s\downarrow -1}]_{i^*}$. Since $[\cdot]$ is traceable, we have $[\sigma'_{\downarrow -1}]_i \neq [\sigma'']_i$ and, again, $(\sigma, \sigma'_{\downarrow -1}, \sigma'', \sigma_i)$ is a fail.

In conclusion, $(\sigma, \sigma'_{\downarrow -1}, \sigma'', \sigma_i)$ is a fail in $\mathcal{F}^{K-1}$, thus contradicting the inductive hypothesis.

- The path σ is new and σ' is old.

  Since $[\cdot]$ is traceable, it must be

  $$[\sigma''_{\downarrow -1}]_{i^*} = [\sigma_{\downarrow -1}]_{i^*} = \sigma_i^* \, .$$

  Since $\mathcal{F}^{K-1}$ has not been spawned at σ', we have $[\sigma']_{i^*} \neq \sigma_i^*$ and so $(\sigma_{\downarrow -1}, \sigma', \sigma''_{\downarrow -1}, \sigma_i^*)$ is a fail in $\mathcal{F}^{K-1}$.

- Both σ and σ' are new.

  It must be $[\sigma''_{\downarrow -1}]_{i^*} = [\sigma_{\downarrow -1}]_{i^*}$ and, since both σ and σ' are new, it holds $[\sigma_{\downarrow -1}]_{i^*} = [\sigma'_{\downarrow -1}]_{i^*}$. However, since A_{i^*} outputs more labels in $\sigma''_{\downarrow -1}$ than in $\sigma'_{\downarrow -1}$ (recall that it outputs the label after $\sigma'_{\downarrow -1}$) and by property (1.11), we have $[\sigma'_{\downarrow -1}]_{i^*} \neq [\sigma''_{\downarrow -1}]_{i^*}$, thus contradicting $[\sigma''_{\downarrow -1}]_{i^*} = [\sigma_{\downarrow -1}]_{i^*} = [\sigma'_{\downarrow -1}]_{i^*}$.

□

[†]It is worth noting that here we are using the fact that $\text{PR}^{\eta^k}((\sigma')^\uparrow) > 0$, since otherwise the last transition in σ' would not be necessarily the one prescribed by η^K.

B

PROOFS OF CHAPTER 6

The following theorem makes explicit the correspondence between P and MDP(P), by showing that the probability of reaching s' from s coincides in the respective schedulers. This correspondence suffices for our purposes, since in this thesis we explore the verification of properties concerning the states.

From Def. 7.1 (p. 122) we have that, given a scheduler η for P, a corresponding scheduler $\eta_{\mathrm{MDP}(P)}$ for MDP(P) can be constructed. Here, we make it precise the correspondence between these schedulers.

THEOREM B.1. *For all* σ,

$$\sum_{\alpha,a'} \eta_{\mathrm{MDP}(P)}(\sigma)(\alpha) \cdot P((a,s),\alpha,(a',(s'_1,\cdots,s'_N))) = \sum_{c} \eta(\sigma)(c) \cdot c(s,s')$$

Proof. For all i, a, let $g_i = \Theta_i(\sigma)$ and $r_{i,a} = \Upsilon(\sigma,a)$. Then,

$$\sum_{\alpha,a'} \eta_{\mathrm{MDP}(P)}(\sigma)(\alpha) \cdot P((a,s),\alpha,(a',(s'_1,\cdots,s'_N))) = \sum_{i,a'} \mathcal{I}(\sigma)(A_i) \cdot g_i(s,a',s'_i) \cdot \prod_{k=1}^{m} r_{j_k,a'}(s,a',s'_{j_k})$$

In addition,

$$\begin{aligned}
&\sum_{c} \eta(\sigma)(c) \cdot c(s,s') \\
&= \sum_{i,a'} \sum_{s''_i} g_i(s,a',s''_i) \cdot \mathcal{I}(\sigma)(A_i) \cdot \frac{1}{\sum_{s''_i} g_i(s,a',s''_i)} \\
&\qquad \cdot g_i(s,a',s_i) \cdot \prod_{k=1}^{m} r_{j_k,a'}(s,a',s'_{j_k}) \\
&= \sum_{i,a'} \mathcal{I}(\sigma)(A_i) \cdot g_i(s,a',s'_i) \cdot \prod_{k=1}^{m} r_{j_k,a'}(s,a',s'_{j_k})
\end{aligned}$$

□

Given a non-randomized scheduler η_M for MDP(P), a corresponding scheduler η for P can be defined as $\mathcal{I}(\sigma) = A_i$, $\Theta_i(\sigma) = g_i$, $\Upsilon_i(\sigma,a) = f_i(a)$, where $(g_i, f_1, \cdots, f_N) = \eta(\sigma)$. It is easy to see that $\eta_{\mathrm{MDP}(P)}(\sigma) = \eta_M$.

THEOREM (6.1). *Let* $\mathcal{S}$ *be a measurable, state-based set of infinite paths. Then,*

$$\sup_{\eta \in \mathrm{SCHED}_P} PR^{\eta}(\mathcal{S}) = \sup_{\eta \in \mathrm{SCHED}_{\mathrm{MDP}(P)}} PR^{\eta}(\mathcal{S}) \,.$$

Proof. For each $\eta \in \mathrm{SCHED}_P$, we have $\eta_{\mathrm{MDP}(P)} \in \mathrm{SCHED}_{\mathrm{MDP}(P)}$. By Theorem B.1, $PR^{\eta}(\mathcal{S}) = PR^{\eta_{\mathrm{MDP}(P)}}(\mathcal{S})$ and so

$$\sup_{\eta \in \mathrm{SCHED}_P} PR^{\eta}(\mathcal{S}) \leqslant \sup_{\eta \in \mathrm{SCHED}_{\mathrm{MDP}(P)}} PR^{\eta}(\mathcal{S}) \,.$$

In addition, since non-randomized schedulers attain supremum probabilities for MDPs, and for all non-randomized $\eta_M \in \text{SCHED}_{\text{MDP}(P)}$ there exists $\eta \in \text{SCHED}_P$ such that $\eta_{\text{MDP}(P)} = \eta_M$ (as shown above), we have

$$\sup_{\eta \in \text{SCHED}_{\text{MDP}(P)}} \Pr^{\eta}(\mathcal{S}) \leqslant \sup_{\eta \in \text{SCHED}_P} \Pr^{\eta}(\mathcal{S}) \,.$$

□

C

PROOFS OF CHAPTER 7

LEMMA 7.1

For all simple IPIOA P*, the triple* $(\mathcal{R}_I^{INV}, \{(\pi^{\alpha,[\![\cdot]\!]}, \pi^{\neg\alpha,[\![\cdot]\!]})\}_{\alpha\in ACTIONS}, [\![\cdot]\!])$ *is an independence structure.*

Proof. Given α, β, a, b, s and s' as in the definition of independence structure, let $s = (s_1, \cdots, s_N)$, and similarly for s' and s''. Let $\alpha = (g_k, f_1, \cdots, f_N)$, $\beta = (g_{k'}, f'_1, \cdots, f'_N)$.

Let's start with the first property. If $\alpha(s, a, s') > 0$, we have $s_i \neq s'_i \implies A_i \in INV(\alpha)$. Since $INV(\alpha) \cap INV(\beta) = \emptyset$, we conclude

$$\forall A_i \in INV(\beta) : s_i = s'_i \,. \tag{C.1}$$

Then,

$$\begin{aligned}
& \beta \in ACTIONS(s) \\
\iff & \{\ P \text{ is simple (and so the generative/reactive structures are as in Def. 1.2 }\} \\
& g_{k'} \in G_{k'}(s_{k'}) \wedge \forall_{A_j \in REACTIVE(\beta), a\in ACTLAB_j} f'_j(a) \in R_j(s_j, a) \\
\iff & \{\ \text{Equation (C.1)}\ \} \\
& g_{k'} \in G_{k'}(s'_{k'}) \wedge \forall_{A_j \in REACTIVE(\beta), a\in ACTLAB_j} R_j(s'_j, a) \\
\iff & \{\ P \text{ is simple}\ \} \\
& \beta \in ACTIONS(s')
\end{aligned}$$

Property *(2)* holds by definition of (probability assigned by a) compound transition, since it must be $s_i = s'_i$ for all A_i such that $a \notin ACTLAB_i$, and $a \notin ACTLAB_i$ for all $A_i \notin INV(\alpha)$.

Property *(3)* holds again by definition of compound transition. As before, it must be $s_i = s'_i$ for all A_i such that $a \notin ACTLAB_i$. Let $A_j \in INV(\alpha)$. Then $A_j \notin INV(\beta)$ (since $INV(\alpha) \cap INV(\beta) = \emptyset$) and so $a \notin ACTLAB_j$. In conclusion $s_j = s'_j$ for all $A_j \in INV(\alpha)$, as desired.

For property *(4)*, let $\mathcal{B} = \{A_i \mid a \in ACTLAB_i\}$. Then,

$$\begin{aligned}
& \alpha((s^\alpha, s^{\neg\alpha}), a, (s'^\alpha, s^{\neg\alpha})) \\
= & \{\ \text{Definition}\ \} \\
& g_k(s_k, a, s'_k) \cdot \prod_{A_i \in \mathcal{B}\setminus\{A_k\}} f_i(a)(s_i, a, s'_i) \\
= & \{\ \mathcal{B} \subseteq INV(\alpha), \text{definition of } s^{\neg\alpha}\ \} \\
& \alpha((s^\alpha, s''^{\neg\alpha}), a, (s'^\alpha, s''^{\neg\alpha}))
\end{aligned}$$

The same proof can be carried out for β, thus implying the property.

Since $ACTIVE(\alpha) \in INV(\alpha)$ and $ACTIVE(\beta) \in INV(\beta)$, property *(5)* follows from $\alpha \mathcal{R}_I^{INV} \beta$.

Property *(6)*: $\alpha^k \mathcal{R}_I^{INV} \beta$ implies

$$A_i \notin INV(\alpha^k) \tag{C.2}$$

for all $A_i \in INV(\beta)$, $k = 1, \cdots, n$, and in particular for all $A_i \in AFFECT(\alpha^k)$. (C.2) implies $a^k \notin ACTLAB_i$, for all k, for all $A_i \in AFFECT(\beta)$. Hence, the projection $[\![\cdot]\!]_i$ hides the steps after σ.

For property *(7)*, if

$$\begin{aligned}&[\![\sigma.\alpha^1.a^1.(s^\beta, s^{1^{\neg\beta}}) \cdots \alpha^n.a^n.(s^\beta, s^{n^{\neg\beta}})]\!]_i \\ \neq\ &[\![\sigma.\alpha'^1.a'^1.(s^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^{n'}.a'^{n'}.(s^\beta, s'^{n'^{\neg\beta}})]\!]_i\end{aligned}$$

then

$$[\![\sigma.\alpha^1.a^1.(s^\beta, s^{1^{\neg\beta}}) \cdots \alpha^n.a^n.(s^\beta, s^{n^{\neg\beta}})]\!]_i = [\![\sigma]\!]_i.b^1.s_i^1.\cdots.b^m.s_i^m$$

and

$$[\![\sigma.\alpha'^1.a'^1.(s^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^{n'}.a'^{n'}.(s^\beta, s'^{n'^{\neg\beta}})]\!]_i = [\![\sigma]\!]_i.b'^1.s_i'^1.\cdots.b'^{m'}.s_i'^{m'}$$

for some b, s_i^k such that $b^1.s_i^1.\cdots.b^m.s_i^m \neq b'^1.s_i'^1.\cdots.b'^{m'}.s_i'^{m'}$. We consider two cases. In case $a \notin \text{ACTLAB}_i$, we have

$$\begin{aligned}&[\![\sigma.\beta.a.(t^\beta, s^{\neg\beta}).\alpha^1.a^1.(t^\beta, s^{1^{\neg\beta}}) \cdots \alpha^n.a^n.(t^\beta, s^{n^{\neg\beta}})]\!]_i \\ =\ &[\![\sigma]\!]_i.b^1.s_i^1.\cdots.b^m.s_i^{m'} \\ \neq\ &[\![\sigma]\!]_i.b'^1.s_i'^1.\cdots.b'^{m'}.s_i'^{m'} \\ =\ &[\![\sigma.\beta.a.(t'^\beta, s^{\neg\beta}).\alpha'^1.a'^1.(t'^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^{n'}.a'^{n'}.(t'^\beta, s'^{n'^{\neg\beta}})]\!]_i\end{aligned}$$

In case $a \in \text{ACTLAB}_i$, there exist local states ν_i, ν_i' such that

$$\begin{aligned}&[\![\sigma.\beta.a.(t^\beta, s^{\neg\beta}).\alpha^1.a^1.(t^\beta, s^{1^{\neg\beta}}) \cdots \alpha^n.a^n.(t^\beta, s^{n^{\neg\beta}})]\!]_i \\ =\ &[\![\sigma]\!]_i.a.\nu_i.b^1.s_i^1.\cdots.b^m.s_i^m \\ \neq\ &\{\ b^1.s_i^1.\cdots.b^m.s_i^m \neq b'^1.s_i'^1.\cdots.b'^{m'}.s_i'^{m'} \\ &\quad \text{implies } a.\nu_i.b^1.s_i^1.\cdots.b^m.s_i^m \neq a.\nu_i'.b'^1.s_i'^1.\cdots.b'^{m'}.s_i'^{m'}\ \} \\ &[\![\sigma]\!]_i.a.\nu_i'.b'^1.s_i'^1.\cdots.b'^{m'}.s_i'^{m'} \\ =\ &[\![\sigma.\beta.a.(t'^\beta, s^{\neg\beta}).\alpha'^1.a'^1.(t'^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^{n'}.a'^{n'}.(t'^\beta, s'^{n'^{\neg\beta}})]\!]_i\end{aligned}$$

For property *(8)*, if

$$\begin{aligned}&\big[\!\big[\sigma.\alpha^1.a^1.(s^\beta, s^{1^{\neg\beta}}) \cdots \alpha^n.a^n.(s^\beta, s^{n^{\neg\beta}}).\beta.a.(t^\beta, s^{n^{\neg\beta}}) \\ &\qquad\qquad .\gamma^1.b^1.u^1.\cdots.\gamma^m.b^m.u^m\big]\!\big]_i \\ \neq\ &\big[\!\big[\sigma.\alpha'^1.a^1.(s^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^{n'}.a'^{n'}.(s^\beta, s'^{n'^{\neg\beta}}).\beta.a.(t'^\beta, s'^{n'^{\neg\beta}}) \\ &\qquad\qquad .\gamma'^1.b'^1.u'^1.\cdots.\gamma'^{m'}.b'^{m'}.u'^{m'}\big]\!\big]_i\end{aligned}$$

let $(t^\beta, s^{n^{\neg\beta}}) = (s_1^*, \cdots, s_{\mathcal{N}}^*)$ and $(t'^\beta, s'^{n'^{\neg\beta}}) = (s_1'^*, \cdots, s_{\mathcal{N}}'^*)$. Note that, by definition of the β projection $\pi^{\beta,[\![\cdot]\!]}$, we have

$$s_i^* = \pi_i(t^\beta, s^{n^{\neg\beta}}) = \pi_i(t^\beta, s^{\neg\beta}) \tag{C.3}$$

$$s_i'^* = \pi_i(t'^\beta, s'^{n'^{\neg\beta}}) = \pi_i(t'^\beta, s^{\neg\beta}) \tag{C.4}$$

for all $A_i \in \text{INV}(\beta)$, since the local state of A_i lies in t^β.

In case $a \in \text{ACTLAB}_i$, then $A_i \in \text{INV}(\beta)$ and so, by definition of $\mathcal{R}_I^{\text{INV}}$, we have $A_i \notin \text{INV}(\alpha^k)$, $A_i \notin \text{INV}(\alpha'^k)$ for all k. Hence, $a^k, a'^k \notin \text{ACTLAB}_i$ for all k. Then

$$\begin{aligned}
&[\![\sigma.\alpha^1.a^1.(s^\beta, s^{1^{\neg\beta}}) \cdots \alpha^n.a^n.(s^\beta, s^{n^{\neg\beta}}).\beta.a.(t^\beta, s^{n^{\neg\beta}}) \\
&\qquad\qquad .\gamma^1.b^1.u^1.\cdots.\gamma^m.b^m.u^m]\!]_i \\
&= [\![\sigma]\!]_i.a.s_i^*.e^1.u_i^1.\cdots.e^q.u_i^q
\end{aligned}$$

and

$$\begin{aligned}
&[\![\sigma.\alpha'^1.a^1.(s^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^{n'}.a'^{n'}.(s^\beta, s'^{n'^{\neg\beta}}).\beta.a.(t'^\beta, s'^{n'^{\neg\beta}}) \\
&\qquad\qquad .\gamma'^1.b'^1.u'^1.\cdots.\gamma'^{m'}.b'^{m'}.u'^{m'}]\!]_i \\
&= [\![\sigma]\!]_i.a.s_i'^*.e'^1.u_i'^1.\cdots.e'^q.u_i'^{q'}
\end{aligned}$$

for some $e^1.u_i^1.\cdots.e^q.u_i^q$, $e'^1.u_i'^1.\cdots.e'^q.u_i'^{q'}$ such that

$$a.s_i^*.e^1.u_i^1.\cdots.e^q.u_i^q \neq a.s_i'^*.e'^1.u_i'^1.\cdots.e'^q.u_i'^{q'} \tag{C.5}$$

Then, if $a \in \text{ACTLAB}_i$, we prove the desired inequality as follows.

$$\begin{aligned}
&[\![\sigma.\beta.a.(t^\beta, s^{\neg\beta}).\alpha^1.a^1.(t^\beta, s^{1^{\neg\beta}}) \cdots \alpha^n.a^n.(t^\beta, s^{n^{\neg\beta}}) \\
&\qquad\qquad .\gamma^1.b^1.u^1.\cdots.\gamma^m.b^m.u^m]\!]_i \\
&= \{\ A_i \notin \text{INV}(\alpha^k) \text{ for all } k\ \} \\
&\quad [\![\sigma]\!]_i.a.\pi_i(t^\beta, s^{\neg\beta}).e^1.u_i^1.\cdots.e^q.e_i^q \\
&= \{\ \text{Equation (C.3)}\ \} \\
&\quad [\![\sigma]\!]_i.a.s_i^*.e^1.u_i^1.\cdots.e^q.e_i^q \\
&\neq \{\ \text{Inequality (C.5)}\ \} \\
&\quad [\![\sigma]\!]_i.a.s_i'^*.e'^1.u_i'^1.\cdots.e'^q.u_i'^{q'} \\
&= \{\ A_i \notin \text{INV}(\alpha'^k) \text{ for all } k\ \} \\
&\quad [\![\sigma.\beta.a.(t'^\beta, s^{\neg\beta}).\alpha'^1.a'^1.(t'^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^{n'}.a'^{n'}.(t'^\beta, s'^{n'^{\neg\beta}}) \\
&\qquad\qquad .\gamma'^1.b'^1.u'^1.\cdots.\gamma'^{m'}.b'^{m'}.u'^{m'}]\!]_i
\end{aligned}$$

If $a \notin \text{ACTLAB}_i$, then

$$\begin{aligned}
&[\![\sigma.\alpha^1.a^1.(s^\beta, s^{1^{\neg\beta}}) \cdots \alpha^n.a^n.(s^\beta, s^{n^{\neg\beta}}).\beta.a.(t^\beta, s^{n^{\neg\beta}}) \\
&\qquad\qquad .\gamma^1.b^1.u^1.\cdots.\gamma^m.b^m.u^m]\!]_i \\
&= [\![\sigma]\!]_i.e^1.u_i^1.\cdots.e^q.u_i^q \\
&= [\![\sigma.\beta.a.(t^\beta, s^{\neg\beta}).\alpha^1.a^1.(t^\beta, s^{1^{\neg\beta}}) \cdots \alpha^n.a^n.(t^\beta, s^{n^{\neg\beta}}) \\
&\qquad\qquad .\gamma^1.b^1.u^1.\cdots.\gamma^m.b^m.u^m]\!]_i
\end{aligned}$$

and

$$\begin{aligned}
&[\![\sigma.\alpha'^1.a^1.(s^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^{n'}.a'^{n'}.(s^\beta, s'^{n'^{\neg\beta}}).\beta.a.(t'^\beta, s'^{n'^{\neg\beta}}) \\
&\qquad\qquad .\gamma'^1.b'^1.u'^1.\cdots.\gamma'^{m'}.b'^{m'}.u'^{m'}]\!]_i \\
&= [\![\sigma]\!]_i.e'^1.u_i'^1.\cdots.e'^q.u_i'^{q'} \\
&= [\![\sigma.\beta.a.(t'^\beta, s^{\neg\beta}).\alpha'^1.a'^1.(t'^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^{n'}.a'^{n'}.(t'^\beta, s'^{n'^{\neg\beta}}) \\
&\qquad\qquad .\gamma'^1.b'^1.u'^1.\cdots.\gamma'^{m'}.b'^{m'}.u'^{m'}]\!]_i
\end{aligned}$$

for some $e^1.u_i^1.\cdots.e^q.u_i^q$ and $e'^1.u_i'^1.\cdots.e'^q.u_i'^{q'}$. The inequality $\left[\sigma^1\right]_i \neq \left[\sigma^2\right]_i$ gives

$$e^1.u_i^1.\cdots.e^q.u_i^q \neq e^1.u_i^1.\cdots.e^q.u_i^q\ ,$$

thus yielding the result. □

LEMMA 7.2

For all simple IPIOA P*, the triple* $(\mathcal{R}_I^{\text{INV}}, \{(\pi^{\alpha,[\![\cdot]\!]}, \pi^{\neg\alpha,[\![\cdot]\!]})\}_{\alpha\in\text{ACTIONS}}, |[\![\cdot]\!]|)$ *is an independence structure.*

We prove this lemma by proving a more general version, in which we consider the full-communication of any projection $[\cdot]$ such that:

$$\begin{aligned} &a \in \text{ACTLAB}_i \iff \\ &\left[\sigma.\beta.a.(s'^{\beta}, s^{\neg\beta}).\alpha^1.a^1.(s'^{\beta}, s^{1^{\neg\beta}}).\cdots.\alpha^n.a^n.(s'^{\beta}, s^{n^{\neg\beta}})\right]_i \\ &\qquad\qquad \neq \left[\sigma.\alpha^1.a^1.(s^{\beta}, s^{1^{\neg\beta}}).\cdots.\alpha^n.a^n.(s^{\beta}, s^{n^{\neg\beta}})\right]_i\ . \end{aligned} \tag{C.6}$$

That is, the projection only allows to see the changes introduced by labels in ACTLAB_i. (A projection that does *not* comply with this property is the visible prefix property —see Def. 6.4).

In addition, we require

$$N^{\text{ACTLAB}_i}(\sigma) \neq N^{\text{ACTLAB}_i}(\sigma') \implies [\sigma]_i \neq \left[\sigma'\right]_i \tag{C.7}$$

where $N^{\text{ACTLAB}_i}(\sigma)$ is the amount of labels in ACTLAB_i, that is,

- $N^{\text{ACTLAB}_i}(\text{INIT}) = 0$
- $N^{\text{ACTLAB}_i}(\sigma.\alpha.a.s) = N^{\text{ACTLAB}_i}(\sigma) + 1$ if $a \in \text{ACTLAB}_i$
- $N^{\text{ACTLAB}_i}(\sigma.\alpha.a.s) = N^{\text{ACTLAB}_i}(\sigma)$, otherwise.

Moreover, we also generalize the independence relation. We consider any $\mathcal{R}_I$ in which, if an atom can see two labels a, a' output by two actions α, α', then α and α' are not independent. In symbols:

$$a, a' \in \text{ACTLAB}_i \wedge \alpha(s, a, t) > 0 \wedge \alpha'(s', a', t') > 0 \implies \alpha\ \not\mathcal{R}_I\ \alpha' \tag{C.8}$$

Properties (C.6) and (C.8) imply that, if we move an action β accross independent actions α^k, then the projection is not affected:

$$\begin{aligned} &\left[\sigma.\beta.a.(s'^{\beta}, s^{\neg\beta}).\alpha^1.a^1.(s'^{\beta}, s^{1^{\neg\beta}}).\cdots.\alpha^n.a^n.(s'^{\beta}, s^{n^{\neg\beta}})\right]_i \\ &\qquad = \left[\sigma.\alpha^1.a^1.(s^{\beta}, s^{1^{\neg\beta}}).\cdots.(s^{\beta}, s^{m^{\neg\beta}}).\beta.a\cdots.\alpha^n.a^n.(s'^{\beta}, s^{n^{\neg\beta}})\right]_i \end{aligned} \tag{C.9}$$

(with possibly $n = m$). This property has a simple explanation: if $a \in \text{ACTLAB}_i$ then (by (C.8)) $a^k \notin \text{ACTLAB}_i$ for all k, and so both projections in Eqn. (C.9) are equal to $\left[\sigma.\beta.a.(s'^{\beta}, s^{\beta})\right]_i$. If $a \notin \text{ACTLAB}_i$, both projections are equal to

$$\left[\sigma.\alpha^1.a^1.(s^{\beta}, s^{1^{\neg\beta}}).\cdots.(s^{\beta}, s^{m^{\neg\beta}}).\cdots.\alpha^n.a^n.(s^{\beta}, s^{n^{\neg\beta}})\right]_i\ .$$

This property also holds for the full-communication version of $[\cdot]$, as stated in the following lemma.

Lemma C.1. *Let $\mathcal{R}_I$ and $|[\cdot]|$ be such that (C.6) and (C.8) hold. If $\alpha^k \mathcal{R}_I \beta$ for all k, then*

$$\left|\left[\sigma.\beta.a.(s'^{\beta}, s^{\neg\beta}).\alpha^1.a^1.(s'^{\beta}, s^{1^{\neg\beta}}).\cdots.\alpha^n.a^n.(s'^{\beta}, s^{n^{\neg\beta}})\right]\right|_i = \left|\left[\sigma.\alpha^1.a^1.(s^{\beta}, s^{1^{\neg\beta}}).\cdots.(s^{\beta}, s^{m^{\neg\beta}}).\beta.a\cdots.\alpha^n.a^n.(s'^{\beta}, s^{n^{\neg\beta}})\right]\right|_i \quad \text{(C.10)}$$

Proof. It suffices to prove

$$\left|\left[\sigma.\beta.a.(s'^{\beta}, s^{\neg\beta}).\alpha^1.a^1.(s'^{\beta}, s^{1^{\neg\beta}}).\cdots.\alpha^n.a^n.(s'^{\beta}, s^{n^{\neg\beta}})\right]\right|_i = \left|\left[\sigma.\alpha^1.a^1.(s^{\beta}, s^{1^{\neg\beta}}).\beta.a.(s'^{\beta}, s^{1^{\neg\beta}})\cdots.\alpha^n.a^n.(s'^{\beta}, s^{n^{\neg\beta}})\right]\right|_i \quad \text{(C.11)}$$

(Equation (C.10) follows from repeated application of Eqn. (C.11).)

We prove Eqn. (C.11) by induction on $n-1$, that is, the number of α^k after the action α^1 that is swapped with β.

If $n-1=0$ we have three cases: if $a, a^1 \notin \text{ACTLAB}_i$, by (C.6), we have

$$\left|\left[\sigma.\beta.a.(s'^{\beta}, s^{\neg\beta}).\alpha^1.a^1.(s'^{\beta}, s^{1^{\neg\beta}})\right]\right|_i = |[\sigma]|_i = \left|\left[\sigma.\alpha^1.a^1.(s^{\beta}, s^{1^{\neg\beta}}).\beta.a.(s'^{\beta}, s^{1^{\neg\beta}})\right]\right|_i$$

If $a \in \text{ACTLAB}_i$ then, by (C.8), we have $a^1 \notin \text{ACTLAB}_i$. Moreover, $a^1 \notin \text{ACTLAB}_{\text{ACTIVE}(\beta)}$, since $a \in \text{ACTLAB}_{\text{ACTIVE}(\beta)}$. Therefore,

$$\begin{aligned}
&\left|\left[\sigma.\beta.a.(s'^{\beta}, s^{\neg\beta}).\alpha^1.a^1.(s'^{\beta}, s^{1^{\neg\beta}})\right]\right|_i \\
&= \left(\left[\sigma.\beta.a.(s'^{\beta}, s^{\neg\beta})\right]_i, |[\sigma]|_{\text{ACTIVE}(\beta)}\right) \\
&= \left(\left[\sigma.\alpha^1.a^1.(s^{\beta}, s^{1^{\neg\beta}}).\beta.a.(s'^{\beta}, s^{\neg\beta})\right]_i, \left|\left[\sigma.\alpha^1.a^1.(s^{\beta}, s^{1^{\neg\beta}})\right]\right|_{\text{ACTIVE}(\beta)}\right) \\
&= \left|\left[\sigma.(s^{\beta}, s^{\neg\beta}).\alpha^1.a^1.(s^{\beta}, s^{1^{\neg\beta}}).\beta.a.(s'^{\beta}, s^{1^{\neg\beta}})\right]\right|_i
\end{aligned}$$

If $a \notin \text{ACTLAB}_i$, $a^1 \in \text{ACTLAB}_i$, then

$$\begin{aligned}
&\left|\left[\sigma.\beta.a.(s'^{\beta}, s^{\neg\beta}).\alpha^1.a^1.(s'^{\beta}, s^{1^{\neg\beta}})\right]\right|_i \\
&= \left(\left[\sigma.\beta.a.(s'^{\beta}, s^{\neg\beta}).\alpha^1.a^1.(s'^{\beta}, s^{1^{\neg\beta}})\right]_i, \left|[\sigma.\beta.a.(s'^{\beta}, s^{\neg\beta})]\right|_{\text{ACTIVE}(\alpha^1)}\right) \\
&= \left(\left[\sigma.\alpha^1.a^1.(s^{\beta}, s^{1^{\neg\beta}})\right]_i, |[\sigma]|_{\text{ACTIVE}(\alpha^1)}\right) \\
&= \left|\left[\sigma.\alpha^1.a^1.(s^{\beta}, s^{1^{\neg\beta}})\right]\right|_i \\
&= \left|\left[\sigma.\alpha^1.a^1.(s^{\beta}, s^{1^{\neg\beta}}).\beta.a.(s'^{\beta}, s^{1^{\neg\beta}})\right]\right|_i
\end{aligned}$$

If $n-1>0$, then we have two cases. If $a^n \notin \text{ACTLAB}_i$, then

$$\left|\left[\sigma.\beta.a.(s'^{\beta}, s^{\neg\beta}).\alpha^1.a^1.(s'^{\beta}, s^{1^{\neg\beta}}).\cdots.\alpha^n.a^n.(s'^{\beta}, s^{n^{\neg\beta}})\right]\right|_i = \left|\left[\sigma.\beta.a.(s'^{\beta}, s^{\neg\beta}).\alpha^1.a^1.(s'^{\beta}, s^{1^{\neg\beta}}).\cdots.\alpha^{n-1}.a^{n-1}.(s'^{\beta}, s^{n-1^{\neg\beta}})\right]\right|_i$$

and

$$\left|\left[\sigma.\alpha^1.a^1.(s^{\beta}, s^{1^{\neg\beta}}).\beta.a.(s'^{\beta}, s^{1^{\neg\beta}})\cdots.\alpha^n.a^n.(s'^{\beta}, s^{n^{\neg\beta}})\right]\right|_i = \left|\left[\sigma.\alpha^1.a^1.(s^{\beta}, s^{1^{\neg\beta}}).\beta.a.(s'^{\beta}, s^{1^{\neg\beta}})\cdots.\alpha^{n-1}.a^{n-1}.(s'^{\beta}, s^{n-1^{\neg\beta}})\right]\right|_i .$$

Hence, if $a^n \notin \text{ACTLAB}_i$ then the property holds by inductive hypothesis.

If $a^n \in \text{ACTLAB}_i$:

$$
\begin{aligned}
&\left|\left[\sigma.\beta.a.(s'^{\beta}, s^{\neg\beta}).\alpha^1.a^1.(s'^{\beta}, s^{1\neg\beta}).\cdots.\alpha^n.a^n.(s'^{\beta}, s^{n\neg\beta})\right]\right|_i \\
&= \Big(\left[\sigma.\beta.a.(s'^{\beta}, s^{\neg\beta}).\alpha^1.a^1.(s'^{\beta}, s^{1\neg\beta}).\cdots.\alpha^n.a^n.(s'^{\beta}, s^{n\neg\beta})\right]_i , \\
&\left|\left[\sigma.\beta.a.(s'^{\beta}, s^{\neg\beta}).\alpha^1.a^1.(s'^{\beta}, s^{1\neg\beta}).\cdots.\alpha^{n-1}.a^{n-1}.(s'^{\beta}, s^{n-1\neg\beta})\right]\right|_{\text{ACTIVE}(\alpha^n)} \Big) \\
&= \{ \text{ The property holds for } [\cdot]. \text{ Inductive hypothesis } \} \\
&\quad \Big(\left[\sigma.\alpha^1.a^1.(s^{\beta}, s^{1\neg\beta}).\beta.a.(s'^{\beta}, s^{1\neg\beta})\cdots.\alpha^n.a^n.(s'^{\beta}, s^{n\neg\beta})\right]_i , \\
&\left|\left[\sigma.\alpha^1.a^1.(s^{\beta}, s^{1\neg\beta}).\beta.a.(s'^{\beta}, s^{1\neg\beta})\cdots.\alpha^{n-1}.a^{n-1}.(s'^{\beta}, s^{n-1\neg\beta})\right]\right|_{\text{ACTIVE}(\alpha^n)} \Big) \\
&= \left|\left[\sigma.\alpha^1.a^1.(s^{\beta}, s^{1\neg\beta}).\beta.a.(s'^{\beta}, s^{1\neg\beta})\cdots.\alpha^n.a^n.(s'^{\beta}, s^{n\neg\beta})\right]\right|_i
\end{aligned}
$$

□

The general version of Lemma 7.2 is the lemma below.

LEMMA C.2. *Let* $(\mathcal{R}_I^{\text{INV}}, \{(\pi^\alpha, \pi^{\neg\alpha})\}_{\alpha\in\text{ACTIONS}}, [\cdot])$ *be an independence structure. If* $\mathcal{R}_I$ *and* $|[\cdot]|$ *comply with (C.6), (C.7) and (C.8), then* $(\mathcal{R}_I^{\text{INV}}, \{(\pi^\alpha, \pi^{\neg\alpha})\}_{\alpha\in\text{ACTIONS}}, |[\cdot]|)$ *is an independence structure.*

Proof. We check the properties related to the projection (namely, properties *(1)*–*(8)*) in Def. 7.5. The other properties (which do not involve the projection) hold since $(\mathcal{R}_I^{\text{INV}}, \{(\pi^\alpha, \pi^{\neg\alpha})\}_\alpha, [\cdot])$ is an independence structure.

For property *(6)*, let $A_i \in \text{AFFECT}(\beta)$. We have to prove $\left|\left[\sigma.\alpha^1.a^1.s^1.\cdots.\alpha^n.a^n.s^n\right]\right|_i = |[\sigma]|_i$. By (C.8) we have $a^k \notin \text{ACTLAB}_i$, and so (C.6) yields

$$\left[\sigma.\alpha^1.a^1.s^1.\cdots.\alpha^k.a^k.s^k\right]_i = [\sigma]_i$$

for all $k \leqslant n$. The definition of $|[\cdot]|$ gives

$$\left|\left[\sigma.\alpha^1.a^1.s^1.\cdots.\alpha^{k-1}.a^{k-1}.s^k\right]\right|_i = |[\sigma]|_i$$

for all $k \leqslant n$, and in particular n.

Property (7) We prove property *(7)* by induction on $n + n'$. If $n + n' = 0$, then the implication holds trivially. If $n + n' > 0$ assume $n > 0$ (the case $n' > 0$ is symmetrical). We have two cases.

- Case $a^n \in \text{ACTLAB}_i$.

If there exists no m such that $a'^m \in \text{ACTLAB}_i$, then we are able to we prove the consequent of Eqn. (7.4), namely

$$
\begin{aligned}
&\left|\left[\sigma.\beta.a.(t^{\beta}, s^{\neg\beta}).\alpha^1.a^1.(t^{\beta}, s^{1\neg\beta})\cdots\alpha^n.a^n.(t^{\beta}, s^{n\neg\beta})\right]\right|_i \\
&\neq \left|\left[\sigma.\beta.a.(t'^{\beta}, s^{\neg\beta}).\alpha'^1.a'^1.(t'^{\beta}, s'^{1\neg\beta})\cdots\alpha'^{n'}.a'^{n'}.(t'^{\beta}, s'^{n'\neg\beta})\right]\right|_i ,
\end{aligned}
\tag{C.12}
$$

The proof of this inequality follows from these two equalities:

$$
\begin{aligned}
&\left|\left[\sigma.\beta.a.(t^{\beta}, s^{\neg\beta}).\alpha^1.a^1.(t^{\beta}, s^{1\neg\beta})\cdots\alpha^n.a^n.(t^{\beta}, s^{n\neg\beta})\right]\right|_i \\
&\qquad = \left([\sigma]_i .\cdots.a^n.\pi_i(t^{\beta}, s^{n\neg\beta})\right), \sigma^{|[\cdot]|}_{\text{ACTIVE}(\alpha^n)} \Big) ,
\end{aligned}
\tag{C.13}
$$

(where $\sigma_j^{|[\cdot]|} \in \text{LOCALPATHS}^{|[\cdot]|}_{\text{ACTIVE}(\alpha^n)}$) and

$$\left|\left[\sigma.\beta.a.(t'^{\beta}, s^{\neg\beta}).\alpha'^1.a'^1.(t'^{\beta}, s'^{1\neg\beta})\cdots\alpha'^{n'}.a'^{n'}.(t'^{\beta}, s'^{n'\neg\beta})\right]\right|_i = \left([\sigma]_i , \sigma'^{|[\cdot]|}_{\text{ACTIVE}(\alpha^n)} \right) .$$

Hence, (C.12) follows from $[\sigma]_i \neq [\sigma]_i.\cdots.a^n.\pi_i(t^\beta, s^{n^{\neg\beta}})$.

Now we explore the case in which m exists. W.l.o.g., let m be the greatest number such that $a'^m \in \text{ACTLAB}_i$, and so

$$\bigwedge_{q=m+1}^{n'} a'^q \notin \text{ACTLAB}_i \,. \tag{C.14}$$

By definition of $|[\cdot]|$, this implies

$$\begin{aligned} &\left|\left[\sigma.\alpha'^1.a'^1.(s^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^{n'}.a'^{n'}.(s^\beta, s'^{n'^{\neg\beta}})\right]\right|_i \\ = &\left|\left[\sigma.\alpha'^1.a'^1.(s^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^{m}.a'^{m}.(s^\beta, s'^{m^{\neg\beta}})\right]\right|_i \end{aligned} \tag{C.15}$$

The definition of $|[\cdot]|$ yields:

$$\begin{aligned} &|\overbrace{[\sigma.\alpha^1.a^1.(s^\beta, s^{1^{\neg\beta}}) \cdots \alpha^n.a^n.(s^\beta, s^{n^{\neg\beta}})]}^{\sigma^1}|_i \\ = &\left(\left[\sigma^1\right]_i, \left|\left[\sigma.\alpha^1.a^1.(s^\beta, s^{1^{\neg\beta}}) \cdots \alpha^{n-1}.a^{n-1}.(s^\beta, s^{n-1^{\neg\beta}})\right]\right|_{\text{ACTIVE}(\alpha^n)}\right) \end{aligned} \tag{C.16}$$

and

$$\begin{aligned} &|\overbrace{[\sigma.\alpha'^1.a'^1.(s^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^{m}.a'^{m}.(s^\beta, s'^{m^{\neg\beta}})]}^{\sigma^2}|_i \\ = &\left(\left[\sigma^2\right]_i, \left|\left[\sigma.\alpha'^1.a'^1.(s^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^{m-1}.a'^{m-1}.(s^\beta, s'^{m-1^{\neg\beta}})\right]\right|_{\text{ACTIVE}(\alpha'^m)}\right) \end{aligned} \tag{C.17}$$

Hence, if $a^n \in \text{ACTLAB}_i$, then

$$\begin{aligned} &\left|\left[\sigma^1\right]\right|_i \neq \left|\left[\sigma.\alpha'^1.a'^1.(s^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^{n'}.a'^{n'}.(s^\beta, s'^{n'^{\neg\beta}})\right]\right|_i \\ \Longrightarrow\ &\{\ \text{Equation (C.15)}\ \} \\ &\left|\left[\sigma^1\right]\right|_i \neq \left|\left[\sigma^2\right]\right|_i \\ \Longrightarrow\ &\{\ \text{Equations C.16 and C.17}\ \} \\ &\left(\left[\sigma^1\right]_i, \left|\left[\sigma^1{\downarrow_{-1}}\right]\right|_{\text{ACTIVE}(\alpha^n)}\right) \neq \left(\left[\sigma^2\right]_i, \left|\left[\sigma^2{\downarrow_{-1}}\right]\right|_{\text{ACTIVE}(\alpha'^m)}\right) \\ \Longrightarrow\ &\{\ \text{Let } \sigma'^1 = \sigma.\beta.a.(t^\beta, s^{\neg\beta}).\alpha^1.a^1.(t^\beta, s^{1^{\neg\beta}}) \cdots \alpha^n.a^n.(t^\beta, s^{n^{\neg\beta}}) \text{ and} \\ &\quad \sigma'^2 = \sigma.\beta.a.(t'^\beta, s^{\neg\beta}).\alpha'^1.a'^1.(t'^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^m.a'^m.(t'^\beta, s'^{m^{\neg\beta}})\,. \\ &\quad \text{If } \text{ACTIVE}(\alpha^n) \neq \text{ACTIVE}(\alpha'^m), \text{ then } \left|\left[\sigma'^1{\downarrow_{-1}}\right]\right|_{\text{ACTIVE}(\alpha^n)} \neq \left|\left[\sigma'^2{\downarrow_{-1}}\right]\right|_{\text{ACTIVE}(\alpha'^m)}\,. \\ &\quad \text{Otherwise, apply the fact that property } \textit{(7)} \text{ holds for } [\cdot].\ \} \\ &\left(\left[\sigma'^1\right]_i, \left|\left[\sigma^1{\downarrow_{-1}}\right]\right|_{\text{ACTIVE}(\alpha^n)}\right) \neq \left(\left[\sigma'^2\right]_i, \left|\left[\sigma^2{\downarrow_{-1}}\right]\right|_{\text{ACTIVE}(\alpha'^m)}\right) \\ \Longrightarrow\ &\{\ \text{If } \text{ACTIVE}(\alpha^n) \neq \text{ACTIVE}(\alpha'^m), \text{ then } \left|\left[\sigma'^1{\downarrow_{-1}}\right]\right|_{\text{ACTIVE}(\alpha^n)} \neq \left|\left[\sigma'^2{\downarrow_{-1}}\right]\right|_{\text{ACTIVE}(\alpha'^m)}\,. \\ &\quad \text{Otherwise, since } \text{LEN}(\sigma'^1{\downarrow_{-1}}) + \text{LEN}(\sigma'^2{\downarrow_{-1}}) = (n-1)+(m-1) < n+n', \\ &\quad \text{the inductive hypothesis applies}\ \} \\ &\left(\left[\sigma'^1\right]_i, \left|\left[\sigma'^1{\downarrow_{-1}}\right]\right|_{\text{ACTIVE}(\alpha^n)}\right) \neq \left(\left[\sigma'^2\right]_i, \left|\left[\sigma'^2{\downarrow_{-1}}\right]\right|_{\text{ACTIVE}(\alpha'^m)}\right) \\ \Longrightarrow\ &\{\ \text{Definition of } |[\cdot]|\ \} \\ &\left|\left[\sigma'^1\right]\right|_i \neq \left|\left[\sigma'^2\right]\right|_i \\ \Longrightarrow\ &\{\ \text{By (C.14), definition of } |[\cdot]|\ \} \\ &\left|\left[\sigma'^1\right]\right|_i \neq \left|\left[\sigma.\beta.a.(t'^\beta, s^{\neg\beta}).\alpha'^1.a'^1.(t'^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^{n'}.a'^{n'}.(t'^\beta, s'^{n'^{\neg\beta}})\right]\right|_i \end{aligned}$$

This implication is (7.4).

- Case $a^n \notin \text{ACTLAB}_i$

$$\|\overbrace{[\sigma.\alpha^1.a^1.(s^\beta, s^{1^{\neg\beta}}) \cdots \alpha^n.a^n.(s^\beta, s^{n^{\neg\beta}})]}^{\sigma^1}\|_i = \left|\left[\sigma^{1}{}_{\downarrow_{-1}}\right]\right|_i \tag{C.18}$$

Then,

$$\begin{aligned}
& \left|\left[\sigma^1\right]\right|_i \neq \left|\left[\sigma.\alpha'^1.a'^1.(s^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^{n'}.a'^{n'}.(s^\beta, s'^{n'^{\neg\beta}})\right]\right|_i \\
\Longrightarrow\ & \{\ \text{Equation (C.18)}\ \} \\
& \left|\left[\sigma^{1}{}_{\downarrow_{-1}}\right]\right|_i \neq \left|\left[\sigma.\alpha'^1.a'^1.(s^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^{n'}.a'^{n'}.(s^\beta, s'^{n'^{\neg\beta}})\right]\right|_i \\
\Longrightarrow\ & \{\ \text{Inductive hypothesis}\ \} \\
& \quad \left|\left[\sigma.\beta.a.(t^\beta, s^{\neg\beta}).\alpha^1.a^1.(t^\beta, s^{1^{\neg\beta}}) \cdots \alpha^{n-1}.a^{n-1}.(t^\beta, s^{n-1^{\neg\beta}})\right]\right|_i \\
& \neq \left|\left[\sigma.\beta.a.(t'^\beta, s^{\neg\beta}).\alpha'^1.a'^1.(t'^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^{n'}.a'^{n'}.(t'^\beta, s'^{n'^{\neg\beta}})\right]\right|_i \\
\Longrightarrow\ & \{\ a^n \notin \text{ACTLAB}_i\ \} \\
& \quad \left|\left[\sigma.\beta.a.(t^\beta, s^{\neg\beta}).\alpha^1.a^1.(t^\beta, s^{1^{\neg\beta}}) \cdots \alpha^{n}.a^{n}.(t^\beta, s^{n^{\neg\beta}})\right]\right|_i \\
& \neq \left|\left[\sigma.\beta.a.(t'^\beta, s^{\neg\beta}).\alpha'^1.a'^1.(t'^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^{n'}.a'^{n'}.(t'^\beta, s'^{n'^{\neg\beta}})\right]\right|_i
\end{aligned}$$

Property (8)

For property *(8)*, let

$$\begin{aligned}
\sigma^{1'} &= \sigma.\beta.a.(t^\beta, s^{\neg\beta}).\alpha^1.a^1.(t^\beta, s^{1^{\neg\beta}}) \cdots \alpha^n.a^n.t^\beta, s^{n^{\neg\beta}}).\gamma^1.b^1.u^1.\cdots.\gamma^m.b^m.u^m \\
\sigma^{2'} &= \sigma.\beta.a.(t'^\beta, s^{\neg\beta}).\alpha'^1.a'^1.(t'^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^{n'}.a'^{n'}.(t'^\beta, s'^{n'^{\neg\beta}}) \\
&\quad .\gamma'^1.b'^1.u'^1.\cdots.\gamma'^{m'}.b'^{m'}.u'^{m'}
\end{aligned}$$

We prove the contrapositive of property *(8)*, namely,

$$\left|\left[\sigma^{1'}\right]\right|_i = \left|\left[\sigma^{2'}\right]\right|_i \implies \left|\left[\sigma^1\right]\right|_i = \left|\left[\sigma^2\right]\right|_i\ .$$

This implication is proven by induction on $m + m'$. If $m + m' = 0$, then for all A_i we have

$$\overbrace{\left|\left[\sigma^{1'}\right]\right|_i = \left|\left[\sigma^{2'}\right]\right|_i}^{(1)} \implies$$

$$\begin{aligned}
& \left|\left[\sigma^1\right]\right|_i \\
=\ & \{\ \text{Lemma C.1}\ \} \\
& \left|\left[\sigma^{1'}\right]\right|_i \\
=\ & \{\ (1)\ \} \\
& \left|\left[\sigma^{2'}\right]\right|_i \\
=\ & \{\ \text{Lemma C.1}\ \} \\
& \left|\left[\sigma^2\right]\right|_i
\end{aligned}$$

For the inductive step, suppose $m + m' > 0$. In case there is no q such that $b^q \in \text{ACTLAB}_i$ and there is no q' such that $b'^{q'} \in \text{ACTLAB}_i$ then

$$\begin{aligned}\left|\left[\sigma^1\right]\right|_i &= \left|\left[\sigma.\alpha^1.a^1.(s^\beta, s^{1^{\neg\beta}}) \cdots \alpha^n.a^n.(s^\beta, s^{n^{\neg\beta}}).\beta.a.(t^\beta, s^{n^{\neg\beta}})\right]\right|_i \\ &= \left|\left[\sigma.\beta.a.(t^\beta, s^{\neg\beta}).\alpha^1.a^1.(t^\beta, s^{1^{\neg\beta}}).\cdots.\alpha^n.a^n.(t^\beta, s^{n^{\neg\beta}})\right]\right|_i \\ &= \left|\left[\sigma^{1'}\right]\right|_i\end{aligned}$$

and

$$\begin{aligned}\left|\left[\sigma^2\right]\right|_i &= \left|\left[\sigma.\alpha'^1.a'^1.(s^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^{n'}.a'^{n'}.(s^\beta, s'^{n'^{\neg\beta}}).\beta.a.(t'^\beta, s'^{n'^{\neg\beta}})\right]\right|_i \\ &= \left|\left[\sigma.\beta.a.(t'^\beta, s^{\neg\beta}).\alpha'^1.a'^1.(t'^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^{n'}.a'^{n'}.(t'^\beta, s'^{n'^{\neg\beta}})\right]\right|_i \\ &= \left|\left[\sigma^{2'}\right]\right|_i\end{aligned}$$

Hence, we have $\left|\left[\sigma^{1'}\right]\right|_i = \left|\left[\sigma^{2'}\right]\right|_i \implies \left|\left[\sigma^1\right]\right|_i = \left|\left[\sigma^2\right]\right|_i$, as desired.

We finish the proof by considering the case in which there exists q such that $b^q \in \text{ACTLAB}_i$ (the case in which $b'^{q'} \in \text{ACTLAB}_i$ for some q' is symmetrical). Let q be the greatest number such that $b^q \in \text{ACTLAB}_i$, and let σ^q

$$\sigma^q = \sigma.\beta.a.(t^\beta, s^{\neg\beta}).\alpha^1.a^1.(t^\beta, s^{1^{\neg\beta}}) \cdots \alpha^n.a^n.t^\beta, s^{n^{\neg\beta}}).\gamma^1.b^1.u^1.\cdots.\gamma^q.b^q.u^q\ .$$

First, we prove:

Claim.

$$\left(b^q \in \text{ACTLAB}_i \wedge \left|\left[\sigma^{1'}\right]\right|_i = \left|\left[\sigma^{2'}\right]\right|_i\right) \implies \exists q' : \left(a'^{q'} \in \text{ACTLAB}_i \vee b'^{q'} \in \text{ACTLAB}_i\right) \quad \text{(C.19)}$$

Proof. Suppose, towards a contradiction, that the antecedent of this implication does hold, and the consequent does not. Then,

$$\begin{aligned}&\left|\left[\sigma^{1'}\right]\right|_i \\ &= \{\ q \text{ is the greatest number such that } b^q \in \text{ACTLAB}_i\ \} \\ &\quad \left|\left[\sigma.\beta.a.(t^\beta, s^{\neg\beta}).\alpha^1.a^1.(t^\beta, s^{1^{\neg\beta}}) \cdots \alpha^n.a^n.t^\beta, s^{n^{\neg\beta}}).\gamma^1.b^1.u^1.\cdots.\gamma^q.b^q.u^q\right]\right|_i \quad \text{(C.20)} \\ &= \{\ b^q \in \text{ACTLAB}_i\ \} \\ &\quad (\ [\sigma^q]_i\ ,\ |[\sigma^{q}{}_{\downarrow -1}]|_{\text{ACTIVE}(\gamma^q)}\ )\end{aligned}$$

If $a \in \text{ACTLAB}_i$, then

$$\begin{aligned}&\left|\left[\sigma^{2'}\right]\right|_i \\ &= \{\ a'^k \notin \text{ACTLAB}_i \wedge b'^k \notin \text{ACTLAB}_i\ \} \\ &\quad \left|\left[\sigma.\beta.a.(t'^\beta, s^{\neg\beta})\right]\right|_i \\ &= \{\ a \in \text{ACTLAB}_i\ \} \\ &\quad (\ \left[\sigma.\beta.a.(t'^\beta, s^{\neg\beta})\right]_i\ ,\ |[\sigma]|_i\ )\end{aligned}$$

From (C.20), we have that $\left|\left[\sigma^{1'}\right]\right|_i = \left|\left[\sigma^{2'}\right]\right|_i$ implies $\left[\sigma.\beta.a.(t'^\beta, s^{\neg\beta})\right]_i = [\sigma^q]_i$, thus contradicting (C.7), since

$$N^{\text{ACTLAB}_i}(\sigma^{2'}) = \overbrace{1}^{a} + N^{\text{ACTLAB}_i}(\sigma) < \overbrace{1}^{a} + \overbrace{1}^{b^q} + N^{\text{ACTLAB}_i}(\sigma) \leqslant N^{\text{ACTLAB}_i}(\sigma^q)\ .$$

If $a \notin \text{ACTLAB}_i$ then, by (C.20) and $\left|\left[\sigma^{1'}\right]\right|_i = \left|\left[\sigma^{2'}\right]\right|_i$ we have $\left|\left[\sigma^{2'}\right]\right|_i \neq \text{INIT}_i$. Hence, $\left|\left[\sigma^{2'}\right]\right|_i = ([\sigma_{\downarrow z}]_i , |[\sigma_{\downarrow z-1}]|_{\text{ACTIVE}(\sigma\langle z\rangle)})$ for some z. Similarly as in the case $a \in \text{ACTLAB}_i$, we have that $\left|\left[\sigma^{1'}\right]\right|_i = \left|\left[\sigma^{2'}\right]\right|_i$ implies $[\sigma_{\downarrow z}]_i = [\sigma^q]_i$, and so

$$N^{\text{ACTLAB}_i}(\sigma^{2'}) = N^{\text{ACTLAB}_i}(\sigma) < \overbrace{1}^{b^q} + N^{\text{ACTLAB}_i}(\sigma) \leqslant N^{\text{ACTLAB}_i}(\sigma^q) ,$$

thus reaching a contradiction for (C.7). □

End of the claim's proof

In conclusion, in case $b^q \in \text{ACTLAB}_i$, we have that either $a'^{q'} \in \text{ACTLAB}_i$ or $b'^{q'} \in \text{ACTLAB}_i$ for some q'. Hence, we split this case into two subcases. We start with the subcase $b'^{q'} \in \text{ACTLAB}_i$. W.l.o.g., assume q' is the greatest number such that $b'^{q'} \in \text{ACTLAB}_i$.

If $b^q \in \text{ACTLAB}_i$, $b'^{q'} \in \text{ACTLAB}_i$:

$$\begin{aligned}
&\left|\left[\sigma^{1'}\right]\right|_i = \left|\left[\sigma^{2'}\right]\right|_i \\
\Longrightarrow\ &\{\ q^z, q'^{z'} \notin \text{ACTLAB}_i \text{ for all } z > q,\, z' > q' \ \} \\
&\quad ([\sigma^q]_i , |[\sigma^q{}_{\downarrow -1}]|_{\text{ACTIVE}(\gamma^q)}) \\
&= ([\overbrace{\sigma.\beta.a.(t'^{\beta}, s^{\neg\beta}) \cdots \alpha'^{n'}.a'^{n'}.(t'^{\beta}, s'^{n'^{\neg\beta}}).\cdots .\gamma'^{q'}.b'^{q'}.u'^{q'}}^{\sigma^{q'}}]_i , \\
&\qquad\qquad \left|\left[\sigma^{q'}{}_{\downarrow -1}\right]\right|_{\text{ACTIVE}(\gamma'^{q'})}) \\
\Longrightarrow\ &\{\ \text{Property } (8) \text{ holds for } [\cdot], \text{ inductive hypothesis } \} \\
&([\overbrace{\sigma \cdots \alpha^n.a^n.(s^{\beta}, s^{n^{\neg\beta}}).\beta.a.(t^{\beta}, s^{n^{\neg\beta}}) \cdots .\gamma^q.b^q.u^q}^{\sigma^{1^q}}]_i , \left|\left[\sigma^{1^q}{}_{\downarrow -1}\right]\right|_{\text{ACTIVE}(\gamma^q)}) \\
&= ([\overbrace{\sigma \cdots \alpha'^{n'}.a'^{n'}.(s^{\beta}, s'^{n'^{\neg\beta}}).\beta.a.(t'^{\beta}, s'^{n'^{\neg\beta}}) \cdots .\gamma'^{q'}.b'^{q'}.u'^{q'}}^{\sigma^{2^{q'}}}]_i , \\
&\qquad\qquad \left|\left[\sigma^{2^{q'}}{}_{\downarrow -1}\right]\right|_{\text{ACTIVE}(\gamma'^{q'})}) \\
\Longrightarrow\ &\{\ q^z, q'^{z'} \notin \text{ACTLAB}_i \text{ for all } z > q,\, z' > q' \ \} \\
&|[\sigma^1]|_i = |[\sigma^2]|_i
\end{aligned}$$

If $b^q \in \text{ACTLAB}_i$, $a'^{q'} \in \text{ACTLAB}_i$ and there is no z such that $b'^z \in \text{ACTLAB}_i$, assume that q' is the greatest number such that $a'^{q'} \in \text{ACTLAB}_i$. Then:

$$\begin{aligned}
&\left|\left[\sigma^{1'}\right]\right|_i = \left|\left[\sigma^{2'}\right]\right|_i \\
\Longrightarrow\ &\{\ \nexists z : b'^z \in \text{ACTLAB}_i \text{ and } a'^{z'} \notin \text{ACTLAB}_i \text{ for all } z' > q' \ \} \\
&|[\sigma^q]|_i = |[\overbrace{\sigma.\beta.a.(t'^{\beta}, s^{\neg\beta}).\alpha'^1.a'^1.(t'^{\beta}, s'^{1^{\neg\beta}}) \cdots \alpha'^{q'}.a'^{q'}.(t'^{\beta}, s'^{q'^{\neg\beta}})}^{\sigma^{q'}}]|_i \\
\Longrightarrow\ &\{\ b^q \in \text{ACTLAB}_i,\, a'^{q'} \in \text{ACTLAB}_i \ \} \\
&([\sigma^q]_i , |[\sigma^q{}_{\downarrow -1}]|_{\text{ACTIVE}(\gamma^q)}) = (\left[\sigma^{q'}\right]_i , |[\sigma^q{}_{\downarrow -1}]|_{\text{ACTIVE}(\alpha'^{q'})}) \\
\Longrightarrow\ &\{\ \text{Inductive hypothesis, property } (8) \text{ holds for } [\cdot] \ \} \\
&([\overbrace{\sigma \cdots \alpha^n.a^n.(s^{\beta}, s^{n^{\neg\beta}}).\beta.a.(t^{\beta}, s^{n^{\neg\beta}}) \cdots .\gamma^q.b^q.u^q}^{\sigma^{1^q}}]_i , \left|\left[\sigma^{1^q}{}_{\downarrow -1}\right]\right|_{\text{ACTIVE}(\gamma^q)})
\end{aligned}$$

$$
\begin{aligned}
&= \Big(\Big[\sigma \cdots \alpha'^{q'}.a'^{q'}.(s^{\beta}, s'^{q'^{\neg\beta}}).\beta.a.(t'^{\beta}, s'^{q'^{\neg\beta}})\Big]_i , \\
&\qquad \Big|\Big[\sigma \cdots \alpha'^{q'-1}.a'^{q'-1}.(s^{\beta}, s'^{q'-1^{\neg\beta}}).\beta.a.(t'^{\beta}, s'^{q'-1^{\neg\beta}})\Big]\Big|_{\text{ACTIVE}(\alpha'^{q'})} \Big) \\
\Longrightarrow\ & \{ \text{ Equation (C.9) } \} \\
&\quad \Big(\Big[\sigma^{1^q}\Big]_i , \Big|\Big[\sigma^{1^q}{\downarrow_{-1}}\Big]\Big|_{\text{ACTIVE}(\gamma^q)} \Big) \\
&= \Big(\overbrace{\big[\sigma \cdots \alpha'^{q'-1}.a'^{q'-1}.(s^{\beta}, s'^{q'-1^{\neg\beta}}).\beta.a.(t'^{\beta}, s'^{q'-1^{\neg\beta}}).\alpha'^{q'}.a'^{q'}.(t'^{\beta}, s'^{q'^{\neg\beta}})\big]}^{\sigma^t}_i , \\
&\qquad\qquad \Big|\big[\sigma^t{\downarrow_{-1}}\big]\Big|_{\text{ACTIVE}(\alpha'^{q'})} \Big) \\
\Longrightarrow\ & \{ \ a'^{q'}, b^q \in \text{ACTLAB}_i \ \} \\
&\quad \Big|\Big[\sigma^{1^q}\Big]\Big|_i = \big|[\sigma^t]\big|_i \\
\Longrightarrow\ & \{ \ a'^{z'} \notin \text{ACTLAB}_i \text{ for all } z' > q' \ \} \\
&\quad \Big|\Big[\sigma^{1^q}\Big]\Big|_i = \Big|\Big[\sigma^t.\alpha'^{q+1}.a'^{q+1}.(t'^{\beta}, s'^{q+1^{\neg\beta}}) \cdots \alpha'^{n'}.a'^{n'}\Big]\Big|_i \\
\Longrightarrow\ & \{ \text{ Lemma C.1 } \} \\
&\quad \Big|\Big[\sigma^{1^q}\Big]\Big|_i = \Big|\Big[\sigma \cdots \alpha'^{n'}.a'^{n'}.(s^{\beta}, s'^{n'^{\neg\beta}}).\beta.a.(t'^{\beta}, s'^{n'^{\neg\beta}})\Big]\Big|_i \\
\Longrightarrow\ & \{ \ b^z \notin \text{ACTLAB}_i \text{ for all } z > q,\ b'^{z'} \notin \text{ACTLAB}_i \text{ for all } z' \ \} \\
&\quad \big|[\sigma^1]\big|_i = \big|[\sigma^2]\big|_i
\end{aligned}
$$

□

Now, we can prove Lemma 7.2.

Proof of Lemma 7.2. We prove that $[\![\cdot]\!]$ complies with (C.6), (C.7) and (C.8), and so the result follows from Lemma C.2.

For (C.6), if $a \in \text{ACTLAB}_i$ then

$$
[\![\sigma.\beta.a.(s'^{\beta}, s^{\neg\beta}).\alpha^1.a^1.(s'^{\beta}, s^{1^{\neg\beta}}).\cdots.\alpha^n.a^n.(s'^{\beta}, s^{n^{\neg\beta}})]\!]_i \\
= [\![\sigma]\!]_i.a.\pi_i(s'^{\beta}, s^{\neg\beta}).b^1.s_i^1.\cdots.b^q.s_i^q \quad \text{(C.21)}
$$

for some $b^1.s_i^1.\cdots.b^q.s^q$ (with possibly $q = 0$). More precisely, the sequence $b^1 \cdots b^q$ is the subsequence of $a^1 \cdots a^n$ comprising only the labels in ACTLAB_i.

By definition of $[\![\cdot]\!]$,

$$
[\![\sigma.\alpha^1.a^1.(s^{\beta}, s^{1^{\neg\beta}}).\cdots.\alpha^n.a^n.(s^{\beta}, s^{n^{\neg\beta}})]\!]_i = [\![\sigma]\!]_i.b'^1.s_i'^1.\cdots.b'^{q'}.s'^{q'}
$$

Note that $q = q'$ and $b^k = b'^k$, since $b'^1 \cdots b'^q$ is same subsequence of $a^1 \cdots a^n$ as before.

Hence,

$$
\Big[\sigma.\beta.a.(s'^{\beta}, s^{\neg\beta}).\alpha^1.a^1.(s'^{\beta}, s^{1^{\neg\beta}}).\cdots.\alpha^n.a^n.(s'^{\beta}, s^{n^{\neg\beta}})\Big]_i \\
\neq \Big[\sigma.\alpha^1.a^1.(s^{\beta}, s^{1^{\neg\beta}}).\cdots.\alpha^n.a^n.(s^{\beta}, s^{n^{\neg\beta}})\Big]_i
$$

follows from $a.\pi_i(s'^{\beta}, s^{\neg\beta}).b^1.s_i^1.\cdots.b^q.s_i^q \neq b^1.s_i'^1.\cdots.b^q.s_i'^q$.

If $a \notin \text{ACTLAB}_i$, we start by pointing out that $\text{LAST}(\sigma) = (s^{\beta}, s^{\neg\beta})$. Since the changes to the state of an atom are caused only by the labels in its alphabet, we have

$$
\pi_i(s^{\beta}, s^{\neg\beta}) = \pi_i(s'^{\beta}, s^{\neg\beta}) \,. \quad \text{(C.22)}
$$

We use this equality to show

$$\pi_i(s^{\beta}, s^{k^{\neg\beta}}) = \pi_i(s'^{\beta}, s^{k^{\neg\beta}}) \,. \tag{C.23}$$

If the state of A_i lies in $s^{\neg\beta}$, Eqn. (C.23) holds since in both terms of the equality the portion corresponding to $s^{\neg\beta}$ coincide. If the state of A_i lies in s^{β}, Eqn. (C.23) holds by Eqn. (C.22).

By definition of $[\![\cdot]\!]$, there exist $j_1 \leqslant \cdots \leqslant j_q$ and $s_i^1, \cdots, s_i^q$ such that

$$[\![\sigma.\beta.a.(s'^{\beta}, s^{\neg\beta}).\alpha^1.a^1.(s'^{\beta}, s^{1^{\neg\beta}}).\cdots.\alpha^n.a^n.(s'^{\beta}, s^{n^{\neg\beta}})]\!]_i = [\![\sigma]\!]_i.a^{j_1}.s_i^1.\cdots.a^{j_q}.s_i^q$$

Note that $j_1 \leqslant \cdots \leqslant j_q$ are the indices such that the sequence $a^{j_1}, \cdots, a^{j_q}$ is the subsequence of $a^1, \cdots, a^q$ comprising only the labels in ACTLAB_i. The local state s_i^k is $\pi_i(s'^{\beta}, s^{k^{\neg\beta}})$.

By definition of $[\![\cdot]\!]$, there exist $t_i^1, \cdots, t_i^q$ such that

$$[\![\sigma.\beta.a.(s'^{\beta}, s^{\neg\beta}).\alpha^1.a^1.(s'^{\beta}, s^{1^{\neg\beta}}).\cdots.\alpha^n.a^n.(s'^{\beta}, s^{n^{\neg\beta}})]\!]_i = [\![\sigma]\!]_i.a^{j_1}.t_i^1.\cdots.a^{j_q}.t_i^q$$

Note that $t_i^k = \pi_i(s^{\beta}, s^{k^{\neg\beta}})$. By Eqn. (C.23), $t_i^k = s_i^k$ for all k, and so

$$[\![\sigma]\!]_i.a^{j_1}.s_i^1.\cdots.a^{j_q}.s_i^q = [\![\sigma]\!]_i.a^{j_1}.t_i^1.\cdots.a^{j_q}.t_i^q \,,$$

thus yielding (C.6).

For (C.7), consider two paths σ, σ' such that

$$N^{\text{ACTLAB}_i}(\sigma) \neq N^{\text{ACTLAB}_i}(\sigma') \,.$$

By definition of $[\![\cdot]\!]$, the projection $[\![\sigma]\!]_i$ is a sequence $s_i^1.a^1.\cdots.a^{n_\sigma}.s_i^{N^{\text{ACTLAB}_i}(\sigma)}$ and the projection $[\![\sigma']\!]_i$ is a sequence $t_i^1.b^1.\cdots.b^{n_\sigma}.t_i^{N^{\text{ACTLAB}_i}(\sigma')}$. Then,

$$N^{\text{ACTLAB}_i}(\sigma) \neq N^{\text{ACTLAB}_i}(\sigma') \implies [\![\sigma]\!]_i \neq [\![\sigma']\!]_i \,.$$

For (C.8), if $a \in \text{ACTLAB}_i$ and $\alpha(s, a, t) > 0$, then $A_i \in \text{INV}(\alpha)$. The same reasoning applies to a' and α', and so $A_i \in \text{INV}(\alpha')$. Then, $\text{INV}(\alpha) \cap \text{INV}(\alpha') \neq \emptyset$ and therefore $\alpha \; \mathcal{R}_I^{\text{INV}} \; \beta$. □

LEMMA 7.3

Let $\mathcal{S}$ be a measurable set of infinite paths, let $[\cdot]$ be a traceable projection and

$$\mathcal{A} = \{A \in \text{ATOMS}(P) \mid \exists \alpha \in \text{AMPLE}(\text{LAST}(\sigma^s)) : A = \text{ACTIVE}(\alpha)\} \,.$$

If $\eta = (\mathcal{I}, \{\Theta_i\}_i, \{\Upsilon_i\}_i)$ is strongly distributed after σ^s under $[\cdot]$, then there exists $A_{i^*} \in \mathcal{A}$, $\eta^* = (\mathcal{I}^*, \{\Theta_i\}_i, \{\Upsilon_i\}_i)$ such that all the following properties hold:

I. $PR^{\eta^*}(\mathcal{S}) \geqslant PR^{\eta}(\mathcal{S})$

II. η^* *is strongly distributed after* σ^s

III. for all $A_j \in \mathcal{A}$, $\sigma' \sqsupseteq \sigma^s$, *we have*

$$PR^{\eta^*}((\sigma')^{\uparrow}) > 0 \wedge \mathcal{I}^*(\sigma')(A_j) > 0 \implies \exists \sigma'' : \sigma^s \sqsubseteq \sigma'' \sqsubseteq \sigma' \wedge \mathcal{I}^*(\sigma'')(A_{i^*}) = 1$$

IV. for all σ, *either* $\mathcal{I}^*(\sigma) = \mathcal{I}(\sigma)$ *or* $\mathcal{I}^*(\sigma)(A_j) = 1$ *for some* A_j. *Moreover* $\eta^*(\sigma) = \eta(\sigma)$ *for all* σ *such that* $\sigma^s \not\sqsubseteq \sigma$.

Proof. We start considering the case $S = \biguplus_{m=1}^{M} (\sigma^m)^\uparrow$, and obtain a scheduler η^l yielding *less* probability than the original one. Then, we show that the result can be extended to arbitrary measurable sets in such a way that the resulting scheduler yields *greater* probability than the original one (the argument is similar to the proofs of Theorem 4.1 and Theorem 4.3).

In order to cope with the sets of the form $\biguplus_{m=1}^{M} (\sigma^m)^\uparrow$, we apply a transformation on schedulers similar to those in Theorem 4.3. Except for some corner cases, the general picture of the proof is as follows: in each step, the transformation starts with a scheduler η complying with properties *(II)* and *(IV)*. This scheduler is transformed into a scheduler η', which in turn complies with with properties *(II)* and *(IV)*. Moreover, $\mathrm{PR}^{\eta'}(\biguplus_{m=1}^{M} (\sigma^m)^\uparrow) \leqslant \mathrm{PR}^{\eta}(\biguplus_{m=1}^{M} (\sigma^m)^\uparrow)$. In order to obtain η', we consider a certain set S_G of global paths, and then we choose a local path σ_{i^*} such that $[\sigma]_{i^*} = \sigma_{i^*}$ for some $\sigma \in S_G$. The new scheduler is defined as $\mathcal{I}'(\sigma)(A_{i^*}) = 1$ for all $\sigma \in S_G$ such that $[\sigma]_{i^*} = \sigma_{i^*}$. The same procedure is repeated at each step, until $A_{i^*} \in \mathcal{A}$. At this point, we let $\eta^l = \eta'$. We prove that η^l complies with all the properties in the theorem statement (except that the probability yielded is *smaller* than that of the original scheduler).

If there exists no σ' such that $\sigma^s \sqsubseteq \sigma'$, $\mathrm{PR}^{\eta}((\sigma')^\uparrow) > 0$, and $\mathcal{I}(\sigma')(A_i) > 0$ for some $A_i \in \mathcal{A}$, then η is the desired η^*. Another corner case occurs if all such σ' such that $\sigma^s \sqsubseteq \sigma'$ have length greater than $\max_{1 \leqslant m \leqslant M} \mathrm{LEN}(\sigma^m)$. In this case, let S_G be the set of minimal paths σ' such that $\sigma^s \sqsubseteq \sigma'$, $\mathrm{PR}^{\eta}((\sigma')^\uparrow) > 0$ and $\mathcal{I}(\sigma')(A_i) > 0$ for some $A_i \in \mathcal{A}$. Since the path in S_G are minimal, we have $(\sigma)^\uparrow \cap (\sigma')^\uparrow = \emptyset$ for all $\sigma, \sigma' \in S_G$. Let

By minimal, we mean minimal wrt. the prefix relation $\sqsubseteq$

$$S_L = \{[\sigma]_i \mid \sigma \in S_G \wedge \mathrm{PR}^{\eta}((\sigma)^\uparrow) > 0 \wedge \left| G_i(\mathrm{LAST}(\sigma)) \right| > 0\} .$$

In this case, let $\eta^l = \mathrm{NR}(\eta, S_G, \sigma_q)$ where $\sigma_{i^*} \in S_L$, $A_q \in \mathcal{A}$ (see page 169) comply

$$\mathcal{I}(\sigma^{>0})(A_q) > 0 ,$$

for some $\sigma^{>0} \in S_G$. We have to show that η^l complies with the properties required in the theorem statement. Since all the paths σ in S_G comply with $\mathrm{LEN}(\sigma) > \max_{1 \leqslant m \leqslant M} \mathrm{LEN}(\sigma^m)$, we have $\mathrm{PR}^{\eta}(\biguplus_{m=1}^{M} (\sigma^m)^\uparrow) = \mathrm{PR}^{\eta^l}(\biguplus_{m=1}^{M} (\sigma^m)^\uparrow)$. For property *(III)*, we can take $A_q = A_{i^*}$. Property *(IV)* holds by $\eta^l = \mathrm{NR}(\eta, S_G, \sigma_q)$ and definition of $\mathrm{NR}(\eta, S_G, \sigma_q)$. It remains to prove that $\mathrm{NR}(\eta, S_G, \sigma_q)$ is strongly distributed after σ^s (property *(II)*).

Claim. Let $\sigma_q \in S_L$ such that $A_q \in \mathcal{A}$, $\mathcal{I}(\sigma^{>0})(A_q) > 0$, for some $\sigma^{>0} \in S_G$. The scheduler $\eta^{NR} = (\mathcal{I}^{NR}, \{\Theta_i\}_i, \{\Upsilon_i\}_i)$ defined as $\eta^{NR} = \mathrm{NR}(\eta, S_G, \sigma_q)$ is strongly distributed after σ^s.

Proof of the claim. First, we prove

$$\forall \sigma \in S_G : \mathcal{I}(\sigma)(A_q) > 0 \tag{C.24}$$

Let $\sigma \in S_G$, $\sigma \neq \sigma^{>0}$. Since $\sigma \in S_G$, we have $\mathcal{I}(\sigma)(A_j) > 0$ for some $A_j \in \mathcal{A}$. Since $\mathcal{I}(\sigma')(A) = 0$ whenever $\sigma^s \sqsubseteq \sigma' \sqsubset \sigma$ and $A \in \mathcal{A}$, property *(6)* of independence structures ensures $[\sigma]_j = \left[\sigma^{>0}\right]_j = [\sigma^s]_j$ and $[\sigma]_q = \left[\sigma^{>0}\right]_q = [\sigma^s]_q$. Then, since η is strongly distributed after σ^s,

$$0 < \frac{\mathcal{I}(\sigma^{>0})(A_q)}{\mathcal{I}(\sigma^{>0})(A_q) + \mathcal{I}(\sigma^{>0})(A_j)} = \frac{\mathcal{I}(\sigma)(A_q)}{\mathcal{I}(\sigma)(A_q) + \mathcal{I}(\sigma)(A_j)} ,$$

and so $\mathcal{I}(\sigma)(A_q) > 0$.

Suppose, towards a contradiction, that η^s is not strongly distributed after σ^s. Then, there exist σ, σ', A_i, $A_{i'}$ violating the condition on strongly distributed schedulers. Let $S_W = \{\sigma \in S_G \mid [\sigma]_q = \sigma_q\}$. Since η is strongly distributed, one of the paths, say σ, must be in S_W. If both paths are in S_W, we have $\mathcal{I}^{NR}(\sigma)(A_q) = \mathcal{I}^{NR}(\sigma')(A_q) = 1$, and so σ, σ' cannot violate the condition. Then, $\sigma \in S_W$ and $\sigma' \notin S_W$. Since $\mathcal{I}^{NR}$ violates the condition for σ, σ', and

$\sigma \in S_W$, we conclude that one of the atoms, say $A_{i'}$, is A_q. Moreover, since σ, σ' violate the condition, $0 < \mathcal{I}^{NR}(\sigma')(A_i) = \mathcal{I}(\sigma')(A_i)$ (the equality holds since $\sigma' \notin S_W$). It cannot be the case that $\mathcal{I}(\sigma')(A_q) = 0$: the projections over A_i and A_q coincide for σ, σ' and $\mathcal{I}(\sigma)(A_q) > 0$ (by Eqn. (C.24)), so it would be $\mathcal{I}(\sigma)(A_q) > 0$, $\mathcal{I}(\sigma')(A_i) > 0$, $\mathcal{I}(\sigma')(A_q) = 0$, contradicting the fact that η is strongly distributed after σ^s. In conclusion, $\mathcal{I}(\sigma')(A_q) > 0$ and $\sigma' \notin S_W$. Then, there exists $\sigma^W \in S_W$ such that $\sigma^W \sqsubseteq \sigma'$. Since $\mathcal{I}^{NR}(\sigma^G)(A_q) = 1$, we have

$$\textsc{Label}(\sigma'\langle \textsc{Len}(\sigma^W)\rangle) \in \textsc{ActLab}_q \ ,$$

and hence property (1.11) implies $[\sigma']_q \neq [\sigma^W]_q$. We use this inequality in the following calculation:

$$\begin{array}{ll} & \sigma_q \\ = & [\sigma^W]_q \qquad \{\ \sigma^W \in S_G\ \} \\ \neq & [\sigma']_q \\ = & [\sigma]_q \qquad \{\ \sigma, \sigma', A_i, q \text{ violate the condition on strongly distributed schedulers}\ \} \\ = & \sigma_q \qquad \{\ \sigma \in S_G\ \} \end{array}$$

End of the claim's proof

This contradiction arises from the assumption that σ, σ' violate the condition on strongly distributed schedulers. Therefore, η^{NR} is strongly distributed after σ^s. □

So far, we have considered the corner cases. In case there exists $A \in \mathcal{A}$ such that $\mathcal{I}(\sigma)(A) > 0$ for some σ complying $\textsc{Len}(\sigma) \leqslant \max_{1 \leqslant m \leqslant M} \sigma^m$, we define S_G, η' and σ_{i^*} using Theorem 4.7 (see page 169), thus ensuring

$$\mathrm{PR}^{\eta'}(\biguplus (\sigma^m)^{\uparrow}) \leqslant \mathrm{PR}^{\eta}(\biguplus (\sigma^m)^{\uparrow}) \ . \tag{C.25}$$

In order to define R_i, we consider any atom $A_r \in \mathcal{A}$ such that such that $\mathcal{I}(\sigma^r)(A_r) > 0$ for some $\sigma^r \in S_G$. By property *(6)* of independence structures (Def. 7.5), we have $[\sigma]_r = [\sigma^s]_r$ for all $\sigma \in S_G$. Since η is strongly distributed and for each $\sigma' \in S_G$ there is at least one $A_j \in \mathcal{A}$ complying with $\mathcal{I}(\sigma')(A_j) > 0$, we can conclude that $\mathcal{I}(\sigma)(A_r) > 0$ for all $\sigma \in S_G$ (otherwise, it should be $\mathcal{I}(\sigma)(A_j) > 0$, $\mathcal{I}(\sigma)(A_r) = 0$, $\mathcal{I}(\sigma^r)(A_r) > 0$ with $[\sigma]_j = [\sigma^s]_j = [\sigma^r]_j$ and $[\sigma]_r = [\sigma^s]_r = [\sigma^r]_r$, thus violating the restriction imposed to strongly distributed schedulers). We define

$$R_i(\sigma_i) = \mathcal{I}(\sigma)(A_i)/\mathcal{I}(\sigma)(A_r) \text{ , for some } \sigma \in S_G \text{ s.t. } [\sigma]_i = \sigma_i \ . \tag{C.26}$$

Note that the particular σ chosen is irrelevant, given that the scheduler is strongly distributed for all the paths extending σ^s. We prove that R_i complies with

$$\forall \sigma \in S_G \ : \ \mathcal{I}(\sigma)(A_i) = \frac{R_i([\sigma]_i)}{\sum_{i'} R_i([\sigma]_{i'})}$$

in the following calculation:

$$
\begin{aligned}
&\frac{R_i([\sigma]_i)}{\sum_{i'} R_i([\sigma]_{i'})} \\
&= \{\ \sigma \in S_G \text{ and } [\sigma]_i = [\sigma]_i \text{ (see Eqn. (C.26))}\ \} \\
&\frac{\mathcal{I}(\sigma)(A_i)/\mathcal{I}(\sigma)(A_r)}{\sum_{i'} \left(\mathcal{I}(\sigma)(A_{i'})/\mathcal{I}(\sigma)(A_r)\right)} \\
&= \frac{\mathcal{I}(\sigma)(A_i)/\mathcal{I}(\sigma)(A_r)}{\left(\sum_{i'} \mathcal{I}(\sigma)(A_{i'})\right)/\mathcal{I}(\sigma)(A_r)} \\
&= \frac{\mathcal{I}(\sigma)(A_i)}{\sum_{i'} \mathcal{I}(\sigma)(A_{i'})} \\
&= \mathcal{I}(\sigma)(A_i)\ .
\end{aligned}
$$

Then, we take η' to be the scheduler whose existence is ensured by Theorem 4.7. The theorem ensures that the scheduler complies with $\Pr^{\eta'}(\biguplus_{m=1}^{M} (\sigma_m)^\uparrow) \leqslant \Pr^{\eta}(\biguplus_{m=1}^{M} (\sigma_m)^\uparrow)$ (later on, we will use this property to construct a scheduler such that $\Pr^{\eta}(\mathcal{S}) \leqslant \Pr^{\eta^*}(\mathcal{S})$ for any measurable set $\mathcal{S}$).

If $A_{i^*} \notin \mathcal{A}$, the same transformation can be repeated until some $A_{i^*} \in \mathcal{A}$ is selected, or $A_q \in \mathcal{A}$ is selected in case $\mathcal{I}(\sigma)(A) = 0$ whenever $\sigma^s \sqsubseteq \sigma$, $\text{LEN}(\sigma) \leqslant \max_{1 \leqslant m \leqslant M} \sigma^m$, and $A \in \mathcal{A}$, or until $\mathcal{I}(\sigma)(A) = 0$ whenever $\sigma^s \sqsubseteq \sigma$, $A \in \mathcal{A}$.

Let η^l be the resulting scheduler after the last transformation. We show that η^l complies with all the properties in the theorem statement (except that the probability yielded is *smaller* than that of the original scheduler). Properties *(II)* and *(IV)* hold since either $\eta^l = \eta'$ or η is obtained by succesive applications of $\text{NR}(\eta, S_G, \sigma_{j^*})$. *(II),(IV)*

In what follows, let S_G, S_L be the sets used to obtain the last η' (that is, the η' such that $\eta' = \eta^l$). With respect to property *(III)*, consider a path $\sigma^s \cdot \sigma$. We split the proof in two cases. *(III)*

- If there exists $\sigma^G \in S_G$ such that $\sigma^G \sqsubseteq \sigma^s \cdot \sigma$, then σ^G is the path whose existence is required.
- If there exists no $\sigma^G \in S_G$ such that $\sigma^G \sqsubseteq \sigma^s \cdot \sigma$, then for all $A_j \in \mathcal{A}$ it must be $\mathcal{I}^l(\sigma^s \cdot \sigma)(A_j) = 0$: if $\mathcal{I}^l(\sigma^s \cdot \sigma)(A_j) > 0$ then, by definition of S_G, either $\sigma^s \cdot \sigma \in S_G$ or there exists a prefix of $\sigma^s \cdot \sigma$ in S_G, thus reaching a contradiction

We have $\Pr^{\eta^l}(\biguplus_m (\sigma^m)^\uparrow) \leqslant \Pr^{\eta}(\biguplus_m (\sigma^m)^\uparrow)$, since Eqn. (C.25) holds in each step. *(I), with $\leqslant$ instead of $\geqslant$, for the particular case $\mathcal{S} = \biguplus_m (\sigma^m)^\uparrow$*

In conclusion, we have proven that there exists η' as in the statement of the theorem, except that:

$$
\Pr^{\eta'}\left(\biguplus_{m=1}^{M} (\sigma)^\uparrow_m\right) \leqslant \Pr^{\eta}\left(\biguplus_{m=1}^{M} (\sigma)^\uparrow_m\right)\ . \tag{C.27}
$$

Next, we show that there exists η' such that

$$
\forall \eta \in \mathbb{S} : \exists \eta' \in \text{SDIST}_P([\cdot], \leqslant) :\ \Pr^{\eta'}(\mathcal{S}) \leqslant \Pr^{\eta}(\mathcal{S}) \tag{C.28}
$$

For all $\mathcal{S} = \biguplus_{m=1}^{\infty} (\sigma^m)^\uparrow$.

Let $S_M = \biguplus_{\{\sigma^m \mid \text{LEN}(\sigma) < M\}} (\sigma^m)^\uparrow$. Then, by Eqn. (C.27), there exists η'_M such that

$$
\Pr^{\eta'_M}(S_M) \leqslant \Pr^{\eta}(S_M)\ .
$$

By *(IV)*, $\{\eta'_M(\sigma)\}_M$ has at most $1 + \left|\text{ATOMS}(P)\right|$ elements: the term 1 corresponds to $\eta(\sigma)$, and the term $\left|\text{ATOMS}(P)\right|$ corresponds to the distributions $1{:}A_i$ assigned by $\mathcal{I}'$, for each $A_i \in$

ATOMS(P). So, Lemma 4.8 ensures the existence of a limit η^d of the scheduler η'_M such that $PR^{\eta^d}(S) \leqslant PR^{\eta}(S)$. It remains to show that this limit meets the requirement in the statement of the theorem.

We show that the set of schedulers such that there exists A_{i^*} complying with properties *(II)*–*(IV)* is finitely falsifiable. Given η', let Z be the set of minimal paths σ satisfying $PR^{\eta'}((\sigma)^\uparrow) > 0$ and $\exists A_i \in \mathcal{A} : 0 < \mathcal{I}'(\sigma)(A_i) \leqslant 1$. In order to prove that η' does not complies with properties *(II)*–*(IV)*, it suffices to find a finite set T complying with certain properties. Such a set T comprises two paths σ^1, σ^2 s.t. $\sigma^s \sqsubseteq \sigma^k$, $PR^{\eta'}((\sigma^k)^\uparrow) > 0$, for $k = 1, 2$. In addition, T comprises all the paths σ'' s.t. $\sigma^s \sqsubseteq \sigma'' \sqsubseteq \sigma^1$, and T must comply with at least one of the following requirements:

- σ^1 and σ^2 violate the condition on strongly distributed schedulers
- $\sigma^1 \in Z$ (this membership depends on the value of $\mathcal{I}'$ for the paths σ'' such that $\sigma^s \sqsubseteq \sigma'' \sqsubseteq \sigma^k$ with $k \in \{1, 2\}$) and there exists $A_i \in \mathcal{A}$ such that $0 < \mathcal{I}'(\sigma)(A_i) < 1$ (note the strict inequality in < 1)
- $\mathcal{I}'(\sigma^1) \neq \mathcal{I}(\sigma)$ and there exists A_i such that $0 < \mathcal{I}'(\sigma)(A_i) < 1$

The verity of each of the items implies the falsity of each of the properties. Conversely, if all items are false for all sets T, they are false in particular for the sets T comprising σ^1 and σ^2 as above, and we can conclude that, for all paths in $\sigma \in Z$, there exists $A_i \in \mathcal{A}$ such that $\mathcal{I}'(\sigma)(A_i) = 1$. In addition, since the condition on strongly distributed schedulers is not violated, these A_i must be the same regardless of the σ, and so we can take such an A_i to be A_{i^*} as in the statement of the theorem. Hence, the falsity of all items implies the verity of properties *(II)*–*(IV)*, and so we have proven that the set of schedulers complying with properties *(II)*–*(IV)* is finitely falsifiable.

Therefore, since the each of the schedulers η'_M complies with such properties, Theorem 3.2 ensures that the limit scheduler η^d also does. By virtue of Eqn. (C.28), we can apply Theorem 4.2, thus yielding the result. □

LEMMA 7.4

Let σ', σ'' be such that, $\sigma' \neq \sigma''$, $PR^{\eta}((\sigma')^\uparrow) > 0$, $PR^{\eta}((\sigma'')^\uparrow) > 0$.

Then, $\mathcal{C}(\sigma') \cap \mathcal{C}(\sigma'') \neq \emptyset$ implies $\sigma' \sim \triangleleft\alpha\triangleright$ (or, symmetrically, $\sigma'' \sim \triangleleft\alpha\triangleright$) and

$$\sigma'' = \sigma'.\beta.a.(s'^{\beta}, s^{\eta^{\neg\beta}})$$

(resp., $\sigma' = \sigma''.\beta.a.(s'^{\beta}.s^{\eta^{\neg\beta}})$) for some s'^{β}. Note that $\sigma'' \sim \triangleleft\alpha\,\beta\triangleright$ (resp., $\sigma' \sim \triangleleft\alpha\,\beta\triangleright$). For such σ', σ'', we have $\mathcal{C}(\sigma') \cap \mathcal{C}(\sigma'') = \mathcal{C}(\sigma'')$ (resp. $\mathcal{C}(\sigma') \cap \mathcal{C}(\sigma'') = \mathcal{C}(\sigma')$).

Proof. If $\sigma' \sim \triangleleft\neg\sigma^s\triangleright$, the fact that $\mathcal{C}$ maps extensions of σ^s to extensions of σ^s yields

$$\mathcal{C}(\sigma') \cap \mathcal{C}(\sigma'') \neq \emptyset \implies \sigma' = \sigma'' .$$

The same argument applies in case $\sigma'' \sim\sim \triangleleft\neg\sigma^s\triangleright$. Hence, we have proven the result for the cases $\sigma' \sim \triangleleft\neg\sigma^s\triangleright$ or $\sigma'' \sim \triangleleft\neg\sigma^s\triangleright$. In what follows, we simply disregard these cases.

If both σ', σ'' are of the form $\triangleleft\alpha\triangleright$ and $\mathcal{C}(\sigma') \cap \mathcal{C}(\sigma'') = \emptyset$, let

$$\sigma^s.\beta.a.(s'^{\beta}, s^{\neg\beta}).\alpha^1.a^1.(s'^{\beta}, s^{1^{\neg\beta}}).\cdots.\alpha^n.a^n.(s'^{\beta}, s^{n^{\neg\beta}})$$

be a path in $\mathcal{C}(\sigma') \cap \mathcal{C}(\sigma'')$. By definition of $\mathcal{C}$, we have

$$\sigma = \sigma' = \sigma^s.\alpha^1.a^1.(s^{\beta}, s^{1^{\neg\beta}}).\cdots.\alpha^n.a^n.(s^{\beta}, s^{n^{\neg\beta}}) .$$

Next, we consider the case in which neither σ' nor σ'' is of the form $\triangleleft\alpha\triangleright$. Let

$$\sigma' = \sigma^s.\alpha'^1.a'^1.(s'^\beta, s'^{1^{\neg\beta}}).\cdots.\alpha'^{n'}.a'^{n'}.(s'^\beta, s'^{n'^{\neg\beta}}) \\ .\beta.a'.(t'^\beta, s'^{n'^{\neg\beta}}).\gamma'^1.b'^1.u'^1.\cdots.\gamma'^{m'}.b'^{m'}.u'^{m'}$$

$$\sigma'' = \sigma^s.\alpha''^1.a''^1.(s''^\beta, s''^{1^{\neg\beta}}).\cdots.\alpha''^{n''}.a''^{n''}.(s''^\beta, s''^{n''^{\neg\beta}}) \\ .\beta.a''.(t''^\beta, s''^{n''^{\neg\beta}}).\gamma''^1.b''^1.u''^1.\cdots.\gamma''^{m''}.b''^{m''}.u''^{m''}$$

with possibly $m' = 0$ and/or $m'' = 0$. W.l.o.g., assume $n \leqslant n'$. Assuming $\mathcal{C}(\sigma') \cap \mathcal{C}(\sigma'') \neq \emptyset$, we prove $\sigma' = \sigma''$. By definition of $\mathcal{C}$, we have $s'^\beta = s''^\beta$, $s'^{k^{\neg\beta}} = s''^{k^{\neg\beta}}$, $\alpha'^k = \alpha''^k$ and $a'^k = a''^k$ for all $k \leqslant n$. In other words,

$$\sigma'{\downarrow_{\text{LEN}(\sigma^s)+n'}} = \sigma''{\downarrow_{\text{LEN}(\sigma^s+n'')}} . \tag{C.29}$$

Since $\text{PR}^\eta((\sigma')^\uparrow) > 0$, we have $\eta(\sigma'{\downarrow_{\text{LEN}(\sigma^s)+n'}}) = 1{:}\beta$. From Eqn. (C.29) and $\text{PR}^\eta((\sigma'')^\uparrow) > 0$, we conclude that β occurs right after $\sigma''{\downarrow_{\text{LEN}(\sigma^s+n'')}}$, and so $n' = n''$. Hence,

$$\sigma'' = \sigma'{\downarrow_{\text{LEN}(\sigma^s)+n'}}.\gamma''^1.b''^1.u''^1.\cdots.\gamma''^{m''}.b''^{m''}.u''^{m''} .$$

From definition of $\mathcal{C}$ and $\mathcal{C}(\sigma') \cap \mathcal{C}(\sigma'') \neq \emptyset$, we get $\sigma' = \sigma''$.

Next, we consider the case $\sigma' \sim \triangleleft\alpha\triangleright$ and $\sigma'' \not\sim \triangleleft\alpha\triangleright$ Let

$$\sigma' = \sigma^s.\alpha'^1.a'^1.(s'^\beta, s'^{1^{\neg\beta}}).\cdots.\alpha'^{n'}.a'^{n'}.(s'^\beta, s'^{n'^{\neg\beta}})$$

$$\sigma'' = \sigma^s.\alpha''^1.a''^1.(s''^\beta, s''^{1^{\neg\beta}}).\cdots.\alpha''^{n''}.a''^{n''}.(s''^\beta, s''^{n''^{\neg\beta}}) \\ .\beta.a''.(t''^\beta, s''^{n''^{\neg\beta}}).\gamma''^1.b''^1.u''^1.\cdots.\gamma''^{m''}.b''^{m''}.u''^{m''}$$

with possibly $m'' = 0$. By definition of $\mathcal{C}$, we have $n' = n'' + m''$, and so $n'' \leqslant n'$. In addition,

$$\sigma'{\downarrow_{\text{LEN}(\sigma^s)+n''}} = \sigma''{\downarrow_{\text{LEN}(\sigma^s)+n''}} . \tag{C.30}$$

Since $\text{PR}^\eta((\sigma'')^\uparrow) > 0$, we have

$$\eta((\sigma''{\downarrow_{\text{LEN}(\sigma^s)+n''}})^\uparrow) = 1{:}\beta .$$

This equality, together with $\text{PR}^\eta((\sigma')^\uparrow) > 0$ and Eqn. (C.30) imply $\sigma' = \sigma''{\downarrow_{\text{LEN}(\sigma^s)+n''}}$. This is equivalent to:

$$\sigma'' = \sigma'.\beta.a''.(t''^\beta, s''^{n''^{\neg\beta}}).\gamma''^1.b''^1.u''^1.\cdots.\gamma''^{m''}.b''^{m''}.u''^{m''}$$

By definition of $\mathcal{C}$, we have $m'' = 0$, and so $\sigma'' = \sigma'.\beta.a''.(t''^\beta, s''^{n''^{\neg\beta}})$, as desired. □

LEMMA 7.5

Let $\eta[\sigma^s \leftarrow \beta]$ be as in Def. 7.8. If $\text{PR}^{\eta[\sigma^s\leftarrow\beta]}((\sigma)^\uparrow) > 0$ then either $\sigma = \sigma^s$ or there exists σ' such that $\sigma \in \mathcal{C}(\sigma')$ and $\text{PR}^\eta((\sigma')^\uparrow) > 0$.

Proof. If σ^s is not a prefix of σ, we have $\text{PR}^\eta((\sigma)^\uparrow) = \text{PR}^{\eta[\sigma^s\leftarrow\beta]}((\sigma)^\uparrow)$. So, in this case, we take $\sigma' = \sigma$.

If $\sigma^s \sqsubset \sigma$, we prove the existence of σ' by induction on $\text{LEN}(\sigma) - \text{LEN}(\sigma^s)$.

If $\text{LEN}(\sigma) - \text{LEN}(\sigma^s) = 1$, since $\text{PR}^\eta(\sigma) > 0$ and $\eta[\sigma^s \leftarrow \beta](\sigma^s) = 1{:}\beta$ we have $\sigma = \sigma^s.\beta.a.s$ for some a, s. In this case, we can take $\sigma' = \sigma^s$ (recall that $\sigma^s \sim \triangleleft\alpha\triangleright$). *Here we use the fact that $\sigma^s \sim \triangleleft\alpha\triangleright$*

If $\text{LEN}(\sigma) - \text{LEN}(\sigma^s) > 1$, let's drop the last state (and the last action) from σ to obtain the path $\sigma{\downarrow_{-1}}$. Let γ, b and s' be the last action, the last action label and the last state in σ (resp.). So, $\sigma{\downarrow_{-1}}\gamma.b.s' = \sigma$. Since $\text{PR}^{\eta[\sigma^s\leftarrow\beta]}(\sigma) > 0$, we have $\text{PR}^{\eta[\sigma^s\leftarrow\beta]}(\sigma{\downarrow_{-1}}) > 0$. By inductive hypothesis, there exists at least one σ'_{-1} such that $\sigma{\downarrow_{-1}} \in \mathcal{C}(\sigma'_{-1})$ and $\text{PR}^\eta((\sigma'_{-1})^\uparrow) > 0$.

- In case σ'_{-1} is of the form $\triangleleft\alpha\,\beta\,\gamma\triangleright$, by definition of $\eta[\sigma^s \leftarrow \beta]$ we have $\eta[\sigma^s \leftarrow \beta](\sigma_{\downarrow_{-1}}) = \eta(\sigma'_{-1})$. We know that $\eta[\sigma^s \leftarrow \beta](\sigma_{\downarrow_{-1}})(\gamma) > 0$. So, $\eta(\sigma'_{-1})(\gamma) > 0$ and the path $\sigma'_{-1}.\gamma.b.s'$ has probability greater than 0 in η. Taking into account the definition of $\mathcal{C}$ for the paths of the form $\triangleleft\alpha\,\beta\,\gamma\triangleright$ and the fact that $\sigma_{\downarrow_{-1}} \in \mathcal{C}(\sigma'_{-1})$, we have $\sigma = \sigma_{\downarrow_{-1}}.\gamma.b.s' \in \mathcal{C}(\sigma'_{-1}.\gamma.b.s')$. So, we can take $\sigma' = \sigma'_{-1}.\gamma.b.s'$.

- In case σ'_{-1} is of the form $\triangleleft\alpha\,\beta\triangleright$, we have $\eta[\sigma^s \leftarrow \beta](\sigma_{\downarrow_{-1}}) = \eta(\sigma'_{-1})$ (even if there are another path such σ_p such that $\sigma_{\downarrow_{-1}} \in \mathcal{C}(\sigma_p)$, the definition of $\eta[\sigma^s \leftarrow \beta]$ uses σ'_{-1}). Again, $\eta(\sigma'_{-1})(\gamma) > 0$ and the path $\sigma'_{-1}.\gamma.b.s'$ has probability greater than 0 in η. Note that $\sigma'_{-1}.\gamma.b.s' \sim \triangleleft\alpha\,\beta\,\gamma\triangleright$. So, we are in the same setting as in the previous case, and we can take $\sigma' = \sigma'_{-1}.\gamma.b.s'$.

- If none of the previous cases holds, then σ'_{-1} is uniquely determined and is of the form $\triangleleft\alpha\triangleright$. Moreover, $\eta(\sigma'_{-1}) \neq 1{:}\beta$: otherwise, $\sigma_{\downarrow_{-1}} \in \mathcal{C}(\sigma'_{-1}.\beta.a.s'')$ for some a, s'', thus contradicting $\sigma_{\downarrow_{-1}} \notin \mathcal{C}(\sigma^{\triangleleft\alpha\,\beta\triangleright})$ for all $\sigma^{\triangleleft\alpha\,\beta\triangleright} \sim \triangleleft\alpha\,\beta\triangleright$. As in the previous cases, we have $\eta[\sigma^s \leftarrow \beta](\sigma_{\downarrow_{-1}}) = \eta(\sigma'_{-1}) \neq 1{:}\beta$. So, $\eta(\sigma'_{-1})(\gamma) > 0$ and the path $\sigma'_{-1}.\gamma.b.s'$ has probability greater than 0 in η. Moreover $\gamma \notin \text{AMPLE}(s)$, and so $\sigma'_{-1}.\gamma.b.s'$ is of the form $\triangleleft\alpha\triangleright$. Then, $\sigma \in \mathcal{C}(\sigma'_{-1}.\gamma.b.s')$, since $\mathcal{C}(\sigma'_{-1}.\gamma.b.s')$ can be obtained by appending $\gamma.b.s'$ to all the paths in $\mathcal{C}(\sigma'_{-1})$. Hence, we can take $\sigma' = \sigma'_{-1}.\gamma.b.s'$.

□

LEMMA 7.6

Let P *be a IPIOA, and* $(\mathcal{R}_I, \{(\pi^\alpha, \pi^{\neg\alpha})\}_\alpha, [\cdot])$ *be an independence structure such that either:*

1. P *is a simple IPIOA,* η *is distributed after* σ^s *and*

$$(\mathcal{R}_I, \{(\pi^\alpha, \pi^{\neg\alpha})\}_\alpha, [\cdot]) = (\mathcal{R}_I^{\text{INV}}, \{(\pi^{\alpha,[\![\cdot]\!]}, \pi^{\neg\alpha,[\![\cdot]\!]})\}_\alpha, [\![\cdot]\!])$$

   or

2. η *is strongly distributed after* σ^s *and* $[\cdot]$ *is traceable.*

For all $\sigma' = \sigma^s.\beta.a.s$ *with* $\text{PR}^{\eta[\sigma^s \leftarrow \beta]}((\sigma')^\uparrow) > 0$, *the scheduler* $\eta[\sigma^s \leftarrow \beta]$ *is distributed (in case (1)) or strongly distributed (in case (2)) after* σ'.

We prove this lemma by resorting to the following one. Roughly speaking, it states that, if η is distributed and $\eta'(\sigma) = \eta(f(\sigma))$ for some f that does not hide information, then η' is also distributed. Intuitively, f does hide information if two paths distinguishable for η are indistinguishable for η', that is, if $[\sigma]_i = [\sigma']_i$, for some σ, σ' such that $[f(\sigma)]_i \neq [f(\sigma')]_i$.

LEMMA C.3. *Given* σ^s, $\sigma^* = \sigma^s.\beta.a.s$, $\eta = (\mathcal{I}, \{\Theta_i\}_i, \{\Upsilon_i\}_i)$, $\eta' = (\mathcal{I}', \{\Theta'_i\}_i, \{\Upsilon'_i\}_i)$ *such that*

- η *is distributed (strongly distributed, resp.) after* σ^s,
- $\text{PR}^{\eta}((\sigma^*)^\uparrow) > 0$,
- $\text{PR}^{\eta'}((\sigma^*)^\uparrow) > 0$,

let S^+ *be the set of finite paths* σ *such that* $\sigma^* \sqsubseteq \sigma$ *and* $\text{PR}^{\eta'}((\sigma)^\uparrow) > 0$, *and let* T^+ *be the set of finite paths* σ *such that* $\sigma^* \sqsubseteq \sigma$ *and* $\text{PR}^{\eta}((\sigma)^\uparrow) > 0$. *If there exists* $f : S^+ \to T^+$ *such that:*

$$\forall \sigma \in S^+ : \eta'(\sigma) = \eta(f(\sigma)) \ ,$$

$$\forall \sigma, \sigma' \in S^+ : \forall A_i : \mathcal{I}'(\sigma)(A_i) > 0 \implies \\ \left([\sigma]_i = \left[\sigma'\right]_i \implies [f(\sigma)]_i = \left[f(\sigma')\right]_i \right) \quad \text{(C.31)}$$

and

$$\begin{aligned} &\forall \sigma,\sigma' \in S^+ : \forall A_i : \forall a \in \text{ACTLAB}_i : \\ &\quad \big(\exists g_j, g_k, s_j, s_k : \mathcal{I}'(\sigma)(A_j) > 0 \wedge \mathcal{I}'(\sigma')(A_k) > 0 \\ &\qquad \wedge \Theta'_j([\sigma]_j)(g_j) > 0 \wedge \Theta'_k([\sigma]_k)(g_k) > 0 \\ &\qquad \wedge g_j(\pi_j(\text{LAST}(\sigma)), a, s_j) > 0 \\ &\qquad \wedge g_k(\pi_k(\text{LAST}(\sigma)), a, s_k) > 0 \\ &\qquad \wedge \left| R_i(\text{LAST}(\sigma), a) \right| > 1 \\ &\qquad \wedge \left| R_i(\text{LAST}(\sigma'), a) \right| > 1 \big) \\ &\quad \implies \left([\sigma]_i = \left[\sigma'\right]_i \implies [f(\sigma)]_i = \left[f(\sigma')\right]_i \right) \end{aligned} \quad \text{(C.32)}$$

then η' is distributed (strongly distributed, resp.) after σ^*.

Proof. In order to prove that η' is distributed after a path σ^*, it suffices to define $\Theta^*_i : \text{LOCALPATHS}_P^{[\cdot]} \to \text{PROB}(T_{G_i})$ and $\Upsilon^*_i : \text{LOCALPATHS}_P^{[\cdot]} \times \text{ACTLAB}_i \to \text{PROB}(T_{R_i})$ such that $\eta^* = (\mathcal{I}', \{\Theta^*_i\}_i, \{\Upsilon^*_i\}_i)$ complies with $\eta^*(\sigma) = \eta'(\sigma)$ whenever $\sigma^* \sqsubseteq \sigma$ and $\text{PR}^{\eta'}((\sigma)^\uparrow) > 0$.

For all A_i, local path σ_i, let $\Theta^*_i(\sigma_i) = \Theta_i(f(h^\Theta(\sigma_i))))$, where $h^\Theta(\sigma_i)$ is a global path such that:

$$\text{PR}^{\eta'}((h^\Theta(\sigma_i))^\uparrow) > 0 \wedge \sigma^* \sqsubseteq h^\Theta(\sigma_i) \wedge \left[h^\Theta(\sigma_i)\right]_i = \sigma_i \wedge \mathcal{I}'(h^\Theta(\sigma_i))(A_i) > 0 \,. \quad \text{(C.33)}$$

If no such global path exist, define $\Theta^*_i(\sigma_i)(g_i) = 1$ for some arbitrary g_i. Similarly, let $\Upsilon^*_i(\sigma_i, a) = \Upsilon_i(f(h^\Upsilon(\sigma_i)), a)$ for some global path $h^\Upsilon(\sigma_i)$ such that there exists A_j, g_j, s_j complying:

$$\begin{aligned} &\text{PR}^{\eta'}((h^\Upsilon(\sigma_i))^\uparrow) > 0 \wedge \sigma^* \sqsubseteq h^\Upsilon(\sigma_i) \wedge \mathcal{I}'(h^\Upsilon(\sigma_i))(A_j) \wedge \\ &\qquad \Theta'_j(\left[h^\Upsilon(\sigma_i)\right]_j)(g_j) > 0 \wedge g(\pi_j(\text{LAST}(h^\Upsilon(\sigma_i))), a, s_j) > 0 \,. \end{aligned} \quad \text{(C.34)}$$

We prove $\eta^*(\sigma) = \eta'(\sigma)$ for all $\sigma^* \sqsubseteq \sigma$ and $\text{PR}^{\eta'}((\sigma)^\uparrow) > 0$ by proving $\eta^*(\sigma)(c) = \eta'(\sigma)(c)$ for all c such that $\eta'(\sigma)(c) > 0$. The equality $\eta^*(\sigma) = \eta'(\sigma)$ follows from $\sum_c \eta^*(\sigma)(c) = \sum_c \eta'(\sigma)(c) = 1$. Since $\eta'(\sigma)(g_i, a, r_{j_1}, \cdots, r_{j_m}) > 0$, then

$$\begin{aligned} &\mathcal{I}'(\sigma)(A_i) > 0 \wedge \Theta'_i(\sigma)(g_i) > 0 \\ &\qquad \wedge \left(\exists s_i : g_i(\pi_i(\text{LAST}(\sigma)), a, s_i) > 0\right) \wedge \left(\underset{k}{\forall} : a \in \text{ACTLAB}_{j_k}\right) \end{aligned} \quad \text{(C.35)}$$

and so σ is a candidate for $h^\Theta([\sigma]_i)$, $h^\Upsilon([\sigma]_{j_k})$. Hence, $h^\Theta([\sigma]_i)$ is defined according to (C.33) and $h^\Upsilon([\sigma]_{j_k})$ is defined according to (C.34), for all k (although $h^\Theta([\sigma]_i)$ is not necessarily equal to σ).

First, we show

$$\Theta_i(f(h^\Theta([\sigma]_i))) = \Theta_i(f(\sigma)) \quad \text{(C.36)}$$

with the following reasoning

$$\begin{aligned} & \left[h^\Theta([\sigma]_i)\right]_i = [\sigma]_i \\ \implies & \{ \text{ Equations (C.33), (C.35) and (C.31) } \} \\ & \left[f(h^\Theta([\sigma]_i))\right]_i = [f(\sigma)]_i \\ \implies & \{ \Theta \text{ is distributed after } \sigma^s \} \\ & \Theta_i(f(h^\Theta([\sigma]_i))) = \Theta_i(f(\sigma)) \end{aligned}$$

The equation

$$\underset{k}{\forall} \Upsilon_{j_k}(f(h^{\Upsilon}([\sigma]_{j_k})), a) = \Upsilon_{j_k}(f(\sigma), a) \tag{C.37}$$

is proven using a similar reasoning:

$$\begin{aligned}
& \left[h^{\Upsilon}([\sigma]_{j_k})\right]_{j_k} = [\sigma]_{j_k} \\
\Longrightarrow\ & \{\ \text{Equations (C.34), (C.35) and (C.32)}\ \} \\
& \left[f(h^{\Upsilon}([\sigma]_i))\right]_i = [f(\sigma)]_i) \\
\Longrightarrow\ & \{\ \Theta \text{ is distributed after } \sigma^s\ \} \\
& \Upsilon_{j_k}(f(h^{\Upsilon}([\sigma]_{j_k})), a) = \Upsilon_{j_k}(f(\sigma), a)
\end{aligned}$$

Then, we calculate:

$$\begin{aligned}
& \eta^*(\sigma)(g_i, a, r_{j_1}, \cdots, r_{j_m}) \\
= & \{\ \eta^* = (\mathcal{I}', \{\Theta_i^*\}_i, \{\Upsilon_i^*\}_i)\ \} \\
& \mathcal{I}'(\sigma)(A_i)\ \Theta_i^*([\sigma]_i)(g_i) \\
& \sum_{s_i} g_i(\pi_i(\textsc{Last}(\sigma)), a, s_i) \prod_{k=1} \Upsilon_{j_k}^*([\sigma]_{j_k}, a)(r_{j_k}) \\
= & \{\ \text{Definition of } \Theta_i^*, \Upsilon_i^*\ \} \\
& \mathcal{I}'(\sigma)(A_i)\ \Theta_i(f(h^{\Theta}([\sigma]_i)))(g_i) \\
& \sum_{s_i} g_i(\pi_i(\textsc{Last}(\sigma)), a, s_i) \prod_{k=1} \Upsilon_{j_k}(f(h^{\Theta}([\sigma]_{j_k})), a)(r_{j_k}) \\
= & \{\ \text{Equations (C.36) and (C.37)}\ \} \\
& \mathcal{I}'(\sigma)(A_i)\ \Theta_i(f(\sigma))(g_i) \\
& \sum_{s_i} g_i(\pi_i(\textsc{Last}(\sigma)), a, s_i) \prod_{k=1} \Upsilon_{j_k}(f(\sigma), a)(r_{j_k}) \\
= & \{\ \eta'(\sigma) = \eta(f(\sigma))\ \} \\
& \eta'(\sigma)(g_i, a, r_{j_1}, \cdots, r_{j_m})
\end{aligned}$$

Next, we prove that, if η is strongly distributed, so η' is. Let σ^1, σ^2, A, A', such that $\text{PR}^{\eta'}((\sigma^1)^{\uparrow}) > 0$ and $\text{PR}^{\eta'}((\sigma^2)^{\uparrow}) > 0$, $[\sigma^1]_A = [\sigma^2]_A$, $[\sigma^1]_{A'} = [\sigma^2]_{A'}$, $\mathcal{I}'(\sigma^1) > 0$, $\mathcal{I}'(\sigma^2) > 0$. By Eqn. (C.31), it must be

$$[f(\sigma^1)]_A = [f(\sigma^2)]_A \ \wedge\ [f(\sigma^1)]_{A'} = [f(\sigma^2)]_{A'} \tag{C.38}$$

We prove the desired equality

$$\frac{\mathcal{I}'(\sigma^1)(A)}{\mathcal{I}'(\sigma^1)(A) + \mathcal{I}'(\sigma^1)(A')} = \frac{\mathcal{I}'(\sigma^2)(A)}{\mathcal{I}'(\sigma^2)(A) + \mathcal{I}'(\sigma^2)(A')}$$

as follows:

$$
\begin{aligned}
&\frac{\mathcal{I}'(\sigma^1)(A)}{\mathcal{I}'(\sigma^1)(A)+\mathcal{I}'(\sigma^1)(A')}\\
&=\{\ \eta'(\sigma)=\eta(f(\sigma))\ \}\\
&=\frac{\mathcal{I}(f(\sigma^1))(A)}{\mathcal{I}(f(\sigma^1))(A)+\mathcal{I}(f(\sigma^1))(A')}\\
&=\{\ \text{Equation (C.38), } \eta \text{ is strongly distributed}\ \}\\
&=\frac{\mathcal{I}(f(\sigma^2))(A)}{\mathcal{I}(f(\sigma^2))(A)+\mathcal{I}(f(\sigma^2))(A')}\\
&=\{\ \eta'(\sigma)=\eta(f(\sigma))\ \}\\
&=\frac{\mathcal{I}'(\sigma^2)(A)}{\mathcal{I}'(\sigma^2)(A)+\mathcal{I}'(\sigma^2)(A')}
\end{aligned}
$$

□

Proof of Lemma 7.6. We show that, for some f, the schedulers η and $\eta[\sigma^s \leftarrow \beta]$ are under the hypotheses of Lemma C.3. Let $f(\sigma) = \sigma_{\not\sim \triangleleft\alpha\triangleright}$ if there exists $\sigma_{\not\sim \triangleleft\alpha\triangleright}$ such that $\sigma_{\not\sim \triangleleft\alpha\triangleright} \not\sim \triangleleft\alpha\triangleright$, $\mathrm{PR}^\eta(\sigma_{\not\sim \triangleleft\alpha\triangleright}) > 0$ and $\mathcal{C}(\sigma_{\not\sim \triangleleft\alpha\triangleright}) = \sigma$. If there exists no such $\sigma_{\not\sim \triangleleft\alpha\triangleright}$, and there exists $\sigma_{\sim \triangleleft\alpha\triangleright}$ such that $\sigma_{\sim \triangleleft\alpha\triangleright} \sim \triangleleft\alpha\triangleright$, $\mathrm{PR}^\eta(\sigma_{\sim \triangleleft\alpha\triangleright}) > 0$ and $\sigma \in \mathcal{C}(\sigma_{\sim \triangleleft\alpha\triangleright})$, let $f(\sigma) = \sigma_{\sim \triangleleft\alpha\triangleright}$. By definition of $\eta[\sigma^s \leftarrow \beta]$ and Lemma 7.5, we have

$$\forall \sigma \in S^+ : \eta[\sigma^s \leftarrow \beta](\sigma) = \eta(f(\sigma))$$

(with S^+ as in Lemma C.3).

Let $\eta = (\mathcal{I}, \{\Theta_i\}_i, \{\Upsilon_i\}_i)$, $\eta' = \eta[\sigma^s \leftarrow \beta] = (\mathcal{I}', \{\Theta'_i\}, \{\Upsilon'_i\})$.

We have to prove (C.31) and (C.32).

To this end, we prove that, if $f(\sigma) \sim \triangleleft\alpha\triangleright$, then $\eta'(\sigma)(\alpha) > 0$ implies $\alpha \notin \textsc{Ample}(\textsc{Last}(\sigma^s))$. To see that this proposition holds, let σ' such that $\sigma \in \mathcal{C}(\sigma')$ and $\sigma' \sim \triangleleft\alpha\triangleright$. Given that $\sigma' \sim \triangleleft\alpha\triangleright$, condition Eqn. (7.5) implies that either $\eta(\sigma') = p_1{:}\alpha_1 + \cdots + p_Q{:}\alpha_Q$ with $\alpha_q \notin \textsc{Ample}(\textsc{Last}(\sigma^s))$ for all $1 \leqslant q \leqslant Q$, or $\eta(\sigma') = 1{:}\beta$. In the former case $\sigma' = f(\sigma)$ and the implication holds. In the latter case $f(\sigma) \sim \triangleleft\alpha\,\beta\triangleright$ and the proposition holds trivially.

Properties (C.31) and (C.32) are proved for each one of three possible cases, which depend on the paths $f(\sigma)$, $f(\sigma')$ appearing in such properties:

- Case $f(\sigma) \sim \triangleleft\alpha\triangleright$, $f(\sigma') \sim \triangleleft\alpha\triangleright$.
  Suppose

$$f(\sigma) = \sigma^s.\alpha^1.a^1.(s^\beta, s^{1^{\neg\beta}}) \cdots \alpha^n.a^n.(s^\beta, s^{n^{\neg\beta}})i\,. \tag{C.39}$$

By definition of f, we have $\sigma \in \mathcal{C}(f(\sigma))$ and so

$$\sigma = \sigma^s.\beta.a.(t^\beta, s^{\neg\beta}).\alpha^1.a^1.(t^\beta, s^{1^{\neg\beta}}) \cdots \alpha^n.a^n.(t^\beta, s^{n^{\neg\beta}}) \tag{C.40}$$

for some a, t^β.

Analogously, if

$$f(\sigma') = \sigma^s.\alpha'^1.a'^1.(s^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^{n'}.a'^{n'}.(s^\beta, s'^{n'^{\neg\beta}})$$

then

$$\sigma' = \sigma^s.\beta.a.(t'^\beta, s^{\neg\beta}).\alpha'^1.a'^1.(t'^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^{n'}.a'^{n'}.(t'^\beta, s'^{n'^{\neg\beta}})$$

for some a, t'^β.

Then, the implication

$$[\sigma]_i = [\sigma']_i \implies [f(\sigma)]_i = [f(\sigma')]_i$$

present in both (C.31) and (C.32) is exactly the contrapositive of property *(7)* of independence structures.

- Case $f(\sigma) \not\sim \triangleleft\alpha\triangleright$, $f(\sigma') \not\sim \triangleleft\alpha\triangleright$.
  Since $\sigma^s \sqsubset \sigma$, we have $f(\sigma) \sim \triangleleft\alpha\,\beta\triangleright$ or $f(\sigma) \sim \triangleleft\alpha\,\beta\,\gamma\triangleright$ (that is, it cannot be the case that $f(\sigma) \sim \triangleleft\neg\sigma^s\triangleright$). Similarly as in the previous case, we have that the implication

  $$[\sigma]_i = [\sigma']_i \implies [f(\sigma)]_i = [f(\sigma')]_i$$

  is property *(8)*.
  Note that the case $m = 0$ in property *(8)* occurs when $f(\sigma) \sim \triangleleft\alpha\,\beta\triangleright$ (the case $m' = 0$ occurs when $f(\sigma') \sim \triangleleft\alpha\,\beta\triangleright$, resp.).

- Case $f(\sigma) \sim \triangleleft\alpha\triangleright$ and $f(\sigma') \not\sim \triangleleft\alpha\triangleright$, or $f(\sigma') \sim \triangleleft\alpha\triangleright$ and $f(\sigma) \not\sim \triangleleft\alpha\triangleright$.
  We split the proof according to whether η is distributed or strongly distributed schedulers (recall the hypotheses for each of the cases in the statement of this lemma).

Distributed schedulers

1. If $f(\sigma) \sim \triangleleft\alpha\triangleright$ and $f(\sigma') = \not\sim \triangleleft\alpha\triangleright$, then σ and $f(\sigma)$ are as in Equations C.40 and C.39,

   $$f(\sigma') = \sigma^s.\alpha'^1.a'^1.(s^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^{m'}.a'^{m'}.(s^\beta, s'^{m'^{\neg\beta}}).\beta.a.(t'^\beta, s'^{n'^{\neg\beta}}) .\gamma'^1.b'^1.u'^1.\cdots.\gamma'^{m'}.b'^{m'}.u'^{m'}$$

   and

   $$\sigma' = \sigma^s.\beta.a.(t'^\beta, s^{\neg\beta}).\alpha'^1.a'^1.(t'^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^{n'}.a'^{n'}.(t'^\beta, s'^{n'^{\neg\beta}}) .\gamma'^1.b'^1.u'^1.\cdots.\gamma'^{m'}.b'^{m'}.u'^{m'}$$

   If $a \in \text{ACTLAB}_i$ then, by definition of $\mathcal{R}_I^{\text{INV}}$ we have $a^k \notin \text{ACTLAB}_i$ for all k (the labels a^k are the ones in Eqn. (C.40)), and so the definition of $[\![\cdot]\!]$ yields

   $$[\![\sigma]\!]_i = [\![\sigma^s]\!]_i.a.\pi_i((t^\beta, s^{\neg\beta}))\ .$$

   Hence, $[\![\sigma]\!]_i = [\![\sigma']\!]_i$ implies $\pi_i((t^\beta, s^{\neg\beta})) = \pi_i((t'^\beta, s^{\neg\beta}))$ and $a'^k, b'^k \notin \text{ACTLAB}_i$ for all k. Note the that former equality implies $\pi_i((t^\beta, s^{n^{\neg\beta}})) = \pi_i((t'^\beta, s'^{n'^{\neg\beta}}))$ since, by the definition of the β-projection $\pi^{\beta,[\![\cdot]\!]}$ (Definitions 7.3 and 7.4), the local state of A_i lies in t^β and $t'\beta$. We use these properties in the following calculation:

   $$\begin{aligned}
   & [\![\sigma]\!]_i = [\![\sigma']\!]_i \\
   \implies & \{\ a^k \notin \text{ACTLAB}_i,\ a'^k, b'^k \notin \text{ACTLAB}_i, \text{definition of } [\![\cdot]\!]\ \} \\
   & [\![f(\sigma)]\!]_i = [\![\sigma^s]\!]_i.a.\pi_i((t^\beta, s^{n^{\neg\beta}})) \wedge [\![f(\sigma')]\!]_i = [\![\sigma^s]\!]_i.a.\pi_i((t'^\beta, s'^{n'^{\neg\beta}})) \\
   \implies & \{\ \pi_i((t^\beta, s^{n^{\neg\beta}})) = \pi_i((t'^\beta, s'^{n'^{\neg\beta}}))\ \} \\
   & [\![f(\sigma)]\!]_i = [\![f(\sigma')]\!]_i
   \end{aligned}$$

   If $a \notin \text{ACTLAB}_i$, then

   $$\begin{aligned}
   [\![\sigma]\!]_i &= [\![\sigma^s]\!]_i.a^{z_1}.\pi_i((s^{z_1^\beta}, t^\beta)).\cdots.a^{z_W}.\pi_i((s^{z_W^\beta}, t^\beta)) \\
   [\![\sigma']\!]_i &= [\![\sigma^s]\!]_i.a'^{z'_1}.\pi_i((s'^{z'^\beta_1}, t'^\beta)).\cdots.a'^{z'_Y}.\pi_i((s'^{z'^\beta_Y}, t'^\beta)) \\
   [\![f(\sigma)]\!]_i &= [\![\sigma^s]\!]_i.a^{z_1}.\pi_i((s^{z_1^\beta}, s^\beta)).\cdots.a^{z_W}.\pi_i((s^{z_W^\beta}, s^\beta)) \\
   [\![f(\sigma')]\!]_i &= [\![\sigma^s]\!]_i.a'^{z'_1}.\pi_i((s'^{z'^\beta_1}, t'^\beta)).\cdots.a'^{z'_Y}.\pi_i((s'^{z'^\beta_Y}, t'^\beta))
   \end{aligned}$$

where $1 \leqslant z_1 < \cdots < z_W \leqslant n$ and $1 \leqslant z'_1 < \cdots \leqslant n' < \cdots < z'_Y \leqslant m'$. Assume $[\![\sigma]\!]_i = [\![\sigma']\!]_i$. Then, we have $W = Y$, $a^k = a'^k$ and $\pi_i((t^\beta, s^{k^{\neg\beta}})) = \pi_i((t'^\beta, s'^{k^{\neg\beta}}))$ for all k. By definition of $\pi^{\beta,[\![\cdot]\!]}$ the latter equality implies $\pi_i((s^\beta, s^{k^{\neg\beta}})) = \pi_i((s'^\beta, s'^{k^{\neg\beta}}))$. Therefore, under the assumption $[\![\sigma]\!]_i = [\![\sigma']\!]_i$ we can prove $[\![f(\sigma)]\!]_i = [\![f(\sigma')]\!]_i$, as desired.

The proof for $f(\sigma) \sim \triangleleft\alpha\triangleright$ and $f(\sigma') =\!\!\not\sim \triangleleft\alpha\triangleright$ is symmetrical.

Strongly distributed schedulers

2. We show that, under the hypotheses of the theorem:

$$[\sigma]_i \neq [\sigma']_i \tag{C.41}$$

Then, both Eqn. (C.31) and Eqn. (C.32) hold, since the antecedent of the implications cannot be true. We assume, towards a contradiction, that (C.41) does not hold. The contradiction we prove is that η is not strongly distributed after σ^s. Since the proof relies only on $f(\sigma) \sim \triangleleft\alpha\triangleright$, $f(\sigma') \not\sim \triangleleft\alpha\triangleright$, and $[\sigma]_i = [\sigma']_i$, the proof for $f(\sigma') \sim \triangleleft\alpha\triangleright$ and $f(\sigma) \not\sim \triangleleft\alpha\triangleright$ is symmetrical.

If $f(\sigma') \not\sim \triangleleft\alpha\triangleright$, then $f(\sigma') \sim \triangleleft\alpha\,\beta\triangleright$ or $f(\sigma') \sim \triangleleft\alpha\,\beta\,\gamma\triangleright$, (and similarly for $f(\sigma)$).

Since $f(\sigma') \sim \triangleleft\alpha\,\beta\triangleright$ or $f(\sigma') \sim \triangleleft\alpha\,\beta\,\gamma\triangleright$, there exists $\sigma_\beta \sqsubseteq f(\sigma')$ such that $\eta(\sigma_\beta)(\beta) = 1$ and $\sigma_\beta \sim \triangleleft\alpha\triangleright$. By property *(6)* of independence structures, we have

$$\left[\sigma_\beta\right]_{\text{ACTIVE}(\beta)} = [\sigma^s]_{\text{ACTIVE}(\beta)} \tag{C.42}$$

Since $[\cdot]$ is traceable and $[f(\sigma)]_i = [f(\sigma')]_i$, then there exist local paths $\sigma_{i_1}, \cdots, \sigma_{i_R}$ such that

$$\begin{aligned}\exists \sigma^1, \cdots, \sigma^R, \sigma'^1, \cdots, \sigma'^R : \quad & \sigma^s \sqsubseteq \sigma^1 \sqsubseteq \cdots \sqsubseteq \sigma^R \sqsubseteq f(\sigma) \\ & \wedge\ \sigma^s \sqsubseteq \sigma'^1 \sqsubseteq \cdots \sqsubseteq \sigma'^R \sqsubseteq f(\sigma') \\ & \wedge\ \left[\sigma^1\right]_{i_1} = \left[\sigma'^1\right]_{i_1} = [\sigma^s]_{i_1} \\ & \wedge \bigwedge_{r=1}^{R} [\sigma^r]_{i_r} = \left[\sigma'^r\right]_{i_r} = \sigma_{i_r} \wedge \mathcal{I}(\sigma^r)(A_{i_r}) > 0 \wedge \mathcal{I}(\sigma'^r)(A_{i_r}) > 0\end{aligned}$$

Intuitively, for the local path σ_{i_R} that leads to the local path $[f(\sigma)]_i$, there are two global paths $\sigma^R \sqsubseteq f(\sigma)$ and $\sigma'^R \sqsubseteq f(\sigma')$ such that $\left[\sigma^R\right]_{i_R} = \left[\sigma'^R\right]_{i_R} = \sigma_{i_R}$, $\mathcal{I}(\sigma^R)(i_R) > 0$ and $\mathcal{I}(\sigma'^R)(i_R) > 0$. Then, we can do the same reasoning for the path $\sigma_{i_{R-1}}$ that leads to σ_{i_R}. Finally, since $\sigma^s \sqsubseteq f(\sigma)$ and $\sigma^s \sqsubseteq f(\sigma')$, we reach σ^1, σ'^1 such that $\left[\sigma^1\right]_{i_1} = \left[\sigma^1\right]_{i_1} = [\sigma^s]_{i_1}$.

Since $\sigma^s \sqsubseteq \sigma_\beta \sqsubseteq f(\sigma)$, there exists r such that $\left[\sigma_\beta\right]_{i_r} = \sigma_{i_r}$, and so

$$[\sigma^r]_{i_r} = \left[\sigma_\beta\right]_{i_r} = \sigma_{i_r} \tag{C.43}$$

The property of strongly distributed schedulers fails for σ^r, σ_β, A_{i_r}, $\text{ACTIVE}(\beta)$: it must be $\text{ACTIVE}(\beta) \neq A_{i_r}$, since $\eta(\sigma^r)(\alpha) > 0$ implies $\alpha\,\mathcal{R}_I\,\beta$, as $\sigma^r \sqsubseteq f(\sigma) \sim \triangleleft\alpha\triangleright$ (condition **A3** of the reduction). Moreover, by property *(6)* of independence structures and $\sigma^r \sqsubseteq f(\sigma)$, we have $[\sigma^r]_{\text{ACTIVE}(\beta)} = [\sigma^s]_{\text{ACTIVE}(\beta)}$, and so by Eqn. (C.42),

$$[\sigma^r]_{\text{ACTIVE}(\beta)} = \left[\sigma_\beta\right]_{\text{ACTIVE}(\beta)} . \tag{C.44}$$

In summary, $[\sigma^r]_{i_r} = \left[\sigma_\beta\right]_{i_r}$ (Eqn. (C.43)), $[\sigma^r]_{\text{ACTIVE}(\beta)} = \left[\sigma_\beta\right]_{\text{ACTIVE}(\beta)}$ (Eqn. (C.44)) and

$$\begin{aligned}\frac{\mathcal{I}(\sigma_\beta)(\text{ACTIVE}(\beta))}{\mathcal{I}(\sigma_\beta)(\text{ACTIVE}(\beta)) + \mathcal{I}(\sigma_\beta)(A_{i_r})} &= \frac{1}{1+0} \\ &\neq \frac{0}{0 + \mathcal{I}(\sigma^r)(A_{i_r})} = \frac{\mathcal{I}(\sigma^r(\text{ACTIVE}(\beta))}{\mathcal{I}(\sigma^r(\text{ACTIVE}(\beta)) + \mathcal{I}(\sigma^r)(A_{i_r})} ,\end{aligned}$$

thus contradicting the fact that η is strongly distributed after σ^s.

□

LEMMA 7.7

For all cylinders $\textsc{Cyl}(\ell_1^+, \cdots, \ell_n^+)$,

$$\mathrm{PR}^{\eta}(\textsc{Cyl}(\ell_1^+, \cdots, \ell_n^+)) = \mathrm{PR}^{\eta[\sigma^s \leftarrow \beta]}(\textsc{Cyl}(\ell_1^+, \cdots, \ell_n^+)) \,.$$

In order to prove this lemma, we first note that $\textsc{Cyl}(\ell_1^+, \cdots, \ell_n^+)$ can be written as a disjoint union:

$$\begin{aligned} \textsc{Cyl}(\ell_1^+, \cdots, \ell_n^+) = \quad & \biguplus_{\{\sigma \mid \textsc{trace}(\sigma) \sim \ell_1^+, \cdots, \ell_n^+ \ell' \,\wedge\, \ell_n \neq \ell'\}} (\sigma)^{\uparrow} \\ \uplus \quad & \biguplus_{\{\sigma \mid \textsc{trace}(\sigma) \sim \ell_1^+, \cdots, \ell_{n-1}^+\}} \mathcal{Z}(\sigma) \end{aligned} \tag{C.45}$$

where $\mathcal{Z}(\sigma) = \{\sigma.\alpha_1.a_1.s^1.\alpha_2.a_2.s^2.\cdots \mid \forall_k \, \mathsf{L}(s^k) = \ell_n\}$.

Equation (C.45) motivates the following lemmata.

LEMMA C.4.

$$\mathrm{PR}^{\eta}((\sigma)^{\uparrow}) = \sum_{\sigma' \in \mathcal{C}(\sigma)} \mathrm{PR}^{\eta[\sigma^s \leftarrow \beta]}((\sigma')^{\uparrow})$$

LEMMA C.5.

$$\mathrm{PR}^{\eta}(\mathcal{Z}(\sigma)) = \sum_{\sigma' \in \mathcal{C}(\sigma)} \mathrm{PR}^{\eta[\sigma^s \leftarrow \beta]}(\mathcal{Z}(\sigma'))$$

$\triangleleft\neg\sigma^s\triangleright$ *Proof of Lemma C.4.* In case $\sigma \sim \triangleleft\neg\sigma^s\triangleright$, the result follows from $\sigma' \sim \triangleleft\neg\sigma^s\triangleright$ for all $\sigma' \sqsubset \sigma$, and so $\eta(\sigma') = \eta[\sigma^s \leftarrow \beta](\sigma')$ for all such σ'.

$\triangleleft\alpha\,\beta\,\gamma\triangleright$ In case $\sigma \sim \triangleleft\alpha\,\beta\,\gamma\triangleright$, suppose

$$\sigma = \sigma^s.\alpha_1.a_1.(s^{\beta}, s^{1^{\neg\beta}}) \cdots \alpha_n.a_n.(s^{\beta}, s^{n^{\neg\beta}}).\beta.a.(s'^{\beta}, s^{n^{\neg\beta}}) \\ .\gamma_1.b_1.(s'^{1^{\beta}}, s'^{1^{\neg\beta}}) \cdots \gamma_{n'}.b_{n'}.(s'^{n'^{\beta}}, s'^{n'^{\neg\beta}})$$

and $\textsc{last}(\sigma^s) = (s^{\beta}, s^{\neg\beta})$. Let

$$\begin{aligned} J_k &= \mathcal{I}(\sigma^s.\alpha_1.a_1.(s^{\beta}, s^{1^{\neg\beta}}) \cdots (s^{\beta}, s^{k-1^{\neg\beta}})) \\ K_k &= \mathcal{I}(\sigma^s.\alpha_1.a_1.(s^{\beta}, s^{1^{\neg\beta}}) \cdots (s^{\beta}, s^{n^{\neg\beta}}).\beta.a.(s'^{\beta}, s^{n^{\neg\beta}}) \\ &\qquad .\gamma_1.b_1.(s'^{1^{\beta}}, s'^{1^{\neg\beta}}) \cdots (s'^{k-1^{\beta}}, s'^{k-1^{\neg\beta}})) \\ J'_k &= \mathcal{I}'(\sigma^s.\beta.a.(s'^{\beta}, s^{\neg\beta}).\alpha_1.a_1.(s'^{\beta}, s^{1^{\neg\beta}}) \cdots (s'^{\beta}, s^{k-1^{\neg\beta}})) \\ K'_k &= \mathcal{I}'(\sigma^s.\beta.a.(s'^{\beta}, s^{\neg\beta}).\alpha_1.a_1 \cdots (s'^{\beta}, s^{n^{\neg\beta}}) \\ &\qquad .\gamma_1.b_1.(s'^{1^{\beta}}, s'^{1^{\neg\beta}}) \cdots \gamma_{n'}.b_{n'}.(s'^{n'^{\beta}}, s'^{n'^{\neg\beta}})) \end{aligned}$$

Since

$$\sigma^s.\beta.a.(s'^{\beta}, s^{\neg\beta}).\alpha_1.a_1.(s'^{\beta}, s^{1^{\neg\beta}}) \cdots (s'^{\beta}, s^{k-1^{\neg\beta}}) \\ \in \mathcal{C}(\sigma^s.\alpha_1.a_1.(s^{\beta}, s^{1^{\neg\beta}}) \cdots (s^{\beta}, s^{k-1^{\neg\beta}}))$$

and

$$\begin{aligned} &\sigma^s.\beta.a.(s'^{\beta}, s^{\neg\beta}).\alpha_1.a_1 \cdots (s'^{\beta}, s^{n^{\neg\beta}}) \\ &\qquad .\gamma_1.b_1.(s'^{1^{\beta}}, s'^{1^{\neg\beta}}) \cdots \gamma_{n'}.b_{n'}.(s'^{n'^{\beta}}, s'^{n'^{\neg\beta}}) \\ &\in \mathcal{C}(\sigma^s.\alpha_1.a_1.(s^{\beta}, s^{1^{\neg\beta}}) \cdots (s^{\beta}, s^{n^{\neg\beta}}).\beta.a.(s'^{\beta}, s^{n^{\neg\beta}}) \\ &\qquad .\gamma_1.b_1.(s'^{1^{\beta}}, s'^{1^{\neg\beta}}) \cdots (s'^{k-1^{\beta}}, s'^{k-1^{\neg\beta}})) \quad , \end{aligned}$$

by definition of $\eta[\sigma^s \leftarrow \beta]$ we get:

$$J_k = J'_k \wedge K_k = K'_k \tag{C.46}$$

for all k.

$$\begin{aligned}
&PR^{\eta}((\sigma)^{\uparrow}) \\
=&\{\ \sigma \text{ is of the form } \triangleleft\alpha\,\beta\,\gamma\triangleright\ \} \\
&PR^{\eta}(\sigma^s.\alpha_1.a_1.(s^{\beta}, s_1^{\neg\beta}) \cdots \alpha_n.a_n.(s^{\beta}, s^{n^{\neg\beta}})\beta.a.(s'^{\beta}, s^{n^{\neg\beta}}) \\
&\qquad .\gamma_1.b_1.(s'^{1^{\beta}}, s'^{1^{\neg\beta}}) \cdots \gamma_{n'}.b_{n'}.(s'^{n'^{\beta}}, s'^{n'^{\neg\beta}})) \\
=&\{\ \eta \text{ is I/O nonrandomized, let } s^{0^{\neg\beta}} = s^{\neg\beta},\ s'^{0^{\beta}} = s'^{\beta}, \text{ and } s'^{0^{\neg\beta}} = s^{n^{\neg\beta}}\ \} \\
&PR^{\eta}(\sigma^s) \cdot \prod_{k=1}^{n} J_k \cdot \alpha_k(\ (s^{\beta}, s^{k^{\neg\beta}}),\ a_k,\ (s^{\beta}, s^{k^{\neg\beta}})\ ) \cdot \beta(\ (s^{\beta}, s_n^{\neg\beta}),\ a,\ (s'^{\beta}, s^{n^{\neg\beta}})\ ) \\
&\qquad \cdot \prod_{k=1}^{n'} K_k \cdot \gamma_k(\ (s'^{k-1^{\beta}}, s'^{k-1^{\neg\beta}}),\ b_k,\ (s'^{k^{\beta}}, s'^{k^{\neg\beta}})\ ) \\
=&\{\ \text{Def. 7.5, property (4). Let } s^{0^{\neg\beta}} = s^{\neg\beta}\ \} \\
&PR^{\eta}(\sigma^s) \cdot \beta(\ (s^{\beta}, s^{\neg\beta}),\ a,\ (s'^{\beta}, s^{\neg\beta})\ ) \cdot \prod_{k=1}^{n} J_k \cdot \alpha_k(\ (s'^{\beta}, s^{k-1^{\neg\beta}}),\ a_k,\ (s'^{\beta}, s^{k^{\neg\beta}})\ ) \\
&\qquad \cdot \prod_{k=1}^{n'} K_k \cdot \gamma_k(\ (s'^{k-1^{\beta}}, s'^{k-1^{\neg\beta}}),\ b_k,\ (s'^{k^{\beta}}, s'^{k^{\neg\beta}})\ ) \\
=&\{\ \text{Equation (C.46)}\ \} \\
&PR^{\eta}(\sigma^s) \cdot \beta(\ (s^{\beta}, s^{\neg\beta}),\ a,\ (s'^{\beta}, s^{\neg\beta})\ ) \cdot \prod_{k=1}^{n} J'_k \cdot \alpha_k(\ (s'^{\beta}, s^{k-1^{\neg\beta}}),\ a_k,\ (s'^{\beta}, s^{k^{\neg\beta}})\ ) \\
&\qquad \cdot \prod_{k=1}^{n'} K'_k \cdot \gamma_k(\ (s'^{k-1^{\beta}}, s'^{k-1^{\neg\beta}}),\ b_k,\ (s'^{k^{\beta}}, s'^{k^{\neg\beta}})\ ) \\
=&\{\ \text{Definition of } \mathcal{C} \text{ (recall that, in this case, } \mathcal{C}(\sigma) \text{ is a singleton set)}\ \} \\
&\sum_{\sigma' \in \mathcal{C}(\sigma)} PR^{\eta[\sigma^s \leftarrow \beta]}((\sigma')^{\uparrow})
\end{aligned}$$

The case in which σ is of the form $\triangleleft\alpha\,\beta\triangleright$ is a particular case of the one above, with $n' = 0$. $\triangleleft\alpha\,\beta\triangleright$
Below, we study the case in which σ is of the form $\triangleleft\alpha\triangleright$. $\triangleleft\alpha\triangleright$

$$\begin{aligned}
&PR^{\eta}((\sigma)^{\uparrow}) \\
=&\, PR^{\eta}((\sigma^s.\alpha_1.a_1.(s^{\beta}, s_1^{\neg\beta}) \cdots \alpha_n.a_n.(s^{\beta}, s^{n^{\neg\beta}}))^{\uparrow}) \\
=&\, PR^{\eta}((\sigma^s)^{\uparrow}) \cdot \prod_{k=1}^{n} J_k \cdot \alpha_k(\ (s^{\beta}, s^{k^{\neg\beta}}),\ a_k,\ (s^{\beta}, s^{k^{\neg\beta}})\ ) \\
=&\{\ \textstyle\sum_{a,s'^{\beta}} \beta((s^{\beta}, s^{n^{\neg\beta}}),\ a,\ (s'^{\beta}, s^{n^{\neg\beta}})) \;=\; 1\ \}
\end{aligned}$$

$$\begin{aligned}
&\sum_{a,s'^{\beta}} \beta((s^{\beta}, s^{n^{\neg\beta}}),\ a,\ (s'^{\beta}, s^{n^{\neg\beta}})) \cdot PR^{\eta}((\sigma^s)^{\uparrow}) \\
&\qquad \cdot \prod_{k=1}^{n} J_k \cdot \alpha_k(\ (s^{\beta}, s^{k^{\neg\beta}}),\ a_k,\ (s^{\beta}, s^{k^{\neg\beta}})\ )
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{a,s'^{\beta}} \mathrm{PR}^{\eta}((\sigma^{s})^{\uparrow}) \\
&\qquad \cdot \prod_{k=1}^{n} J_k \cdot \alpha_k(\,(s^{\beta}, s^{k^{\neg\beta}}),\, a_k,\, (s^{\beta}, s^{k^{\neg\beta}})\,) \;\cdot\; \beta((s^{\beta}, s^{n^{\neg\beta}}),\, a,\, (s'^{\beta}, s^{n^{\neg\beta}})) \\
&= \sum_{a,s'^{\beta}} \beta((s^{\beta}, s^{\neg\beta}),\, a,\, (s'^{\beta}, s^{\neg\beta})) \cdot\; \mathrm{PR}^{\eta}((\sigma^{s})^{\uparrow}) \\
&\qquad\qquad \cdot \prod_{k=1}^{n} J_k \cdot \alpha_k(\,(s^{\beta}, s^{k^{\neg\beta}}),\, a_k,\, (s^{\beta}, s^{k^{\neg\beta}})\,) \\
&= \sum_{\{a,s'^{\beta} \mid \beta((s^{\beta},s^{\neg\beta}),\, a,\, (s'^{\beta},s^{\neg\beta}))>0\}} \beta((s^{\beta}, s^{\neg\beta}),\, a,\, (s'^{\beta}, s^{n^{\neg\beta}})) \;\cdot\; \mathrm{PR}^{\eta}((\sigma^{s})^{\uparrow}) \\
&\qquad\qquad \cdot \prod_{k=1}^{n} J_k \cdot \alpha_k(\,(s^{\beta}, s^{k^{\neg\beta}}),\, a_k,\, (s^{\beta}, s^{k^{\neg\beta}})\,) \\
&= \sum_{\sigma' \in \mathcal{C}(\sigma)} \mathrm{PR}^{\eta[\sigma^{s} \leftarrow \beta]}((\sigma')^{\uparrow})
\end{aligned}
$$

□

Proof of Lemma C.5. For all $m > \mathrm{LEN}(\sigma)$, let Z_m be the set of infinite paths

$$
\{\sigma.\gamma_1.b_1.s^1 \cdots \gamma_{m-\mathrm{LEN}(\sigma)}.b_{m-\mathrm{LEN}(\sigma)}.s^{m-\mathrm{LEN}(\sigma)} \cdot \gamma_{m-\mathrm{LEN}(\sigma)+1}.b_{m-\mathrm{LEN}(\sigma)+1} \cdots \mid \bigwedge_{k=1}^{m-\mathrm{LEN}(\sigma)} L(s^k) = \ell_n\} .
$$

Then, $\mathcal{Z}(\sigma) = \bigcap_{m=\mathrm{LEN}(\sigma)+1}^{\infty} Z_m$. This equality, together with $Z_{m+1} \supseteq Z_m$, implies

$$
\mathrm{PR}^{\eta}(\mathcal{Z}(\sigma)) = \lim_{m\to\infty} \mathrm{PR}^{\eta}(Z_m) . \tag{C.47}
$$

Note that $L(s^q) = \ell_n$ is not required for $q > m - \mathrm{LEN}(\sigma)$.

Let E_m be the set of finite paths

$$
\{\sigma_Z \mid \sigma_Z \sim \sigma.\gamma_1.b_1.s^1 \cdots \gamma_{m-\mathrm{LEN}(\sigma)}.b_{m-\mathrm{LEN}(\sigma)}.s^{m-\mathrm{LEN}(\sigma)} \wedge \bigwedge_{k=1}^{m-\mathrm{LEN}(\sigma)} L(s^k) = \ell_n \wedge \mathrm{PR}^{\eta}((\sigma_Z)^{\uparrow}) \neq 0\} .
$$

Note that the paths in E_m have length m.

Then,

$$
\mathrm{PR}^{\eta}(Z_m) = \sum_{\sigma_Z \in E_m} \mathrm{PR}^{\eta}((\sigma_Z)^{\uparrow}) . \tag{C.48}
$$

Similarly, for $\mathcal{C}(\sigma)$ we define

$$
Z'_m = \biguplus_{\sigma' \in \mathcal{C}(\sigma)} \{\sigma'.\gamma_1.b_1.s_1 \cdots \gamma_{m-\mathrm{LEN}(\sigma')}.b_{m-\mathrm{LEN}(\sigma')}.s^{m-\mathrm{LEN}(\sigma')} \cdot \gamma_{m-\mathrm{LEN}(\sigma')+1} \cdots \mid \bigwedge_{k=1}^{m-\mathrm{LEN}(\sigma')} L(s^k) = \ell_n\}
$$

for all $m > \text{LEN}(\sigma) + 1$. Then,

$$\text{PR}^{\eta}(\biguplus_{\sigma' \in \mathcal{C}(\sigma)} \mathcal{Z}(\sigma')) = \lim_{n \to \infty} \text{PR}^{\eta}(Z'_m)\,. \tag{C.49}$$

By defining

$$E'_m = \{\sigma_{Z'} \mid \sigma_{Z'} \sim \sigma'.\gamma_1.b_1 \cdots \gamma_{m-\text{LEN}(\sigma')}.b_{m-\text{LEN}(\sigma')}.s^{m-\text{LEN}(\sigma')} \\ \wedge\, \sigma' \in \mathcal{C}(\sigma_Z) \wedge \bigwedge_{k=1}^{m-\text{LEN}(\sigma')} L(s^k) = \ell_n \wedge \text{PR}^{\eta[\sigma^s \leftarrow \beta]}((\sigma_{Z'})^{\uparrow}) \neq 0\}$$

we obtain

$$\text{PR}^{\eta[\sigma^s \leftarrow \beta]}(Z'_m) = \sum_{\sigma_{Z'} \in E'_m} \text{PR}^{\eta[\sigma^s \leftarrow \beta]}((\sigma_{Z'})^{\uparrow})\,. \tag{C.50}$$

Next, we show that

$$\sum_{\sigma_{Z'} \in E'_{m+1}} \text{PR}^{\eta[\sigma^s \leftarrow \beta]}((\sigma_{Z'})^{\uparrow}) \leqslant \sum_{\sigma_Z \in E_m} \text{PR}^{\eta}((\sigma_Z)^{\uparrow}) \leqslant \sum_{\sigma_{Z'} \in E'_m} \text{PR}^{\eta[\sigma^s \leftarrow \beta]}((\sigma_{Z'})^{\uparrow})\,. \tag{C.51}$$

By Eqn. (C.48) and Eqn. (C.50), this inequality implies $\text{PR}^{\eta[\sigma^s \leftarrow \beta]}(Z'_{m+1}) \leqslant \text{PR}^{\eta}(Z_m) \leqslant \text{PR}^{\eta[\sigma^s \leftarrow \beta]}(Z'_m)$. By Equations (C.47) and (C.49), we have

$$\text{PR}^{\eta}(\mathcal{Z}(\sigma)) = \sum_{\sigma' \in \mathcal{C}(\sigma)} \text{PR}^{\eta[\sigma^s \leftarrow \beta]}(\mathcal{Z}(\sigma'))\,,$$

which is what we want to prove.

In order to prove Eqn. (C.51) we start by the inequality

$$\sum_{\sigma_Z \in E_m} \text{PR}^{\eta}((\sigma_{Z'})^{\uparrow}) \leqslant \sum_{\sigma_Z \in E'_m} \text{PR}^{\eta[\sigma^s \leftarrow \beta]}((\sigma_{Z'})^{\uparrow})\,.$$

To this end, we explore how E_m relates to E'_m. Let's take $\sigma_Z \in E_m$. If $\sigma_Z \not\sim \triangleleft\alpha\triangleright$, then $\mathcal{C}(\sigma_Z) \subseteq E'_m$ (recall that, in these cases, $\mathcal{C}(\sigma_Z)$ is a singleton set). If σ_Z is of the form $\triangleleft\alpha\triangleright$, then $\mathcal{C}(\sigma_Z) \subseteq E'_{m+1}$ (since, in this case, $\mathcal{C}$ inserts β in σ_Z) and, in addition, $\sigma_{\downarrow_{-1}} \in E'_m$ for all $\sigma \in \mathcal{C}(\sigma_Z)$. Hence, the set

$$I = \{\sigma_{\downarrow_{-1}} \mid \exists \sigma_Z \in E_m : \sigma_Z \sim \triangleleft\alpha\triangleright \wedge \sigma \in \mathcal{C}(\sigma_Z)\}$$

obtained by dropping the last state of every path in $\bigcup_{\sigma_Z \in E_m \wedge \sigma_Z \sim \triangleleft\alpha\triangleright} \mathcal{C}(\sigma_Z)$ complies with

$$\biguplus_{\sigma Z \in E_m \wedge \sigma_Z \sim \triangleleft\alpha\triangleright} \biguplus_{\sigma_{Z'} \in \mathcal{C}(\sigma_Z)} (\sigma_{Z'})^{\uparrow} \subseteq \biguplus_{\sigma_{Z'} \in I} (\sigma_{Z'})^{\uparrow}\,. \tag{C.52}$$

In addition

$$\forall \sigma_Z \in E_M, \sigma_Z \not\sim \triangleleft\alpha\triangleright \;:\; I \cap \mathcal{C}(\sigma_Z) = \emptyset\,. \tag{C.53}$$

To see this, suppose that $\sigma_Z \not\sim \triangleleft\alpha\triangleright$, $\mathcal{C}(\sigma_Z) = \{\sigma_{Z'}\}$ and $\sigma_{Z'} \in I$. Since $\sigma_{Z'} \in I$, we know that $\sigma_{Z'} = \sigma'_{\alpha\downarrow_{-1}}$ for some path σ'_α such that $\sigma'_\alpha \in \mathcal{C}(\sigma_\alpha)$, where $\sigma_\alpha \sim \triangleleft\alpha\triangleright$ and $\sigma_\alpha \in E_m$. Since $\sigma'_\alpha \in \mathcal{C}(\sigma_\alpha)$, and σ_α is of the form $\triangleleft\alpha\triangleright$, and $\sigma_{Z'} = \sigma'_{\alpha\downarrow_{-1}}$, we have $\sigma_{Z'} \in \mathcal{C}(\sigma_{\alpha\downarrow_{-1}})$. In addition, since (by Lemma 7.4) $\mathcal{C}(\sigma_Z) \cap \mathcal{C}(\sigma_{\alpha\downarrow_{-1}}) = \{\sigma_{Z'}\}$, it must be $\sigma_Z = \sigma_{\alpha\downarrow_{-1}}.\beta.a.s'$ for some s'. However, $\eta(\sigma_{\alpha\downarrow_{-1}}) \neq 1{:}\beta$ (because σ_α is of the form $\triangleleft\alpha\triangleright$). So $\text{PR}^{\eta}(\sigma_Z) = 0$, which implies $\sigma_Z \notin E_m$. Hence, Eqn. (C.53) holds.

In addition to Eqn. (C.53), we need a disjointness property. Since in E_m we cannot have two paths σ_Z and $\sigma_Z.\beta.a.s$, Lemma 7.4 ensures that $\mathcal{C}$ maps distinct paths in E_m to disjoint set of paths, that is:

$$\forall \sigma_Z, \sigma_Q \in E_m : \sigma_Z \neq \sigma_Q \implies \mathcal{C}(\sigma_Z) \cap \mathcal{C}(\sigma_Q) = \emptyset\,. \tag{C.54}$$

These observations allow the following calculation:

$$\begin{aligned}
& \textstyle\sum_{\sigma_Z \in E_m} \mathrm{PR}^{\eta}((\sigma_Z)^{\uparrow}) \\
= \;& \{\ \text{Split sum}\ \} \\
& \textstyle\sum_{\sigma_Z \in E_m \wedge \sigma_Z \not\sim \triangleleft\alpha\triangleright} \mathrm{PR}^{\eta}((\sigma_Z)^{\uparrow}) + \sum_{\sigma_Z \in E_m \wedge \sigma_Z \sim \triangleleft\alpha\triangleright} \mathrm{PR}^{\eta}((\sigma_Z)^{\uparrow}) \\
= \;& \{\ \text{Lemma C.4}\ \} \\
& \textstyle\sum_{\sigma_Z \in E_m \wedge \sigma_Z \not\sim \triangleleft\alpha\triangleright} \mathrm{PR}^{\eta[\sigma^s \leftarrow \beta]}((\mathcal{C}(\sigma_Z))^{\uparrow}) \\
+ \;& \textstyle\sum_{\sigma_Z \in E_m \wedge \sigma_Z \sim \triangleleft\alpha\triangleright} \sum_{\sigma_{Z'} \in \mathcal{C}(\sigma_Z)} \mathrm{PR}^{\eta[\sigma^s \leftarrow \beta]}((\sigma_{Z'})^{\uparrow}) \\
\leqslant \;& \{\ \text{Inclusion (C.52)}\ \} \\
& \textstyle\sum_{\sigma_{Z'} \in E'_m \wedge \exists \sigma_Z \not\sim \triangleleft\alpha\triangleright : \{\sigma_{Z'}\} = \mathcal{C}(\sigma_Z)} \mathrm{PR}^{\eta[\sigma^s \leftarrow \beta]}((\sigma_{Z'})^{\uparrow}) + \sum_{\sigma_{Z'} \in I} \mathrm{PR}^{\eta[\sigma^s \leftarrow \beta]}((\sigma_{Z'})^{\uparrow}) \\
\leqslant \;& \{\ \text{Equation (C.53)}\ \} \\
& \textstyle\sum_{\sigma_{Z'} \in E'_m} \mathrm{PR}^{\eta[\sigma^s \leftarrow \beta]}((\sigma_{Z'})^{\uparrow})
\end{aligned}$$

Next, we consider the inequation $\sum_{\sigma_Z \in E_m} \mathrm{PR}^{\eta}((\sigma_Z)^{\uparrow}) \geqslant \sum_{\sigma_{Z'} \in E_{m+1}} \mathrm{PR}^{\eta[\sigma^s \leftarrow \beta]}((\sigma_{Z'})^{\uparrow})$. For the proof, we need the following inclusions:

$$\begin{aligned}
&\biguplus\{(\sigma_Z)^{\uparrow} \mid \sigma_Z \in E_m \wedge (\ \sigma_Z \sim \triangleleft\alpha\,\beta\triangleright \ \vee\ \sigma_Z \sim \triangleleft\alpha\,\beta\,\gamma\triangleright\ )\} \\
&\qquad\qquad \subseteq \biguplus\{(\sigma_Z)^{\uparrow} \mid \sigma_Z \in E_{m+1} \wedge \sigma \sim \triangleleft\alpha\,\beta\,\gamma\triangleright\}
\end{aligned} \tag{C.55}$$

$$\begin{aligned}
&\biguplus\{(\sigma_Z)^{\uparrow} \mid \sigma_Z \in E_m \wedge \sigma_Z \sim \triangleleft\alpha\triangleright \ \wedge \eta(\sigma_Z) = 1{:}\beta\} \\
&\qquad\qquad \subseteq \biguplus\{(\sigma_Z)^{\uparrow} \mid \sigma_Z \in E_{m+1} \wedge \sigma \sim \triangleleft\alpha\,\beta\triangleright\}
\end{aligned} \tag{C.56}$$

Inclusion (C.55) holds since, for all $\sigma_Z \in E_{m+1}$, $\sigma_Z \sim \triangleleft\alpha\,\beta\,\gamma\triangleright$, the path $\sigma_{\downarrow -1}$ is in E_m and is of the form $\triangleleft\alpha\,\beta\triangleright$ or $\triangleleft\alpha\,\beta\,\gamma\triangleright$. Inclusion (C.56) holds since, for all $\sigma_Z \in E_{m+1}$, $\sigma_Z \sim \triangleleft\alpha\,\beta\triangleright$, the path $\sigma_{\downarrow -1}$ is in E_m, is of the form $\triangleleft\alpha\triangleright$ and $\eta(\sigma_{\downarrow -1}) = 1{:}\beta$ (recall that the paths in E_m have positive probability in η).

In addition, we need to prove the following statement for all $\sigma_{Z'} \in E_{m+1}$

$$\begin{aligned}
&\exists \sigma_Z \in E_{m+1} \mid \sigma_Z \not\sim \triangleleft\alpha\triangleright \ \wedge \{\sigma_{Z'}\} = \mathcal{C}(\sigma_Z) \\
\vee\ &\exists \sigma_Z \in E_m \mid \sigma_Z \sim \triangleleft\alpha\triangleright \ \wedge \eta(\sigma_Z) \neq 1{:}\beta \wedge \{\sigma_{Z'}\} \in \mathcal{C}(\sigma_Z)
\end{aligned} \tag{C.57}$$

In order to prove this claim, note that $\mathrm{PR}^{\eta[\sigma^s \leftarrow \beta]}(\sigma_{Z'}) \neq 0$ (since $\sigma_{Z'} \in E'_m$). So, Lemma 7.5 ensures the existence of some σ_Z with nonzero probability such that $\sigma_{Z'} \in \mathcal{C}(\sigma_Z)$. Since the changes introduced by $\mathcal{C}$ yield a stuttering equivalent labelling, and $\mathcal{C}$ can insert one action, we have that $\sigma_Z \in E_m \cup E_{m+1}$. If $\sigma_Z \in E_{m+1}$, and $\sigma_{Z'} \in \mathcal{C}(\sigma_Z)$, we conclude that $\sigma_{Z'} \sim \triangleleft\alpha\,\beta\triangleright$ or $\sigma_Z \sim \triangleleft\alpha\,\beta\,\gamma\triangleright$, and

$$\sigma_Z \in E_{m+1} \wedge \sigma_Z \not\sim \triangleleft\alpha\triangleright \ \wedge \{\sigma_{Z'}\} = \mathcal{C}(\sigma_Z) \tag{C.58}$$

holds. If $\sigma_Z \in E_m$, then $\sigma_Z \sim \triangleleft\alpha\triangleright$. If $\eta(\sigma_Z) = 1{:}\beta$, we have a path $\sigma'' \in E_{m+1}$ of the form $\sigma_Z.\beta.a.s'$ such that $\sigma_{Z'} \in \mathcal{C}(\sigma'')$. So, if $\eta(\sigma_Z) = 1{:}\beta$, the case (C.58) holds. If $\eta(\sigma_Z) \neq 1{:}\beta$, we are in the case

$$\sigma_Z \in E_m \wedge \sigma_Z \sim \triangleleft\alpha\triangleright \ \wedge \eta(\sigma_Z) \neq 1{:}\beta \wedge \{\sigma_{Z'}\} \in \mathcal{C}(\sigma_Z)$$

Now, we are able to prove the inequality.

$$
\begin{aligned}
& \textstyle\sum_{\sigma_Z \in E_m} \mathrm{PR}^{\eta}((\sigma_Z)^{\uparrow}) \\
\geqslant\ & \{\ \text{Split sum}\ \} \\
& \quad \textstyle\sum_{\sigma_Z \in E_m \wedge \sigma_Z \sim \lhd \neg \sigma^s \rhd} \mathrm{PR}^{\eta}((\sigma_Z)^{\uparrow}) \\
& + \textstyle\sum_{\sigma_Z \in E_m \wedge (\sigma_Z \sim \lhd \alpha\ \beta \rhd \ \vee\ \sigma_Z \sim \lhd \alpha\ \beta\ \gamma \rhd)} \mathrm{PR}^{\eta}((\sigma_Z)^{\uparrow}) \\
& + \textstyle\sum_{\sigma_Z \in E_m \wedge \sigma_Z \sim \lhd \alpha \rhd \wedge \eta(\sigma_Z) = 1 : \beta} \mathrm{PR}^{\eta}((\sigma_Z)^{\uparrow}) \\
& + \textstyle\sum_{\sigma_Z \in E_m \wedge \sigma_Z \sim \lhd \alpha \rhd \wedge \eta(\sigma_Z) \neq \beta} \mathrm{PR}^{\eta}((\sigma_Z)^{\uparrow}) \\
\geqslant\ & \{\ \text{Inclusions (C.55) and (C.56)}\ \} \\
& \quad \textstyle\sum_{\sigma_Z \in E_m \wedge \sigma_Z \sim \lhd \neg \sigma^s \rhd} \mathrm{PR}^{\eta[\sigma^s \leftarrow \beta]}((\sigma_Z)^{\uparrow}) \\
& + \textstyle\sum_{\sigma_Z \in E_{m+1} \wedge \sigma_Z \sim \lhd \alpha\ \beta\ \gamma \rhd} \mathrm{PR}^{\eta}((\sigma_Z)^{\uparrow}) \\
& + \textstyle\sum_{\sigma_Z \in E_{m+1} \wedge \sigma_Z \sim \lhd \alpha\ \beta \rhd} \mathrm{PR}^{\eta}((\sigma_Z)^{\uparrow}) \\
& + \textstyle\sum_{\sigma_Z \in E_m \wedge \sigma_Z \sim \lhd \alpha \rhd \wedge \eta(\sigma_Z) \neq 1 : \beta} \mathrm{PR}^{\eta}((\sigma_Z)^{\uparrow}) \\
= \ & \{\ \text{Lemma C.4}\ \} \\
& \quad \textstyle\sum_{\sigma_Z \in E_m \wedge \sigma_Z \sim \lhd \neg \sigma^s \rhd} \mathrm{PR}^{\eta[\sigma^s \leftarrow \beta]}(\mathcal{C}(\sigma_Z)) \\
& + \textstyle\sum_{\sigma_Z \in E_{m+1} \wedge (\sigma_Z \sim \lhd \alpha\ \beta \rhd \ \vee\ \sigma_Z \sim \lhd \alpha\ \beta\ \gamma \rhd)} \mathrm{PR}^{\eta[\sigma^s \leftarrow \beta]}((\mathcal{C}(\sigma_Z))^{\uparrow}) \\
& + \textstyle\sum_{\sigma_Z \in E_m \wedge \sigma_Z \sim \lhd \alpha \rhd \wedge \eta(\sigma_Z) \neq 1 : \beta} \sum_{\sigma_{Z'} \in \mathcal{C}(\sigma_Z)} \mathrm{PR}^{\eta[\sigma^s \leftarrow \beta]}((\sigma_{Z'})^{\uparrow}) \\
\geqslant\ & \{\ \text{Rewrite}\ \} \\
& \quad \textstyle\sum_{\sigma_{Z'} \in E'_{m+1} \wedge \exists \sigma_Z\, :\, \{\sigma_{Z'}\} = \mathcal{C}(\sigma_Z) \wedge \sigma_Z \not\sim \lhd \alpha \rhd} \mathrm{PR}^{\eta[\sigma^s \leftarrow \beta]}((\sigma_{Z'})^{\uparrow}) \\
& + \textstyle\sum_{\sigma_{Z'} \in E'_{m+1} \wedge \exists \sigma_Z\, :\, \sigma_Z \sim \lhd \alpha \rhd \wedge \eta(\sigma_Z) \neq 1 : \beta} \mathrm{PR}^{\eta[\sigma^s \leftarrow \beta]}((\sigma_{Z'})^{\uparrow}) \\
\geqslant\ & \{\ \text{Statement (C.57)}\ \} \\
& \quad \textstyle\sum_{\sigma_{Z'} \in E'_{m+1}} \mathrm{PR}^{\eta[\sigma^s \leftarrow \beta]}((\sigma_{Z'})^{\uparrow})
\end{aligned}
$$

So, we have proven Eqn. (C.51), which implies

$$\mathrm{PR}^{\eta}(\mathcal{Z}(\sigma)) = \sum_{\sigma' \in \mathcal{C}(\sigma)} \mathrm{PR}^{\eta[\sigma^s \leftarrow \beta]}(\mathcal{Z}(\sigma'))\ .$$

□

Proof of Lemma 7.7. Let $\mathcal{U} = \{(\sigma)^{\uparrow} \mid \textsc{Trace}(\sigma) \sim \ell_1^+, \cdots, \ell_n^+ \ell' \ \wedge\ \ell_n \neq \ell'\} \cup \{\mathcal{Z}(\sigma) \mid \textsc{Trace}(\sigma) \sim \ell_1^+, \cdots, \ell_{n-1}^+\}$. From Eqn. (C.45), it suffices to prove:

$$\sum_{U \in \mathcal{U}} \mathrm{PR}^{\eta}(U) = \sum_{U \in \mathcal{U}} \mathrm{PR}^{\eta[\sigma^s \leftarrow \beta]}(U)\ .$$

To this end, we define a function $\mathcal{M}$ mapping the positive terms in $\sum_{U \in \mathcal{U}} \mathrm{PR}^{\eta}(U)$ to sets of positive terms in $\sum_{U \in \mathcal{U}} \mathrm{PR}^{\eta[\sigma^s \leftarrow \beta]}(U)$. Using $\mathcal{M}$, we show that $\sum_{U \in \mathcal{U}} \mathrm{PR}^{\eta}(U)$ is a reordering of $\sum_{U \in \mathcal{U}} \mathrm{PR}^{\eta[\sigma^s \leftarrow \beta]}(U)$.

We define $\mathcal{M}$ as $\mathcal{M}((\sigma)^{\uparrow}) = \{(\sigma')^{\uparrow} \mid \sigma' \in \mathcal{C}(\sigma)\}$, $\mathcal{M}(\mathcal{Z}(\sigma)) = \{\mathcal{Z}(\sigma') \mid \sigma' \in \mathcal{C}(\sigma)\}$. This function complies with the following properties:

1. $\mathrm{PR}^{\eta}(U) \neq 0 \wedge \mathrm{PR}^{\eta}(U') \neq 0 \wedge U \in \mathcal{U} \wedge U' \in \mathcal{U} \wedge \mathcal{M}(U) \cap \mathcal{M}(U') \neq \emptyset \implies U = U'$
2. $\mathrm{PR}^{\eta}(U) = \sum_{U' \in \mathcal{M}(U)} \mathrm{PR}^{\eta[\sigma^s \leftarrow \beta]}(U')$
3. for all U' such that $\mathrm{PR}^{\eta[\sigma^s \leftarrow \beta]}(U') > 0$ there exists U such that $\mathrm{PR}^{\eta}(U) > 0$ and $U' \in \mathcal{M}(U)$.

For the proof of property *(1)* suppose, towards a contradiction, that $\mathcal{M}(\mathcal{U}) \cap \mathcal{M}(\mathcal{U}') \neq \emptyset$, $\mathrm{PR}^{\eta}(\mathcal{U}) \neq 0$, $\mathrm{PR}^{\eta}(\mathcal{U}') \neq 0$, and $\mathcal{U} \neq \mathcal{U}'$. Then, by definition of $\mathcal{M}$, it must be either ($\mathcal{U} = (\sigma)^{\uparrow}$ and $\mathcal{U}' = (\sigma')^{\uparrow}$) or ($\mathcal{U} = \mathcal{Z}(\sigma)$ and $\mathcal{U}' = \mathcal{Z}(\sigma')$). In the first case, $\sigma = \sigma'$ by Lemma 7.4, since neither σ nor σ' can be of the form $\triangleleft\alpha\,\beta\triangleright$ (because $\textsc{Trace}(\sigma) \sim \ell_1^+, \cdots, \ell_n^+\ell'$ with $\ell_n \neq \ell'$, and β is stutter). In the second case, it must be $\mathcal{C}(\sigma) \cap \mathcal{C}(\sigma') \neq \emptyset$. By Lemma 7.4, it is possible only if $\sigma = \sigma^s.\alpha_1.a_1.(s^{1^\beta}, s^{1^{\neg\beta}}) \cdots \alpha_n.a_n.(s^{n^\beta}, s^{n^{\neg\beta}})$ and $\sigma' = \sigma.\beta.(s'^{\beta}, s'^{\neg\beta})$ for some α_k, s^k (or, symmetrically, σ is the path that finishes with β). Since $\mathrm{PR}^{\eta}(\sigma') > 0$, we have that $\eta(\sigma^s.\alpha_1.a_1.(s^{1^\beta}, s^{1^{\neg\beta}}) \cdots \alpha_n.a_n.(s_n^{n^\beta}, s^{n^{\neg\beta}})) = \beta$. In addition, $\mathsf{L}((s^{n\beta}, s^{n^{\neg\beta}})) = \ell_{n-1}$ (see the definition of $\mathcal{U}$ above) and, since β is stutter, the paths of the form

$$\sigma^s.\alpha_1.a_1.(s^{1^\beta}, s^{1^{\neg\beta}}) \cdots \alpha_n.a_n.(s^{n^\beta}, s^{n^{\neg\beta}})\beta.a.s'$$

with $\mathsf{L}(s') \neq \ell_n$ have probability 0. Then, all the paths in $\mathcal{Z}(\sigma)$ start with a prefix having probability 0. Hence, $\mathrm{PR}^{\eta}(\mathcal{U}) = \mathrm{PR}^{\eta}(\mathcal{Z}(\sigma)) = 0$.

Property *(2)* holds by Lemma C.4 (in case $\mathcal{U} = (\sigma)^{\uparrow}$) and Lemma C.5 (in case $\mathcal{U} = \mathcal{Z}(\sigma)$).

For property *(3)*, in case $\mathcal{U}' = (\sigma')^{\uparrow}$ the claim becomes Lemma 7.5. In case $\mathcal{U}' = \mathcal{Z}(\sigma')$, since $\mathrm{PR}^{\eta[\sigma^s \leftarrow \beta]}((\sigma')^{\uparrow}) > 0$, there exists σ such that $\sigma' \in \mathcal{C}(\sigma)$ and $\mathrm{PR}^{\eta}((\sigma)^{\uparrow}) > 0$. Then, $\mathcal{U}' \in \mathcal{M}(\mathcal{Z}(\sigma))$. Because of the property **(2)** for $\mathcal{M}$ proven above, we have the inequality

$$0 < \mathrm{PR}^{\eta[\sigma^s \leftarrow \beta]}(\mathcal{U}') \leqslant \sum_{\mathcal{U}'' \in \mathcal{C}(\mathcal{Z}(\sigma))} \mathrm{PR}^{\eta[\sigma^s \leftarrow \beta]}(\mathcal{U}'') = \mathrm{PR}^{\eta}(\mathcal{Z}(\sigma)) \,.$$

So, we can take $\mathcal{U} = \mathcal{Z}(\sigma)$.

By properties *(1)* and *(3)* we have

$$\sum_{\mathcal{U}' \in \mathcal{U}} \mathrm{PR}^{\eta[\sigma^s \leftarrow \beta]}(\mathcal{U}') = \sum_{\{\mathcal{U}' \in \mathcal{U} \,|\, \mathrm{PR}^{\eta[\sigma^s \leftarrow \beta]}(\mathcal{U}') > 0\}} \mathrm{PR}^{\eta[\sigma^s \leftarrow \beta]}(\mathcal{U}') = \sum_{\{\mathcal{U} \in \mathcal{U} \,|\, \mathrm{PR}^{\eta}(\mathcal{U}) > 0\}} \sum_{\mathcal{U}' \in \mathcal{M}(\mathcal{U})} \mathrm{PR}^{\eta[\sigma^s \leftarrow \beta]}(\mathcal{U}') \,.$$

By property *(2)*,

$$\sum_{\{\mathcal{U} \in \mathcal{U} \,|\, \mathrm{PR}^{\eta}(\mathcal{U}) > 0\}} \sum_{\mathcal{U}' \in \mathcal{M}(\mathcal{U})} \mathrm{PR}^{\eta[\sigma^s \leftarrow \beta]}(\mathcal{U}') = \sum_{\{\mathcal{U} \in \mathcal{U} \,|\, \mathrm{PR}^{\eta}(\mathcal{U}) > 0\}} \mathrm{PR}^{\eta}(\mathcal{U}) = \sum_{\mathcal{U} \in \mathcal{U}} \mathrm{PR}^{\eta}(\mathcal{U}) \,.$$

□

D

PROOFS OF CHAPTER 9

D.1 THEOREM 9.1

Given an MDP M, *let* $P = \|_{\alpha \in \text{ACTIONS}_M} A_\alpha$. *Moreover, let* $\widehat{P}$ *be a reduction of* MDP(P) *complying with conditions* **A1**–**A4**, *by taking the independence relation to be* $\mathbb{I}$. *Then,*

The definition of MDP(P) *is Def. 6.2 in p. 108*

$$\sup_{\eta \in \text{SDIST}_P(\langle\!|\cdot|\!\rangle)} \Pr^\eta(\phi) \leqslant \sup_{\eta \in \text{SCHED}_{\widehat{P}}} \Pr^\eta(\phi) \,.$$

We prove this theorem in the same way as we did for Theorem 7.2, that is, we resort to Theorem 7.3 using the full-communication projection $\langle\!|\cdot|\!\rangle$. To this aim, we propose an α projection $(\pi^\alpha, \pi^{\neg\alpha})$ and prove that $(\mathbb{I}, \{(\pi^\alpha, \pi^{\neg\alpha})\}_\alpha, \langle\!|\cdot|\!\rangle)$ is an independence structure. First, we show that $(\mathbb{I}, \{(\pi^\alpha, \pi^{\neg\alpha})\}_\alpha, (\!|\cdot|\!))$ is an independence structure, and then we extend the result to the projection $\langle\!|\cdot|\!\rangle$ using Lemma C.2.

The following definition introduces the α-projections used in the proof.

Definition D.1. For all $\alpha \in \text{ACTIONS}_M$, $s \in S_M$, we define $\pi^\alpha(s)$ to be the valuation $V \in \mathbb{V}(\text{WRITEVAR}(\alpha))$ such that $V(v) = s(v)$ for all $v \in \text{WRITEVAR}(\alpha)$. In addition, $\pi^{\neg\alpha}(s)$ is defined as the valuation $V \in \mathbb{V}(\mathcal{V}_M \setminus \text{WRITEVAR}(\alpha))$ such that $V(s) = s(v)$ for all $v \in \mathcal{V}_M \setminus \text{WRITEVAR}(\alpha)$.

Lemma D.1. *The triple* $(\mathbb{I}, \{(\pi^\alpha, \pi^{\neg\alpha})\}_\alpha, (\!|\cdot|\!))$ *is an independence structure.*

Proof. In this proof, α and β are any two actions such that $\alpha \mathcal{R}_I \beta$.

We prove each of the properties that an independence structure must comply with:

Property *(1)*: if $\alpha(s, a, s') > 0$, then $s'(v) = s(v)$ for all $v \notin \text{WRITEVAR}(\alpha)$. Since α and β are independent, we have $v \notin \text{WRITEVAR}(\alpha)$ for all $v \in \text{READVAR}(\beta)$. Hence, $s'(v) = s(v)$ for all $v \in \text{READVAR}(\beta)$, and property *(1)* follows from Equation (9.1).

Property *(2)*: if $\alpha(\overbrace{(s^\alpha, s^{\neg\alpha})}^{s}, a, \overbrace{(s'^\alpha, s'^{\neg\alpha})}^{s'}) > 0$, then $s'(v) = s(v)$ for all $v \notin \text{WRITEVAR}(\alpha)$. Therefore, $\pi^{\neg\alpha}(s) = \pi^{\neg\alpha}(s')$.

Property *(3)*: if $\beta(\overbrace{(s^\alpha, s^{\neg\alpha})}^{s}, a, \overbrace{(s'^\alpha, s'^{\neg\alpha})}^{s'}) > 0$, then $s'(v) = s(v)$ for all $v \notin \text{WRITEVAR}(\beta)$. By $\alpha \mathbb{I} \beta$, we have

$$v \in \text{WRITEVAR}(\alpha) \implies v \notin \text{WRITEVAR}(\beta) \,.$$

Hence, $s'(v) = s(v)$ for all $v \in \text{WRITEVAR}(\alpha)$. Therefore $\pi^\alpha(s) = \pi^\alpha(s')$.

Property *(4)*: first, we consider the case in which $s''^{\neg\alpha}(v) \neq s^{\neg\alpha}(v)$, for some $v \in \text{VAR}(\alpha)$. Since $\alpha \mathbb{I} \beta$ implies $v \notin \text{WRITEVAR}(\beta)$, we have

$$\begin{aligned} & \beta(\,(s'^\alpha, s^{\neg\alpha})\,, b\,, (s'^\alpha, s''^{\neg\alpha})\,) \\ = \; & \beta(\,(s^\alpha, s^{\neg\alpha})\,, b\,, (s^\alpha, s''^{\neg\alpha})\,) \\ = \; & 0 \end{aligned}$$

and the required equality holds.

The second case we consider is that in which $s'^{\alpha}(v) \neq s^{\alpha}(v)$ for some $v \in \text{VAR}(\beta)$. Since $\alpha \mathbb{I} \beta$ implies $v \notin \text{WRITEVAR}(\alpha)$, we have

$$\begin{aligned} & \alpha(\,(s^{\alpha}, s^{\neg\alpha}),\, a,\, (s'^{\alpha}, s^{\neg\alpha})\,) \\ = \;& \alpha(\,(s^{\alpha}, s''^{\neg\alpha}),\, a,\, (s'^{\alpha}, s''^{\neg\alpha}) \\ = \;& 0 \end{aligned}$$

and the required equality holds.

Now, we consider the case in which

$$\forall v \in \text{VAR}(\alpha) \,:\, s''^{\neg\alpha}(v) = s^{\neg\alpha}(v) \tag{D.1}$$

$$\forall v \in \text{VAR}(\beta) \,:\, s'^{\alpha}(v) = s^{\alpha}(v) \tag{D.2}$$

Recall that the labels in the system are of the form a_V (definition of ACTLAB_α). Suppose that $a = c_V$ and $b = d_{V'}$. If $V \neq (s'^{\alpha})\big|_{\text{WRITEVAR}(\alpha)}$ or $V' \neq (s''^{\alpha})\big|_{\text{WRITEVAR}(\beta)}$, then both sides of the equality is 0. Hence, assume $V = (s'^{\alpha})\big|_{\text{WRITEVAR}(\alpha)}$ and $V' = (s''^{\alpha})\big|_{\text{WRITEVAR}(\beta)}$.

$$\begin{aligned} & \alpha(\,(s^{\alpha}, s^{\neg\alpha}),\, a_V,\, (s'^{\alpha}, s^{\neg\alpha})\,) \cdot \beta(\,(s'^{\alpha}, s^{\neg\alpha}),\, b_{V'},\, (s'^{\alpha}, s''^{\neg\alpha})\,) \\ = \;& \{\; \text{Equations (D.1),(9.3),} \\ & \quad \alpha \text{ only writes variables in } \text{WRITEVAR}(\alpha), \text{ definition of } s^{\neg\alpha} \;\} \\ & \alpha(\,(s^{\alpha}, s''^{\neg\alpha}),\, a_V,\, (s'^{\alpha}, s''^{\neg\alpha})\,) \cdot \beta(\,(s'^{\alpha}, s^{\neg\alpha}),\, b_{V'},\, (s'^{\alpha}, s''^{\neg\alpha})\,) \\ = \;& \{\; \text{Equations (D.2), (9.3),} \\ & \quad \text{WRITEVAR}(\beta) \cap \text{WRITEVAR}(\alpha) = \emptyset, \text{ definition of } s^{\alpha} \;\} \\ & \alpha(\,(s^{\alpha}, s''^{\neg\alpha}),\, a_V,\, (s'^{\alpha}, s''^{\neg\alpha})\,) \cdot \beta(\,(s^{\alpha}, s^{\neg\alpha}),\, b_{V'},\, (s^{\alpha}, s''^{\neg\alpha})\,) \\ = \;& \beta(\,(s^{\alpha}, s^{\neg\alpha}),\, b_{V'},\, (s^{\alpha}, s''^{\neg\alpha})\,) \cdot \alpha(\,(s^{\alpha}, s''^{\neg\alpha}),\, a_V,\, (s'^{\alpha}, s''^{\neg\alpha})\,) \end{aligned}$$

Actually, it might be difficult to find where we have restricted $\mathbb{I}$ so that it cannot be $\alpha \mathbb{I} \alpha$. The answer is (9.2)

Property *(5)*: if $\alpha \mathbb{I} \beta$, we have $\alpha \neq \beta$. Hence, $\text{ACTIVE}(\alpha) = A_\alpha \neq A_\beta = \text{ACTIVE}(\beta)$.

Property *(6)*: the only atom in $\text{AFFECT}(\beta)$ is A_β: although there might some atoms A_j such that $a_V \in \text{ACTLAB}_j$, we have $\left|\, R_j(s, a_V) \,\right| = 1$ for all s, j, and so $A_j \notin \text{AFFECT}(\beta)$. For all k, let $a^k = d^k_{V^k}$, where $d^k \in \mathcal{L}_M$ and $V^k \in \text{WRITEVAR}(\alpha^k)$. Since $\alpha^k \mathbb{I} \beta$, we have

$$d^k \notin \text{READLAB}(\beta) \;\wedge\; \text{WRITEVAR}(\alpha^k) \cap \text{VAR}(\beta) = \emptyset\,. \tag{D.3}$$

Hence

$$\begin{aligned} & (\!|\sigma.\alpha^1.a^1.s^1.\cdots.\alpha^n.a^n.s^n|\!)_\beta \\ = \;& (\sigma.\alpha^1.a^1.s^1.\cdots.\alpha^n.a^n.s^n)\big|_\beta \\ = \;& \{\; \text{Definition of } (\cdot)\big|, \text{ Eqn. (D.3)} \;\} \\ = \;& (\sigma)\big|_\beta \\ = \;& (\!|\sigma|\!)_\beta \end{aligned}$$

Property *(7)*: if

$$\begin{aligned} & (\!|\sigma.\alpha^1.a^1.(s^\beta, s^{1\neg\beta}) \cdots \alpha^n.a^n.(s^\beta, s^{n\neg\beta})|\!)_\gamma \\ \neq \;& (\!|\sigma.\alpha'^1.a'^1.(s^\beta, s'^{1\neg\beta}) \cdots \alpha'^{n'}.a'^{n'}.(s^\beta, s'^{n'\neg\beta})|\!)_\gamma \end{aligned}$$

then

$$(\!|\sigma.\alpha^1.a^1.(s^\beta, s^{1\neg\beta}) \cdots \alpha^n.a^n.(s^\beta, s^{n\neg\beta})|\!)_\gamma = (\sigma)\big|_\gamma.b^1.s^1_\gamma.\cdots.b^m.s^m_\gamma$$

and

$$(\!|\sigma.\alpha'^1.a'^1.(s^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^{n'}.a'^{n'}.(s^\beta, s'^{n'^{\neg\beta}})|\!)_\gamma = (\sigma)\big|_\gamma.b'^1.s'^1_\gamma.\cdots.b'^m.s'^m_\gamma$$

for some $b^k, b'^k \in \mathcal{L}_M \cup \{\kappa\}$, $s^k_\gamma, s'^k_\gamma \in \mathbb{V}(\text{VAR}(\gamma))$ such that

$$b^1.s^1_\gamma.\cdots.b^m.s^m_\gamma \neq b'^1.s'^1.\cdots.b'^{m'}.s'^{m'}\,.$$

We consider two cases. In case $a \notin \text{ACTLAB}_\gamma$, we have

$$\begin{aligned}
&(\!|\sigma.\beta.a.(t^\beta, s^{\neg\beta}).\alpha^1.a^1.(t^\beta, s^{1^{\neg\beta}}) \cdots \alpha^n.a^n.(t^\beta, s^{n^{\neg\beta}})|\!)_\gamma \\
&= (\sigma)\big|_\gamma.b^1.s^1_\gamma.\cdots.b^m.s^{m'}_\gamma \\
&\neq (\sigma)\big|_\gamma.b'^1.s'^1_\gamma.\cdots.b'^{m'}.s'^{m'}_\gamma \\
&= (\!|\sigma.\beta.a.(t'^\beta, s^{\neg\beta}).\alpha'^1.a'^1.(t'^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^{n'}.a'^{n'}.(t'^\beta, s'^{n'^{\neg\beta}})|\!)_\gamma
\end{aligned}$$

In case $a \in \text{ACTLAB}_\gamma$, there exist local states $v_\gamma, v'_\gamma \in \mathbb{V}(\text{VAR}(\gamma))$, $d, d' \in \mathcal{L}_M \cup \{\kappa\}$ such that

$$\begin{aligned}
&(\!|\sigma.\beta.a.(t^\beta, s^{\neg\beta}).\alpha^1.a^1.(t^\beta, s^{1^{\neg\beta}}) \cdots \alpha^n.a^n.(t^\beta, s^{n^{\neg\beta}})|\!)_\gamma \\
&\qquad\qquad = (\sigma)\big|_\gamma.d.v_\gamma.b^1.s^1_\gamma.\cdots.b^m.s^m_\gamma
\end{aligned}$$

and

$$\begin{aligned}
&(\!|\sigma.\beta.a.(t'^\beta, s^{\neg\beta}).\alpha'^1.a'^1.(t'^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^{n'}.a'^{n'}.(t'^\beta, s'^{n'^{\neg\beta}})|\!)_\gamma \\
&\qquad\qquad = (\sigma)\big|_\gamma.d'.v'_\gamma.b'^1.s'^1_\gamma.\cdots.b'^{m'}.s'^{m'}_\gamma
\end{aligned}$$

Then,

$$\begin{aligned}
&(\!|\sigma.\beta.a.(t^\beta, s^{\neg\beta}).\alpha^1.a^1.(t^\beta, s^{1^{\neg\beta}}) \cdots \alpha^n.a^n.(t^\beta, s^{n^{\neg\beta}})|\!)_\gamma \\
&= (\sigma)\big|_\gamma.d.v_\gamma.b^1.s^1_\gamma.\cdots.b^m.s^m_\gamma \\
&\neq \{\ b^1.s^1_\gamma.\cdots.b^m.s^m_\gamma \neq b'^1.s'^1_\gamma.\cdots.b'^{m'}.s'^{m'}_\gamma \text{ implies} \\
&\qquad d.v_\gamma.b^1.s^1_\gamma.\cdots.b^m.s^m_\gamma \neq d'.v'_\gamma.b'^1.s'^1_\gamma.\cdots.b'^{m'}.s'^{m'}_\gamma\ \} \\
&\quad (\sigma)\big|_\gamma.d'.v'_\gamma.b'^1.s'^1_\gamma.\cdots.b'^{m'}.s'^{m'}_\gamma \\
&= (\!|\sigma.\beta.a.(t'^\beta, s^{\neg\beta}).\alpha'^1.a'^1.(t'^\beta, s'^{1^{\neg\beta}}) \cdots \alpha'^{n'}.a'^{n'}.(t'^\beta, s'^{n'^{\neg\beta}})|\!)_\gamma
\end{aligned}$$

Property *(8)* follows as in Lemma 7.1. □

The same argument used to prove that $[\![\cdot]\!]$ complies with (C.6), (C.7) and (C.8) can be applied for $(\!|\cdot|\!)$, and so Lemma C.2 implies:

LEMMA D.2. *The triple* $(\mathbb{I}, \{(\pi^\alpha, \pi^{\neg\alpha})\}_\alpha, |\!(\cdot)\!|)$ *is an independence structure.*

Proof of Theorem 9.1. The independence structure in Lemma D.2 complies with the hypotheses of Theorem 7.3 since, by Theorem 4.5, the projection $|\!(\cdot)\!|$ is traceable. By applying Theorem 7.3 to $|\!(\cdot)\!|$) we get

$$\sup_{\eta \in \text{SDIST}_P(|\!(\cdot)\!|)} \text{PR}^\eta(\phi) \leqslant \sup_{\eta \in \text{SCHED}_{\widehat{P}}} \text{PR}^\eta(\phi)\,. \tag{D.4}$$

By Theorems 4.4 and 2.3:

$$\sup_{\eta \in \text{SDIST}_P((\!|\cdot|\!))} \text{PR}^\eta(\phi) \leqslant \sup_{\eta \in \text{SDIST}_P(|\!(\cdot)\!|)} \text{PR}^\eta(\phi)\,. \tag{D.5}$$

From (D.4) and (D.5), we obtain the desired inequality:

$$\sup_{\eta \in \mathrm{SDIST}_{\mathcal{P}}(\!(\cdot)\!)} \mathrm{PR}^{\eta}(\phi) \leqslant \sup_{\eta \in \mathrm{SCHED}_{\widehat{\mathcal{P}}}} \mathrm{PR}^{\eta}(\phi)\,.$$

□

GLOSSARY (INCLUDING SYMBOLS AND NOTATIONS)

For each entry, we include a brief explanation and a list of page numbers. The first number corresponds to the page in which the symbol in the entry is introduced. The remaining numbers correspond to selected occurrences of the symbol. In page xv, we introduce basic notation that you should have no problem to remember.

$\sigma\langle k\rangle$	19, k-th transition in σ.
$\textsc{Last}(\sigma)$	19, Last state in σ.
$\textsc{Len}(\sigma)$	19, Number of states in σ.
$\|$	17, $P \parallel Q$ is the system whose atoms are $\textsc{atoms}(P) \cup \textsc{atoms}(Q)$. Given two atoms A and B, $A \parallel B$ denotes the system P having $\textsc{atoms}(P) = \{A, B\}$.
$\sigma_{\downarrow k}$	20, k-th prefix of σ, ending in $\sigma(k)$. If k is negative, it ends in $\sigma(\textsc{Len}(\sigma) + k)$.
$\sigma^{\uparrow k}$	20, 122, k-th suffix of σ, starts from $s(k)$.
$\sigma' \sqsubseteq \sigma$	20, σ' is a prefix of (or is equal to) σ.
$\sigma(k)$	19, k-th state in σ.
$\sigma \cdot \sigma'$	20, Concatenation of σ and σ'. Assumes $\textsc{Last}(\sigma) = \textsc{First}(\sigma')$.
$(\sigma)^{\uparrow}$	20, Set comprising all infinite paths ω such that σ is a prefix of ω.
ς	19, Fictitious transition executed when no generative transitions are enabled in the system. $\varsigma(s, s') = 1$ iff $s = s'$.
$\textsc{active}(c)$	18, Atom that generates the output in compound transition c.
$\textsc{actlab}(g_i)$	108, Labels that are output by g_i with positive probability.
$\textsc{apaths}(A_i)$	17, 21, Paths in the atom A_i.
$c(s, s')$	18, Probability of reaching s' from s using c.
Cylinder	20, see $(\sigma)^{\uparrow}$.
$\textsc{dist}_P$	22, Set of distributed schedulers under projection $[\![\cdot]\!]$.
Extension set	20, Set of the form $(\sigma)^{\uparrow}$ for some σ.

Fringe	84, Finite set $\{\sigma^m\}_m$ such that $\forall_{m \neq m'} (\sigma)^\uparrow \cap (\sigma')^\uparrow = \emptyset$ and $\biguplus_m (\sigma^m)^\uparrow = (\text{INIT})^\uparrow$.
G_i	17, Generative structure of atom A_i.
Generative structure	26, 212, Structure describing the generative (output) transitions enabled in each state. $G(s)$ is the set of generative transitions enabled at state s.
Global enabledness conditions	27, A system has *global enabledness conditions* iff the set of transitions enabled at a given atom is a function of the global state.
Global path	19, Path $s^1.c^1.s^2.c^2 \cdots c^{n-1}.s^n$ where the s^k are global states and the c^k are compound transitions.
Glossary	xv, This glossary.
$\text{LABEL}(c)$	18, Label a of a compound transition $c = (g_i, a, r_{j_1}, \cdots, r_{j_m})$.
Local enabledness conditions	27, A system has *local enabledness conditions* iff the set of transitions enabled at a given atom is a function of the state of the atom.
$\mathcal{N}$	17, Number of a atoms in a given system.
Path in an atom	17, Sequence $s_i^1.a^1.\cdots.a^{n-1}.s_i^n$ where the s_i^k are local states of the atom A_i and a^k in ACTLAB_i.
Projection equivalent for a set of atoms	77, 102, See Def. 4.6.
R_i	17, Reactive structure of atom A_i.
Rate scheduler	47, 212, Scheduler that chooses a rate for each local path.
Rate-based interleaving scheduler	47, Interleaving scheduler that can be obtained by composing rate schedulers.
Rate-based scheduler	47, Scheduler whose corresponding interleaving scheduler is rate-based.
$\text{REACH}(U)$	24, 90, Set comprising all infinite paths ω such that $\omega(k) \in U$ for some k.
Reachability set	24, Set $\mathcal{S}$ of infinite paths such that as $\mathcal{S} = \text{REACH}(U)$ for some U.
$\text{REACTIVE}(c)$	18, Atoms that react to the output in the compound transition c. Formally, $\text{REACTIVE}(c) = \{A_i \mid \text{LABEL}(c) \in \text{ACTLAB}_i\}$.
Reactive structure	26, 212, Structure describing the reactive (input) transitions enabled for each state s and action label a. $R(s, a)$ is the set of reactive transitions enabled at state s for the action label a.

SCHED_P	35, Set comprising all schedulers for system P, for all projections.
Strongly distributed scheduler	42, Distributed scheduler in which the interleaving scheduler is also restricted.
T_{G_i}	17, Set of generative transitions on (S_i, ACTLAB_i).
T_{R_i}	17, Set of reactive transitions on (S_i, ACTLAB_i).
Total order-based interleaving scheduler	49, 213, Interleaving scheduler that selects the atom having the minimum local path, according to a total order on local paths.
Total order-based scheduler	49, Scheduler whose corresponding interleaving scheduler is total order-based.
Traceable projection	75, See Def. 4.5, p. 75.

BIBLIOGRAPHY

[1] Prism semantics. Available at: www.prismmodelchecker.org/doc/semantics.pdf. (Cited on page 149.)

[2] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Annals of Mathematics. Second Series*, 160(2):781–793, 2004. (Cited on page 7.)

[3] James H. Anderson, Giuseppe Prencipe, and Roger Wattenhofer, editors. *Principles of Distributed Systems, 9th International Conference, OPODIS 2005, Pisa, Italy, December 12-14, 2005, Revised Selected Papers*, volume 3974 of *Lecture Notes in Computer Science*. Springer, 2006. (Cited on page 218.)

[4] Suzana Andova. Process Algebra with Probabilistic Choice. In Katoen [106], pages 111–129. (Cited on page 8.)

[5] Suzana Andova and Sonja Georgievska. On Compositionality, Efficiency, and Applicability of Abstraction in Probabilistic Systems. In Nielsen et al. [124], pages 67–78. (Cited on page 9.)

[6] Jesús Arias-Fisteus, Luis Sánchez Fernández, and Carlos Delgado Kloos. Applying model checking to BPEL4WS business collaborations. In Hisham Haddad, Lorie M. Liebrock, Andrea Omicini, and Roger L. Wainwright, editors, *SAC*, pages 826–830. ACM, 2005. (Cited on page 1.)

[7] Yonatan Aumann. Efficient Asynchronous Consensus with the Weak Adversary Scheduler. In *PODC*, pages 209–218, 1997. (Cited on page 163.)

[8] Yonatan Aumann and Michael A. Bender. Efficient low-contention asynchronous consensus with the value-oblivious adversary scheduler. *Distributed Computing*, 17(3):191–207, 2005. (Cited on page 163.)

[9] Yonatan Aumann and Avivit Kapah-Levy. Cooperative Sharing and Asynchronous Consensus Using Single-Reader Single-Writer Registers. In *SODA*, pages 61–70, 1999. (Cited on page 163.)

[10] Jos C. M. Baeten. A brief history of process algebra. *Theor. Comput. Sci.*, 335(2-3):131–146, 2005. (Cited on page 8.)

[11] Jos C. M. Baeten, Jan A. Bergstra, and Scott A. Smolka. Axiomatizing Probabilistic Processes: ACP with Generative Probabilities. *Inf. Comput.*, 121(2):234–255, 1995. (Cited on page 8.)

[12] 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. (Cited on pages 2, 8, 121, 123, 126, 137, 138, and 140.)

[13] C. Baier, M. Größer, and F. Ciesinski. Quantitative Analysis of Distributed Randomized Protocols. In *Proc. of FMICS'05*, pages 2–7. ACM, 2005.

[14] C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008. (Cited on pages 1 and 8.)

[15] Christel Baier. Polynomial Time Algorithms for Testing Probabilistic Bisimulation and Simulation. In Rajeev Alur and Thomas A. Henzinger, editors, *CAV*, volume 1102 of *Lecture Notes in Computer Science*, pages 50–61. Springer, 1996. (Cited on page 8.)

[16] Christel Baier, Nathalie Bertrand, and Marcus Größer. On decision problems for probabilistic büchi automata. In Roberto M. Amadio, editor, *FoSSaCS*, volume 4962 of *Lecture Notes in Computer Science*, pages 287–301. Springer, 2008. (Cited on page 102.)

[17] Christel Baier, Frank Ciesinski, and Marcus Größer. ProbMela and verification of Markov decision processes. *SIGMETRICS Perform. Eval. Rev.*, 32(4):22–27, 2005.

[18] Christel Baier, Edmund M. Clarke, Vassili Hartonas-Garmhausen, Marta Z. Kwiatkowska, and Mark Ryan. Symbolic Model Checking for Probabilistic Processes. In Degano et al. [67], pages 430–440. (Cited on page 1.)

[19] Christel Baier, Pedro R. D'Argenio, and Marcus Größer. Partial Order Reduction for Probabilistic Branching Time. *Electr. Notes Theor. Comput. Sci.*, 153(2):97–116, 2006. (Cited on page 140.)

[20] Christel Baier and Marcus Größer. Recognizing omega-regular Languages with Probabilistic Automata. In *LICS*, pages 137–146. IEEE Computer Society, 2005. (Cited on page 102.)

[21] Christel Baier and Marta Z. Kwiatkowska. Model Checking for a Probabilistic Branching Time Logic with Fairness. *Distributed Computing*, 11(3):125–155, 1998. (Cited on page 1.)

[22] Thomas Ball and Robert B. Jones, editors. *Computer Aided Verification, 18th International Conference, CAV 2006, Seattle, WA, USA, August 17-20, 2006, Proceedings*, volume 4144 of *Lecture Notes in Computer Science*. Springer, 2006. (Cited on page 222.)

[23] Marco Bernardo and Roberto Gorrieri. A Tutorial on EMPA: A Theory of Concurrent Processes with Nondeterminism, Priorities, Probabilities and Time. *Theor. Comput. Sci.*, 202(1-2):1–54, 1998. (Cited on page 8.)

[24] Dirk Beyer, Thomas A. Henzinger, Ranjit Jhala, and Rupak Majumdar. The software model checker Blast. *STTT*, 9(5-6):505–525, 2007. (Cited on page 1.)

[25] Andrea Bianco and Luca de Alfaro. Model Checking of Probabalistic and Nondeterministic Systems. In Thiagarajan [143], pages 499–513. (Cited on pages 1, 4, 8, 62, 91, 111, 116, and 140.)

[26] Armin Biere, Edmund M. Clarke, Richard Raimi, and Yunshan Zhu. Verifiying Safety Properties of a Power PC Microprocessor Using Symbolic Model Checking without BDDs", booktitle = CAV. In Nicolas Halbwachs and Doron Peled, editors, *CAV*, volume 1633 of *Lecture Notes in Computer Science*, pages 60–71. Springer, 1999. (Cited on page 1.)

[27] G. Brassard and P. Bratley. *Fundamentals of Algorithmics*. Prentice-Hall, 1995. (Cited on page 116.)

[28] Mario Bravetti, Marco Bernardo, and Roberto Gorrieri. Towards Performance Evaluation with General Distributions in Process Algebras. In Sangiorgi and de Simone [132], pages 405–422. (Cited on page 8.)

[29] Mario Bravetti and Pedro R. D'Argenio. Tutte le Algebre Insieme: Concepts, Discussions and Relations of Stochastic Process Algebras with General Distributions. In Christel Baier, Boudewijn R. Haverkort, Holger Hermanns, Joost-Pieter Katoen, and Markus Siegle, editors, *Validation of Stochastic Systems*, volume 2925 of *Lecture Notes in Computer Science*, pages 44–88. Springer, 2004. (Cited on page 8.)

[30] Robert Bringhurst. *The Elements of Typographic Style*. Version 2.5. Hartley & Marks, Publishers, Point Roberts, WA, USA, 2002. (Cited on page 225.)

[31] James E. Burns, Paul Jackson, Nancy A. Lynch, Michael J. Fischer, and Gary L. Peterson. Data Requirements for Implementation of N-Process Mutual Exclusion Using a Single Shared Variable. *J. ACM*, 29(1):183–205, 1982. (Cited on page 7.)

[32] Nadia Busi. Analysis issues in Petri nets with inhibitor arcs. *Theor. Comput. Sci.*, 275(1-2):127–177, 2002. (Cited on page 60.)

[33] G. Calafiore, F. Dabbene, and R. Tempo. A Survey of Randomized Algorithms for Control Synthesis and Performance Verification. *COMPLEXITY: Journal of Complexity*, 23, 2007. (Cited on page 7.)

[34] Ran Canetti, Ling Cheung, Dilsun Kirli Kaynar, Nancy A. Lynch, and Olivier Pereira. Compositional Security for Task-PIOAs. In *CSF*, pages 125–139. IEEE Computer Society, 2007. (Cited on pages 37 and 49.)

[35] A. Cassandra. The POMDP page. `www.pomdp.org`. (Cited on pages 9 and 37.)

[36] Nathalie Chabrier and François Fages. Symbolic Model Checking of Biochemical Networks. In Corrado Priami, editor, *CMSB*, volume 2602 of *Lecture Notes in Computer Science*, pages 149–162. Springer, 2003. (Cited on page 1.)

[37] Tushar Deepak Chandra. Polylog Randomized Wait-Free Consensus. In *PODC*, pages 166–175, 1996. (Cited on page 163.)

[38] K. Chatterjee, R. Majumdar, and M. Jurdzinski. On Nash equilibria in stochastic games. In *CSL '04*, pages 26–40, 2004. (Cited on pages 4, 5, and 92.)

[39] K. Chatzikokolakis, S. Knight, and P. Panangaden. Epistemic Strategies and Games on Concurrent Processes. In *Proceedings of SOFSEM '09*, 2009. (Cited on page 8.)

[40] K. Chatzikokolakis and C. Palamidessi. A framework for analyzing probabilistic protocols and its application to the Partial Secrets Exchange. *Theor. Comput. Sci.*, 389(3):512–527, 2007. (Cited on page 6.)

[41] Konstantinos Chatzikokolakis. *Probabilistic and Information-Theoretic Approaches to Anonymity*. PhD thesis, LIX, École Polytechnique, 2007. (Cited on page 1.)

[42] Konstantinos Chatzikokolakis, Gethin Norman, and David Parker. Bisimulation for Demonic Schedulers. In Luca de Alfaro, editor, *FOSSACS*, volume 5504 of *Lecture Notes in Computer Science*, pages 318–332. Springer, 2009. (Cited on pages 1, 8, and 21.)

[43] Konstantinos Chatzikokolakis and Catuscia Palamidessi. Making Random Choices Invisible to the Scheduler. In Luís Caires and Vasco Thudichum Vasconcelos, editors, *CONCUR*, volume 4703 of *Lecture Notes in Computer Science*, pages 42–58. Springer, 2007. (Cited on pages 6 and 8.)

[44] D. Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *J. Cryptology*, 1(1):65–75, 1988. (Cited on pages 3, 40, 149, and 157.)

[45] L. Cheung. *Reconciling Nondeterministic and Probabilistic Choices*. PhD thesis, Radboud Universiteit Nijmegen, 2006. (Cited on pages 1, 9, 21, 42, 50, 60, 93, 161, and 163.)

[46] 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. (Cited on pages 1, 6, 8, 13, 20, 21, 37, 38, 50, and 102.)

[47] Ling Cheung. Randomized Wait-Free Consensus Using an Atomicity Assumption. In Anderson et al. [3], pages 47–60. (Cited on page 3.)

[48] Benny Chor, Amos Israeli, and Ming Li. On Processor Coordination Using Asynchronous Hardware. In *PODC*, pages 86–97, 1987. (Cited on page 163.)

[49] Benny Chor, Amos Israeli, and Ming Li. Wait-Free Consensus Using Asynchronous Hardware. *SIAM J. Comput.*, 23(4):701–712, 1994. (Cited on pages 3, 7, and 163.)

[50] F. Ciesinski and C. Baier. LiQuor: A tool for Qualitative and Quantitative Linear Time analysis of Reactive Systems. In *QEST'06*, pages 131–132. IEEE CS, 2006. (Cited on pages 1, 5, and 160.)

[51] Alessandro Cimatti, Edmund M. Clarke, Fausto Giunchiglia, and Marco Roveri. NUSMV: A New Symbolic Model Checker. *STTT*, 2(4):410–425, 2000. (Cited on page 1.)

[52] E. M. Clarke, O. Grumberg, M. Minea, and D. Peled. State Space Reduction Using Partial Order Techniques. *STTT*, 2(3):279–287, 1999. (Cited on page 8.)

[53] Edmund M. Clarke and E. Allen Emerson. Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic. In Kozen [110], pages 52–71. (Cited on page 1.)

[54] Edmund M. Clarke, Orna Grumberg, Hiromi Hiraishi, Somesh Jha, David E. Long, Kenneth L. McMillan, and Linda A. Ness. Verification of the futurebus+ cache coherence protocol. *Formal Methods in System Design*, 6(2):217–232, 1995. (Cited on page 1.)

[55] Edmund M. Clarke, Somesh Jha, and Wilfredo R. Marrero. Efficient verification of security protocols using partial-order reductions. *STTT*, 4(2):173–188, 2003. (Cited on page 1.)

[56] E.M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 2000. (Cited on pages 1, 8, 121, and 126.)

[57] 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. (Cited on pages 2, 8, 121, 123, and 140.)

[58] Pedro R. D'Argenio, Bertrand Jeannet, Henrik Ejersbo Jensen, and Kim Guldstrand Larsen. Reachability Analysis of Probabilistic Systems by Successive Refinements. In Luca de Alfaro and Stephen Gilmore, editors, *PAPM-PROBMIV*, volume 2165 of *Lecture Notes in Computer Science*, pages 39–56. Springer, 2001. (Cited on page 9.)

[59] Pedro R. D'Argenio and Joost-Pieter Katoen. A theory of Stochastic systems. Part II: Process algebra. *Inf. Comput.*, 203(1):39–74, 2005. (Cited on page 8.)

[60] Pedro R. D'Argenio, Joost-Pieter Katoen, and Ed Brinksma. An algebraic approach to the specification of stochastic systems. In David Gries and Willem P. de Roever, editors, *PROCOMET*, volume 125 of *IFIP Conference Proceedings*, pages 126–147. Chapman & Hall, 1998. (Cited on page 8.)

[61] P.R. D'Argenio. *Algebras and Automata for Timed and Stochastic Systems*. PhD thesis, Department of Computer Science, University of Twente, 1999. (Cited on page 2.)

[62] P.R. D'Argenio and J.-P. Katoen. A Theory of Stochastic Systems, Part I: Stochastic Automata. *Information and Computation*, 203(1):1–38, 2005.

[63] L. de Alfaro. The verification of probabilistic systems under memoryless partial-information policies is hard. In *PROBMIV'99. TR CSR-99-8*, pages 19–32. University of Birmingham, 1999. (Cited on pages 6, 9, 37, 38, 50, and 102.)

[64] L. de Alfaro, T. A. Henzinger, and R. Jhala. Compositional Methods for Probabilistic Systems. In *CONCUR'01*, LNCS 2154, pages 351–365. Springer, 2001. (Cited on pages 6, 9, 13, 20, 21, 37, 38, 50, 93, 102, 161, and 163.)

[65] Luca de Alfaro. *Formal Verification of Probabilistic Systems*. PhD thesis, Stanford University, 1997. (Cited on pages 87, 88, and 139.)

[66] Luca de Alfaro, Thomas A. Henzinger, and Orna Kupferman. Concurrent reachability games. *Theor. Comput. Sci.*, 386(3):188–217, 2007. (Cited on page 5.)

[67] Pierpaolo Degano, Roberto Gorrieri, and Alberto Marchetti-Spaccamela, editors. *Automata, Languages and Programming, 24th International Colloquium, ICALP'97, Bologna, Italy, 7-11 July 1997, Proceedings*, volume 1256 of *Lecture Notes in Computer Science*. Springer, 1997. (Cited on page 216.)

[68] Yuxin Deng, Catuscia Palamidessi, and Jun Pang. Weak Probabilistic Anonymity. *Electron. Notes Theor. Comput. Sci.*, 180(1):55–76, 2007. (Cited on page 157.)

[69] Mariangiola Dezani-Ciancaglini and Ugo Montanari, editors. *International Symposium on Programming, 5th Colloquium, Torino, Italy, April 6-8, 1982, Proceedings*, volume 137 of *Lecture Notes in Computer Science*. Springer, 1982. (Cited on page 223.)

[70] Martin Dietzfelbinger. *Primality Testing in Polynomial Time, From Randomized Algorithms to "PRIMES Is in P"*, volume 3000 of *Lecture Notes in Computer Science*. Springer, 2004. (Cited on page 7.)

[71] Javier Esparza. Decidability and Complexity of Petri Net Problems - An Introduction. In Reisig and Rozenberg [131], pages 374–428.

[72] A. Fehnker and P. Gao. Formal Verification and Simulation for Performance Analysis for Probabilistic Broadcast Protocols. In *Proc. 5th International Conference on Ad-Hoc, Mobile, and Wireless Networks (ADHOC-NOW'06)*, volume 4104 of *LNCS*, pages 128–141. Springer, 2006. (Cited on page 1.)

[73] Luis María Ferrer Fioriti, Pedro R. D'Argenio (as advisor), and Sergio Giro (as co advisor). Implementación de técnicas de Reducción de Orden Parcial para sistemas probabilísticos (trabajo final de licenciatura), 2009. (Cited on page 149.)

[74] M. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, 1985. (Cited on page 7.)

[75] A. Giacalone, C.-C. Jou, and S. A. Smolka. Algebraic reasoning for probabilistic concurrent systems. In *Proceedings IFIP TC2 Working Conference on Programming Concepts and Methods*, pages 443–458. North-Holland, 1990. (Cited on page 8.)

[76] Stephen Gilmore and Jane Hillston. The PEPA Workbench: A Tool to Support a Process Algebra-based Approach to Performance Modelling. In Haring and Kotsis [90], pages 353–368. (Cited on page 8.)

[77] 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. (Cited on page 10.)

[78] Sergio Giro and Pedro R. D'Argenio. Quantitative model checking revisited: neither Decidable nor Approximable. In J.-F. Raskin and P.S. Thiagarajan, editors, *FORMATS*, volume 4763 of *Lecture Notes in Computer Science*, pages 179–194. Springer, 2007. (Cited on pages 10, 92, 95, and 102.)

[79] 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. (Cited on pages 10 and 20.)

[80] Sergio Giro and Pedro R. D'Argenio. On the verification of probabilistic I/O automata with unspecified rates. In Sung Y. Shin and Sascha Ossowski, editors, *SAC*, pages 582–586. ACM, 2009. (Cited on page 10.)

[81] 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. (Cited on page 10.)

[82] 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. (Cited on pages 8 and 15.)

[83] P. Godefroid. *Partial-Order Methods for the Verification of Concurrent Systems - An Approach to the State-Explosion Problem*. LNCS 1032. Springer, 1996. (Cited on pages 8 and 121.)

[84] Patrice Godefroid. *Partial-Order Methods for the Verification of Concurrent Systems – An Approach to the State-Explosion Problem*. PhD thesis, University of Liège, 1994.

[85] B. L. Rothschild R. L. Graham and J. H. Spencer. *Ramsey Theory*. John Wiley & Sons, 1990. (Cited on page 60.)

[86] Marcus Größer. *Reduction Methods for Probabilistic Model Checking*. PhD thesis, Technische Universität Dresden, 2008. (Cited on pages 102 and 138.)

[87] Marcus Größer, Gethin Norman, Christel Baier, Frank Ciesinski, Marta Z. Kwiatkowska, and David Parker. On Reduction Criteria for Probabilistic Reward Models. In S. Arun-Kumar and Naveen Garg, editors, *FSTTCS*, volume 4337 of *Lecture Notes in Computer Science*, pages 309–320. Springer, 2006. (Cited on page 140.)

[88] R. Gupta, S. A. Smolka, and S. Bhaskar. On Randomization in Sequential and Distributed Algorithms. *ACM Computing Surveys*, 26(1), 1994. (Cited on page 7.)

[89] Theo Härder, Hartmut Wedekind, and Gerhard Zimmermann, editors. *Entwurf und Betrieb verteilter Systeme, Fachtagung des Sonderforschungsbereiche 124 und 182, Dagstuhl, 19.-21. September 1990, Proceedings*, volume 264 of *Informatik-Fachberichte*. Springer, 1990. (Cited on page 221.)

[90] Günter Haring and Gabriele Kotsis, editors. *Computer Performance Evaluation, Modeling Techniques and Tools, 7th International Conference, Vienna, Austria, May 3-6, 1994, Proceedings*, volume 794 of *Lecture Notes in Computer Science*. Springer, 1994. (Cited on page 220.)

[91] Peter G. Harrison and B. Strulo. SPADES - a process algebra for discrete event simulation. *J. Log. Comput.*, 10(1):3–42, 2000. (Cited on page 8.)

[92] Klaus Havelund and Thomas Pressburger. Model Checking JAVA Programs using JAVA PathFinder. *STTT*, 2(4):366–381, 2000. (Cited on page 1.)

[93] Holger Hermanns. *Interactive Markov Chains: The Quest for Quantified Quality*, volume 2428 of *Lecture Notes in Computer Science*. Springer, 2002. (Cited on page 8.)

[94] Holger Hermanns, Vassilis Mertsiotakis, and Michael Rettelbach. A Construction and Analysis Tool Based on the Stochastic Process Algebra TIPP. In Margaria and Steffen [120], pages 427–430. (Cited on page 8.)

[95] Holger Hermanns, Björn Wachter, and Lijun Zhang. Probabilistic CEGAR. In Aarti Gupta and Sharad Malik, editors, *CAV*, volume 5123 of *Lecture Notes in Computer Science*, pages 162–175. Springer, 2008. (Cited on page 163.)

[96] Ulrich Herzog. Formal Description, Time and Performance Analysis. A Framework. In Härder et al. [89], pages 172–190. (Cited on page 8.)

[97] 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. (Cited on pages 1, 5, 25, and 149.)

[98] M. Hofmann and M. Felleisen, editors. *Proceedings of the 34th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2007, Nice, France, January 17-19, 2007*. ACM, 2007.

[99] Gerard J. Holzmann. The Model Checker SPIN. *IEEE Trans. Software Eng.*, 23(5):279–295, 1997. (Cited on page 1.)

[100] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Language, and Computation*. Addison–Wesley, 1979.

[101] Juraj Hromkovic. Randomized Communication Protocols (A Survey). In *International Symposium on Stochastic Algorithms: Foundations and Applications*, volume 1 of *Lecture Notes in Computer Science*, 2001. (Cited on page 7.)

[102] B. Jeannet, P.R. D'Argenio, and K.G. Larsen. Rapture: A tool for verifying Markov Decision Processes. In I. Cerna, editor, *Tools Day'02, Brno, Czech Republic*, Technical Report. Faculty of Informatics, Masaryk University Brno, 2002. (Cited on page 149.)

[103] Neil D. Jones and Markus Müller-Olm, editors. *Verification, Model Checking, and Abstract Interpretation, 10th International Conference, VMCAI 2009, Savannah, GA, USA, January 18-20, 2009. Proceedings*, volume 5403 of *Lecture Notes in Computer Science*. Springer, 2009. (Cited on page 222.)

[104] Richard M. Karp. An introduction to randomized algorithms. *Discrete Appl. Math.*, 34:165–201, 1991. (Cited on page 7.)

[105] J.-P. Katoen, M. Khattri, and I. S. Zapreev. A Markov Reward Model Checker. In *Quantitative Evaluation of Systems (QEST)*, pages 243–244, Los Alamos, CA, USA, 2005. IEEE Computer Society. (Cited on page 1.)

[106] Joost-Pieter Katoen, editor. *Formal Methods for Real-Time and Probabilistic Systems, 5th International AMAST Workshop, ARTS'99, Bamberg, Germany, May 26-28, 1999. Proceedings*, volume 1601 of *Lecture Notes in Computer Science*. Springer, 1999. (Cited on page 215.)

[107] Mark Kattenbelt, Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Abstraction Refinement for Probabilistic Software. In Jones and Müller-Olm [103], pages 182–197. (Cited on page 9.)

[108] Robert M. Keller. Formal Verification of Parallel Programs. *Commun. ACM*, 19(7):371–384, 1976. (Cited on page 8.)

[109] J. G. Kemeny, J. L. Snell, and A. W. Knapp. *Denumerable Markov Chains*. Van Nostrand Company, 1966. (Cited on page 24.)

[110] Dexter Kozen, editor. *Logics of Programs, Workshop, Yorktown Heights, New York, May 1981*, volume 131 of *Lecture Notes in Computer Science*. Springer, 1982. (Cited on page 218.)

[111] J. B. Kruskal. The theory of well-quasi-ordering: A frequently discovered concept. *Journal of Combinatorial Theory*, pages 297–305, 1972. (Cited on page 60.)

[112] M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic Model Checking and Power-Aware Computing. In *Proc. 7th International Workshop on Performability Modeling of Computer and Communication Systems (PMCCS'05)*, pages 6–9, 2005. (Cited on page 1.)

[113] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Symmetry Reduction for Probabilistic Model Checking. In Ball and Jones [22], pages 234–248. (Cited on pages 2, 9, and 163.)

[114] M. Littman. POMDP information page. `www.cs.duke.edu/~mlittman/topics/pomdp-page.html`. (Cited on pages 9 and 37.)

[115] N. Lynch, R. Segala, and F. Vaandrager. Compositionality for Probabilistic Automata. In *Proc. of CONCUR 03*, LNCS 2761, pages 208–221. Springer, 2003.

[116] Nancy A. Lynch, Roberto Segala, and Frits W. Vaandrager. Compositionality for Probabilistic Automata. In Roberto M. Amadio and Denis Lugiez, editors, *CONCUR*, volume 2761 of *Lecture Notes in Computer Science*, pages 204–222. Springer, 2003. (Cited on page 163.)

[117] Nancy A. Lynch and Mark R. Tuttle. An introduction to input/output automata. *CWI Quarterly*, 2(3):219–246, 1989. (Cited on page 17.)

[118] 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. (Cited on pages 93, 94, and 95.)

[119] P. Malacaria. Assessing security threats of looping constructs. In M. Hofmann and M. Felleisen, editors, *Proc. of POPL 2007*, pages 225–235. ACM, 2007.

[120] Tiziana Margaria and Bernhard Steffen, editors. *Tools and Algorithms for Construction and Analysis of Systems, Second International Workshop, TACAS '96, Passau, Germany, March 27-29, 1996, Proceedings*, volume 1055 of *Lecture Notes in Computer Science*. Springer, 1996. (Cited on page 221.)

[121] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. John Wiley & Sons, 1995.

[122] D. A. Martin. Borel determinacy. *Annals of Mathematics*, 102(2):363–371, 1975. (Cited on page 92.)

[123] Peter Niebert. Personal communication. (Cited on page 97.)

[124] Mogens Nielsen, Antonín Kucera, Peter Bro Miltersen, Catuscia Palamidessi, Petr Tuma, and Frank D. Valencia, editors. *SOFSEM 2009: Theory and Practice of Computer Science, 35th Conference on Current Trends in Theory and Practice of Computer Science, Spindleruv Mlýn, Czech Republic, January 24-30, 2009. Proceedings*, volume 5404 of *Lecture Notes in Computer Science*. Springer, 2009. (Cited on page 215.)

[125] 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. (Cited on page 8.)

[126] M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley, 1994. (Cited on pages 4 and 8.)

[127] Jean-Pierre Queille and Joseph Sifakis. Specification and verification of concurrent systems in CESAR. In Dezani-Ciancaglini and Montanari [69], pages 337–351. (Cited on page 1.)

[128] Michael O. Rabin. Probabilistic algorithms. In *Algorithms and Complexity: New Directions and Results*, pages 21–39. Academic Press, 1976. (Cited on page 7.)

[129] Michael O. Rabin. N-Process Mutual Exclusion with Bounded Waiting by $4 \log_2 N$-Valued Shared Variable. *J. Comput. Syst. Sci.*, 25(1):66–75, 1982. (Cited on page 7.)

[130] J.-F. Raskin and P. S. Thiagarajan, editors. *Formal Modeling and Analysis of Timed Systems, 5th International Conference, FORMATS 2007, Salzburg, Austria, October 3-5, 2007, Proceedings*, volume 4763 of *Lecture Notes in Computer Science*. Springer, 2007.

[131] Wolfgang Reisig and Grzegorz Rozenberg, editors. *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets, the volumes are based on the Advanced Course on Petri Nets, held in Dagstuhl, September 1996*, volume 1491 of *Lecture Notes in Computer Science*. Springer, 1998. (Cited on page 219.)

[132] Davide Sangiorgi and Robert de Simone, editors. *CONCUR '98: Concurrency Theory, 9th International Conference, Nice, France, September 8-11, 1998, Proceedings*, volume 1466 of *Lecture Notes in Computer Science*. Springer, 1998. (Cited on page 216.)

[133] R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, Laboratory for Computer Science, MIT, 1995. (Cited on pages 4, 6, 8, 37, 38, and 91.)

[134] Roberto Segala. A Compositional Trace-Based Semantics for Probabilistic Automata. In Insup Lee and Scott A. Smolka, editors, *CONCUR*, volume 962 of *Lecture Notes in Computer Science*, pages 234–248. Springer, 1995. (Cited on page 163.)

[135] Sven Seuken and Shlomo Zilberstein. Formal models and algorithms for decentralized decision making under uncertainty. *Autonomous Agents and Multi-Agent Systems*, 17(2):190–250, 2008. (Cited on pages 9 and 37.)

[136] A. Sokolova and E.P. de Vink. Probabilistic automata: system types, parallel composition and comparison. In C. Baier, B.R. Haverkort, H. Hermanns, J.-P. Katoen, and M. Siegle, editors, *Validation of Stochastic Systems: A Guide to Current Research*, LNCS 2925, pages 1–43, 2004. (Cited on page 15.)

[137] E. Stark and G. Pemmasani. Implementation of a compositional performance analysis algorithm for probabilistic I/O automata. In *Proc. of PAPM '99*, pages 3–24. Prensas Universitarias de Zaragoza, 1999. (Cited on pages 46, 47, and 148.)

[138] E. W. Stark, R. Cleaveland, and S. A. Smolka. A process-algebraic language for probabilistic I/O automata. In Roberto M. Amadio and Denis Lugiez, editors, *CONCUR '03*, volume 2761 of *LNCS*, pages 189–203. Springer-Verlag, 2003. (Cited on pages 46 and 47.)

[139] E. W. Stark and S. Smolka. Compositional analysis of expected delays in networks of probabilistic I/O automata. In *LICS 98*, pages 466–477. IEEE CS Press, 1998. (Cited on pages 46 and 47.)

[140] Eugene W. Stark. On behaviour equivalence for probabilistic I/O automata and its relationship to probabilistic bisimulation. *J. Autom. Lang. Comb.*, 8(2):361–395, 2003. (Cited on pages 46 and 47.)

[141] M. Stoelinga. *Alea jacta est: Verification of Probabilistic, Real-time and Parametric Systems*. PhD thesis, Katholieke Universiteit Nijmegen, 2002.

[142] Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall, fourth edition, 2003. (Cited on page 158.)

[143] P. S. Thiagarajan, editor. *Foundations of Software Technology and Theoretical Computer Science, 15th Conference (FSTTCS), Bangalore, India, December 18-20, 1995, Proceedings*, volume 1026 of *Lecture Notes in Computer Science*. Springer, 1995. (Cited on page 216.)

[144] Stavros Tripakis. Undecidable problems of decentralized observation and control. In *Proc. of the 40th IEEE Conference on Decision and Control*, volume 5, pages 4104–4109, 2001. (Cited on page 99.)

[145] Noel Vaillant. probability.net. Probability tutorials on line. Tutorial 2. (Cited on pages 62 and 63.)

[146] D. Varacca and M. Nielsen. Probabilistic Petri nets and mazurkiewicz equivalence, 2003. Unpublished draft.

[147] 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. (Cited on pages 8, 37, 46, 47, 162, and 163.)

[148] H. Younes, M. Kwiatkowska, G. Norman, and D. Parker. Numerical vs. Statistical Probabilistic Model Checking: An Empirical Study. In K. Jensen and A. Podelski, editors, *Proc. 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'04)*, volume 2988 of *LNCS*, pages 46–60. Springer, 2004. (Cited on page 1.)

COLOPHON

This thesis was typeset with LaTeX 2$_\varepsilon$ using Hermann Zapf's *Palatino* and *Euler* type faces (Type 1 PostScript fonts *URW Palladio L* and *FPL* were used). The listings are typeset in *Bera Mono*, originally developed by Bitstream, Inc. as "Bitstream Vera". (Type 1 PostScript fonts were made available by Malte Rosenau and Ulrich Dirr.)

The typographic style was inspired by 's genius as presented in *The Elements of Typographic Style* [30]. It is available for LaTeX via CTAN as "`classicthesis`".

Final Version as of March 28, 2010 at 18:59.

DECLARATION

I declare that this thesis is my own work except where explicitly indicated.

Córdoba, March 2010

Sergio Giro