Weak Bisimulation for Fully Probabilistic Processes

Christel Baier [a] and Holger Hermanns [b]

[a] *Fakultät für Mathematik & Informatik, Universität Mannheim, Seminargebäude A 5, 68131 Mannheim, Germany*
baier@pi2.informatik.uni-mannheim.de

[b] *Systems Validation Centre, FMT/CTIT, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands*
hermanns@cs.utwente.nl

Abstract

Bisimulations that abstract from internal computation have proven to be useful for verification of compositionally defined transition systems. In the literature of probabilistic extensions of such transition systems, similar bisimulations are rare. In this paper, we introduce *weak and branching bisimulation* for *fully probabilistic systems*, transition systems where nondeterministic branching is replaced by probabilistic branching. In contrast to the nondeterministic case, both relations coincide. We give an *algorithm* to decide weak (and branching) bisimulation with a time complexity cubic in the number of states of the fully probabilistic system. This meets the worst case complexity for deciding branching bisimulation in the nondeterministic case. In addition, the relation is shown to be a congruence with respect to the operators of *PLSCCS*, a *lazy synchronous* probabilistic variant of CCS. We illustrate that due to these properties, weak bisimulation provides all the crucial ingredients for *mechanised compositional verification* of probabilistic transition systems.

Centre for Telematics and Information Technology ***TR-CTIT-12, June 1999***

Contents

1 Introduction

In recent years, the need to formally reason about probabilistic phenomena in software and hardware systems has stimulated the study of probabilistic models of computation. A variety of models has been proposed in the literature, many of them based on transition systems. These models can be classified with respect to their treatment of nondeterminism. Several approaches *replace* the concept of nondeterministic branching by probabilistic branching, e.g. [19,33,48,73,27]. Following [27], this model can be subdivided according to the relationship between occurrences of actions and transition probabilities. In 'reactive' systems, transition probability distributions are dependent on the occurrences of actions. In contrast, in 'generative' – also called *fully probabilistic* – systems, these distributions implicitly assign probabilities also to occurrences of actions. 'Stratified' systems allow for levelwise probabilistic branching. Other authors, e.g. [71,60,32,44,64], allow for *both*, nondeterministic as well as probabilistic branching, mainly because this allows one to express the concurrent (interleaved) execution of independent (probabilistic) activities by means of nondeterminism. Accordingly, we subsume these models as *concurrent* probabilistic systems.

Verification techniques for probabilistic models have been inspired by successful experiences in the non-probabilistic case. This includes probabilistic variants of temporal logics, e.g. [4,12,21,32–34,60,61,71,72]. Another research strand focusses on equivalences and preorders used to established that one system 'implements' another, according to some notion of implementation, such as strong bisimulation [48], simulation [41,64], testing preorders [19,44,74,73], trace, failure and ready equivalence [43].

In the context of specification and verification of distributed (non-probabilistic) systems, weak [58,53] or branching [28] bisimulation are the basis for a variety of verification methods because of three crucial properties. These equivalences (1) exploit abstraction from internal computation, they are (2) compositional with respect to parallel composition and other operators, and (3) efficient algorithms exist to minimise components with respect to their internal behaviour [57,45]. In this way, the omnipresent phenomenon of *state-space explosion* can be alleviated. Instead of building the global transition system underlying some (specification or) implementation, the transition system is built up componentwise, where weak (or branching) bisimulation algorithms are applied to minimise the intermediate state spaces of components. This can enormously reduce the size of the global transition system without affecting properties to be verified. An impressive example of this strategy is given in [14].

Several authors mentioned that the definition of a weak bisimulation that

abstracts from internal computation is desirable, but problematic in a probabilistic setting [43,32]. Segala&Lynch [64,65] introduce notions of weak and branching bisimulation for *concurrent* probabilistic systems. Their definition of weak and branching bisimulation replaces Milner's *weak transition relation* $\stackrel{\alpha}{\Longrightarrow}$ (which is defined with the help of the transitive, reflexive closure $(\stackrel{\tau}{\longrightarrow})^*$ of internal transitions) by assigning a (possibly infinite) set of distributions to each state. Their definitions are natural extensions of non-probabilistic weak and branching bisimulation to probabilistic systems with non-determinism (concurrent probabilistic systems) but it is not obvious how a definition for the fully probabilistic case can be derived.

In this paper, we introduce notions of weak bisimulation and branching bisimulation for *fully probabilistic* systems that arise as rather natural extensions of the corresponding relations in the non-probabilistic case. The basic idea is to replace Milner's weak transitions $s \stackrel{\alpha}{\Longrightarrow} t$ (i.e. $s(\stackrel{\tau}{\longrightarrow})^* \stackrel{\alpha}{\longrightarrow} (\stackrel{\tau}{\longrightarrow})^* t$) by the probabilities to reach state t from s via a sequence of transitions labelled by a trace of the form $\tau^*\alpha\tau^*$. In contrast to the non-probabilistic case where branching bisimulation equivalence is strictly finer than weak bisimulation equivalence, the two equivalences coincide for finite fully probabilistic systems. Moreover, we show that this notion of equivalence enjoys a central position relative to other weak relations for fully probabilistic systems found in the literature.

For mechanised verification purposes, the decidability of such equivalences, together with efficient algorithms for finite state systems is a crucial aspect. In the non-probabilistic case, for instance, *strong* bisimulation can be decided in time $\mathcal{O}(m \cdot \log n)$ [57] where n is the number of states and m the number of transitions in the underlying (finite) transition system. *Weak* bisimulation can be computed by means of the same algorithm, but on an induced transition system where the weak relation $\Longrightarrow$ replaces the strong relation $\longrightarrow$. In other words, the problem of deciding (or minimising with respect to) weak bisimulation is reduced to the computation of the transitive, reflexive closure $(\stackrel{\tau}{\longrightarrow})^*$ of internal transitions and deciding strong bisimulation in a finite system. Using the transitive closure operation from [20] and the partitioning/splitter technique by [57] the time complexity for deciding weak bisimulation is $\mathcal{O}(n^{2.3})$ where n is the number of states. For computing *branching* bisimulation of a non-probabilistic system, Groote&Vaandrager [30] propose an algorithm which works with a variant of the partitioning/splitter technique à la [45]. It uses both transition relations ($\longrightarrow$ and $\Longrightarrow$), and runs in time $\mathcal{O}(nm)$ where n is the number of states and m the number of transitions, i.e. the size of $\longrightarrow$.

As far as the authors know, the question whether weak or branching bisimulation on *concurrent* probabilistic systems (i.e. in the style of [64,65]) is decidable is still open, even though some effort in this direction has been carried out recently [59,9]. The main problem appears to be that the induced weak transi-

tion relation $\Longrightarrow$ might be infinite, even for finite systems; hence, an adaption of the methods used in the non-probabilistic case fails. However, we show that in the *fully probabilistic* case, weak (or branching) bisimulation equivalence *is* decidable for finite systems. We present an algorithm to compute the weak bisimulation equivalence classes with a modification of the partitioning/splitter technique à la [45,57]. The time complexity of our method is *cubic* in the number of states; thus, it meets the worst case complexity for deciding branching bisimulation in the non-probabilistic case [30] (where, in the worst case, $\mathcal{O}(m) = \mathcal{O}(n^2)$).

This provides us with an equivalence that (1) exploits abstraction from internal computation together with (3) an efficient algorithm to minimise components with respect to their internal behaviour. In order to cover all three main ingredients of a compositional verification method for fully probabilistic systems, we introduce a probabilistic calculus, *PLSCCS*. The calculus provides the standard CCS-style operators to specify systems, where probabilistic choice replaces non-deterministic choice, and parallel composition is *lazy synchronous*. Lazy synchrony refers to a form of synchronisation, where visible actions are forced to proceed in *lockstep*, i.e. synchronously, while internal actions are supposed to happen locally in some component, and hence are excluded from synchronisation, they happen asynchronously. Similar kinds of composition are present in *synchronous languages*, such as Esterelle [11], Lustre [31], or others [18,13,36]. We show that weak and branching bisimulation are congruences with respect to the operators of *PLSCCS*, with the exception (inherited from the non-probabilistic setting) of the probabilistic choice operator. So, *PLSCCS* provides means for a compositional specification of fully probabilistic systems. In addition, the algorithm for weak bisimulation can be used to minimise components of a complex fully probabilistic system with respect to their internal behaviour, without affecting properties to be verified.

Organisation of the paper: In Section 2 we introduce the fully probabilistic model together with basic notations used in the sequel. Section 3 introduces weak and branching bisimulation, and shows that both coincide for finite systems, It also discusses the relation between weak (and branching) bisimulation and other equivalences for fully probabilistic systems found in the literature. In Section 4 we present our algorithm for deciding weak bisimulation. Section 5 is devoted to the calculus *PLSCCS* and the proof of congruence. In Section 6 we illustrate how the congruence property and the algorithm can be exploited for a compositional verification technique. Section 7 concludes the paper. To enhance readability, most of the proofs are given in the appendix.

2 Fully probabilistic systems

In this section we introduce the *fully probabilistic* model, the model we focus on throughout this paper. It can be regarded as a specialisation of non-probabilistic *transition systems* where probabilities are used to resolve nondeterminism. From a slightly different point of view, it can also be interpreted as an action-labelled *Markov chain* with discrete parameter set [46].

In non-probabilistic labelled transition systems, the possible steps from states to successor states are described by a transition relation $\longrightarrow \subseteq S \times Act \times S$ (where Act stands for the underlying set of actions), i.e. the state changes are associated with action labels. Intuitively, $s \xrightarrow{a} t$ asserts that, in state s, it is possible to perform the action a and to reach state t afterwards. The probability of choosing one particular transition is unspecified. In contrast, fully probabilistic systems quantify the probability of each possible transition by means of a transition probability function $\mathbf{P}$, where $\mathbf{P}(s, a, t)$ determines the probability to perform the action a from state s and to reach t in doing so.

We first introduce some standard notations for actions and action sequences. For $L \subseteq Act$, L^* denotes the set of finite sequences over L. The empty sequence is denoted by ε. L^+ denotes the set of finite nonempty sequences over Act, i.e. $L^+ = L^* \setminus \{\varepsilon\}$. We further assume that Act contains a distinguished symbol τ. Intuitively, τ stands for any 'internal' activity of the system which is invisible for an observer (or the environment of the system). We refer to τ as the *internal* action. The other actions are called *visible*. We use greek letters $\alpha, \beta, \ldots$ to denote visible actions and arabic letters $a, b, \ldots$ to range over arbitrary actions.

Definition 2.1 *A* fully probabilistic system *is a tuple* $(S, Act, \mathbf{P})$ *consisting of a set* S *of states, a nonempty set* Act *of actions and a function* $\mathbf{P} : S \times Act \times S \longrightarrow [0, 1]$ *(called the* transition probability function*) such that, for each* $s \in S$, $\mathbf{P}(s, a, t) > 0$ *for at most countably many pairs* $(a, t) \in Act \times S$ *and* $\sum_{a,t} \mathbf{P}(s, a, t) \in \{0, 1\}$.

Let $(S, Act, \mathbf{P})$ be a fully probabilistic system. $\mathcal{S}$ is said to be *finite* iff S and Act are finite. A state s of $\mathcal{S}$ is called *terminal* iff $\sum_{a,t} \mathbf{P}(s, a, t) = 0$.

Example 2.2 *We consider a simple communication protocol similar to that in [33]. The system consists of two entities: a sender working with an unreliable medium to transmit messages to a receiver. The sender, having produced a message, passes the message to the medium, which in turn tries to deliver it to the receiver. With probability 0.01, the messages gets lost, in this case the medium retries to deliver it correctly. With probability 0.99, the message is delivered correctly. Once this has occured the sender waits for the acknowl-*

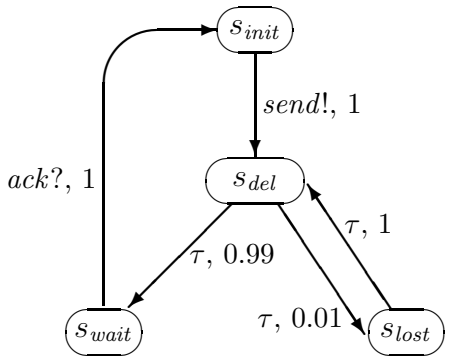


Fig. 1. The sender with action labels

edgement from the receiver's side and then returns to the initial state. For simplicity, we assume that the acknowledgement cannot be corrupted or lost. We describe the behaviour of the sender by the fully probabilistic system depicted in Figure 1, using four states:

- s_{init}*: the state in which the sender produces a message and passes the message to the medium. This is modelled by an output action (send!).*
- s_{del}*: the state in which the medium tries to deliver the message,*
- s_{lost}*: the state reached when the message is lost, and retransmission is necessary,*
- s_{wait}*: the state reached when the message is delivered correctly and in which the system waits for the acknowledgement by the receiver, modelled by an input action (ack?).*

For instance, the transition $s_{wait} \xrightarrow{ack?} s_{init}$ *stands for the case where the sender gets the acknowledgement of the receipt of the message;* $s_{del} \xrightarrow{\tau} s_{lost}$ *for the case where the medium looses the messages. This event, as well as retransmission and correct delivery, is supposed to be invisible from the point of view of the sender. In total, the behaviour of the sender can be specified by the fully probabilistic system* $(S, Act, \mathbf{P})$ *where* $S = \{s_{init}, s_{del}, s_{lost}, s_{wait}\}$ *and* $\mathbf{P}(s_{init}, send!, s_{del}) = \mathbf{P}(s_{lost}, \tau, s_{del}) = \mathbf{P}(s_{wait}, ack?, s_{init}) = 1$, $\mathbf{P}(s_{del}, \tau, s_{init}) = 0.01$, $\mathbf{P}(s_{del}, \tau, s_{wait}) = 0.99$ *and* $\mathbf{P}(\cdot) = 0$ *in all other cases.*

For $s \in S$, $C \subseteq S$, $a \in Act$ and $L \subseteq Act$, we define $\mathbf{P}(s, a, C) = \sum_{t \in C} \mathbf{P}(s, a, t)$, $\mathbf{P}(s, a) = \mathbf{P}(s, a, S)$, and $\mathbf{P}(s, L) = \sum_{a \in L} \mathbf{P}(s, a)$. An *execution fragment* or *finite path* is a nonempty finite sequence

$$\sigma = s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_2} \ldots \xrightarrow{a_k} s_k$$

such that $s_0, s_1, \ldots, s_k \in S$, $a_1, \ldots, a_k \in Act$ and $\mathbf{P}(s_{i-1}, a_i, s_i) > 0$, $i = 1, \ldots, k$. We use following notations,

- $|\sigma|$ denotes the *length* of σ, i.e. $|\sigma| = k$.
- $first(\sigma)$ denotes the first state of σ, i.e. $first(\sigma) = s_0$.
- $last(\sigma)$ denotes the last state of σ, i.e. $last(\sigma) = s_k$.

- $\sigma(i)$ denotes the $(i+1)$-st state of σ, i.e. $\sigma(i) = s_i$, $i = 0, 1, \ldots, k$.
- $\sigma^{(i)}$ denotes the i-th prefix of σ, i.e. $\sigma^{(i)} = s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \ldots \xrightarrow{a_i} s_i$, $i = 0, 1, \ldots, k$. If $i > k = |\sigma|$ then we let $\sigma^{(i)} = \sigma$.
- $\mathit{trace}(\pi) = a_1 a_2 \ldots a_k$.
- If $k = |\sigma| = 0$ then we put $\mathbf{P}(\sigma) = 1$. For $k \geq 1$, we define

$$\mathbf{P}(\sigma) = \mathbf{P}(s_0, a_1, s_1) \cdot \mathbf{P}(s_1, a_2, s_2) \cdot \ldots \cdot \mathbf{P}(s_{k-1}, a_k, s_k).$$

- σ is called *maximal* iff $\mathit{last}(\sigma)$ is terminal.
- state t is called *reachable* from state s if there exists a finite path σ with $\mathit{first}(\sigma) = s$ and $\mathit{last}(\sigma) = t$.

Example 2.3 *For the system in Example 2.2,* $\sigma = s_{init} \xrightarrow{send!} s_{del} \xrightarrow{\tau} s_{lost} \xrightarrow{\tau} s_{del} \xrightarrow{\tau} s_{wait}$ *is an execution fragment (finite path) with* $|\sigma| = 4$, $\mathit{first}(\sigma) = s_{init}$, $\mathit{last}(\sigma) = s_{wait}$, $\sigma(2) = s_{lost}$, $\sigma^{(2)} = s_{init} \longrightarrow s_{del} \longrightarrow s_{lost}$ *and* $\mathbf{P}(\sigma) = 1 \cdot 0.01 \cdot 1 \cdot 0.99 = 0.0099$. *It has no finite maximal execution fragment as there are no terminal states.*

An *execution* or *fulpath* in $(S, Act, \mathbf{P})$ is either a maximal execution fragment or an infinite sequence $\pi = s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_2} \ldots$ where $s_0, s_1, \ldots, \in S$, $a_1, a_2, \ldots \in Act$ and $\mathbf{P}(s_{i-1}, a_i, s_i) > 0$, $i = 1, 2, \ldots$. A *path* denotes an execution fragment or an execution. For path π to be an infinite sequence, $\pi(i)$, $\pi^{(i)}$ and $\mathit{first}(\pi)$ are defined as for execution fragments, $\mathit{trace}(\pi) = a_1 a_2 \ldots$, $|\pi| = \infty$, and we define

$$\mathit{inf}(\pi) = \{s \in S : \pi(i) = s \text{ for infinitely many indices } i \geq 0\}.$$

If, on the other hand, σ is a finite path then we define $\mathit{inf}(\sigma) = \emptyset$.

Example 2.4 *For the system of Figure 1,*

$$\pi = s_{init} \xrightarrow{send!} s_{del} \xrightarrow{\tau} s_{lost} \xrightarrow{\tau} s_{del} \xrightarrow{\tau} s_{lost} \xrightarrow{\tau} \ldots$$

is an execution (fulpath) with $\pi^{(2)} = s_{init} \xrightarrow{send!} s_{del} \xrightarrow{\tau} s_{lost}$, $\mathit{first}(\pi) = s_{init}$, $\pi(3) = s_{del}$, $\mathbf{P}(\pi^{(4)}) = 1 \cdot 0.01 \cdot 1 \cdot 0.01 = 0.0001$ *and* $\mathit{trace}(\pi^{(4)}) = send!\ \tau\ \tau\ \tau$.

We let $\mathit{Path}^{\mathcal{S}}_{ful}$ ($\mathit{Path}^{\mathcal{S}}_{ful}(s)$) denote the set of fulpaths in $\mathcal{S}$ (starting in state s). Similarly, $\mathit{Path}^{\mathcal{S}}_{fin}$ ($\mathit{Path}^{\mathcal{S}}_{fin}(s)$) denotes the set of finite paths in $\mathcal{S}$ (starting in state s). The set of states which are reachable from s are subsumed in $\mathit{Reach}^{\mathcal{S}}(s)$. If the underlying fully probabilistic system $\mathcal{S}$ is clear from the context we usually omit the superscript $\mathcal{S}$. For each state s, $\mathbf{P}$ induces a probability space on $\mathit{Path}_{ful}(s)$ as follows. Let $\sigma\uparrow$ denote the *basic cylinder* induced by σ, i.e.

$$\sigma\uparrow \;=\; \{\pi \in \mathit{Path}_{ful}(s) : \sigma \leq_{prefix} \pi\}\,.$$

where $\leq_{prefix}$ is the usual prefix relation on paths. We define $\sigma\mathit{Field}^{\mathcal{S}}(s)$ to

be the smallest sigma-field on $Path_{ful}(s)$ which contains all basic cylinders $\sigma\uparrow$ where $\sigma \in Path_{fin}(s)$, i.e. σ ranges over all finite paths starting in s. The probability measure $Prob$ on $\sigma Field^{\mathcal{S}}(s)$ is the unique measure with $Prob\,(\sigma\uparrow) = \mathbf{P}(\sigma)$.

Lemma 2.5 *Let $(S, Act, \mathbf{P})$ be a fully probabilistic system and $\Sigma \subseteq Path_{fin}$ such that σ, $\sigma' \in \Sigma$, $\sigma \neq \sigma'$ implies $\sigma \not\leq_{prefix} \sigma'$. Then, for all $s \in S$,*

$$Prob\,(\Sigma(s)\uparrow) = \sum_{\sigma\in\Sigma(s)} \mathbf{P}(\sigma).$$

3 Weak and branching bisimulation

In this section we define weak and branching bisimulation for fully probabilistic systems. While in the non-probabilistic case branching bisimulation equivalence is strictly finer than weak bisimulation equivalence, these two relations coincide for finite fully probabilistic systems (Theorem 3.9). As a starting point, we recall strong probabilistic bisimulation, an elegant generalisation of non-probabilistic bisimulation [51,58]. Originally introduced by Larsen&Skou [48] for reactive systems, it has been reformulated by van Glabbeek *et al* [27] for fully probabilistic systems.

Definition 3.1 *A* strong bisimulation *on a fully probabilistic system $(S, Act, \mathbf{P})$ is an equivalence relation R on S such that $(s, s') \in R$ implies for all $a \in Act$ and all $C \in S/R$.*

$$\mathbf{P}(s, a, C) = \mathbf{P}(s', a, C)$$

Two states s, s' are called strongly bisimulation equivalent *(denoted $s \sim s'$) iff $(s, s') \in R$ for some strong bisimulation R.*

3.1 Weak bisimulation

In order to abstract from internal moves, non-probabilistic weak bisimulation neglects these moves as far as they do not influence the future behaviour of a process. To do so, Milner [53] introduces a notion of observable steps of a process, that consist of a single *visible* action α preceded and followed by an arbitrary number (including zero) of internal steps. Technically this is achieved by deriving a 'weak' transition relation $\Longrightarrow$ from the 'strong' relation $\longrightarrow$, by setting $\stackrel{\alpha}{\Longrightarrow} = (\stackrel{\tau}{\longrightarrow})^* \stackrel{\alpha}{\longrightarrow} (\stackrel{\tau}{\longrightarrow})^*$, and $\stackrel{\tau}{\Longrightarrow} = (\stackrel{\tau}{\longrightarrow})^*$. Note that a weak internal transition $\stackrel{\tau}{\Longrightarrow}$ is possible without actually performing an internal action.

For the definition of weak bisimulation in the fully probabilistic setting, we replace Milner's weak internal transitions $s \stackrel{\tau}{\Longrightarrow} t$ by the probability $Prob(s, \tau^*, t)$ to reach state t from s via internal actions. Similarly, for visible actions α. we deal with the probabilities $Prob(s, \tau^*\alpha\tau^*, t)$ rather than the weak transition relation $\stackrel{\alpha}{\Longrightarrow}$.

Definition 3.2 *A* weak bisimulation *on a fully probabilistic system* $(S, Act, \mathbf{P})$ *is an equivalence relation* R *on* S *such that* $(s, s') \in R$ *implies for all* $a \in Act$ *and all* $C \in S/R$*:*

(1) $Prob(s, \tau^*, C) \;=\; Prob(s', \tau^*, C)$
(2) $Prob(s, \tau^*\alpha\tau^*, C) \;=\; Prob(s', \tau^*\alpha\tau^*, C)$ *for all* $\alpha \in Act \setminus \{\tau\}$.

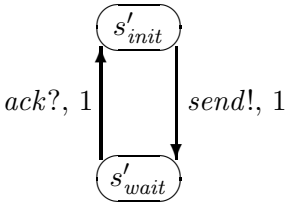
Two states s, s' *are called* weakly bisimulation equivalent *(denoted* $s \;\approx\; s'$*) iff* $(s, s') \in R$ *for some weak bisimulation* R.

Note that $Prob(s, \tau^*, C) = 1$ if $s \in C$. Hence, condition (1) is always fulfilled for the equivalence class C of s (and s'). It is easy to see that any strong bisimulation is a weak bisimulation, and hence $\sim \subseteq \approx$.

Lemma 3.3 *For finite systems,* $\approx$ *is a weak bisimulation.*

The proof is given in the appendix (cf. Lemma A.17).

Example 3.4 *We compare the fully probabilistic system describing the sender's behaviour shown in Figure 1 with a simple specification of a communication protocol depicted below.*



Using weak bisimulation equivalence as the underlying implementation relation the sender can be verified against this specification. To illustrate this, we show that the initial states s_{init} *and* s'_{init} *are weakly bisimulation equivalent. Let* R *be the equivalence on* $S = \{s_{init}, s_{del}, s_{wait}, s_{lost}, s'_{init}, s'_{wait}\}$ *such that* $S/R = \{C_I, C_W\}$ *where* $C_I = \{s_{init}, s'_{init}\}$ *is the equivalence class of the initial states and* $C_W = \{s_{del}, s_{wait}, s_{lost}, s'_{wait}\}$ *the equivalence class of the other states. For*

$s_I \in C_I$ and $s_W \in C_W$, we have:

$$
\begin{array}{ll}
Prob(s_I, \tau^*, C_I) = 1, & Prob(s_W, \tau^*, C_I) = 0, \\
Prob(s_I, \tau^*, C_W) = 0, & Prob(s_W, \tau^*, C_W) = 1, \\
Prob(s_I, \tau^*\ send!\ \tau^*, C_I) = 0, & Prob(s_W, \tau^*\ send!\ \tau^*, C_I) = 0, \\
Prob(s_I, \tau^*\ ack?\ \tau^*, C_I) = 0, & Prob(s_W, \tau^*\ ack?\ \tau^*, C_I) = 1, \\
Prob(s_I, \tau^*\ send!\ \tau^*, C_W) = 1, & Prob(s_W, \tau^*\ send!\ \tau^*, C_W) = 0, \\
Prob(s_I, \tau^*\ ack?\ \tau^*, C_W) = 0, & Prob(s_W, \tau^*\ ack?\ \tau^*, C_W) = 0.
\end{array}
$$

Hence, R is a weak bisimulation. In particular, the initial states s_{init} of the sender and s'_{init} of its specification are weakly bisimulation equivalent.

In the non-probabilistic case, it holds for weakly bisimulation equivalent states s, s' that if $s \stackrel{\alpha_1 \ldots \alpha_k}{\Longrightarrow} t$ then there is some t' such that $s' \stackrel{\alpha_1 \ldots \alpha_k}{\Longrightarrow} t'$, and that t, t' are weakly bisimulation equivalent. Here, $\stackrel{\alpha_1 \ldots \alpha_k}{\Longrightarrow}$ denotes $(\stackrel{\tau}{\longrightarrow})^* \stackrel{\alpha_1}{\longrightarrow} (\stackrel{\tau}{\longrightarrow})^* \ldots (\stackrel{\tau}{\longrightarrow})^* \stackrel{\alpha_k}{\longrightarrow} (\stackrel{\tau}{\longrightarrow})^*$. This result carries over to finite fully probabilistic systems.

Theorem 3.5 *Let $(S, Act, \mathbf{P})$ be a finite fully probabilistic system and Ω a regular expression of the form $\tau^*\alpha_1\tau^*\alpha_2\tau^* \ldots \tau^*\alpha_k$ or $\tau^*\alpha_1\tau^*\alpha_2\tau^* \ldots \tau^*\alpha_k\tau^*$. Then:*

$$\textit{If } s \approx s' \textit{ then } Prob(s, \Omega, C) = Prob(s', \Omega, C) \textit{ for all } C \in S/\approx.$$

Proof: see Section A.1, Theorem A.18. ■

3.2 Branching bisimulation

From a specific point of view, non-probabilistic weak bisimulation appears too coarse. Strong bisimulation has the property that any computation in one process corresponds to a computation in the other in such a way that all intermediate steps correspond as well. This is not true for weak bisimulation. The standard counterexample (taken from [28]) is depicted in Figure 2. In this example, s and s' are weakly bisimulation equivalent, but the computations $s \stackrel{\alpha}{\Longrightarrow} t \stackrel{\beta}{\Longrightarrow} w$ and $s' \stackrel{\alpha}{\Longrightarrow} v'' \stackrel{\beta}{\Longrightarrow} w''$ pass through intermediate states (t and v'') that are not equivalent. In particular, the latter computation does not pass through any state where γ is possible.

In order to overcome this lack, van Glabbeek&Weijland postulate a branching condition [28]. The basic idea is that in order to simulate a step $s \stackrel{\alpha}{\longrightarrow} t$ by an equivalent state s' with a sequence $s' \stackrel{\tau}{\Longrightarrow} \stackrel{\alpha}{\longrightarrow} \stackrel{\tau}{\Longrightarrow} t'$, all steps preceding

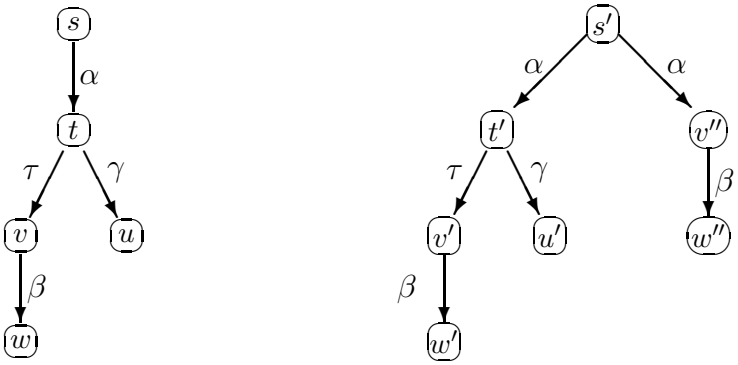


Fig. 2. Distinguishing weak and branching bisimulation in the non-probabilistic case

the transition $\xrightarrow{\alpha}$ have to remain inside the equivalence class of s (and s') and all steps following this transition have to stay inside the class of t (and hence of t'). To achieve this property it is sufficient to demand that s' performs arbitrarily many internal actions leading to a state s'' which is still equivalent to s' (and s) and then to directly reach the equivalence class of t by performing action α. This implies that all intermediate states on the path from s' to s'' also belong to equivalence class of s', s, and s'' [28].

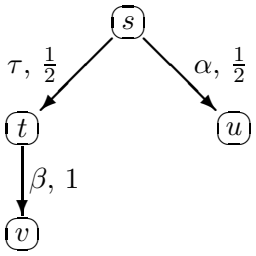
To adapt this idea to the fully probabilistic case, we deal with the probability to reach an equivalence class C (of a given equivalence relation R) from a state s by means of a trace labelled $\tau^*\alpha$, where the τ steps only pass through states equivalent to s (with respect to R). This probability will be denoted $Prob_R(s, \tau^*\alpha, C)$, and is defined below. For notational convenience, we shall use $\widehat{\tau}$ to denote ε, whereas for visible actions, $\widehat{\alpha} = \alpha$. Recall that ε denotes the empty word in Act^*. Hence, $\tau^*\widehat{a} = \tau^*$ if $a = \tau$.

Definition 3.6 *Let $(S, Act, \mathbf{P})$ be a fully probabilistic system, R an equivalence relation on S, $s \in S$, $C \subseteq S$ and $a \in Act$. Then, $Path^R_{ful}(s, \tau^*\widehat{a}, C)$ denotes the set of fulpaths $\pi \in Path_{ful}(s)$ such that there is some $k \geq 0$ with*

- $(s, \pi(i)) \in R$, $i = 1, \ldots, k-1$,
- $trace(\pi^{(k)}) \in \tau^*\widehat{a}$,
- $\pi(k) \in C$.

Let $Prob_R(s, \tau^*\widehat{a}, C) = Prob(Path^R_{ful}(s, \tau^*\widehat{a}, C))$, and furthermore $Prob_R(s, \tau^*\widehat{a}, t) = Prob((s, \tau^*\widehat{a}, \{t\})$.

Example 3.7 *The reader is invited to check that for the relation R in Example 3.4, we have $Prob_R(s, \tau^*\widehat{a}\tau^*, C) = Prob(s, \tau^*\widehat{a}\tau^*, C)$ for all states s, classes $C \in \{C_I, C_W\}$ and $a \in \{\tau, send!, ack?\}$.*



For the system shown above and the (identity) relation R with $(x, y) \in R$ iff $x = y$, we have $Prob_R(s, \tau^*\beta, v) = 0$ while $Prob(s, \tau^*\beta, v) = 1/2$.

It is worth noticing that for arbitrary equivalences R on S, and $C \subseteq S$, $s \in C$ implies $Path_{ful}(s) = Path^R_{ful}(s, \tau^*, C)$, and hence $Prob_R(s, \tau^*, C) = 1$ if $s \in C$. We now have the necessary means to lift branching bisimulation to the fully probabilistic case.

Definition 3.8 *Let $(S, Act, \mathbf{P})$ be a fully probabilistic system. A* branching bisimulation *on $(S, Act, \mathbf{P})$ is an equivalence relation R on S such that for all $(s, s') \in R$, $C \in S/R$:*

(1) $Prob_R(s, \tau^*, C) = Prob_R(s, \tau^*, C)$
(2) $Prob_R(s, \tau^*\alpha, C) = Prob_R(s, \tau^*\alpha, C)$ *for all* $\alpha \in Act \setminus \{\tau\}$.

Two states s, s' are called branching bisimulation equivalent *(denoted $s \approx_{br} s'$) iff $(s, s') \in R$ for some branching bisimulation R.*

For finite systems, branching bisimulation equivalence $\approx_{br}$ is a branching bisimulation, as shown in the appendix (cf. Lemma A.16). In contrast to the non-probabilistic case, where branching bisimulation equivalence is strictly finer than weak bisimulation equivalence, the branching condition *does not* add any distinguishing power to the relation in the fully probabilistic setting; weak and branching bisimulation equivalence coincide for finite systems:

Theorem 3.9 *Let $(S, Act, \mathbf{P})$ be a finite fully probabilistic system and s, $s' \in S$. Then, $s \approx s'$ iff $s \approx_{br} s'$.*

Proof: The proof is given in the appendix (cf. Corollary A.14). ■

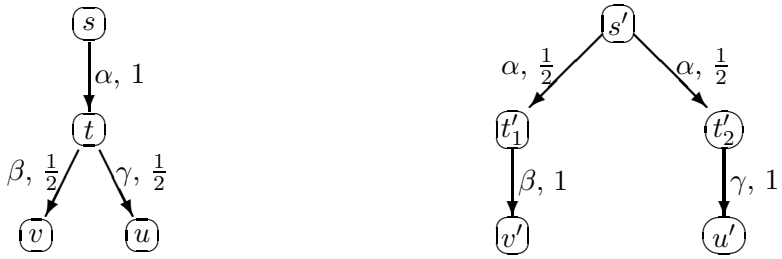
To exemplify the difference to the non-probabilistic setting, we return to the standard example depicted in Figure 2. In the non-probabilistic case, s and s' are weakly but not branching bisimulation equivalent. However, if we add non-zero probabilities (which turns the system into a fully probabilistic system) then s and s' are neither branching, nor weakly bisimulation equivalent. This can be seen as follows. Assume that $s \approx s'$ holds. Then, $1 = Prob(s, \tau^*\alpha\tau^*, T) = Prob(s', \tau^*\alpha\tau^*, T)$ has to hold, where T denotes the weak bisimulation equivalence class of t. Clearly, v'' does not belong to T, because t can perform γ (with non-zero probability) while v'' can-

not. Hence $Prob(s', \tau^*\alpha\tau^*, T) = 1$ implies $\mathbf{P}(s', \alpha, t') = 1$, and therefore $\mathbf{P}(s', \alpha, v'') = 0$. This contradicts our decision to add *non-zero* probabilities to the non-probabilistic system of Figure 2, in particular transition $s' \xrightarrow{\alpha} v''$.

3.3 Connection to other equivalences

In this section we discuss how weak (and branching) bisimulation relates to other equivalences for fully probabilistic systems studied in the literature. First, it is not surprising that weak bisimulation equivalence $\approx$ is strictly coarser than strong bisimulation equivalence $\sim$ (Definition 3.1) since the latter does not abstract from internal moves. Formally, if $(S, Act, \mathbf{P})$ is a fully probabilistic system and s, s' are strongly bisimulation equivalent states then s and s' are also weakly bisimulation equivalent. Moreover, if the system is τ-free (i.e. $\mathbf{P}(t, \tau) = 0$ for all states t) then weak and strong bisimulation equivalence coincide. (Note that, for a τ-free system $(S, Act, \mathbf{P})$ we have $Prob(s, \tau^*\alpha\tau^*, C) = \mathbf{P}(s, \alpha, C)$ for arbitrary states s, and $C \subseteq S$.)

Of course, we cannot expect weak bisimulation to be comparable with *strong* trace, failure or ready equivalence in the sense of Jou&Smolka [43]. The reason is that weak bisimulation equivalence abstracts from internal steps while the equivalences of [43] do not treat these τ-steps in a distinct way and are strictly coarser than strong (and hence weak) bisimulation equivalence for τ-free systems. For instance, the states s and s' of the system below are strongly trace equivalent but not (strongly or weakly) bisimulation equivalent.



Vice versa, the states s and s' of the system $(\{s, s', t\}, \{\tau, \alpha\}, \mathbf{P})$ where $\mathbf{P}(s, \tau, s') = \mathbf{P}(s', \alpha, t) = 1$ and $\mathbf{P}(\cdot) = 0$ in all other cases are weakly bisimulation equivalent but neither strong trace, nor failure nor ready equivalent in the sense of [43]. So, we turn our attention to the *weak* counterparts of the equivalences proposed in [43]. Theorem 3.5 implies that for finite systems, $\approx$ is strictly finer than weak trace, failure or ready equivalence. Here, e.g. s, s' are called *weakly trace equivalent* iff

$$Prob(s, \tau^*\alpha_1\tau^* \ldots \tau^*\alpha_k\tau^*) = Prob(s', \tau^*\alpha_1\tau^* \ldots \tau^*\alpha_k\tau^*)$$

for all $k \geq 0$ and $\alpha_1, \ldots, \alpha_k \in Act \setminus \{\tau\}$.

Notions of probabilistic testing have also been introduced in the literature. Christoff [15] and Cleaveland et al [19] (see also [73]) introduce *testing equivalences* for finite fully probabilistic processes that relate two processes in terms of the reliability in certain environments. While [15] deal with deterministic environments [19] consider probabilistic testing scenarios. Both abstract from internal computations. In the remainder of this section we discuss the relation between these testing preorders and our notion of weak bisimulation. To do so, we fix a finite fully probabilistic system $(S, Act, \mathbf{P})$.

Testing equivalence of Christoff: We show that weak bisimulation $\approx$ is stronger than the testing equivalences introduced by Christoff [15] (see also [16,17]). In [15], fully probabilistic processes are distinguished by means of the conditional probabilities of certain deterministic testing scenarios. The several testing scenarios lead to the definitions of *probabilistic trace equivalence* $=_{tr}$, *weak probabilistic testing equivalence* $=_{wte}$ and *strong probabilistic testing equivalence* $=_{ste}$. As shown in [15], $=_{tr} \supseteq =_{wte} \supseteq =_{ste}$. We show that weak bisimulation equivalence $\approx$ is stronger than strong probabilistic testing equivalence $=_{ste}$ (and thus, it is also stronger than $=_{wte}$ and $=_{tr}$).

We briefly recall the definition of strong probabilistic testing equivalence. More precise, we use an equivalent characterisation of $=_{ste}$ which is given in [17], see there for more explanations. Let *Offr* be the set of nonempty subsets of $Act \setminus \{\tau\}$ (the set of *offerings*) and $Offr^*$ the set of (finite) strings of offerings. ε_{Off} denotes the empty string of offerings. Now, for $L_1, \ldots, L_k \in Offr$ and $\alpha_1, \ldots, \alpha_r \in Act \setminus \{\tau\}$, we use $Q(s, L_1 \ldots L_k, \alpha_1 \ldots \alpha_r, t)$to denote the probability of performing the string $\tau^*\alpha_1 \ldots \tau^*\alpha_r$ ending up in state t when offered a string of $L_1 \ldots L_k$. The formal definition of $Q(\cdot)$ is as follows.

Definition 3.10 *The function*

$$Q : S \times Offr^* \times (Act \setminus \{\tau\})^* \times 2^S \longrightarrow [0,1]$$

is defined as follows. Let $s \in S$, $C \subseteq S$, $L \in Offr$, $\alpha \in Act \setminus \{\tau\}$, $\tilde{L} \in Offr^*$, $\tilde{\alpha} \in (Act \setminus \{\tau\})^*$.

$$\begin{aligned}
Q(s, \varepsilon_{Off}, \tilde{\alpha}, C) &= 0 \quad \textit{if } \tilde{\alpha} \neq \varepsilon \\
Q(s, \tilde{L}, \varepsilon, C) &= \begin{cases} 1 : \textit{if } s \in C \\ 0 : \textit{otherwise} \end{cases} \\
Q(s, L\tilde{L}, \alpha\tilde{\alpha}, C) &= \sum_{u \in S} Q(s, L, \alpha, C) \cdot Q(u, \tilde{L}, \tilde{\alpha}, C) \\
Q(s, L, \alpha, C) &= 0 \quad \textit{if } \alpha \notin L
\end{aligned}$$

If $\alpha \in L$ then the values $Q(s, L, \alpha, C)$, $s \in S$, $C \subseteq S$ can be computes as the

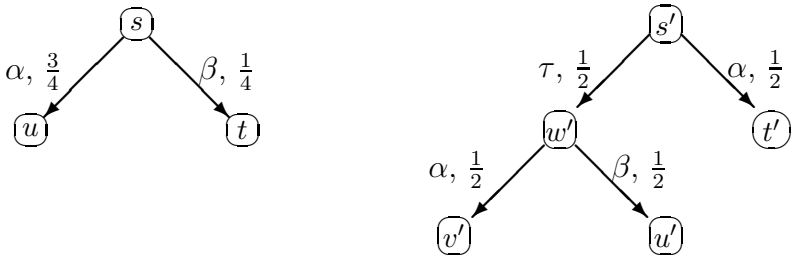


Fig. 3. $s =_{ste} s'$ but $s \not\approx s'$

unique solution of the following linear equation system.

1. $Q(s, L, \alpha, C) = 0$ if $Prob(s, \tau^*\alpha, C) = 0$.
2. If $Prob(s, \tau^*\alpha, C) > 0$ then

$$Q(s, L, \alpha, C) \;=\; \frac{\mathbf{P}(s, \alpha, C)}{\mathbf{P}(s, \tau) + \mathbf{P}(s, L)} + \sum_{u \in S} \frac{\mathbf{P}(s, \tau, u)}{\mathbf{P}(s, \tau) + \mathbf{P}(s, L)} \cdot Q(s, L, \alpha, C).$$

Note that $Prob(s, \tau^*\alpha, C) > 0$, $\alpha \in L$ implies $\mathbf{P}(s, \tau) + \mathbf{P}(s, L) > 0$. If $\tilde{L} \in Offr^*$ and $\tilde{\alpha} \in (Act \setminus \{\tau\})^*$ then we put $Q(s, \tilde{L}, \tilde{\alpha}) \;=\; Q(s, \tilde{L}, \tilde{\alpha}, S)$. This probability $Q(s, \tilde{L}, \tilde{\alpha})$ denotes the probability of performing the sequence $\tilde{\alpha}$ (interweaved with arbitrary many internal steps) from state s when offered a string of $\tilde{L}$. It is the basis of *strong testing equivalence*, $=_{ste}$.

Definition 3.11 *$s =_{ste} s'$ iff for all $\tilde{L} \in Offr^*$, $\tilde{\alpha} \in (Act \setminus \{\tau\})^*$,*

$$Q(s, \tilde{L}, \tilde{\alpha}) = Q(s', \tilde{L}, \tilde{\alpha})$$

Theorem 3.12 *$\approx$ is strictly finer than $=_{ste}$.*

Proof: In Appendix A.2, Theorem A.20 we show that $\approx$ is finer than $=_{ste}$. To see that $=_{ste}$ and $\approx$ do not coincide consider the fully probabilistic system of Figure 3. Then, $s =_{ste} s'$ as, for instance,

$$Q(s, \{\alpha, \beta\}, \alpha) \;=\; \tfrac{3}{4} \;=\; \tfrac{1}{2} \;+\; \tfrac{1}{2} \cdot \tfrac{1}{2} \;=\; Q(s', \{\alpha, \beta\}, \alpha)$$

and $Q(s, \{\alpha\}, \alpha) = 1 = Q(s', \{\alpha\}, \alpha)$. On the other hand,

$$Prob(s', \tau^*\alpha, S) = 3/4 > 1/2 = Prob(w', \tau^*\alpha, S).$$

Hence, $s' \not\approx w'$. Thus, $Prob(s', \tau^*, W) = 1/2 > 0 = Prob(s, \tau^*, W)$ where W is the weak bisimulation equivalence class of w'. Thus, $s \not\approx s'$. ■

Algorithms for deciding the three kinds of testing equivalences are presented in [17]. They are based on iteratively solving linear equation systems, and run in time $\mathcal{O}(n^4)$ where n is the number of states of the underlying system. As we will see in Section 4, using our (finer) notion of weak bisimulation instead results in a reduction of complexity, since our algorithm runs in cubic time. The reason is that our algorithm uses conditional probabilities which can

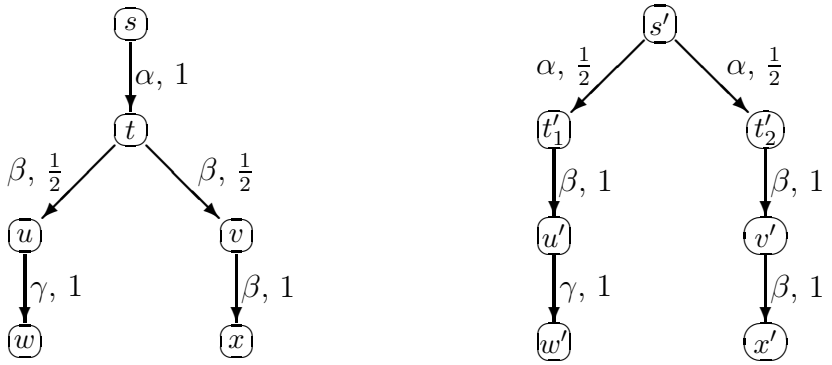


Fig. 4. $s \equiv s'$ but $s \not\approx s'$

be computed by simple arithmetic operations rather than by solving linear equation systems.

Testing equivalences of Cleaveland et al: In [19] (see also [73]) quantitative extensions of the non-probabilistic testing preorders by de Nicola&Hennessy [55,38] are discussed. Given a test $\mathcal{T}$ – which is represented by a fully probabilistic system equipped with a set of *success* states – the probability for a fully probabilistic process $\mathcal{P}$ to *pass* the test $\mathcal{T}$ is defined as the probability measure of the set of 'interaction sequences' leading to a success state. Intuitively, given a class of tests, two fully probabilistic processes $\mathcal{P}$, $\mathcal{P}'$ are testing equivalent with respect to a certain class of tests iff $\mathcal{P}$ and $\mathcal{P}'$ pass all tests $\mathcal{T}$ of that class with the same probability. Two classes of tests are considered in [19]:

- the class Tests_0 of τ-free tests which yields the testing equivalence $\equiv_0$,
- the class *Tests* of all tests that do not contain 'τ-loops' which yields the testing equivalence $\equiv$.

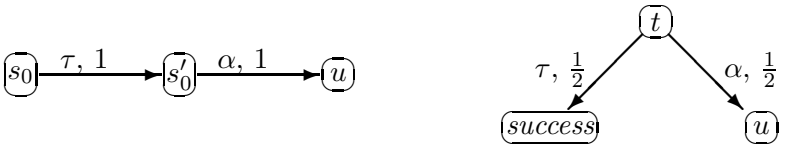
The exact definition (more precise, an alternative characterisation) of $\equiv_0$ is given in Appendix A.2 where we prove that $\equiv_0$ is coarser than $\approx$. For the precise definition of $\equiv$ see [19] or [73].

Theorem 3.13

(a) $\approx$ *is strictly finer than* $\equiv_0$.
(b) $\approx$ *and* $\equiv$ *are incomparable.*

Proof: In Appendix A.2, Theorem A.27 we show that $\approx$ is finer than $\equiv_0$. As pointed out in [73], the states s and s' of the system shown in Figure 4 are testing equivalent with respect to $\equiv$ (and hence, testing equivalent with respect to $\equiv_0$). On the other hand, s and s' are *not* weakly bisimulation equivalent, because $\mathit{Prob}(s, \tau^*\alpha\tau^*, T) = 1$ while $\mathit{Prob}(s', \tau^*\alpha\tau^*, T) = 0$ where T denotes the weak bisimulation equivalence class of t. (Note that neither t'_1 nor t'_2 is weakly bisimulation equivalent to t.) To illustrate that $\approx$ and $\equiv$ are incomparable, we remark that the states s_0 and s'_0 of the system shown below

are weakly bisimulation equivalent, but $s_0 \not\equiv s_0'$.



For instance, the test $\mathcal{T}$ shown on the right distinguishes the states s_0 and s_0'. The probability for s_0 to pass the test $\mathcal{T}$ is $3/4$ while the probability for s_0' to pass $\mathcal{T}$ is $1/2$. ■

4 Deciding weak bisimulation equivalence

In this section we develop an algorithm to compute the weak bisimulation equivalence classes. As pointed out earlier, an efficient algorithm for finite state systems is crucial for mechanised verification purposes, since it allows to minimise systems with respect to their internal behaviour. The general idea of our algorithm is to use a partition refinement technique similar to the ones proposed by Kanellakis&Smolka [45] resp. Paige&Tarjan [57] for deciding strong bisimulation equivalence in the non-probabilistic case.

For a given finite set S of states, a *partition* $\mathcal{X}$ of S is a set $\{C_1, \ldots, C_n\}$ such that $\bigcup_i C_i = S$ and where the C_i are pairwise disjoint. The elements of a partition $\mathcal{X}$ are sets of states, and are called *blocks* in the sequel. Each partition $\mathcal{X}$ of S induces an equivalence relation $\bigcup_{C \in \mathcal{X}} C \times C$ on S. Vice versa, the set of equivalence classes of some equivalence relation on S form a partition of S. Therefore, in the sequel we blur the distinction between between a partition and its induced equivalence, and between blocks and equivalence classes.

The general idea of partition refinement to compute weak bisimulation equivalence is as follows: The algorithm starts with some 'simple' initial partition $\mathcal{X}_{init}$ that is coarser than $\approx$ and then successively refines the given partition $\mathcal{X}$ with the help of a 'splitter' of $\mathcal{X}$, eventually resulting in the set of weak bisimulation equivalence classes. The crucial point is the definition of a splitter. A possible candidate for a 'splitter' of a partition $\mathcal{X}$ is a pair $(a, C) \in Act \times \mathcal{X}$ that violates the condition for $\mathcal{X}$ to be a weak bisimulation, i.e.

$$(*)\ \ Prob(s, \tau^* \widehat{a} \tau^*, C) \ \neq\ Prob(s', \tau^* \widehat{a} \tau^*, C) \text{ for some } B \in \mathcal{X} \text{ and } s,\, s' \in B.$$

One approach for a partition refinement algorithm would be to refine $\mathcal{X}$ according to a splitter in the sense of (*), i.e. to replace $\mathcal{X}$ by $Refine'(\mathcal{X}, a, C) = \{B/ \simeq_{(a,C)} \colon B \in \mathcal{X}\}$ where

$$s \simeq_{(a,C)} s' \ \text{ iff } \ Prob(s, \tau^* \widehat{a} \tau^*, C) = Prob(s', \tau^* \widehat{a} \tau^*, C).$$

The probabilities $Prob(s, \tau^*\widehat{a}\tau^*, C)$ can be computed by solving the linear equation system (cf. [8]).

$$
\begin{aligned}
x_s &= 1 \qquad \text{if } a = \tau \text{ and } s \in C \\
x_s &= 0 \qquad \text{if } Path_{ful}(s, \tau^*\widehat{a}\tau^*, C) = \emptyset \\
x_s &= \sum_{t \in S} \mathbf{P}(s, \tau, t) \cdot x_t \; + \; \mathbf{P}(s, a, C) \\
&\qquad \text{if } Path_{ful}(s, \tau^*\widehat{a}\tau^*, C) \neq \emptyset \text{ and } a \neq \tau \vee s \notin C.
\end{aligned}
$$

The test whether $Path_{ful}(s, \tau^*\widehat{a}\tau^*, C) = \emptyset$ can be done by a reachability analysis of the underlying directed graph, e.g. with a depth first search like method. Then, $\Omega(n^{3.8})$ is an asymptotic lower bound for the time complexity of this method, where, n is the number of states. Note that in the worst case we need n refinement steps and in each refinement step we have to solve a linear equation system with n variables and n equations (which takes $\mathcal{O}(n^{2.8})$ time with the method of [2]).

As a more efficient alternative, we develop an algorithm that runs in time $\mathcal{O}(n^3)$ in the sequel. The basic idea is to replace (*) by a condition that asserts that $\mathcal{X}$ violates the conditions of a *branching* bisimulation. To realise this idea, we use an alternative definition of a splitter that is based on an characterisation of branching bisimulations which uses the *conditional probability* $\mathbf{P}_{\mathcal{X}}(s, a, C)$, the probability to reach a block $C \in \mathcal{X}$ from state s via action a within one step *under the condition* that the system does not make an internal move inside the block (of $\mathcal{X}$) that contains s. These conditional probabilities can be computed by simple *arithmetic operations*. Thus, the use of this kind of splitters has the advantage that in the refinement steps we do not have to solve linear equation systems.

4.1 The algorithm

In what follows, we fix a finite fully probabilistic system $(S, Act, \mathbf{P})$ and a partition $\mathcal{X}$ of S. $[s]_{\mathcal{X}}$ denotes the unique block in $\mathcal{X}$ that contains s. We say that $\mathcal{X}$ is a weak (branching) bisimulation iff the induced equivalence relation $R_{\mathcal{X}} = \bigcup_{C \in \mathcal{X}} C \times C$ is a weak (branching) bisimulation. We use S_{term} to denote the set of terminal states (i.e. all states $s \in S$ where $\mathbf{P}(s, a, t) = 0$ for all $a \in Act$ and $t \in S$). Furthermore,

Definition 4.1 *We say that state s is* silent*, if s is terminal or $\mathbf{P}(s, \tau, [s]_{\mathcal{X}}) = 1$, otherwise we call state s* non-silent*. We let*

$$S_{ns}^{\mathcal{X}} \;=\; \{s \in S \setminus S_{term} : \mathbf{P}(s, \tau, [s]_{\mathcal{X}}) < 1\}$$

denote the set of all non-silent states with respect to $\mathcal{X}$.

Thus a non-silent states will with non-zero probability either perform something visible (in case that $\mathbf{P}(s,\tau) < 1$) or silently step into a different class (in case that $\mathbf{P}(s,\tau,t) > 0$ for some $t \notin [s]_{\mathcal{X}}$).

Definition 4.2 *If $s \in S_{ns}^{\mathcal{X}}$ and $(a,C) \in Act \times \mathcal{X}$ with $(a,C) \neq (\tau,[s]_{\mathcal{X}})$ then we define*

$$\mathbf{P}_{\mathcal{X}}(s,a,C) \;=\; \frac{\mathbf{P}(s,a,C)}{1-\mathbf{P}(s,\tau,[s]_{\mathcal{X}})}.$$

$\mathbf{P}_{\mathcal{X}}(s,a,C)$ is the conditional probability for non-silent state s to reach block C via action a under the condition that being in state s the system does not make a silent move inside the block $[s]_{\mathcal{X}}$ (i.e. it performs a visible action or makes a τ-move to another block). With this conditional probability, we get the following alternative characterisation of branching bisimulations that refers to the probabilities $\mathbf{P}_{\mathcal{X}}(\cdot)$ rather than the values $Prob_{R_{\mathcal{X}}}(\cdot)$.

Lemma 4.3 *$\mathcal{X}$ is a branching bisimulation iff, for all $B \in \mathcal{X}$ with $B \cap S_{ns}^{\mathcal{X}} \neq \emptyset$:*

(1) $\mathbf{P}_{\mathcal{X}}(s,a,C) = \mathbf{P}_{\mathcal{X}}(s',a,C)$
for all s, $s' \in B \cap S_{ns}^{\mathcal{X}}$ and $(a,C) \in Act \times \mathcal{X}$ with $(a,C) \neq (\tau,B)$.
(2) *If $s_0 \in B \setminus S_{ns}^{\mathcal{X}}$ then there exists a finite path σ with*
- *$first(\sigma) = s_0$,*
- *$\sigma(i) \in B \setminus S_{ns}^{\mathcal{X}}$, $i = 0,1,\ldots,|\sigma|-1$,*
- *$last(\sigma) \in B \cap S_{ns}^{\mathcal{X}}$.*

Proof: see Appendix A.1, Lemma A.10. ■

This lemma says that a block B (containing non-silent states) of a branching bisimulation consists of (1) all non-silent states exhibiting the same conditional values $\mathbf{P}_{\mathcal{X}}(\cdot,a,C)$, together with (2) those silent states that can silently evolve only to non-silent states of the same block.

Example 4.4 *Consider the system of Figure 5 and the partition $\mathcal{X} = \{B_1,B_2,B_3\}$ where $B_1 = \{s_0,s,s'\}$, $B_2 = \{t,t'\}$ and $B_3 = \{w,w',v,v'\}$. We show that all blocks of $\mathcal{X}$ satisfy the conditions of Lemma 4.3. We have $S_{term} = \{w,w',v\}$, $\mathbf{P}(s_0,\tau,[s_0]_{\mathcal{X}}) = 1$ and $\mathbf{P}(v',\tau,[v']_{\mathcal{X}}) = 1$. Thus, $S_{ns}^{\mathcal{X}} = \{s,s',t,t'\}$.*

- *For the block B_1, we first consider the states s and s'. We have:*

$$\mathbf{P}_{\mathcal{X}}(s,\tau,B_2) = \mathbf{P}_{\mathcal{X}}(s',\tau,B_2) = \frac{1}{2}, \quad \mathbf{P}_{\mathcal{X}}(s,\alpha,B_3) = \mathbf{P}_{\mathcal{X}}(s',\alpha,B_3) = \frac{1}{2}$$

and $\mathbf{P}_{\mathcal{X}}(s,a,C) = \mathbf{P}_{\mathcal{X}}(s',a,C) = 0$ for all $(a,C) \notin \{(\tau,B_2),(\alpha,B_3)\}$. Hence, B_1 satisfies condition (1). Second we show condition (2) for B_1. For this, we have to consider the state $s_0 \in B_1 \setminus S_{ns}^{\mathcal{X}}$. The finite path leading from s_0 to a state of $B_1 \cap S_{ns}^{\mathcal{X}}$ is given by $s_0 \xrightarrow{\tau} s$.

- *For the block* $B_2 = \{t, t'\}$ *we get:* $\mathbf{P}_{\mathcal{X}}(t, \beta, B_3) = \mathbf{P}_{\mathcal{X}}(t', \beta, B_3) = 1$ *and* $\mathbf{P}_{\mathcal{X}}(t, a, C) = \mathbf{P}_{\mathcal{X}}(t', a, C) = 0$ *for all* $(a, C) \neq (\beta, B_3)$*. Hence,* B_2 *satisfies (1). As* $B_2 \cap S^{\mathcal{X}}_{ns} = \emptyset$*,* B_2 *fulfills condition (2).*
- *As* $B_3 \cap S^{\mathcal{X}}_{ns} = \emptyset$ *for the block* B_3 *there is nothing to show.*

Lemma 4.3 yields that $\mathcal{X}$ *is a branching bisimulation.*

The lemma does not impose constraints on those classes of a branching bisimulation that consist of silent states only. So, let $\mathcal{X}$ be a branching bisimulation and $B \in \mathcal{X}$ such that $B \cap S^{\mathcal{X}}_{ns} = \emptyset$. By the definition of silence, $s \in B$ implies either that s is terminal or $\mathbf{P}(s, \tau, B) = 1$. In either case, $Prob_{R_{\mathcal{X}}}(s, \tau^* \alpha, C) = 0$ and $Prob_{R_{\mathcal{X}}}(s, \tau^*, C) = 0$ if $C \in \mathcal{X}$ is distinct from B. Having this in mind, we can show how the $Prob_{R_{\mathcal{X}}}(s, \tau^* \widehat{a}, C)$ can be derived from the conditional probabilities $\mathbf{P}_{\mathcal{X}}(t, a, C)$ if $\mathcal{X}$ is a branching bisimulation.

Lemma 4.5 *If* $\mathcal{X}$ *is a branching bisimulation then for all* $B \in \mathcal{X}$*,* $s \in B$ *implies*

- $Prob_{R_{\mathcal{X}}}(s, \tau^*, C) = \mathbf{P}_{\mathcal{X}}(B, \tau, C)$ *for all* $C \in \mathcal{X}$*,*
- $Prob_{R_{\mathcal{X}}}(s, \tau^* \alpha, C) = \mathbf{P}_{\mathcal{X}}(B, \alpha, C)$ *for all* $\alpha \in Act \setminus \{\tau\}$ *and* $C \in \mathcal{X}$*, where*

$$\mathbf{P}_{\mathcal{X}}(B, a, C) \;=\; \begin{cases} \mathbf{P}_{\mathcal{X}}(t, a, C) & : \textit{if } (a, C) \neq (\tau, B),\ B \cap S^{\mathcal{X}}_{ns} \neq \emptyset, \textit{ and } t \in B \cap S^{\mathcal{X}}_{ns} \\ 0 & : \textit{if } (a, C) \neq (\tau, B) \textit{ and } B \cap S^{\mathcal{X}}_{ns} = \emptyset \\ 1 & : \textit{if } (a, C) = (\tau, B). \end{cases}$$

Proof: see Appendix A.1, Lemma A.11. ■

Since by Lemma 4.3 each non-silent state s in class B exhibits the same

$$\mathbf{P}_{\mathcal{X}}(s, a, C) = \frac{\mathbf{P}(s, a, C)}{1 - \mathbf{P}(s, a, C)}$$

value, the definition of $\mathbf{P}_{\mathcal{X}}(B, a, C)$ is well-defined. It seems worth to point out that the above fraction may not return the same values if $(a, C) = (\tau, B)$. In other words, if $\mathcal{X}$ is a branching bisimulation, $B \in \mathcal{X}$ and s, $s' \in B \cap S^{\mathcal{X}}_{ns}$ then

$$\frac{\mathbf{P}(s, \tau, B)}{1 - \mathbf{P}(s, \tau, B)} \;\neq\; \frac{\mathbf{P}(s', \tau, B)}{1 - \mathbf{P}(s', \tau, B)}$$

is possible. For instance, for the states s and s' in Example 4.4 (Figure 5) we have $s \approx_{br} s'$ but $\mathbf{P}(s, \tau, B_1)/(1 - \mathbf{P}(s, \tau, B_1)) = 0$ while $\mathbf{P}(s', \tau, B_1)/(1 - \mathbf{P}(s', \tau, B_1)) = 1/2$ (where $B_1 = \{s_0, s, s'\}$ is the branching bisimulation equivalence class of s and s').

As indicated above, the algorithm to compute branching (and hence weak) bisimulation equivalence is based on the values of $\mathbf{P}_{\mathcal{X}}$ to refine partitions by

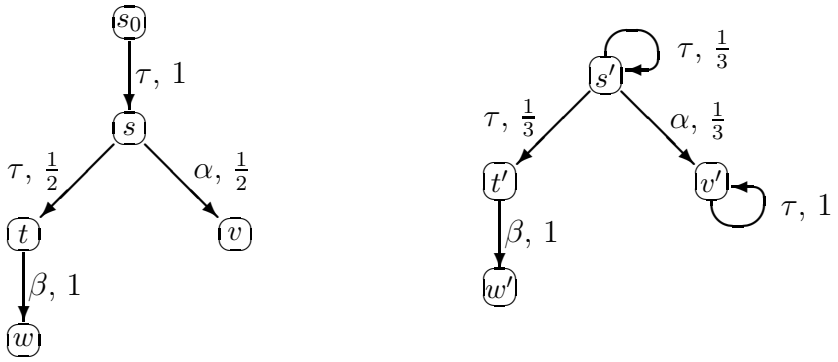


Fig. 5. Branching bisimulation equivalent fully probabilistic systems

means of splitters:

Definition 4.6 *A* splitter *of a partition $\mathcal{X}$ is a tuple (a, C) consisting of an action $a \in Act$ and some $C \in \mathcal{X}$ such that there exists some $B \in \mathcal{X}$ with $(\tau, B) \neq (a, C)$ and $\mathbf{P}_{\mathcal{X}}(s, a, C) \neq \mathbf{P}_{\mathcal{X}}(s', a, C)$ for some states s, $s' \in B \cap S^{\mathcal{X}}_{ns}$.*

The main idea for refining a given partition $\mathcal{X}$ via a splitter (a, C) is to isolate in each $B \in \mathcal{X}$ with $(\tau, B) \neq (a, C)$ those states *non-silent* states s, $s' \in B$ where $\mathbf{P}_{\mathcal{X}}(s, a, C) = \mathbf{P}_{\mathcal{X}}(s', a, C)$, in order to ensure condition (1) of Lemma 4.3. By condition (2) of the lemma, each such equivalence class A of $B \cap S^{\mathcal{X}}_{ns}$ has to be enriched with exactly those *silent states* states of B that can reach A via internal actions and that cannot reach any other equivalence class A' of $B \cap S^{\mathcal{X}}_{ns}$ without passing A. So, we define the splitting of a block B by means of a splitter (a, C) for the non-silent members of B first. This is performed by means of a function $\mathit{NonSilentSplit}(B, a, C)$, that only groups non-silent states. Silent states are treated afterwards, by means of a *silence closure* operation. In short, if A is a class of equivalent non-silent states in B then $\mathit{SilenceClose}(A, B)$ enriches A by all silent states of B for which all finite paths of internal steps that end up in a non-silent state actually end up in A.

Definition 4.7 *Let (a, C) be a splitter of a partition $\mathcal{X}$ and $B \in \mathcal{X}$ such that $(\tau, B) \neq (a, C)$. We define*

$$\mathit{NonSilentSplit}(B, a, C) \;=\; (B \cap S^{\mathcal{X}}_{ns}) / \equiv_{\mathcal{X}}$$

where, for s, $s' \in B \cap S^{\mathcal{X}}_{ns}$, $s \equiv_{\mathcal{X}} s'$ iff $\mathbf{P}_{\mathcal{X}}(s, a, C) = \mathbf{P}_{\mathcal{X}}(s', a, C)$.

Furthermore, if $A \in \mathit{NonSilentSplit}(B, a, C)$ then we define the silence closure *$\mathit{SilenceClose}(A, B)$ of A to be the largest subset $\overline{A}$ of B which contains A and such that for all $s \in \overline{A} \setminus A$:*

- $\mathbf{P}(s, \tau, \overline{A}) = 1$
- *There exists a finite path σ with*
  - $\mathit{first}(\sigma) = s$,
  - $\sigma(i) \in \overline{A} \setminus A$, $i = 0, 1, \ldots, |\sigma| - 1$,
  - $\mathit{last}(\sigma) \in A$.

The residuum *of B with respect to (a, C) is given by*

$$Res(B, a, C) = \{B \setminus B'\} \setminus \{\emptyset\} \quad \textit{where} \quad B' = \bigcup_{A \in NonSilentSplit(B,a,C)} SilenceClose(A, B).$$

The residuum contains a singleton set of all silent states that are not covered by any silence closure $SilenceClose(A, B)$ if there are such states. Otherwise, the residuum is an empty set. The states contained in this singleton set possess non-zero probability of silently evolving to distinct, non-equivalent states inside B, Therefore, they do not belong to a particular silence closure, and hence have to be separated. The refinement operator *Refine* of a partition by means of a splitter is now easy to define as a collection of the sets we have defined above.

Definition 4.8 *Let $\mathcal{X}$ be a partition, (a, C) a splitter of $\mathcal{X}$. For $B \in \mathcal{X}$, we define:*

- *If $(a, C) = (\tau, B)$ then $Refine(B, a, C) = \{B\}$.*
- *If $(a, C) \neq (\tau, B)$ then*

$$Refine(B, a, C) = \{\ SilenceClose(A, B)\ :\ A \in NonSilentSplit(B, a, C)\ \} \cup\ Res(B, a, C).$$

We define $Refine(\mathcal{X}, a, C) = \bigcup_{B \in \mathcal{X}} Refine(B, a, C)$.

Clearly, for each partition $\mathcal{X}$ which is coarser than $S/\approx_{br}$ and each splitter (a, C) of $\mathcal{X}$, the partition $Refine(\mathcal{X}, a, C)$ is coarser than $S/\approx_{br}$ and strictly finer than $\mathcal{X}$. Our refinement operator preserves condition (2) of Lemma 4.3. More precisely, if $B \in \mathcal{X}$ such that $B \cap S^{\mathcal{X}}_{ns} = \emptyset$ and condition (2) of Lemma 4.3 is fulfilled then all blocks $A \in Refine(B, a, C)$ fulfill condition (2). Moreover, if $\mathcal{X}$ is a partition that is coarser than $S/\approx_{br}$, and fulfills condition (2) of Lemma 4.3, and there does not exist a splitter for $\mathcal{X}$ then $\mathcal{X} = S/\approx_{br}$. Hence, $\mathcal{X}$ contains the weak bisimulation equivalence classes (by Theorem 3.9). These observations lead to the following algorithm. We start with a 'simple' partition $\mathcal{X}$ that satisfies condition (2) of Lemma 4.3 and that is coarser than $\approx$. Then we apply the refinement operator to $\mathcal{X}$, as long as $\mathcal{X}$ can be refined, i.e. as long as there exists a splitter for $\mathcal{X}$, eventually resulting in the partition $\mathcal{X} = S/\approx$.

Since the initial partition has to fulfill condition (2) of Lemma 4.3 the initialisation of the algorithm requires care. We cannot start with the 'trivial' partition $\mathcal{X} = \{S\}$ (that identifies all states) as it might actually violate condition (2). For instance, for a system with two states, a terminal state t and a state s with $\mathbf{P}(s, \alpha, t) = 1$, the trivial partition $\mathcal{X}_{trivial} = \{\{s, t\}\}$ does not have a splitter. Hence, if we would start with $\mathcal{X}_{trivial}$ then our algorithm would

Computing the weak bisimulation equivalence classes

Input: a finite fully probabilistic system $(S, Act, \mathbf{P})$

Output: the set $S/\approx$ of weak bisimulation equivalence classes

Method:

Compute the set S_{div} of divergent states;

$\mathcal{X} := \{S_{div}, S \setminus S_{div}\} \setminus \{\emptyset\}$;

While $\mathcal{X}$ contains a splitter (a, C) do $\mathcal{X} := \mathit{Refine}(\mathcal{X}, a, C)$;

Return $\mathcal{X}$.

Fig. 6. Algorithm computing the weak bisimulation equivalence classes

return that s and t are weakly bisimulation equivalent which is not the case. To avoid this phenomenon, our initial partition consists of two blocks: the weak bisimulation equivalence class of the terminal states and its complement. (Of course, if one of these blocks is empty then we only start with one block.) To be precise, the weak bisimulation equivalence class of the terminal states consists of all silent states that cannot reach a state where a visible action can be performed with some non-zero probability. We refer to those states as *divergent* states.

Definition 4.9 *A state s is called* divergent *iff $Path_{ful}(s, \tau^*\alpha) = \emptyset$ for all $\alpha \in Act \setminus \{\tau\}$. Let S_{div} be the set of divergent states.*

Note that $S_{term} \subseteq S_{div} \subseteq S \setminus S_{ns}^{\{S\}}$. Our algorithm for computing weak bisimulation equivalence is sketched in Figure 6.

Example 4.10 *For the system described in Example 4.4 (Figure 5) the algorithm proceeds as follows. Since $S_{div} = \{w, w', v, v'\}$, the while-loop is initialised with the partition $\mathcal{X} = \{\{s_0, s, s', t, t'\}, S_{div}\}$. In the first iteration, the non-silent states are given by $S_{ns}^{\mathcal{X}} = \{s, s', t, t'\}$. The pairs (α, S_{div}) and $(\beta, \{s_0, s, s', t, t'\})$ are splitters of $\mathcal{X}$. For the splitter (α, S_{div}) we obtain*

$$\mathbf{P}_{\mathcal{X}}(s, \alpha, S_{div}) \;=\; \frac{1}{2} \;=\; \frac{\frac{1}{3}}{1 - \frac{1}{3}} \;=\; \mathbf{P}_{\mathcal{X}}(s', \alpha, S_{div}),$$

and $\mathbf{P}_{\mathcal{X}}(t, \alpha, S_{div}) = 0 = \mathbf{P}_{\mathcal{X}}(t', \alpha, S_{div})$. Hence, the split operator of non-silent states separates s and s' from t and t'. More precisely, we get:

$$NonSilentSplit(\{s_0, s, s', t, t'\}, \alpha, S_{div}) = \{\{s, s'\}, \{t, t'\}\},$$
$$NonSilentSplit(S_{div}, \alpha, S_{div}) = \emptyset.$$

The silence closure operator yields

$$SilenceClose(\{s, s'\}, \{s_0, s, s', t, t'\}) = \{s_0, s, s'\},$$
$$SilenceClose(\{t, t'\}, \{s_0, s, s', t, t'\}) = \{t, t'\},$$
$$SilenceClose(\emptyset, S_{div}) = \emptyset,$$

and $Res(S_{div}, \alpha, S_{div}) = S_{div}$. *Hence, we get the partition*

$$\mathcal{X} := Refine(\mathcal{X}, \alpha, S_{div}) = \{\{s_0, s, s'\}, \{t, t'\}, \{w, w', v, v'\}\}$$

as a result of the first iteration of the loop. No further splitter exists for this partition. So, our algorithm returns $\mathcal{X}$ *containing the weak bisimulation equivalence classes.*

4.2 Complexity of the algorithm

In order to establish the time and space complexity of the algorithm, we let $n = |S|$. We suppose that the alphabet Act is fixed, i.e. we treat the size $|Act|$ as a constant.

Theorem 4.11 *The algorithm of Figure 6 can be implemented in time* $\mathcal{O}(n^3)$ *and space* $\mathcal{O}(n^2)$.

Proof: The initialisation of the algorithm has quadratic time (and space) complexity, since S_{div} can be computed by a reachability analysis in the underlying directed graph. We compute all states that can reach a state of $\{t \in S : \mathbf{P}(t, \alpha) > 0 \text{ for some } \alpha \in Act \setminus \{\tau\}\}$, e.g. by a standard depth first search. Then, the computation of the initial partition $\mathcal{X}$ needs $\mathcal{O}(n^2)$ time and space. To establish a complexity result for the complete algorithm requires more details concerning the implementation of a refinement step. Each refinement step consists of two phases, a preparatory phase and a splitting phase, where refinement, i.e. splitting of blocks, actually takes place.

Preparatory phase: Let $\mathcal{X}$ be the current partition. We compute the values $\mathbf{P}(s, a, C)$ and $\mathbf{P}_{\mathcal{X}}(s, a, C)$ for each $s \in S$, $a \in Act$ and $C \in \mathcal{X}$. The set $S^{\mathcal{X}}_{ns}$ can be derived from the probabilities $\mathbf{P}_{\mathcal{X}}(s, \tau, C)$, $s \in C$. For each pair (a, C) (where $a \in Act$ and $C \in \mathcal{X}$) and $A \in \mathcal{X}$ we compute

$$min(A, a, C) = \min_{s \in A} \mathbf{P}_{\mathcal{X}}(s, a, C), \quad max(A, a, C) = \max_{s \in A} \mathbf{P}_{\mathcal{X}}(s, a, C).$$

Then, (a, C) is a splitter of $\mathcal{X}$ iff $min(A, a, C) < max(A, a, C)$ for some A with $(a, C) \neq (\tau, A)$. If there is no splitter of $\mathcal{X}$ then $\mathcal{X} = S/\approx$. Otherwise we choose some splitter (a, C) of $\mathcal{X}$.

Splitting phase: For all $B \in \mathcal{X}$ with $(\tau, B) \neq (a, C)$ we compute the set $\mathit{Refine}(B, a, C)$ as follows. We construct an ordered binary tree $\mathit{Tree}(B)$ by successively inserting the values $\mathbf{P}_{\mathcal{X}}(s, a, C)$, $s \in B \cap S^{\mathcal{X}}_{ns}$. Each node v of $\mathit{Tree}(B)$ is represented as a record with components $v.\mathit{key}$ and $v.\mathit{states}$. For each state $s \in B \cap S^{\mathcal{X}}_{ns}$, we traverse the tree $\mathit{Tree}(B)$ starting in the root and we search for the value $\mathbf{P}_{\mathcal{X}}(s, a, C)$.

- If we reach a node v with $v.\mathit{key} = \mathbf{P}_{\mathcal{X}}(s, a, C)$ then we insert s into $v.\mathit{states}$.
- Otherwise, $\mathbf{P}_{\mathcal{X}}(s, a, C)$ is not yet represented in $\mathit{Tree}(B)$ and we insert a node v with $v.\mathit{key} = \mathbf{P}_{\mathcal{X}}(s, a, C)$ and $v.\mathit{states} = \{s\}$.

In the final tree, $v.\mathit{states}$ is the set of states $s \in B \cap S^{\mathcal{X}}_{ns}$ with $\mathbf{P}_{\mathcal{X}}(s, a, C) = v.\mathit{key}$. Thus, the nodes of the final tree $\mathit{Tree}(B)$ represent the sets $A \in \mathit{NonSilentSplit}(B, a, C)$. More precisely,

$$\mathit{NonSilentSplit}(B, a, C) \;=\; \{v.\mathit{states} : v \text{ is a node in } \mathit{Tree}(B)\}.$$

We derive $\mathit{Refine}(B, a, C)$ by means of a topological sorting of silent states in B. Let G_B be the directed graph (B, E_B) where $(s, t) \in E_B$ iff $\mathbf{P}(t, \tau, s) > 0$ and $t \in B \setminus S^{\mathcal{X}}_{ns}$ (note that edges are 'reversed' in this graph). We compute the sets $\mathit{SilenceClose}(A, B)$, $A \in \mathit{NonSilentSplit}(B, a, C)$, by the following breadth first search like method. We use three kinds of labels for the states:

- $\mathit{label}(s) = \bot$ iff $s \in B \setminus S^{\mathcal{X}}_{ns}$ and s is not yet visited.
- $\mathit{label}(s) = A \in \mathit{NonSilentSplit}(B, a, C)$ iff s is reachable in G_B from some state in A but there is no other $A' \in \mathit{NonSilentSplit}(B, a, C)$ where a path from a state of A' to s in G_B is already detected.
- $\mathit{label}(s) = *$ iff there are two sets A, $A' \in \mathit{NonSilentSplit}(B, a, C)$ such that s is reachable from a state in A and from a state in A'. (In particular, all successors of a $*$-labelled state in G_B are also labelled by $*$.)

Initially, we define $\mathit{label}(s) = \bot$ for all $s \in B \setminus S^{\mathcal{X}}_{ns}$ and $\mathit{label}(s) = A$ for all $s \in A$ and $A \in \mathit{NonSilentSplit}(B, a, C)$. (The implementation might use the respective $v.\mathit{key}$ values computed before as identifiers (labels) for different A).

We use a queue Q which initially contains the states $s \in A$, $A \in \mathit{NonSilentSplit}(B, a, C)$. While Q is not empty we take the first element s of Q, remove s from Q and, if $\mathit{label}(s) \neq *$ then, for all $t \in B \setminus S^{\mathcal{X}}_{ns}$ with $(s, t) \in E_B$, we do:

(1) If $\mathit{label}(t) = \bot$ then we add t to Q and set $\mathit{label}(t) = \mathit{label}(s)$.
(2) If $\mathit{label}(t) \in \mathit{NonSilentSplit}(B, a, C)$, $\mathit{label}(t) \neq \mathit{label}(s)$, then we set

$label(u) = *$ for $u = t$ and all successors u of t in G_B.[1]

Then, $SilenceClose(A, B) = \{s \in B : label(s) = A\}$, $Res(B, a, C) = \{\{s \in B : label(s) \in \{\bot, *\}\}\} \setminus \{\emptyset\}$.

Complexity: It is clear that the method described above can be implemented in space $\mathcal{O}(n^2)$. We show that the time complexity of our method is $\mathcal{O}(n^3)$. First, we observe that there are at most n iterations of the refinement step. Thus, it suffices to show that each refinement step takes time $\mathcal{O}(n^2)$.

First we observe that for each iteration (i.e. each refinement step), the preparatory phase requires $\mathcal{O}(n^2)$ time,[2] in order to compute the values $\mathbf{P}(s, a, C)$, and to decide whether a splitter exist. If so, the splitting phase is entered, where the trees $Tree(B)$ are subsequently constructed. Ranging over all B, the construction of the trees, and hence the computation of the sets $NonSilentSplit(B, a, C)$ takes $\mathcal{O}(n \log n)$ time if one resorts to some kind of ordered balanced trees. We show that, ranging over all $B \in \mathcal{X}$, the sets $SilenceClose(\cdot, B)$ and $Res(B, a, C)$ can be derived in time $\mathcal{O}(n^2)$: For fixed $B \in \mathcal{X}$, the directed graph G_B can be constructed in time $\mathcal{O}(|B|^2)$. Each state $s \in B$ is added to Q at most once. (Note that only states with label $\bot$ can be added to Q.) Each state t which is visited by a depth first search in step (2) is labelled by $*$. Thus, it can never be visited in step (2) once again. As a consequence, each state causes time costs (at most) of order $2n$ in the computation of $Refine(B, a, C)$: as an element of Q and as a state with label $\neq *$ that is visited in step (2). Either case involves $\mathcal{O}(n)$ computations. Summing up over all $s \in B$, the computation of $Refine(B, a, C)$ has time complexity $\mathcal{O}(|B| \cdot n)$. So, we obtain $Refine(\mathcal{X}, a, C)$ in time $\mathcal{O}(n^2)$. ■

5 Lazy synchronous CCS

Process calculi such as Milner's CCS or SCCS [51–53], Hoare's CSP [39], Bergstra&Klop's ACP [10] or the ISO standard LOTOS [40] are important high-level specification means for compositional design and analysis of parallel systems. Such process calculi are usually equipped with an interleaving semantics, where (apart from synchronous calculi) the concurrent execution of independent actions is modelled by *interleaving* the actions in all possible orders. Nondeterminism is used to describe the freedom in the choice which

[1] This may be realised by a depth first search starting in t to find all successors of t. States that are already labelled by $*$ are ignored.

[2] Note that for each tuple (s, a, C) we have to calculate $\sum_{t \in C} \mathbf{P}(s, a, t)$. Hence, for fixed a and ranging over all $s \in S$, $C \in \mathcal{X}$, we get the time complexity $\mathcal{O}(n^2)$. Since we suppose Act to be fixed, the values $\mathbf{P}(s, a, C)$ can be computed in time $\mathcal{O}(n^2)$.

independent action is executed next.

During the last decade, these calculi have been extended in several ways in order to provide means to specify the probabilistic behaviour of parallel systems within the same formalism [32,50,54,37,35,3]. The essential ingredient is the incorporation of a *probabilistic choice* operator. Distinguishing fine points in this variety of approaches are (1) the interplay of probabilistic choice with nondeterministic choice, and (2) the semantics of parallel composition. [3]

In the fully probabilistic setting considered in this paper, nondeterminism is completely ruled out. So, addressing the first of the above two issues, probabilistic choice *replaces* nondeterministic choice, instead of letting both choice operators coexist. The second issue, semantics of parallel composition has received considerable attention in the fully probabilistic setting [23]. Due to the absence of nondeterminism, the model is not closed under the usual nondeterministic interleaving of all independent actions.

Instead, we introduce in the sequel a new, lazy synchronous parallel composition operator. Before we explain lazy synchronous parallel composition, we briefly review two other main strands followed in the definition of fully probabilistic parallel composition. One approach replaces nondeterministic interleaving by *probabilistically scheduled* interleaving [5,67,56,66,29]. For this purpose, the parallel composition operator is decorated with a set of parameters. These parameters are used to quantify the probability of each nondeterministic alternative that arises in a parallel composition. In this way, the fully probabilistic model remains closed under parallel composition, but an intuitive interpretation of the parameters involved is often difficult. [23] contains a classification of these approaches, and defines criteria to be fulfilled by an *appropriate* probabilistically scheduled parallel composition.

The other major approach, *synchronous parallelism*, has been proposed by [25,27]. Synchrony refers to a drastically different view on the evolution of processes. Instead of assuming that actions may occur independently and concurrently, synchrony means that all parallel components have to interact (or explicitly signal that they intend to idle) in each time step, they proceed in *lockstep* [52]. In this way one avoids to make a scheduling decision since all components must perform a step. Several approaches that have appeared in the literature [25,43,68,49,70,27,47] deal with the synchronous product $\mathcal{P}_1 \times \mathcal{P}_2$ in the style of Milner's SCCS where each step of $\mathcal{P}_1 \times \mathcal{P}_2$ is composed by exactly *one* action of $\mathcal{P}_1$ and $\mathcal{P}_2$. The probability of such a step is determined by the product of the individual probabilities of each action involved.

[3] A third, less obvious, distinguishing feature lies in the semantics of probabilistic choice, where the probabilistic decision of the process may either be conditional on the actions offered by the environment (external probabilistic choice) or totally independent of what is offered (internal probabilistic choice) [50,54].

$$\mathsf{P} \quad ::= \quad nil \;\Big|\; Z \;\Big|\; a.\mathsf{P} \;\Big|\; \bigoplus_{i\in I} [p_i]\mathsf{P}_i \;\Big|\; \mathsf{P}_1 \otimes \mathsf{P}_2 \;\Big|\; \mathsf{P} \setminus L \;\Big|\; \mathsf{P}[\ell]$$

Fig. 7. Syntax of *PLSCCS* expressions

We deviate from these approaches and introduce a form of synchrony that is less rigid. In fact, our concept of *lazy synchrony* has been used in the probabilistic setting by other authors (e.g. [24,36]) as well. In these approaches, the processes $\mathcal{P}_1$ and $\mathcal{P}_2$ have to synchronise at certain synchronisation points but perform *sequences* of independent actions in isolation between these synchronisation points. In the non-probabilistic setting, similar kinds of composition are present in *synchronous languages*, such as Esterelle [11], Lustre [31], or others [18,13,36]. In what follows, we build upon this intuition, and define a lazy synchronous probabilistic variant of CCS, denoted *PLSCCS*, where visible actions are forced to proceed in lockstep, while internal actions τ are executed independently. We show that weak (as well as strong) bisimulation is a congruence for the operators of *PLSCCS* (except for a rather standard deficiency with respect to choice).

5.1 *PLSCCS*: a lazy synchronous calculus

In *PLSCCS*, internal actions are executed independently, while visible actions are forced to proceed in lockstep. In other words, each step of the lazy product $\mathsf{P}_1 \otimes \mathsf{P}_2$ is composed by sequences of steps of P_1 and P_2, where each of them starts with an arbitrary number of internal transitions (labelled τ) and ends up with a visible action α. (Recall our convention to use greek letters for visible actions only.) Hence, the probabilities for the transitions of $\mathsf{P}_1 \otimes \mathsf{P}_2$ are given by the product of the individual probabilities where we deal with the cumulative effect of τ-transitions. Syntax and semantics of all other operators are similar to SCCS; apart from the probabilistic choice operator $\bigoplus_{i\in I} [p_i]\mathsf{P}_i$, replacing the nondeterministic choice $\sum_{i\in I} \mathsf{P}_i$ of SCCS.

Syntax of *PLSCCS*: Let *Var* be a set of process variables. *PLSCCS* expressions are given by the grammar shown in Figure 7. Here, $Z \in Var$, $a \in Act$, $L \subseteq Act$, I is a countable indexing set, p_i are real numbers $p_i \in]0,1]$ with $\sum p_i = 1$ and $\ell : Act \longrightarrow Act$ is a relabelling function with $\ell(\tau) = \tau$. PLSCCS denotes the collection of all *PLSCCS* expressions. For finite indexing set $I = \{i_1, \ldots, i_n\}$, we also write $[p_{i_1}]\mathsf{P}_{i_1} \oplus \ldots \oplus [p_{i_n}]\mathsf{P}_{i_n}$ instead of $\bigoplus_{i\in I} [p_i]\mathsf{P}_i$.

As it is common practice in process calculi, we will use *process equations* of the form $Z_j \stackrel{\text{def}}{=} \mathsf{P}_j$, $j = 1, \ldots, k$, to specify recursive behaviour. In the above equations, $Z_1, \ldots, Z_k$ are pairwise distinct process variables (taken from

Var), and $\mathsf{P}_1, \ldots, \mathsf{P}_k$ are expressions given by the above grammar. They may itself contain process variables. Technically, process equations are denoted by means of a function Δ that assign to each process variable Z an expression $\Delta(Z)$ (i.e. $\Delta : \mathit{Var} \longrightarrow \mathsf{PLSCCS}$). A *PLSCCS process* is a pair $\mathcal{P} = \langle \Delta, \mathsf{P} \rangle$ consisting of a declaration Δ and an expression P. The intended meaning of a process $\mathcal{P} = \langle \Delta, \mathsf{P} \rangle$ is that the behaviour of $\mathcal{P}$ is given by the expression P where each occurrence of a process variable Z in P is viewed as a recursive call of expression $\Delta(Z)$. If *op* is a n-ary operator symbol of the calculus and $\mathcal{P}_i = \langle \Delta, \mathsf{P}_i \rangle$, $i = 1, \ldots, n$, are processes then we write $op(\mathcal{P}_1, \ldots, \mathcal{P}_n)$ as shorthand for the process $\langle \Delta, op(\mathsf{P}_1, \ldots, \mathsf{P}_n) \rangle$. *PLSCCS* denotes the set of all *PLSCCS* processes, i.e. pairs $\mathcal{P} = \langle \Delta, \mathsf{P} \rangle$ consisting of a declaration Δ and an expression $\mathsf{P} \in \mathsf{PLSCCS}$.

Example 5.1 *The expressions*

$$\begin{aligned} \mathsf{P} &= \tau.(\ [1/2]\ \tau.\beta.\mathit{nil} \oplus [1/2]\ \alpha.\mathit{nil}\ ), \\ \mathsf{P}' &= [1/3]\ \tau.Z \oplus [1/3]\ \tau.\beta.\mathit{nil} \oplus [1/3]\ \alpha.Y, \text{ and} \\ \mathsf{P}'' &= \tau.Y. \end{aligned}$$

are elements of PLSCCS. *For a declaration* Δ' *such that* $\Delta'(Z) = \mathsf{P}'$ *and* $\Delta'(Y) = \mathsf{P}''$, $\langle \Delta', \mathsf{P}' \rangle$ *is a PLSCCS process.*

The intended meanings of inaction *nil*, prefixing $a.\mathsf{P}$, restriction $\mathsf{P} \setminus L$ and relabelling $\mathsf{P}[\ell]$ are as in the case of standard (S)CCS. The probabilistic choice operator $\bigoplus_{i \in I}\ [p_i]\mathsf{P}_i$ is internal: if P_i, $i \in I$, are pairwise distinct then, with probability p_i, $\bigoplus_{i \in I}\ [p_i]\mathsf{P}_i$ behaves as P_i, independent of what is offered by the environment.

As already mentioned, $\mathsf{P}_1 \otimes \mathsf{P}_2$ represents the lazy product of P_1 and P_2. Since $\otimes$ treats τ as a local action and only synchronises visible actions, we assume a function

$$(\mathit{Act} \setminus \{\tau\}) \times (\mathit{Act} \setminus \{\tau\}) \longrightarrow \mathit{Act}, \quad (\alpha, \beta) \mapsto \alpha \star \beta.$$

where, $\alpha \star \beta$ stands for the result of the synchronisation on the *visible* actions α and β. Note that visible actions have to synchronise, but may become invisible in doing so, because $\alpha \star \beta = \tau$ *is* possible. Each a-labelled transition of the lazy product $\mathsf{P}_1 \otimes \mathsf{P}_2$ is then composed by *sequences of steps* of the components P_1 and P_2 that are labelled by strings of the form $\tau^*\alpha$ and $\tau^*\beta$ respectively such that a is the result of the synchronised execution of the visible actions α and β (i.e. $a = \alpha \star \beta$). In other words, the probability of $\mathsf{P}_1 \otimes \mathsf{P}_2$ to move via the action a to $\mathsf{P}'_1 \otimes \mathsf{P}'_2$ is cumulated from all probabilities $\mathit{Prob}^{\Delta}(\mathsf{P}_1, \tau^*\alpha, \mathsf{P}'_1) \cdot \mathit{Prob}^{\Delta}(\mathsf{P}_2, \tau^*\beta, \mathsf{P}'_2)$ where (α, β) ranges over all pairs of visible actions satisfying that $\alpha \star \beta = a$. Here, $\mathit{Prob}^{\Delta}(\mathsf{P}, \tau^*\alpha, \mathsf{P}')$ is the probability for P to perform a sequence of internal actions followed by α ending

$$\mathbf{P}^{\Delta}(a.\mathsf{P}, a, \mathsf{P}) = 1$$

$$\mathbf{P}^{\Delta}\Big(\bigoplus_{i\in I}[p_i]\mathsf{P}_i, a, \mathsf{P}'\Big) = \sum_{i\in I} p_i \cdot \mathbf{P}^{\Delta}(\mathsf{P}_i, a, \mathsf{P}')$$

$$\mathbf{W}^{\Delta}(\mathsf{P}, \alpha, \mathsf{P}') \;=\; \mathbf{P}^{\Delta}(\mathsf{P}, \alpha, \mathsf{P}') + \sum_{\mathsf{P}''\in\mathsf{PLSCCS}} \mathbf{P}^{\Delta}(\mathsf{P}, \tau, \mathsf{P}'') \cdot \mathbf{W}^{\Delta}(\mathsf{P}'', \alpha, \mathsf{P}')$$

$$\mathbf{P}^{\Delta}(\mathsf{P}_1 \otimes \mathsf{P}_2, a, \mathsf{P}'_1 \otimes \mathsf{P}'_2) = \sum_{\beta\star\gamma = a} \mathbf{W}^{\Delta}(\mathsf{P}_1, \beta, \mathsf{P}'_1) \cdot \mathbf{W}^{\Delta}(\mathsf{P}_2, \gamma, \mathsf{P}'_2)$$

$$\mathbf{P}^{\Delta}(\mathsf{P} \setminus L, a, \mathsf{P}' \setminus L) = \mathbf{P}^{\Delta}(\mathsf{P}, a, \mathsf{P}') \;\text{ if } a \notin L$$

$$\mathbf{P}^{\Delta}(\mathsf{P}[\ell], a, \mathsf{P}'[\ell]) = \sum_{b\in\ell^{-1}(a)} \mathbf{P}^{\Delta}(\mathsf{P}, b, \mathsf{P}')$$

$$\mathbf{P}^{\Delta}(Z, a, \mathsf{P}') = \mathbf{P}^{\Delta}(\Delta(Z), a, \mathsf{P}')$$

Fig. 8. Equations for $\mathbf{P}^{\Delta}$ and $\mathbf{W}^{\Delta}$ for the semantics of *PLSCCS*.

up in the state P', with $Prob^{\Delta}$ denoting the probability measure obtained by declaration Δ.

Operational semantics for *PLSCCS*: We supply *PLSCCS* with an operational semantics with the help of a higher-order operator on function pairs the form $\langle \mathbf{P}^{\Delta}, \mathbf{W}^{\Delta}\rangle$. The first function, $\mathbf{P}^{\Delta}$, in these pairs is an element of the function space $\mathsf{PLSCCS} \times Act \times \mathsf{PLSCCS} \longrightarrow [0,1]$, while the second one, $\mathbf{W}^{\Delta}$, is from $\mathsf{PLSCCS} \times (Act \setminus \{\tau\}) \times \mathsf{PLSCCS} \longrightarrow [0,1]$. In the definition of the lazy product, $\mathbf{W}^{\Delta}$ is used to cumulate the probability of moving with a trace $\tau^*\alpha$ from P to P', while $\mathbf{P}^{\Delta}$ represents the actual one-step transition probabilities. Formally,

Definition 5.2 *For a fixed declaration $\Delta : Var \longrightarrow \mathsf{PLSCCS}$, the semantics of* PLSCCS *is defined as the fully probabilistic system* $(\mathsf{PLSCCS}, Act, \mathbf{P}^{\Delta})$ *where the transition probability function*

$$\mathbf{P}^{\Delta} : \mathsf{PLSCCS} \times Act \times \mathsf{PLSCCS} \longrightarrow [0,1]$$

is given by the least function pair $\langle \mathbf{P}^{\Delta}, \mathbf{W}^{\Delta}\rangle$ satisfying the equations of Figure 8.

The existence of a least function pair satisfying these equations can be derived with standard methods of domain theory, see e.g. [1].

Example 5.3 *For the expressions* P, P', *and* P'' *of Example 5.1, and declaration Δ' of Example 5.1, the operational semantics of $\mathcal{P} = \langle \Delta', \mathsf{P}\rangle$ and $\mathcal{P}' = \langle \Delta', \mathsf{P}'\rangle$ are depicted in Figure 9 (represented as trees, and where some intermediate states are not labelled.). The reader is invited to compare this*

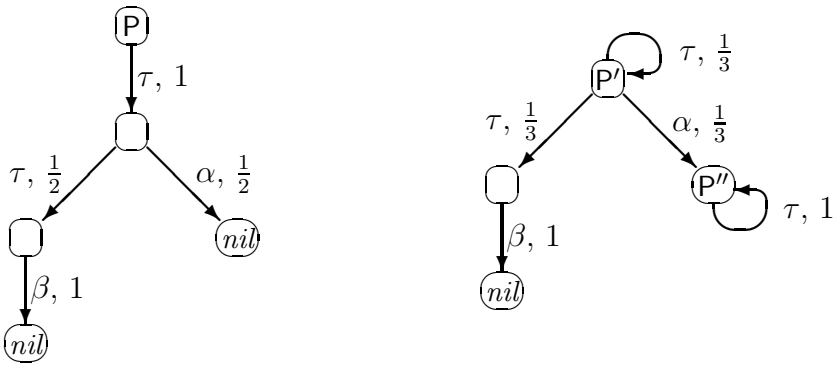


Fig. 9. Operational semantics of $\mathcal{P} = \langle \Delta', \mathsf{P} \rangle$ and $\mathcal{P}' = \langle \Delta', \mathsf{P}' \rangle$.

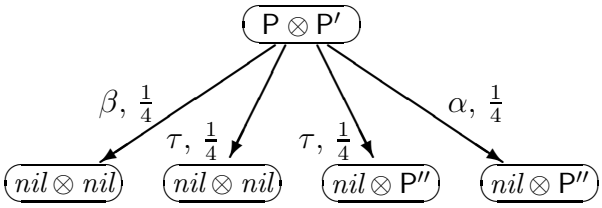


Fig. 10. Operational semantics of $\mathcal{P} \otimes \mathcal{P}'$.

figure with Figure 5.

To illustrate the lazy product, Figure 10 contains the operational semantics of $\mathcal{P} \otimes \mathcal{P}'$, where we assume a synchronisation function $\star$, such that $\alpha \star \alpha = \alpha$, $\beta \star \beta = \beta$, and $\alpha \star \beta = \beta \star \alpha = \tau$. For instance, the operational semantics defines $\mathbf{P}^{\Delta}(\mathsf{P} \oplus \mathsf{P}', \tau, nil \oplus \mathsf{P}'') = 1/4$ as a result of

$$\begin{aligned}\mathbf{P}^{\Delta}(\mathsf{P} \oplus \mathsf{P}', \tau, nil \oplus \mathsf{P}'') &= \sum_{\beta \star \gamma = \tau} \mathbf{W}^{\Delta}(\mathsf{P}, \beta, nil) \cdot \mathbf{W}^{\Delta}(\mathsf{P}', \gamma, \mathsf{P}'') \\ &= \mathbf{W}^{\Delta}(\mathsf{P}, \beta, nil) \cdot \mathbf{W}^{\Delta}(\mathsf{P}', \alpha, \mathsf{P}'') \; + \\ &\quad \mathbf{W}^{\Delta}(\mathsf{P}, \alpha, nil) \cdot \mathbf{W}^{\Delta}(\mathsf{P}', \beta, \mathsf{P}'').\end{aligned}$$

Obviously, $\mathbf{W}^{\Delta}(\mathsf{P}', \beta, \mathsf{P}'') = 0$ (since action β does not occur on any path from P' to P''), and hence only first summand might contribute non-zero probability. We calculate

$$\begin{aligned}\mathbf{W}^{\Delta}(\mathsf{P}', \alpha, \mathsf{P}'') &= \mathbf{P}^{\Delta}(\mathsf{P}', \alpha, \mathsf{P}'') + \sum_{\mathsf{P}''' \in \mathsf{PLSCCS}} \mathbf{P}^{\Delta}(\mathsf{P}, \tau, \mathsf{P}''') \cdot \mathbf{W}^{\Delta}(\mathsf{P}''', \alpha, \mathsf{P}'') \\ &= \frac{1}{3} + \mathbf{P}^{\Delta}(\mathsf{P}, \tau, \mathsf{P}) \cdot \mathbf{W}^{\Delta}(\mathsf{P}', \alpha, \mathsf{P}'') \\ &= \frac{1}{3} + \frac{1}{3} \cdot \mathbf{W}^{\Delta}(\mathsf{P}', \alpha, \mathsf{P}''),\end{aligned}$$

whose (least) solution is $1/2$. Furthermore, we obtain $\mathbf{W}^{\Delta}(\mathsf{P}, \beta, nil) = 1/2$,

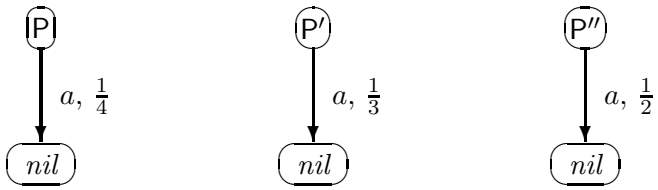


Fig. 11. Three sub-stochastic processes

and hence $\mathbf{P}^{\Delta}(\mathsf{P} \oplus \mathsf{P}', \tau, \mathit{nil} \oplus \mathsf{P}'') = 1/4$.

Our transition probability function is potentially *sub-stochastic* in the sense that $\sum_{a,t}\ \mathbf{P}^{\Delta}(s,a,t)$ might be strictly between 0 and 1 (violating Definition 2.1). We identify any sub-stochastic probabilistic system $(S, \mathit{Act}, \mathbf{P})$ with the fully probabilistic system $(S \cup \{\dagger\}\, \mathit{Act} \cup \{\dagger\}, \mathbf{P}_{\dagger})$ obtained by identifying $\mathbf{P}_{\dagger}$ with $\mathbf{P}$ on $S \times \mathit{Act} \times S$, by setting

$$\mathbf{P}_{\dagger}(s,\dagger,\dagger) \;=\; 1 \;-\; \sum_{a \in \mathit{Act}, t \in S} \mathbf{P}(s,a,t),$$

and $\mathbf{P}_{\dagger}(\cdot) = 0$ in all remaining cases. The distinguished state † is a means to represent a deadlock in the system. So, a state s in a sub-stochastic system may possess the potential of deadlocking with a certain probability, say p. In this case, $\mathbf{P}_{\dagger}(s,\dagger,\dagger) = p$. Remark that, by definition, state † can only be reached via action †, and no other state can be reached via action †.

In the context of *PLSCCS*, sub-stochasticity can arise as a result of restriction, unguarded probabilistic choice, and recursion.

Example 5.4 *For a declaration Δ with $\Delta(Z) = Z$, Figure 11 presents three sub-stochastic probabilistic systems obtained via the operational semantics for the processes*

$$\mathsf{P} = ([1/4]\, a.\mathit{nil} \oplus [3/4]\, b.\mathit{nil}) \setminus \{a\},$$
$$\mathsf{P}' = [1/3]\, a.\mathit{nil} \oplus [2/3]\, \mathit{nil}, \text{ and}$$
$$\mathsf{P}'' = [1/2]\, a.\mathit{nil} \oplus [1/2]\, Z.$$

Accordingly, we obtain the deadlock probabilities

$$\mathbf{P}_{\dagger}^{\Delta}(\mathsf{P},\dagger,\dagger) \;=\; \frac{3}{4}, \qquad \mathbf{P}_{\dagger}^{\Delta}(\mathsf{P}',\dagger,\dagger) \;=\; \frac{2}{3}, \text{ and} \qquad \mathbf{P}_{\dagger}^{\Delta}(\mathsf{P}'',\dagger,\dagger) \;=\; \frac{1}{2}.$$

As far as we know, the use of fixed-point equations is novel when defining an operational semantics. We have chosen this style of definition, because the defining equations *as such* are easy to understand, and the domain-theoretic arguments are not much involved. Alternatively, the semantics could be given in terms of an (indexed) probabilistic transition relation using standard operational rules, as it is done, for instance, for synchronous probabilistic CCS [25].

Lemma 5.5 *For all* P, $\mathsf{P}' \in \mathsf{PLSCCS}$ *and* $\alpha \in Act$,

$$\mathbf{W}^{\Delta}(\mathsf{P}, \alpha, \mathsf{P}') \;=\; Prob^{\Delta}(\mathsf{P}, \tau^*\alpha, \mathsf{P}').$$

Proof: easy verification. Uses structural induction on the syntax of P, and the fact that $Prob^{\Delta}(\mathsf{P}, \tau^*\alpha, \mathsf{P}')$ is the least solution of the equation

$$Prob^{\Delta}(\mathsf{P}, \alpha, \mathsf{P}') \;=\; \mathbf{P}^{\Delta}(\mathsf{P}, \alpha, \mathsf{P}') + \sum_{\mathsf{P}'' \in \mathsf{PLSCCS}} \mathbf{P}^{\Delta}(\mathsf{P}, \tau, \mathsf{P}'') \cdot Prob^{\Delta}(\mathsf{P}'', \alpha, \mathsf{P}')$$

in the interval $[0, 1]$. ■

So, $\mathbf{W}^{\Delta}(\mathsf{P}, \tau^*\alpha, \mathsf{P}')$ determines the probability for the process $\langle \Delta, \mathsf{P} \rangle$ to behave as $\langle \Delta, \mathsf{P}' \rangle$ after performing a sequence of τ-steps followed by visible action α. As exemplified in Example 5.3, the calculation of this probability boils down to the solution of a linear equation system (similar to the one appearing in Section 4 to compute $Prob(s, \tau^*\widehat{a}\tau^*, C)$), as long as we are dealing with a finite process, i.e. where the operational semantics maps on a fully probabilistic process that is finite (or can be identified with a finite process).

Corollary 5.6 *For all* P_1, P_2, P'_1, $\mathsf{P}'_2 \in \mathsf{PLSCCS}$ *and* $a \in Act$,

$$\mathbf{P}^{\Delta}(\mathsf{P}_1 \otimes \mathsf{P}_2, a, \mathsf{P}'_1 \otimes \mathsf{P}'_2) \;=\; \sum_{\beta \star \gamma = a} Prob^{\Delta}(\mathsf{P}_1, \tau^*\beta, \mathsf{P}'_1) \cdot Prob^{\Delta}(\mathsf{P}_2, \tau^*\gamma, \mathsf{P}'_2).$$

Proof: Follows immediately from Lemma 5.5. ■

Comparison to synchronous probabilistic CCS: To facilitate the inspection of the semantics, we highlight the distinguishing features of *PLSCCS*, compared to the well-studied fully probabilistic variant of synchronous CCS [25,27], called *PSCCS* in the sequel.

- The internal action τ does not play any specific role in *PSCCS*. In particular, $(Act, \star)$ is supposed to form an Abelian monoid, where τ is treated as any other action.
- The product $\mathsf{P}_1 \times \mathsf{P}_2$ of *PSCCS* is strictly synchronous. In terms of a higher order operator, this semantics is obtained by replacing the equation for $\mathsf{P}_1 \otimes \mathsf{P}_2$ in Figure 8 by

  $$\mathbf{P}^{\Delta}(\mathsf{P}_1 \times \mathsf{P}_2, a, \mathsf{P}'_1 \times \mathsf{P}'_2) \;=\; \sum_{b \star c \,=\, a} \mathbf{P}^{\Delta}(\mathsf{P}_1, b, \mathsf{P}'_1) \cdot \mathbf{P}^{\Delta}(\mathsf{P}_2, c, \mathsf{P}_2).$$

  Then, the use of $\mathbf{W}^{\Delta}$ is superfluous.
- Probabilistic choice in *PLSCCS* is internal, while it is external in *PSCCS*. Since, in the (S)CCS context, the influence of the environment is made explicit by means of restriction, this implies that the semantics of *restriction*

differs between *PSCCS* and *PLSCCS*. *PSCCS* restriction (denoted $\mathsf{P}\lceil L$) involves renormalisation, and relies on the equation

$$\mathbf{P}^{\Delta}(\mathsf{P}\lceil L, a, \mathsf{P}'\lceil L) \;=\; \frac{\mathbf{P}^{\Delta}(\mathsf{P}, a, \mathsf{P}')}{\sum\limits_{b\in L, \mathsf{P}'\in \mathsf{PLSCCS}} \mathbf{P}^{\Delta}(\mathsf{P}, b, \mathsf{P}')} \;\text{ if } a \notin L.$$

In contrast, *PLSCCS* restriction $\mathsf{P} \setminus L$ may induce nonzero deadlock probabilities, in case that an action is randomly chosen that is not offered by the environment.

5.2 Compositionality

In this section, we establish the congruence result (Theorem 5.8) stating the compositionality of weak bisimulation with respect to the operators of *PLSCCS*. More precisely, we show that weak bisimulation equivalence $\approx$ is a congruence with respect to the *PLSCCS* operators action prefixing, restriction, relabelling, lazy product and *guarded* probabilistic choice. In what follows, we shrink our attention to *finite PLSCCS* processes.

Definition 5.7 *A PLSCCS process $\mathcal{P} = \langle \Delta, \mathsf{P}\rangle$ is called* finite *iff there are only finitely many expressions $\mathsf{P}' \in \mathsf{PLSCCS}$ that are reachable from P in $(\mathsf{PLSCCS}, Act, \mathbf{P}^{\Delta})$ with nonzero probability. A declaration $\Delta : Var \longrightarrow \mathsf{PLSCCS}$ is called finite iff, for each $Z \in Var$, $\langle \Delta, Z\rangle$ is finite.*

If $\langle \Delta, \mathsf{P}\rangle$ is finite then $(\mathsf{PLSCCS}, Act, \mathbf{P}^{\Delta})$ can be identified with the *finite* fully probabilistic process that arises by removing all expressions that are not reachable from the initial state P. Obviously, if Δ is finite then $\langle \Delta, \mathsf{P}\rangle$ is, for arbitrary expressions P. A sufficient condition which guarantees that Δ is finite is 'regularity' of Δ in the sense that, for all process variables Z there is no occurrence of Z' in $\Delta(Z)$ that is contained in a subexpression of the form $\mathsf{P}[\ell]$, $\mathsf{P} \setminus L$ or $\mathsf{P}_1 \otimes \mathsf{P}_2$.

To establish compositionality, we adapt weak bisimulation equivalence $\approx$ (and strong bisimulation equivalence $\sim$) to *PLSCCS* processes as follows. For fixed declaration Δ, we define the relations $\approx^{\Delta}$ by $\mathsf{P} \approx^{\Delta} \mathsf{P}'$ iff $\mathsf{P} \approx \mathsf{P}'$ holds in $(\mathsf{PLSCCS}, Act, \mathbf{P}^{\Delta})$. Similarly, we define $\mathsf{P} \sim^{\Delta} \mathsf{P}'$ iff $\mathsf{P} \sim \mathsf{P}'$ in $(\mathsf{PLSCCS}, Act, \mathbf{P}^{\Delta})$, to lift strong bisimulation equivalence $\sim$.[4]

Theorem 5.8 *Weak bisimulation equivalence is preserved by the PLSCCS*

[4] Remark that in order to compare $\langle \Delta', \mathsf{P}\rangle$ and $\langle \Delta'', \mathsf{P}'\rangle$, where both Δ' and Δ'' are finite, but distinct, we may always use α-conversion to join Δ' and Δ'' to some equipotent declaration Δ_α and expressions P'_α and P''_α such that $\langle \Delta', \mathsf{P}\rangle$ and $\langle \Delta_\alpha, \mathsf{P}'_\alpha\rangle$ (as well as $\langle \Delta'', \mathsf{P}'\rangle$ and $\langle \Delta_\alpha, \mathsf{P}''_\alpha\rangle$) coincide.

operators action prefixing, restriction, relabelling and lazy product. More precisely, if Δ is a finite declaration, then for all PLSCCS expressions s, s', P_i, P'_i:

(a) *If* $\mathsf{P} \approx^{\Delta} \mathsf{P}'$ *then* $a.\mathsf{P} \approx^{\Delta} a.\mathsf{P}'$, $\mathsf{P} \setminus L \;\approx^{\Delta}\; \mathsf{P}' \setminus L$, *and* $\mathsf{P}[\ell] \;\approx^{\Delta}\; \mathsf{P}'[\ell]$.
(b) *If* $\mathsf{P}_i \approx^{\Delta} \mathsf{P}'_i$, $i = 1, 2$, *then* $\mathsf{P}_1 \otimes \mathsf{P}_2 \;\sim^{\Delta}\; \mathsf{P}'_1 \otimes \mathsf{P}'_2$; *and thus,*

$$\mathsf{P}_1 \otimes \mathsf{P}_2 \;\approx^{\Delta}\; \mathsf{P}'_1 \otimes \mathsf{P}'_2.$$

(c) *Weak bisimulation equivalence is a congruence with respect to* guarded *probabilistic choice, i.e. if* $\mathsf{P}_i \approx^{\Delta} \mathsf{P}'_i$, $i \in I$, *then*

$$\bigoplus_{i\in I} [p_i]a_i.\mathsf{P}_i \;\approx^{\Delta}\; \bigoplus_{i\in I} [p_i]a_i.\mathsf{P}'_i.$$

Proof: Part (a) is an easy verification. We show (b) and (c). More precisely, we fix a finite declaration Δ, some finite subsets $\mathcal{S}_1$, S_2, of PLSCCS that are closed with respect to the transition relation induced by $\mathbf{P}^{\Delta}$ (i.e. if $\mathsf{P} \in S_i$ and $\mathbf{P}^{\Delta}(\mathsf{P}, a, \mathsf{P}') > 0$ then $\mathsf{P}' \in S_i$). We show that

$$R \;=\; \Big\{(\mathsf{P}_1 \otimes \mathsf{P}_2, \mathsf{P}'_1 \otimes \mathsf{P}'_2) : \mathsf{P}_i, \mathsf{P}'_i \in S_i, \mathsf{P}_i \approx^{\Delta} \mathsf{P}'_i, i = 1, 2\Big\}$$

is a *strong* bisimulation, and use that $\sim \subseteq \approx$. For subsets C_1 of S_1 and C_2 of S_2, we define

$$C_1 \otimes C_2 = \{\mathsf{P}_1 \otimes \mathsf{P}_2 : \mathsf{P}_1 \in C_1, \mathsf{P}_2 \in C_2\}.$$

Clearly, R is an equivalence relation on $S = S_1 \otimes S_2$ (the state space induced by the product of S_1 and S_2). Each equivalence class $C \in S/R$ is of the form $C = C_1 \otimes C_2$ where $C_i \in \mathsf{S}_i/\approx^{\Delta}$, $i = 1, 2$. Let $a \in Act$, $C = C_1 \otimes C_2 \in S/R$ and $(s, s') \in R$ where $s = \mathsf{P}_1 \otimes \mathsf{P}_2$, $s' = \mathsf{P}'_1 \otimes \mathsf{P}'_2$, $\mathsf{P}_i \approx^{\Delta} \mathsf{P}'_i$, $i = 1, 2$. Then, by Theorem 3.5 and Corollary 5.6 we obtain

$$\mathbf{P}^{\Delta}(s, a, C) \;=\; \sum_{(\alpha_1,\alpha_2)\in sync_a} Prob^{\Delta}(\mathsf{P}_1, \tau^*\alpha_1, C_1) \cdot Prob^{\Delta}(\mathsf{P}_2, \tau^*\alpha_2, C_2)$$

$$= \sum_{(\alpha_1,\alpha_2)\in sync_a} Prob^{\Delta}(\mathsf{P}'_1, \tau^*\alpha_1, C_1) \cdot Prob^{\Delta}(\mathsf{P}'_2, \tau^*\alpha_2, C_2) \;=\; \mathbf{P}^{\Delta}(s', a, C).$$

We conclude $\mathbf{P}^{\Delta}(\mathsf{P}, a, C) = \mathbf{P}^{\Delta}(\mathsf{P}', a, C)$ for all $a \in Act$ and $C \in S/R$. Hence, R is a strong bisimulation. In particular, whenever $(\mathsf{P}, \mathsf{P}') \in R$ then $\mathsf{P} \approx^{\Delta} \mathsf{P}'$. This yields the claim of part (b). Part (c) can be derived from Theorem 3.5 and the fact that, for $C \subseteq$ PLSCCS and $\mathsf{P} = \bigoplus_{i\in I} [p_i]a_i.\mathsf{P}_i$,

$$Prob^{\Delta}(\mathsf{P}, \tau^*a, C) \;=\; \sum_{i\in I_\tau} p_i \cdot Prob^{\Delta}(\mathsf{P}_i, \tau^*a, C) \;+\; \sum_{j\in J} p_j$$

where $I_\tau = \{i \in I : a_i = \tau\}$ and $J = \{i \in I : a_i = a, \mathsf{P}_i \in C\}$. ■

The counterexample that demonstrates that weak bisimulation is not a congruence for the probabilistic choice operator is almost the same as the counterexample in the non-probabilistic case showing that weak bisimulation fails

to be preserved by non-deterministic choice [53]. Consider the *PLSCCS* expressions

$$\mathsf{P} \;=\; [1/2]\;\mathsf{P}_1 \;\oplus\; [1/2]\;\mathsf{P}_2, \text{ and } \mathsf{P}' \;=\; [1/2]\;\mathsf{P}'_1 \;\oplus\; [1/2]\;\mathsf{P}_2.$$

where $\mathsf{P}_1 = \alpha.nil$, $\mathsf{P}'_1 = \tau.\mathsf{P}_1$, $\mathsf{P}_2 = \beta.nil$. Then, (for an arbitrary declaration Δ) $\mathsf{P}_1 \approx^{\Delta} \mathsf{P}'_1$ but $\mathsf{P} \not\approx^{\Delta} \mathsf{P}'$ as P reaches (via internal actions) the weak bisimulation equivalence class C of P_1 with probability $1/2$ while P' cannot move to a state that is weakly bisimulation equivalent to P_1. Formally, we have

$$Prob^{\Delta}(\mathsf{P}, \tau^*, C) \;=\; \frac{1}{2} \;>\; 0 \;=\; Prob^{\Delta}(\mathsf{P}', \tau^*, C)$$

where C is the weak bisimulation equivalence class of P_1 (and Δ an arbitrary declaration). This deficiency can be fixed in the standard way, by defining the coarsest congruence contained in $\approx^{\Delta}$, see [53]. Remark, however, that the distinction between P and P' vanishes (i.e. $\mathsf{P}'' \oplus \mathsf{P} \approx \mathsf{P}'' \oplus \mathsf{P}'$), if they are considered in the context of a lazy product with some arbitrary process P''.

By a simple rework of the proof in [27], together with clause (b) of the above theorem, we get that in addition to weak bisimulation equivalence, *strong* bisimulation equivalence is a congruence for *PLSCCS*, as well. On the other hand, it is worth to highlight that weak (as opposed to strong) bisimulation equivalence is *not* a congruence for the synchronous product $\times$ of *PSCCS*, because that operator does not treat τ-actions in any distinguished way.

6 Putting it all together

In this section, we draw the complete picture how all ingredients introduced thus far provide a truly compositional method for specifying and verifying probabilistic systems. We consider a variant of the simple communication protocol of Example 2.2 (Figure 1). The specification of the intended behaviour is simple, using visible actions *produce* and *consume*:

$$Spec \stackrel{\text{def}}{=} produce \;.\; consume \;.\; Spec.$$

In order to realise this behaviour we implement it as the the *lazy product* of a sender (who tries to send messages) and a receiver (who waits for the messages of the sender). The sender works with an unreliable medium that might lose message (with probability 0.01). If a message gets lost, the sender retries to deliver the message. In case where the message is delivered correctly, the sender waits for an acknowledgement of the receipt. For simplicity, we assume that the acknowledgement is transmitted by a safe medium that does not loose messages. The behaviour of the sender can be specified using process equations as explained above.

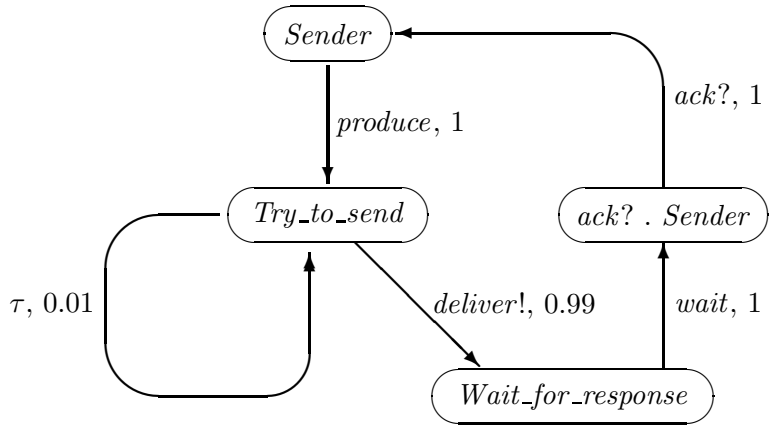


Fig. 12. The operational semantics of the sender

$$Sender \stackrel{\text{def}}{=} produce \;.\; Try\_to\_send$$
$$Try\_to\_send \stackrel{\text{def}}{=} [0.01]Lost \;\oplus\; [0.99]Deliver$$
$$Lost \stackrel{\text{def}}{=} \tau \;.\; Try\_to\_send$$
$$Deliver \stackrel{\text{def}}{=} deliver! \;.\; Wait\_for\_response$$
$$Wait\_for\_response \stackrel{\text{def}}{=} wait \;.\; ack? \;.\; Sender$$

We use the visible actions *deliver*! (the output action by which the medium transmits the message to the receiver), *ack*? (an input action that denotes that the sender reads the acknowledgement) and the action *wait* that is used to force the sender to be idle in the step where the receiver works up the message. The invisible action τ is used to describe the activities that are needed for preparing the next attempt to deliver the message. The operational semantics of the *Sender* is shown in Figure 12. The receiver is specified as follows.

$$Receiver \stackrel{\text{def}}{=} wait \;.\; Get\_message$$
$$Get\_message \stackrel{\text{def}}{=} deliver? \;.\; consume \;.\; Acknowledge$$
$$Acknowledge \stackrel{\text{def}}{=} ack! \;.\; Receiver$$

We use the actions *deliver*? (the input action that stands for the receipt of the message), *consume*, *ack*! (the output action by which the receiver acknowledges the receipt of the message) and the action *wait* (which ensures that the receiver is idle while the sender generates the next message). We suppose that $deliver! \star deliver? = ack? \star ack! = \tau$, and $wait \star \alpha = \alpha \star wait = \alpha$ for all visible α. Applying the operational semantics of *PLSCCS* to $Sender \otimes Receiver$, we obtain the fully probabilistic process $Sender \otimes Receiver$ shown in Figure 13. [5]

[5] Note that the probability for the sender to reach the state *Wait_for_response* from *Try_to_send* via a path labelled by a trace of $\tau^* deliver!$ is 1.

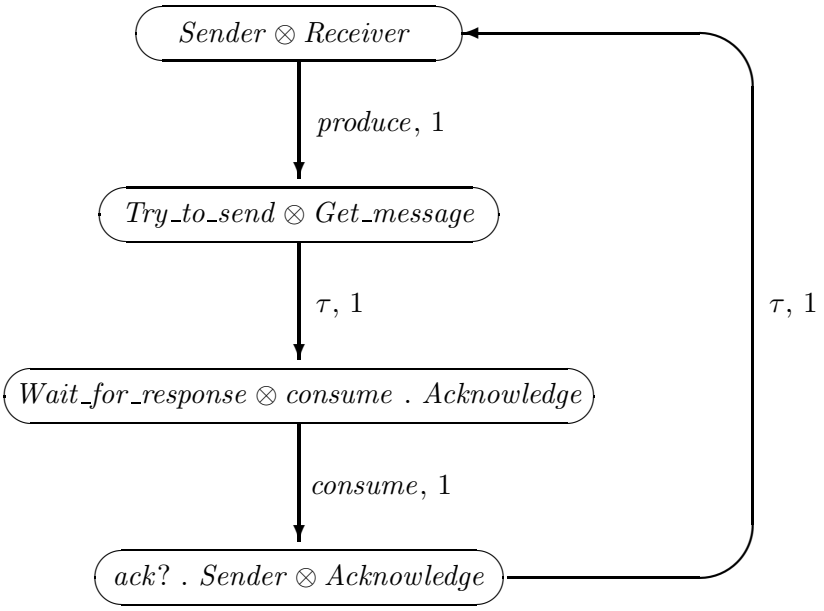


Fig. 13. The operational semantics of *Sender* ⊗ *Receiver*

Now, we may whish to verify the implementation *Sender* ⊗ *Receiver* against the specification, *Spec*, and we can do so, using the algorithm of Section 4. The algorithm returns that *Sender* ⊗ *Receiver* is indeed weakly bisimulation equivalent to the specification.

On the other hand, we may apply weak bisimulation equivalence to, say, *Sender* in isolation. The weak bisimulation algorithm computes the equivalence classes of *Sender*, which afterwards (by standard means) can be turned into a minimised representation of the senders behaviour,

$$Sender_{\min} \stackrel{\text{def}}{=} produce \;.\; deliver! \;.\; wait \;.\; ack? \;.\; Sender_{\min}$$

Now, our congruence result (part (b) of Theorem 5.8) justifies to investigate the lazy product $Sender_{\min} \otimes Receiver$ instead of $Sender \otimes Receiver$, since both are weakly bisimulation equivalent. So, in order to verify the implementation against the specification *Spec*, it is sufficient to establish

$$Sender_{\min} \otimes Receiver \;\approx\; Spec.$$

This is easily verified by means of our algorithm.

7 Conclusion

In this paper, we have developed the three main ingredients for a compositional verification technique for fully probabilistic systems. We have (1) introduced weak and branching bisimulation, equivalence notions that abstract

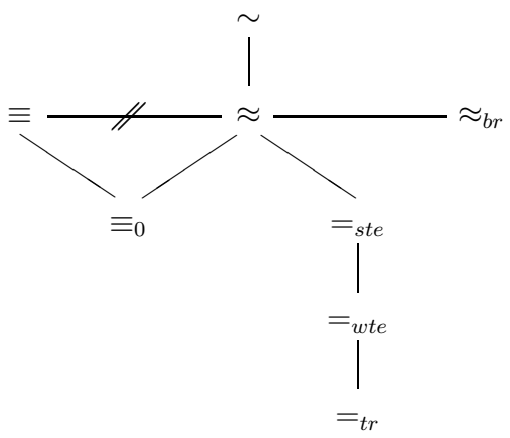


Fig. 14. Position of weak bisimulation equivalence ($\approx$) compared to other weak equivalences, cf. Section 3.3.

from internal computations. We have (2) introduced a calculus *PLSCCS* that can be used for specifying fully probabilistic systems, and that enjoys the compositionality property with respect to weak (and branching) bisimulation. In addition, we have (3) illustrated that mechanised verification (and minimisation) is feasible, by developing an algorithm to compute the weak (and branching) bisimulation equivalence classes. The algorithm has a cubic time complexity, which meets the worst case complexity of branching bisimulation equivalence in the non-probabilistic setting.

These three ingredients form the basis to alleviate the *state-space explosion* problem by generating the state space in a compositional fashion, where the algorithm is applied to minimize the intermediate state spaces of components. Furthermore, the concept of lazy synchrony in isolation already helps to alleviate the state space explosion, since the effect of internal moves is by definition cumulated with successing visible actions. We are intending to study in what sense this beneficial side result can be transferred back to the non-probabilistic setting.

Apart from its practical interest for compositional analysis of fully probabilistic systems, weak bisimulation equivalence enjoys a central position with respect to other weak relations studied in the literature. Summarising the results of Section 3.3 yields the lattice in Figure 14, where two relations are connected by a line if the upper is contained in the lower. Weak and branching bisimulation equivalence are connected by a horizontal line, since they agree. The crossed line between $\equiv$ and $\approx$ indicates that both are incomparable.

The proofs of our main results rely on the regularity of certain matrices (with columns and rows for each state of the underlying system). Thus, the main results are only established for *finite* systems. It is an open question whether our results carry over to arbitrary fully probabilistic systems (with possibly infinitely many states).

A Proofs of the main theorems

In this appendix we give the proofs of the main results of this paper. In what follows, we fix a finite fully probabilistic system $(S, Act, \mathbf{P})$. We use the following notations: If R is an equivalence relation on S and Ω a regular expression such that $Prob(s, \Omega, C) = Prob(s', \Omega, C)$ for all $(s, s') \in R$ and $C \in S/R$ then we define for all $A \in S/R$: $Prob(A, \Omega, C) = Prob(s, \Omega, C)$ where $s \in A$. We simply write $[s]$ to denote the weak bisimulation equivalence class of s (i.e. $[s] = [s]_{\approx}$).

A.1 Weak and branching bisimulation

In this section we give the proof of Theorem 3.9 stating that weak and branching bisimulation equivalence coincide, and the proof of Theorem 3.5.

Definition A.1 *Let R be a weak bisimulation. R is called* complete *iff*

- *Whenever $s \in S$, $C \in S/R$ and $Prob(s, \tau^*, C) = 1$ then $s \in C$.*
- *If $S_{div} \neq \emptyset$ then $S_{div} \in S/R$.*

Note that, if R is a complete weak bisimulation and $A \in S/R$, $A \neq S_{div}$, then $A \cap S_{div} = \emptyset$ (in particular, A does not contain terminal states) and there is a state $s \in A$ with $\mathbf{P}(s, \tau, A) < 1$. Thus, $A \cap S^{\mathcal{X}}_{ns} \neq \emptyset$ where $\mathcal{X} = S/R$ is the induced partition.(Recall that $S^{\mathcal{X}}_{ns}$ is the set of non-silent states with respect to $\mathcal{X}$.)

Lemma A.2 *$s \approx s'$ iff $(s, s') \in R$ for some complete weak bisimulation R.*

Proof: It suffices to show that each weak bisimulation is contained in some complete weak bisimulation. For R to be a weak bisimulation, we define $\mathcal{J}(R)$ to be the smallest equivalence relation on S which contains R and such that:

- If $Prob(s, \tau^*, [s']_R) = 1$ then $(s, s') \in \mathcal{J}(R)$
- If $S_{div} \neq \emptyset$ then $(s, s') \in \mathcal{J}(R)$ for all s, $s' \in S_{div}$.

Let $R_0 = R$, $R_{i+1} = \mathcal{J}(R_i)$ and $R' = \bigcup_i R_i$. It is easy to see that R' is a complete weak bisimulation. ■

Definition A.3 *For R to be a complete weak bisimulation, we define matrices $\mathbf{A}_R$ and $\mathbf{A}^0_R$ as follows: Let $A_1, \ldots, A_k$ be an enumeration of those equivalence classes $A_i \in S/R$ which contain some state $s_i \in S \setminus S_{term}$ with $\mathbf{P}(s_i, \tau, A_i) < 1$. (Since R is complete, $\{A_1, \ldots, A_k\} = S/R \setminus \{S_{div}\}$.) Let $\mathbf{A}_R$ be the following*

matrix:

$$\mathbf{A}_R \;=\; (\; Prob(A_i, \tau^*, A_j)\;)_{1\le i,j\le k}$$

We define $A_0 = S_{div}$. *In the case where* $A_0 = \emptyset$ *we define* $Prob(A_0, \tau^*, A_0) = 1$ *and*

$$Prob(A_0, \tau^*, A_j) = Prob(A_j, \tau^*, A_0) = 0$$

if $j \ge 1$. *Let* $\mathbf{A}^0_R \;=\; (\; Prob(A_i, \tau^*, A_j)\;)_{0\le i,j\le k}$.

Regardless of whether S_{div} is empty or not, the matrix $\mathbf{A}^0_R$ is of the form

$$\mathbf{A}^0_R \;=\; \left(\begin{array}{c|ccc} 1 & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & \mathbf{A}_R & \\ 0 & & & \end{array}\right).$$

Note that $S_{term} \subseteq S_{div} = A_0$. Hence, $A_i \cap S_{term} = \emptyset$, $i = 1, \ldots, k$.

Lemma A.4 *If* R *is a complete weak bisimulation then* $\mathbf{A}_R$ *and* $\mathbf{A}^0_R$ *are regular. Moreover: For all* $l \in \{1, \ldots, k\}$ *and all* $b_0, \ldots, b_{l-1}, b_{l+1}, \ldots, b_k \in [0, 1]$, *the equation system*

$$x_l = 0, \quad \sum_{i=0}^{k} x_i \cdot Prob(A_i, \tau^*, A_j) \;=\; b_j, \quad j = 0, \ldots, k, j \ne l$$

has at most one solution.

Proof: Let $A_0, \ldots, A_k$ be as in Definition A.3. For each $h \in \{1, \ldots, k\}$ we fix some state $s_h \in A_h$ with $\mathbf{P}(s_h, \tau, A_h) < 1$. Let

$$\mathbf{C} \;=\; (\; \mathbf{P}(s_h, \tau, A_i)\;)_{1\le h,i\le k} \quad \text{and} \quad \mathbf{E} \;=\; \begin{pmatrix} 1-e_1 & 0 & 0 & \cdots & 0 \\ 0 & 1-e_2 & 0 & \cdots & 0 \\ 0 & 0 & 1-e_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1-e_k \end{pmatrix}$$

where

$$e_j \;=\; \sum_{i=1}^{k} \mathbf{P}(s_j, \tau, A_i) \cdot Prob(A_i, \tau^*, A_j).$$

We show that $\mathbf{C} \cdot \mathbf{A}_R \ + \ \mathbf{E} \ = \ \mathbf{A}_R$. Let $d_{h,j}$ be the element of $\mathbf{C} \cdot \mathbf{A}_R \ + \ \mathbf{E}$ in the h-th row and j-th column.

- For $j = 1, \ldots, k$ we have:

$$d_{j,j} \ = \ \sum_{i=1}^{k} \mathbf{P}(s_j, \tau, A_i) \cdot \mathit{Prob}(A_i, \tau^*, A_j) \ + \ 1 - e_j \ = \ 1 \ = \ \mathit{Prob}(A_j, \tau^*, A_j).$$

- For h, $j = 1, \ldots, k$ and $h \neq j$:

$$\mathit{Prob}(A_h, \tau^*, A_j) \ = \ \mathit{Prob}(s_h, \tau^*, A_j) \ = \ \sum_{i=1}^{k} \mathbf{P}(s_h, \tau, A_i) \cdot \mathit{Prob}(A_i, \tau^*, A_j) \ = \ d_{h,j}.$$

  Note that $\mathit{Prob}(A_0, \tau^*, A_j) = 0$ for all $j \geq 1$.

This yields $\mathbf{C} \cdot \mathbf{A}_R \ + \ \mathbf{E} \ = \ \mathbf{A}_R$. Thus, $\mathbf{E} \ = \ (\mathbf{I} - \mathbf{C}) \cdot \mathbf{A}_R$ where $\mathbf{I}$ denotes the $k \times k$-identity matrix. Next we show that $e_j < 1$, $j = 1, \ldots, k$.

- If $\mathit{Prob}(A_{i_0}, \tau^*, A_j) \neq 0$ for some $i_0 \in \{1, \ldots, k\} \setminus \{j\}$ with $\mathbf{P}(s_j, \tau, A_{i_0}) > 0$ then

$$e_j \ \leq \ \sum_{\substack{i=1 \\ i \neq i_0}}^{k} \mathbf{P}(s_j, \tau, A_i) \ + \ \mathbf{P}(s_j, \tau, A_{i_0}) \cdot \mathit{Prob}(A_{i_0}, \tau^*, A_j) \ < \ \mathbf{P}(s_j, \tau) \ \leq \ 1$$

  since $\mathit{Prob}(A_{i_0}, \tau^*, A_j) < 1$ because R is complete.
- If $\mathit{Prob}(A_i, \tau^*, A_j) = 0$ for all $i \in \{1, \ldots, k\} \setminus \{j\}$ with $\mathbf{P}(s_j, \tau, A_i) > 0$ then

$$e_j \ = \ \mathbf{P}(s_j, \tau, A_j) \cdot \mathit{Prob}(A_j, \tau^*, A_j) \ = \ \mathbf{P}(s_j, \tau, A_j) < 1.$$

Thus, in both cases, $e_j < 1$. Hence, $\mathbf{E}$ is regular which yields the regularity of $\mathbf{A}_R$. It is clear that the regularity of $\mathbf{A}_R$ implies the regularity of the matrix $\mathbf{A}_R^0$. Recall that $\mathbf{A}_R^0$ (and thus the inverse matrix $({\mathbf{A}_R}^0)^{-1}$ of $\mathbf{A}_R^0$) are of the form:

$$\mathbf{A}_R^0 \ = \ \left(\begin{array}{c|ccc} 1 & 0 & \ldots & 0 \\ \hline 0 & & & \\ \vdots & & \mathbf{A}_R & \\ 0 & & & \end{array}\right) \qquad ({\mathbf{A}_R}^0)^{-1} \ = \ \left(\begin{array}{c|ccc} 1 & 0 & \ldots & 0 \\ \hline 0 & & & \\ \vdots & & \mathbf{A}_R^{-1} & \\ 0 & & & \end{array}\right)$$

Moreover, $\mathbf{E} \cdot \mathbf{A}_R^{-1} = \mathbf{I} - \mathbf{C}$. Thus, $(1 - e_j) \cdot a_{j,j} \ = \ 1 - \mathbf{P}(s_j, \tau, A_j) > 0$ where $({\mathbf{A}_R}^0)^{-1} = (a_{i,j})_{0 \leq i,j \leq 1}$. Hence,

(*) $a_{j,j} > 0$, $j = 1, \ldots, k$.

Next we show that the equation system of above has at most one solution. Let

$$L = \{(b_0, \ldots, b_{l-1}, t, b_{l+1}, \ldots, b_k) : t \in I\!R\}$$

where $\mathbb{R}$ denotes the set of real numbers. Let $H = \{(x_0, \ldots, x_k) \in \mathbb{R}^{k+1} : x_l = 0\}$ and $L' = \{\mathbf{y} \cdot (\mathbf{A}_R{}^0)^{-1} : \mathbf{y} \in L\}$. Then, $H \cap L'$ is the set of solution of the equation system under consideration. We show that either $H \cap L' = \emptyset$ or $H \cap L'$ consists of a single point. First we observe that H and L (and thus L' and $H \cap L'$) are affine spaces with $dim(H) = k$ and $dim(L) = dim(L') = 1$ where $dim(X)$ denotes the dimension of X. Hence, if $H \cap L' \neq \emptyset$ then

- either $dim(H \cap L') = 0$ (iff $H \cap L'$ consists of a single point)
- or $dim(H \cap L') = 1$ (iff $L' \subseteq H$).

Therefore, it suffices to show that $L' \not\subseteq H$. We suppose that $L' \subseteq H$. Then, there are real vectors $\mathbf{a}$, $\mathbf{c}$ such that $L' = \{\mathbf{a} + t \cdot \mathbf{c} : t \in \mathbb{R}\}$ where $\mathbf{a} = (a_0, \ldots, a_k)$ and $\mathbf{c} = (c_0, \ldots, c_k)$ with $a_l = c_l = 0$ and $\mathbf{c} \neq 0$. By definition of L' we have $L = \{\mathbf{x} \cdot \mathbf{A}_R^0 : \mathbf{x} \in L'\}$. Hence,

$$b_j \;=\; \sum_{i=0}^{k} a_i \cdot Prob(A_i, \tau^*, A_j) \;+\; t \cdot \sum_{i=0}^{k} c_i \cdot Prob(A_i, \tau^*, A_j)$$

for all $j = 0, \ldots, k$, $j \neq l$ and $t \in \mathbb{R}$. Thus,

$$\sum_{i=0}^{k} c_i \cdot Prob(A_i, \tau^*, A_j) = 0 \;\text{ if } j \neq l.$$

Hence,

$$c \;\stackrel{\text{def}}{=}\; \sum_{i=0}^{k} c_i \cdot Prob(A_i, \tau^*, A_l) \;\neq\; 0$$

(since, otherwise the rows of $\mathbf{A}_R^0$ would be linear dependent in contradiction to the regularity of $\mathbf{A}_R^0$). W.l.o.g. $c = 1$. Then, $\mathbf{c}$ is the l-th row of $(\mathbf{A}_R{}^0)^{-1}$. In particular, $0 = c_l = a_{l,l}$. But this contradicts the constraint $a_{l,l} > 0$ from (*). ■

We show that $\approx$ coincides with another kind of bisimulation equivalence that we call delay bisimulation, in analogy to [26].

Definition A.5 *A* delay bisimulation *is an equivalence relation R on S such that $Prob(s, \tau^*\widehat{a}, C) = Prob(s', \tau^*\widehat{a}, C)$ for all $(s, s') \in R$, $a \in Act$ and all equivalence classes $C \in S/R$. $s \approx_{dl} s'$ iff $(s, s') \in R$ for some delay bisimulation R.*

Lemma A.6 *$s \approx_{dl} s'$ implies $s \approx s'$. More precisely: Each delay bisimulation is a weak bisimulation.*

Proof: Let R be a delay bisimulation. We show that R is a weak bisimulation. Let $\alpha \in Act \setminus \{\tau\}$, $s \in S$ and $C \in S/R$. Then, for all $B \in S/R$ and $s \in B$:

$$Prob(s,\tau^*\alpha\tau^*,C) \;=\; \sum_{t\in S} Prob(s,\tau^*\alpha,t)\cdot Prob(t,\tau^*,C)$$
$$= \sum_{A\in S/R} Prob(B,\tau^*\alpha,A)\cdot Prob(A,\tau^*,C).$$

Hence, if $(s,s')\in R$ then $Prob(s,\tau^*\alpha\tau^*,C) = Prob(s',\tau^*\alpha\tau^*,C)$ for all $C\in S/R$ and $\alpha\in Act\setminus\{\tau\}$. ■

Lemma A.7 *$s \approx s'$ implies $s \approx_{dl} s'$. More precisely: Each complete weak bisimulation is a delay bisimulation.*

Proof: Let R be a complete weak bisimulation. We fix some $\alpha\in Act\setminus\{\tau\}$ and show that $Prob(s,\tau^*\alpha,A) = Prob(s',\tau^*\alpha,A)$ for all $s\approx s'$ and all $A\in S/R$. (Ranging over all α, we obtain that R is a delay bisimulation. Thus, $\approx\subseteq\approx_{dl}$.)

By the regularity of $\mathbf{A}^0_R$ (Lemma A.4): Whenever we fix a real vector $\mathbf{a}$ (of length $k+1$) then the linear equation system $\mathbf{x}\cdot\mathbf{A}^0_R=\mathbf{a}$ has a unique solution. For $s\in S$ and $j=0,1,\ldots,k$ we have:

$$Prob(s,\tau^*\alpha\tau^*,A_j) = \sum_{t\in S} Prob(s,\tau^*\alpha,t)\cdot Prob(t,\tau^*,A_j)$$
$$= \sum_{i=0}^{k} Prob(s,\tau^*\alpha,A_i)\cdot Prob(A_i,\tau^*,A_j).$$

Thus, for fixed l: For all states $s\in A_l$, the vector $(Prob(s,\tau^*\alpha,A_i))_{0\le i\le k}$ is a solution of the linear equation system $\mathbf{x}\cdot\mathbf{A}^0_R=\mathbf{a}$ where $\mathbf{a}=(a_j)_{0\le j\le k}$ and $a_j=Prob(A_l,\tau^*\alpha\tau^*,A_j)$. By the regularity of $\mathbf{A}^0_R$: If $(s,s')\in R$ (i.e. $s,s'\in A_l$ for some l) then $Prob(s,\tau^*\alpha,A_i) = Prob(s',\tau^*\alpha,A_i)$, $i=0,\ldots,k$. ■

Proposition A.8 *$s \approx s'$ iff $s \approx_{dl} s'$*

Proof: follows by Lemma A.6 and Lemma A.7. ■

Lemma A.9 *$s \approx_{br} s'$ implies $s \approx s'$. More precisely: Each branching bisimulation is a weak bisimulation.*

Proof: Let R be a branching bisimulation. It suffices to show that R is a delay bisimulation (Lemma A.6). For $r\ge 1$ and $A,C\in S/R$, $A\ne C$, let Γ_r be the set of tuples $(C_0,\ldots,C_r)$ such that

- $C_i\in S/R$, $i=0,1,\ldots,r$,
- $C_0=A$, $C_r=C$,
- $C_i\ne C_{i+1}$, $i=0,\ldots,r-1$.

Then, for all $s \in A$:

$$Prob(s, \tau^*, C) \;=\; \sum_{r=1}^{\infty} \sum_{(C_1,\ldots,C_r)\in\Gamma_r} \prod_{i=0}^{r-1} Prob_R(C_i, \tau^*, C_{i+1})$$

Hence, $Prob(s, \tau^*, C) = Prob(s', \tau^*, C)$ for all s, $s' \in A$. Similarly,

$$Prob(s, \tau^*\alpha, C) = Prob(s', \tau^*\alpha, C)$$

for all s, $s' \in A$, $\alpha \in Act \setminus \{\tau\}$ and $C \in S/R$. ■

Lemma A.10 *Let R be an equivalence relation on S. Then, R is a branching bisimulation if and only if for all $C \in S/R$, $a \in Act$ and $(s, s') \in R$:*

(1) *If $\mathbf{P}(s, \tau, [s]_R)$, $\mathbf{P}(s', \tau, [s']_R) < 1$ and $(a, C) \neq (\tau, [s]_R)$ then*

$$\frac{\mathbf{P}(s, a, C)}{1 - \mathbf{P}(s, \tau, [s]_R)} \;=\; \frac{\mathbf{P}(s', a, C)}{1 - \mathbf{P}(s', \tau, [s']_R)}.$$

(2) *If $\mathbf{P}(s, \tau, [s]_R) = 1$ then*
- *either $\mathbf{P}(t, \tau, [t]_R) = 1$ for all $t \in [s]_R$*
- *or there exists a finite path σ starting in s with $\sigma(i) \in [s]_R$, $i = 1, \ldots, |\sigma|$ and $\mathbf{P}(last(\sigma), \tau, [s]_R) < 1$.*

Proof: Let $T = S_{term} \cup \{t \in S : \mathbf{P}(t, \tau, [t]_R) = 1\}$ (i.e. T is the set of *silent* states $T = S \setminus S_{ns}^{\mathcal{X}}$ with respect to $\mathcal{X} = S/R$).

"if": Let $A \in S/R$ with $A \subseteq T$. Then, for all $s \in A$:

$$Prob_R(s, \tau^*\widehat{a}, C) \;=\; \begin{cases} 1 : \text{if } a = \tau \text{ and } C = A \\ 0 : \text{otherwise} \end{cases}$$

Then, $Prob_R(t, \tau^*\widehat{a}, C) = Prob_R(t', \tau^*\widehat{a}, C)$ for all t, $t' \in A$. Let A, $C \in S/R$ such that $A \not\subseteq T$. Then, the matrix $\mathbf{I} - \mathbf{C}_A$ is regular where $\mathbf{I}$ is the $|A| \times |A|$-identity matrix and $\mathbf{C}_A = (\mathbf{P}(s, \tau, t))_{s,t\in A}$. This can be seen as follows: If $(\mathbf{I} - \mathbf{C}_A) \cdot \mathbf{x} = 0$, $\mathbf{x} = (x_s)_{s\in A}$, then $x_s = \sum_{t\in A} \mathbf{P}(s, \tau, t) \cdot x_t$ for all $s \in A$. If we suppose that $\mathbf{x} \neq 0$ then we may assume w.l.o.g. that $x_s > 0$ for some $s \in A$ (otherwise we deal with $-\mathbf{x}$ rather than $\mathbf{x}$). Let W be the set of states $s \in A$ where x_s is maximal. Then, for all $s \in W$, $\mathbf{P}(s, \tau, A) = 1$ and $x_s = x_t$ if $\mathbf{P}(s, \tau, t) > 0$. Thus, $W \subseteq T$ and, if $s \in W$ then $\mathbf{P}(s, \tau, t) > 0$ implies $t \in W$. Hence, there is no finite path σ starting in s with $\sigma(i) \in A$, $i = 0, 1, \ldots, |\sigma|-1$, and $\mathbf{P}(last(\sigma), \tau, A) < 1$. This contradicts (2).

Let $a \in Act$ such that $(a, C) \neq (\tau, A)$. Then,

$$Prob_R(s, \tau^*\widehat{a}, C) = \sum_{t\in A} \mathbf{P}(s, \tau, t) \cdot Prob_R(t, \tau^*\widehat{a}, C) \;+\; \mathbf{P}(s, a, C).$$

Hence, the vector $(Prob_R(s, \tau^*\widehat{a}, C))_{s\in A}$ solves the equation system $(\mathbf{I} - \mathbf{C}_A) \cdot \mathbf{x} = \mathbf{b}$ where $\mathbf{b} = (b_s)_{s\in A}$ and $b_s = \mathbf{P}(s, a, C)$. On the other hand, for all $s \in A$:

$$(*) \quad \mathbf{P}(s,a,C) \;=\; = \sum_{t\in A} \mathbf{P}(s,\tau,t)\cdot \mathbf{P}(s,a,C) \;+\; \mathbf{P}(s,a,C)(1-\mathbf{P}(s,\tau,A)).$$

Let $x = \mathbf{P}(s, a, C)/(1 - \mathbf{P}(s, \tau, A))$ where $s \in A \setminus T$. (*) yields:

$$x \;=\; \sum_{t\in A} \mathbf{P}(s,\tau,t)\cdot x \;+\; \mathbf{P}(s,a,C) \quad \text{for all } s \in A.$$

(Note that $\mathbf{P}(s, a, C) = 0$ and $\mathbf{P}(s, \tau, A) = 1$ for $s \in A \cap T$.) Hence, the vector $\mathbf{x} = (x_s)_{s\in A}$ where $x_s = x$ for all $s \in A$ solves the equation system $(\mathbf{I} - \mathbf{C}_A) \cdot \mathbf{x} = \mathbf{b}$. By the regularity of $\mathbf{I} - \mathbf{C}_A$ we get $Prob_R(s, \tau^*\widehat{a}, C) = x$ for all $s \in A$. This yields that R is a branching bisimulation and

$$Prob_R(s,\tau^*\widehat{a},C) \;=\; x \;=\; \frac{\mathbf{P}(s,a,C)}{1-\mathbf{P}(s,\tau,A)}$$

for all $s \in A \setminus T$.

"only if": Let R be a branching bisimulation, A, $C \in S/R$ and $a \in Act$ such that $(a, C) \neq (\tau, A)$. Let

$$x_{a,C} \;=\; Prob_R(A, \tau^*\widehat{a}, C).$$

Then, $x_{a,C} \;=\; \mathbf{P}(s, \tau, A) \cdot x_{a,C} \;+\; \mathbf{P}(s, a, C)$ for all $s \in A$. Hence, if $s \in A \cap T$ then

$$x_{a,C} \;=\; \frac{\mathbf{P}(s,a,C)}{1-\mathbf{P}(s,\tau,A)}.$$

If $s \in A \cap T$ and $A \not\subseteq T$ then $x_{a,C} \neq 0$ for some pair (a, C) as above. Thus,

$$Prob_R(s, \tau^*\widehat{a}, C) = x_{a,C} > 0.$$

Then, there exists a finite path σ' starting in s of length $r \geq 1$ with $trace(\sigma') \in \tau^*a$, $\sigma'(i) \in A$, $i = 0, 1, \ldots, r - 1$ and $last(\sigma') \in C$. Let σ be the $(r - 1)$-th prefix of σ'. Then, $\sigma(i) \in A$, $i = 1, \ldots, r - 1$ and $\mathbf{P}(last(\sigma), \tau, A) < 1$ (since $\mathbf{P}(last(\sigma), a, C) > 0$). ■

Lemma A.11 *Let R be a branching bisimulation on S. If $s \in S$ with $\mathbf{P}(s, \tau, [s]_R) < 1$ then*

$$Prob_R(s,\tau^*\widehat{a},C) \;=\; \frac{\mathbf{P}(s,a,C)}{1-\mathbf{P}(s,\tau,C)}$$

for all $a \in Act$, $C \in S/R$ with $(a, C) \neq (\tau, [s]_R)$.

Definition A.12 *Let R be an equivalence relation on S.*

$$\mathbf{P}_R : S \times Act \times S \longrightarrow [0, 1]$$

is given by: $\mathbf{P}_R(s,a,t) = 0$ *if* $\mathbf{P}(s,\tau,[s]_R) = 1$ *or* $s \in S_{term}$. *For* $s \in S \setminus S_{term}$ *with* $\mathbf{P}(s,\tau,[s]_R) < 1$,

$$\mathbf{P}_R(s,a,t) \;=\; \frac{\mathbf{P}(s,a,t)}{1-\mathbf{P}(s,\tau,[s]_R)} \quad \textit{if } a \neq \tau \textit{ or } t \notin [s]_R.$$

and $\mathbf{P}_R(s,\tau,t) = 0$ *if* $t \in [s]_R$. *For* $C \subseteq S$, $a \in Act$ *and* $s \in S$, *let*

$$\mathbf{P}_R(s,a,C) \;=\; \sum_{t \in C} \mathbf{P}_R(s,a,t).$$

Clearly, $\mathbf{P}_R(\cdot) = \mathbf{P}_{\mathcal{X}}(\cdot)$ (defined as in Definition 4.2 for the induced partition $\mathcal{X} = S/R$.

Lemma A.13 $s \approx s'$ *implies* $s \approx_{br} s'$. *More precisely: Each complete weak bisimulation is a branching bisimulation.*

Proof: Let R be a complete weak bisimulation. It suffices to show that R fulfills the conditions (1), (2) of Lemma A.10. As in the proof of that lemma, let $T = S_{term} \cup \{t \in S : \mathbf{P}(t,\tau,[t]_R) = 1\}$.

Condition (2): Let $A \in S/R$, $A \neq S_{div}$. There exists $a \in Act$ and $C \in S/R$ with $(\tau, A) \neq (a, C)$ such that $Prob(A, \tau^* \widehat{a}, C) \neq 0$. Hence, for each $s \in T$ there is a finite path σ with $trace(\sigma) \in \tau^* \widehat{a}$ and $last(\sigma) \in C$. Let i be the smallest index such that $\sigma(i) \notin T$. (Such an index i exists by definition of T.) Then, $i \geq 1$ and $\sigma(i) \in A \setminus T$. (Note that $\sigma(i-1) \in T$ implies $\sigma(i) \in A$.)

Condition (1): Let $A_0, \ldots, A_k$ be as in Definition A.3. Let $C \in S/R$, $A_j \neq C$, $s \in A_j$. Then,

$$Prob(s,\tau^*,C) \;=\; \sum_{i=0}^{k} \mathbf{P}(s,\tau,A_i) \cdot Prob(A_i,\tau^*,C).$$

Note that $Prob(C,\tau^*,C) = 1$. Now we suppose that $s \in A_j \setminus T$. Then,

$$\begin{aligned}
\frac{Prob(s,\tau^*,C)}{1-\mathbf{P}(s,\tau,A_j)} \;&=\; \sum_{i=0}^{k} \frac{\mathbf{P}(s,\tau,A_i)}{1-\mathbf{P}(s,\tau,A_j)} \cdot Prob(A_i,\tau^*,C) \\
&= \sum_{\substack{i=0 \\ i \neq j}}^{k} \frac{\mathbf{P}(s,\tau,A_i)}{1-\mathbf{P}(s,\tau,A_j)} \cdot Prob(A_i,\tau^*,C) \;+\; \frac{\mathbf{P}(s,\tau,A_j)}{1-\mathbf{P}(s,\tau,A_j)} \cdot Prob(A_j,\tau^*,C) \\
&= \sum_{\substack{i=0 \\ i \neq j}}^{k} \mathbf{P}_R(s,\tau,A_i) \cdot Prob(A_i,\tau^*,C) \;+\; \frac{\mathbf{P}(s,\tau,A_j)}{1-\mathbf{P}(s,\tau,A_j)} \cdot Prob(s,\tau^*,C)
\end{aligned}$$

Here, we use the fact that $Prob(s,\tau^*,C) = Prob(A_j,\tau^*,C)$. We obtain:

$$Prob(s,\tau^*,C) \;=\; Prob(s,\tau^*,C)\cdot\left(\frac{1}{1-\mathbf{P}(s,\tau,A_j)}-\frac{\mathbf{P}(s,\tau,A_j)}{1-\mathbf{P}(s,\tau,A_j)}\right)$$

$$=\;\sum_{\substack{i=0\\ i\neq j}}^{k}\mathbf{P}_R(s,\tau,A_i)\cdot Prob(A_i,\tau^*,C)$$

Thus, for each $s\in A_j\setminus T$, the vector $(\mathbf{P}_R(s,\tau,A_i))_{0\leq i\leq k}$ solves the equation system

$$x_j=0,\quad \sum_{i=0}^{k}x_i\cdot Prob(A_i,\tau^*,C)\;=\;Prob(A_j,\tau^*,C).$$

Lemma A.4 yields $\mathbf{P}_R(s,\tau,C)=\mathbf{P}_R(s',\tau,C)$ for all $s,s'\in A_j$ and $C\in S/R$, $C\neq A_j$. Let

$$\mathbf{P}_R(A_j,\tau,C)=\mathbf{P}_R(s,\tau,C)\;\text{ where } s\in A_j\cap T.$$

For all $\alpha\in Act\setminus\{\tau\}$ and $s\in A_j$:

$$Prob(s,\tau^*\alpha,C)\;=\;\sum_{i=0}^{k}\mathbf{P}(s,\tau,A_i)\cdot Prob(A_i,\tau^*\alpha,C)\;+\;\mathbf{P}(s,\alpha,C)$$

As before, we obtain for $s\in A_j\setminus T$:

$$Prob(s,\tau^*\alpha,C)\;=\;\sum_{\substack{i=0\\ i\neq j}}^{k}\mathbf{P}_R(s,\tau,A_i)\cdot Prob(A_i,\tau^*\alpha,C)\;+\;\mathbf{P}_R(s,\alpha,C).$$

Then, for all $s\in A_j\setminus T$, $\alpha\in Act$ and $C\in S/R$:

$$Prob(A_j,\tau^*\alpha,C)\;=\;\sum_{\substack{i=0\\ i\neq j}}^{k}\mathbf{P}_R(A_j,\tau,A_i)\cdot Prob(A_i,\tau^*\alpha,C)\;+\;\mathbf{P}_R(s,\alpha,C)$$

We obtain $\mathbf{P}_R(s,\alpha,C)=\mathbf{P}_R(s',\alpha,C)$ for all $s,\,s'\in A_j\setminus T$. ■

Corollary A.14 *$s\;\approx\;s'$ iff $s\;\approx_{br}\;s'$.*

Lemma A.15 *Let R_1, R_2 be branching bisimulations. Then, $R=(R_1\cup R_2)^*$ is a branching bisimulation.*

Proof: We show that R fulfills the conditions (1) and (2) of Lemma A.10. First we observe that for $j\in\{1,2\}$, each equivalence class $C'\in S/R$ can be written as $C'=C_0\cup\ldots\cup C_r$ where $C_i\in S/R_j$.

Condition (1): Let B, $C_0,\ldots,C_r\in S/R_j$ such that $C_i\cap C_h=\emptyset$ if $i\neq h$. Let s, $s'\in C_0$ with $\mathbf{P}(s,\tau,C_0)$, $\mathbf{P}(s',\tau,C_0)<1$ and $a\in Act$ with $a\neq\tau$ if $C_0=B$ and $C'=C_0\cup\ldots\cup C_r$. Then:

$$\mathbf{P}(s,a,B)\cdot(1-\mathbf{P}(s',\tau,C')) \;=\; \mathbf{P}(s,a,B)\cdot\left(1-\sum_{i=0}^{r}\mathbf{P}(s',\tau,C_i)\right)$$

$$=\mathbf{P}(s,a,B)\cdot\left(1-\mathbf{P}(s',\tau,C_0)-\sum_{i=1}^{r}\frac{\mathbf{P}(s,\tau,C_i)}{1-\mathbf{P}(s,\tau,C_0)}\cdot(1-\mathbf{P}(s',\tau,C_0))\right)$$

$$=(1-\mathbf{P}(s',\tau,C_0))\cdot\mathbf{P}(s,a,B)\cdot\left(1-\sum_{i=1}^{r}\frac{\mathbf{P}(s,\tau,C_i)}{1-\mathbf{P}(s,\tau,C_0)}\right)$$

$$=\mathbf{P}(s',a,B)\cdot(1-\mathbf{P}(s,\tau,C_0))\cdot\left(1-\sum_{i=1}^{r}\frac{\mathbf{P}(s,\tau,C_i)}{1-\mathbf{P}(s,\tau,C_0)}\right)$$

$$=\mathbf{P}(s',a,B)\cdot\left(1-\sum_{i=0}^{r}\mathbf{P}(s,\tau,C_i)\right) \;=\; \mathbf{P}(s',a,B)\cdot(1-\mathbf{P}(s,\tau,C'))$$

Now we assume that $C'\in S/R$. Hence, for all $B'\in S/R$, $a\in Act$ such that $a\neq\tau$ if $C_0\subseteq B'$: If $\mathbf{P}(s,\tau,C')$, $\mathbf{P}(s',\tau,C')<1$. then:

$$\frac{\mathbf{P}(s,a,B')}{1-\mathbf{P}(s,\tau,C')} \;=\; \sum_{i=0}^{r}\frac{\mathbf{P}(s,a,B_i)}{1-\mathbf{P}(s,\tau,C')} \;=\; \sum_{i=0}^{r}\frac{\mathbf{P}(s',a,B_i)}{1-\mathbf{P}(s',\tau,C')} \;=\; \frac{\mathbf{P}(s',a,B')}{1-\mathbf{P}(s',\tau,C')}$$

if $B'=B_0\cup\ldots\cup B_m$ where $B_i\in S/R_j$ and $B_i\cap B_h=\emptyset$ if $i\neq h$.

Condition (2): Let $\mathbf{P}(s,\tau,[s]_R)=1$. We may assume that $\mathbf{P}(s,\tau,[s]_{R_j})<1$, $j=1,2$, and that $\mathbf{P}(t,\tau,C')<1$ for some $t\in C'$ where $C'=[s]_R$. (In the case where $\mathbf{P}(s,\tau,[s]_{R_j})=1$ for some j we apply Lemma A.10 to R_j and obtain the claim.) There is some $a\in Act$, $B'\in S/R$ with $B'\neq C'$ if $a=\tau$ and $\mathbf{P}(t,a,B')>0$. By definition of R there is a sequence $s=s_0,s_1,\ldots,s_l=t$ with $(s_i,s_{i+1})\in R_1\cup R_2$, $i=1,\ldots,l$. It can easily shown by induction on i that $\mathit{Prob}(s_i,\tau^*\widehat{a},B')>0$, $i=0,\ldots,l$, Hence, there is a finite path σ with $\mathit{first}(\sigma)=s$, $\sigma(i)\in C'$, $i=0,1,\ldots,|\sigma|$ and $\mathbf{P}(\mathit{last}(\sigma),\tau,C')<1$. ■

Lemma A.16 *$\approx_{br}$ is a branching bisimulation.*

Proof: Let $R_1,\ldots,R_r$ be an enumeration of all branching bisimulations (on the fixed fully probabilistic system $(S,Act,\mathbf{P})$). Let $R=(\bigcup_i R_i)^*$. By induction on r and using Lemma A.15, it can be shown that R is a branching bisimulation. Thus, $R\subseteq\approx_{br}$. On the other hand, $\approx_{br}=\bigcup_i R_i$ (by definition of $\approx_{br}$). Hence, $\approx_{br}\subseteq R$. Thus, $\approx_{br}=R$ is a branching bisimulation. ■

Lemma A.17 *$\approx$ is a weak bisimulation.*

Proof: Lemma A.9, Lemma A.16 and Corollary A.14 yield that $\approx=\approx_{br}$ is a weak bisimulation. ■

Theorem A.18 *If $s\approx s'$. then, for all $C\in S/\approx$, $k\geq 1$ and $\alpha_1,\ldots,\alpha_k\in Act\setminus\{\tau\}$:*

(a) $Prob(s, \tau^*\alpha_1\tau^*\alpha_2\tau^* \ldots \tau^*\alpha_k, C) = Prob(s', \tau^*\alpha_1\tau^*\alpha_2\tau^* \ldots \tau^*\alpha_k, C)$,
(b) $Prob(s, \tau^*\alpha_1\tau^*\alpha_2\tau^* \ldots \tau^*\alpha_k\tau^*, C) = Prob(s', \tau^*\alpha_1\tau^*\alpha_2\tau^* \ldots \tau^*\alpha_k\tau^*, C)$.

Proof: We prove part (a) by induction on k. In the basis of induction ($k = 1$) we have to show that $Prob(s, \tau^*\alpha, C) = Prob(s', \tau^*\alpha, C)$ for all visible actions α and all weak bisimulation equivalence classes C. This follows immediately by Proposition A.8 and Lemma A.17. In the induction step $k-1 \Longrightarrow k$ we assume that $k \geq 2$, $\alpha_1, \ldots, \alpha_k \in Act \setminus \{\tau\}$ and $\Omega = \tau^*\alpha_2\tau^* \ldots \tau^*\alpha_k$. Then,

- $Prob(t, \Omega, C) \;=\; Prob(t', \Omega, C)$ for all $t \approx t'$ and $C \in S/\approx$
- $Prob(s, \tau^*\alpha_1, A) = Prob(s', \tau^*\alpha_1, A)$ for all $A \in S/\approx$

(induction hypothesis). Thus:

$$Prob(s, \tau^*\alpha_1\Omega, C) \;=\; \sum_{A \in S/\approx} Prob(s, \tau^*\alpha_1, A) \cdot Prob(A, \Omega, C)$$

$$= \sum_{A \in S/\approx} Prob(s', \tau^*\alpha_1, A) \cdot Prob(A, \Omega, C) \;=\; Prob(s', \tau^*\alpha_1\Omega, C).$$

Here, we use the fact that $Path_{ful}(u, \tau^*\alpha_1\Omega, C)$ can be written as disjoint union of the sets $\Pi_A(u)$, $A \in S/\approx$, where Π_A is the set of fulpaths π such that

- $trace(\pi^{(k)}) \in \tau^*\alpha_1$,
- $\pi(k) \in A$,
- $\pi = \pi^{(k)} \circ \gamma$ where $\gamma \in \; Path_{ful}(\pi(k), \Omega, C)$

for some $k \geq 0$. Part (b) can be derived from (a):

$$Prob(s, \Omega, C) \;=\; \sum_{A \in S/\approx} Prob(s, \tau^*\alpha_1\tau^* \ldots \tau^*\alpha_k, A) \cdot Prob(A, \tau^*, C)$$

$$= \sum_{A \in S/\approx} Prob(s', \tau^*\alpha_1\tau^* \ldots \tau^*\alpha_k, A) \cdot Prob(A, \tau^*, C) \;=\; Prob(s', \Omega, C).$$

where $\Omega = \tau^*\alpha_1\tau^* \ldots \tau^*\alpha_k\tau^*$. ■

A.2 Weak bisimulation and the testing equivalences $=_{ste}$ and $\equiv_0$

We complete the proofs of Theorem 3.12 and Theorem 3.13 by showing that $\approx$ is finer than the testing equivalences $=_{ste}$ and $\equiv_0$.

Lemma A.19 *If A, $C \in S/\approx$, s, $s' \in A$, $L \subseteq Act \setminus \{\tau\}$ and $\alpha \in L$ then*

$$Q(s, L, \alpha, C) \;=\; Q(s', L, \alpha, C).$$

Proof: First we observe that, for all $A,\ B \in S/\approx\ =\ S/\approx_{br}$, $s,\ s' \in A$ with $\mathbf{P}(s,\tau,A)$, $\mathbf{P}(s',\tau,A) < 1$ and $\alpha \in Act \setminus \{\tau\}$:

- $\dfrac{\mathbf{P}(s,\tau,S\setminus A)\ +\ \mathbf{P}(s,L)}{1-\mathbf{P}(s,\tau,A)}\ =\ \dfrac{\mathbf{P}(s',\tau,S\setminus A)\ +\ \mathbf{P}(s',L)}{1-\mathbf{P}(s',\tau,A)}$
- $\dfrac{\mathbf{P}(s,\alpha,B)}{1-\mathbf{P}(s,\tau,A)}\ =\ \dfrac{\mathbf{P}(s',\alpha,B)}{1-\mathbf{P}(s',\tau,A)}$

In particular, $\mathbf{P}(s,\tau,S\setminus A)\ +\ \mathbf{P}(s,L) = 0$ iff $\mathbf{P}(s',\tau,S\setminus A)\ +\ \mathbf{P}(s',L) = 0$. If $A = S_{div}$ then we put $\mathbf{P}'(A,\alpha,B) = 0$ and $r_A = 1$. For $A \in S/\approx$, $A \neq S_{div}$, we define

$$\mathbf{P}'(A,\alpha,B)\ =\ \frac{\mathbf{P}(s,\alpha,B)}{1-\mathbf{P}(s,\tau,A)},\quad r_A\ =\ \frac{\mathbf{P}(s,\tau,S\setminus A)+\mathbf{P}(s,L)}{1-\mathbf{P}(s,\tau,A)}$$

where $s \in A$ such that $\mathbf{P}(s,\tau,A) < 1$. Let $(q_A)_{A\in S/\approx}$ be the unique solution of the following equation system.

1. $q_A = 0$ if $r_A = 0$ or $Prob(A,\tau^*\alpha,C) = 0$.
2. If $r_A > 0$ and $Prob(A,\tau^*\alpha,C) > 0$ then

$$q_A =\ \frac{1}{r_A}\cdot\left(\mathbf{P}'(A,\alpha,C)\ +\ \sum_{\substack{B\in S/\approx\\ B\neq A}}\mathbf{P}'(A,\tau,B)\cdot q_B\right).$$

The uniqueness of the equation system above is an easy verification. For all $A \in S/\approx$ such that $r_A > 0$ and $Prob(A,\tau^*\alpha,C) > 0$ and $s \in A$ with $\mathbf{P}(s,\tau,A) > 0$ we have:

$$\mathbf{P}(s,\tau,S\setminus A)+\mathbf{P}(s,L) > 0$$

and

$$\begin{aligned}
&\left(1-\frac{\mathbf{P}(s,\tau,A)}{\mathbf{P}(s,\tau)+\mathbf{P}(s,L)}\right)\cdot q_A\ =\ \frac{\mathbf{P}(s,\tau,S\setminus A)+\mathbf{P}(s,L)}{\mathbf{P}(s,\tau)+\mathbf{P}(s,L)}\cdot q_A\\
&=\ \frac{\mathbf{P}(s,\tau,S\setminus A)+\mathbf{P}(s,L)}{\mathbf{P}(s,\tau)+\mathbf{P}(s,L)}\cdot\frac{1}{r_A}\cdot\left(\mathbf{P}'(A,\alpha,C)+\sum_{\substack{B\in S/\approx\\ B\neq A}}\mathbf{P}'(A,\tau,B)\cdot q_B\right)\\
&=\ \frac{1-\mathbf{P}(s,\tau,A)}{\mathbf{P}(s,\tau)+\mathbf{P}(s,L)}\cdot\left(\mathbf{P}'(A,\alpha,C)+\sum_{\substack{B\in S/\approx\\ B\neq A}}\mathbf{P}'(A,\tau,B)\cdot q_B\right)
\end{aligned}$$

$$= \frac{1}{\mathbf{P}(s,\tau)+\mathbf{P}(s,L)} \cdot \left(\mathbf{P}(s,\alpha,C) + \sum_{\substack{B\in S/\approx \\ B\neq A}} \mathbf{P}(A,\tau,B)\cdot q_B \right)$$

$$= \frac{\mathbf{P}(s,\alpha,C)}{\mathbf{P}(s,\tau)+\mathbf{P}(s,L)} + \sum_{\substack{B\in S/\approx \\ B\neq A}} \frac{\mathbf{P}(s,\tau,B)}{\mathbf{P}(s,\tau)+\mathbf{P}(s,L)} \cdot q_B.$$

Thus,

$$q_A = \frac{\mathbf{P}(s,\alpha,C)}{\mathbf{P}(s,\tau)+\mathbf{P}(s,L)} + \sum_{B\in S/\approx} \frac{\mathbf{P}(s,\tau,B)}{\mathbf{P}(s,\tau)+\mathbf{P}(s,L)} \cdot q_B.$$

For $A \in S/\approx$ and $s \in A$, let $q_s = q_A$ and $r_s = r_A$. Then, the vector $(q_s)_{s\in S}$ solves the following regular linear equation system. If $\mathit{Prob}(s,\tau^*\alpha,C) = 0$ or $r_s = 0$ then $q_s = 0$. Otherwise,

$$q_s = \frac{\mathbf{P}(s,\alpha,C)}{\mathbf{P}(s,\tau)+\mathbf{P}(s,L)} + \sum_{u\in S} \frac{\mathbf{P}(s,\tau,u)}{\mathbf{P}(s,\tau)+\mathbf{P}(s,L)} \cdot q_u.$$

It is easy to see that, if $r_s = 0$ then $Q(s,L,\alpha,C) = 0$. Thus, the vector $(Q(s,L,\alpha,C))_{s\in S}$ is also a solution of the equation system above. Hence,

$$q_s = Q(s,L,\alpha,C) \text{ for all } s \in S.$$

We conclude: $Q(s,L,\alpha,C) = q_A = Q(s',L,\alpha,C)$ for all s, $s' \in A$, $A \in S/\approx$. ■

Theorem A.20 *$\approx$ is finer than $=_{ste}$.*

Proof: As observed in [15], $s =_{ste} s'$ iff

$$Q(s,L_1\ldots L_k,\alpha_1\ldots\alpha_k) = Q(s',L_1\ldots L_k,\alpha_1\ldots\alpha_k)$$

for all $L_1,\ldots,L_k \in \mathit{Offr}$ and $\alpha_1,\ldots,\alpha_k \in \mathit{Act}\setminus\{\tau\}$. By induction on k and using Lemma A.19 we obtain that, if $s \approx s'$ then

$$Q(s,L_1\ldots L_k,\alpha_1\ldots\alpha_k,C) = Q(s',L_1\ldots L_k,\alpha_1\ldots\alpha_k,C)$$

for all $C \in S/\approx$. Summing up over all $C \in S/\approx$ we obtain

$$Q(s,L_1\ldots L_k,\alpha_1\ldots\alpha_k) = Q(s',L_1\ldots L_k,\alpha_1\ldots\alpha_k).$$

Hence, $s =_{ste} s'$. ■

Definition A.21 *For X to be a set, let $\mathit{Distr}_0(X)$ be the set consisting of all distributions on X and the function $\rho : X \longrightarrow [0,1]$ with $\rho(x) = 0$ for all $x \in X$.*

Definition A.22 *A* probabilistic trace *is a finite sequence*

$$\theta \;=\; \langle \rho_1, \alpha_1 \rangle \langle \rho_2, \alpha_2 \rangle \;\ldots \langle \rho_k, \alpha_k \rangle$$

over $Distr_0(Act \setminus \{\tau\}) \times (Act \setminus \{\tau\})$. ε_{PrTr} *denotes the empty probabilistic trace,* $ProbTraces$ *the collection of all probabilistic traces.*

Definition A.23 *For* $\rho \in Distr_0(Act \setminus \{\tau\})$ *and* $s \in S$, *the* normalizator *of* s *and* ρ *is defined by*

$$norm(s,\rho) \;=\; \sum_{\alpha \in Act \setminus \{\tau\}} \mathbf{P}(s,\alpha) \cdot \rho(\alpha) \;+\; \mathbf{P}(s,\tau).$$

[19,73] define the probabilities $N(s,\alpha,\rho,C)$ that from state s the state t is reached via a finite path labelled by $\tau^*\alpha$ given that the environment is enabling actions accordance with ρ. Here, we use a slightly different way to define $N(\cdot)$ (which yields the same values).

Definition A.24 *Let*

$$N : S \times (Act \setminus \{\tau\}) \times Distr_0(Act \setminus \{\tau\}) \times 2^S \longrightarrow [0,1]$$

be defined as follows. The vector $(N(s,\alpha,\rho,C))_{s \in S}$ *is the unique solution of the following linear equation system:*

1. *If* $Prob(s,\tau^*\alpha,C) = 0$ *or* $norm(s,\rho) = 0$ *then* $N(s,\alpha,\rho,C) = 0$.
2. *If* $Prob(s,\tau^*\alpha,C) > 0$ *and* $norm(s,\rho) > 0$ *then*

$$N(s,\alpha,\rho,C) \;=\; \frac{\rho(\alpha)}{norm(s,\rho)} \cdot \mathbf{P}(s,\alpha,C) \;+\; \sum_{t \in S} \frac{\mathbf{P}(s,\tau,t)}{norm(s,\rho)} \cdot N(t,\alpha,\rho,C).$$

Clearly, if $\rho(\alpha) = 0$ for all α then $N(s,\alpha,\rho,C) = 0$ for all states s and $C \subseteq S$.

Definition A.25 *Let* $M : S \times ProbTraces \longrightarrow [0,1]$ *be given by:* $M(s,\varepsilon_{PrTr}) = 1$ *and*

$$M(s,\langle \rho,\alpha \rangle \theta) \;=\; \sum_{t \in S} N(s,\alpha,\rho,t) \cdot M(t,\theta)$$

We define $s \;\equiv_0\; s'$ *iff* $M(s,\theta) = M(s',\theta)$ *for all* $\theta \in ProbTraces$.

Lemma A.26 *If* $s \approx s'$ *then* $N(s,\alpha,\rho,C) \;=\; N(s',\alpha,\rho,C)$ *for all* $\alpha \in Act \setminus \{\tau\}$, $\rho \in Distr_0(Act \setminus \{\tau\}$ *and* $C \in S/\approx$.

Proof: If $\rho(\alpha) = 0$ for all α then $N(s,\alpha,\rho,C) = N(s',\alpha,\rho,C) = 0$. Now we assume that $\rho \in Distr(Act \setminus \{\tau\})$. For $A \in S/\approx$, $A \neq S_{div}$, we choose some $s \in A$ with $\mathbf{P}(s,\tau,A) < 1$ and define

$$\mathbf{P}'(A,a,C) \;=\; \frac{\mathbf{P}(s,\alpha,C)}{1-\mathbf{P}(s,\tau,A)} \quad \text{for } (a,C) \neq (\tau,A),$$

$$\mathbf{P}'(A,\tau) \;=\; \sum_{\substack{C\in S/\approx \\ C\neq A}} \mathbf{P}'(A,\tau,C), \qquad \mathbf{P}'(A,\alpha) \;=\; \sum_{C\in S/\approx} \mathbf{P}'(A,\alpha,C).$$

We define $\mathbf{P}'(S_{div},\tau) = 0$ and $\mathbf{P}'(S_{div},a,C) = 0$ if $(a,C) \neq (\tau,S_{div})$. For all $A \in S/\approx$ we define:

$$norm(A,\rho) \;=\; \sum_{\alpha\in Act\setminus\{\tau\}} \mathbf{P}'(A,\alpha)\cdot\rho(\alpha) \;+\; \mathbf{P}'(A,\tau).$$

First we show that

(1) $norm(A,\rho) = 0$ implies $Prob(s,\tau^*\alpha) = 0$ for all $s \in A$ and $\alpha \in Act \setminus \{\tau\}$ with $\rho(\alpha) > 0$.

Let $norm(A,\rho) = 0$ and $\alpha \in Act \setminus \{\tau\}$ with $\rho(\alpha) > 0$. Then:

- $\mathbf{P}'(A,\alpha,C) = 0$ for all $C \in S/\approx$,
- $\mathbf{P}'(A,\tau,C) = 0$ for all $C \in S/\approx$, $C \neq A$.

Hence, $\mathbf{P}(s,\alpha) = 0$ and $\mathbf{P}(s,\tau,S\setminus A) = 0$ for all $s \in A$. In particular, $Prob(s,\tau^*,S\setminus A) = 0$ for all $s \in A$. This yields $Prob(s,\tau^*\alpha) = 0$.

For A, $C \in S/\approx$, $\alpha \in Act\setminus\{\tau\}$ and $\rho \in Distr_0(Act\setminus\{\tau\})$, we define $N(A,\alpha,\rho,C)$ as follows. The vector $(N(A,\alpha,\rho,C))_{A\in S/\approx}$ is the unique solution of the linear equation system:

1. If $norm(A,\rho) = 0$ then $N(A,\alpha,\rho,C) = 0$.
2. If $norm(A,\rho) > 0$ then

$$N(A,\alpha,\rho,C) \;=\; \frac{\rho(\alpha)}{norm(A,\rho)}\cdot\mathbf{P}'(A,\alpha,C) + \sum_{\substack{B\in S/\approx \\ B\neq A}} \frac{\mathbf{P}'(A,\tau,t)}{norm(A,\rho)}\cdot N(B,\alpha,\rho,C).$$

In what follows, we suppose α, ρ and C to be fixed. It suffices to show that

(*) $N(s,\alpha,\rho,C) = N(A,\alpha,\rho,C)$ for all $s \in A$ and $A \in S/\approx$.

For all $s \in S$ we define $x_s = N([s],\alpha,\rho,C)$. (Recall that $[s]$ denotes the weak bisimulation equivalence class of s.) Clearly, (*) holds if $norm(A,\rho) = 0$. Now we assume $A \in S/\approx$ and $norm(A,\rho) > 0$. (In particular, $A \neq S_{div}$.) Then:

(2) If $s \in A$ with $\mathbf{P}(s,\tau,A) < 1$ then

$$norm(s,\rho) - \mathbf{P}(s,\tau,A) = norm(A,\rho)\cdot(1-\mathbf{P}(s,\tau,A)).$$

(3) If $s \in A$ with $\mathbf{P}(s,\tau,A) < 1$ then $norm(s,\rho) = \mathbf{P}(s,\tau,A)$ iff $norm(A,\rho) = 0$.

First we assume that $norm(A,\rho) = 0$. By (1), $Prob(s,\tau^*\alpha) = 0$ for all $s \in A$. Thus, by the definition of $N(\cdot)$, $N(s,\alpha,\rho,C) = 0 = N(A,\alpha,\rho,C) = x_s$.

Next we assume that $norm(A,\rho) > 0$. It is easy to see that, if $Prob(A,\tau^*\alpha,C) = 0$ then $x_s = 0 = N(A,\alpha,\rho,C)$ for all $s \in A$. In what follows, we suppose $Prob(A,\tau^*\alpha,C) > 0$. By (2) and (3),

- $norm(s,\rho) > 0$ for all $s \in A$,
- $norm(s,\rho) > \mathbf{P}(s,\tau,A)$ for all $s \in A$ with $\mathbf{P}(s,\tau,A) < 1$.

Let $s \in A$ with $\mathbf{P}(s,\tau,A) < 1$. Then:

$$
\begin{aligned}
x_s &= N(A,\alpha,\rho,C) \\
&= \frac{\rho(\alpha)}{norm(A,\rho)} \cdot \sum_{t\in C} \frac{\mathbf{P}(s,\alpha,t)}{1-\mathbf{P}(s,\tau,A)} \;+\; \frac{1}{norm(A,\rho)} \cdot \sum_{t\in S\setminus A} \frac{\mathbf{P}(s,\tau,t)}{1-\mathbf{P}(s,\tau,A)} \cdot x_t \\
&= \rho(\alpha) \cdot \frac{1-\mathbf{P}(s,\tau,A)}{norm(s,\rho)-\mathbf{P}(s,\tau,A)} \cdot \sum_{t\in C} \frac{\mathbf{P}(s,\alpha,C)}{1-\mathbf{P}(s,\tau,A)} \\
&\quad + \frac{1-\mathbf{P}(s,\tau,A)}{norm(s,\rho)-\mathbf{P}(s,\tau,A)} \cdot \sum_{t\in S\setminus A} \frac{\mathbf{P}(s,\tau,t)}{1-\mathbf{P}(s,\tau,A)} \cdot x_t \\
&= \frac{\rho(\alpha)}{norm(s,\rho)-\mathbf{P}(s,\tau,A)} \cdot \mathbf{P}(s,\alpha,C) \\
&\quad + \frac{1}{norm(s,\rho)-\mathbf{P}(s,\tau,A)} \cdot \sum_{t\in S} \mathbf{P}(s,\tau,t) \cdot x_t \;-\; \frac{\mathbf{P}(s,\tau,A)}{norm(s,\rho)-\mathbf{P}(s,\tau,A)} \cdot x_s
\end{aligned}
$$

Thus,

$$
\begin{aligned}
x_s \cdot \frac{norm(s,\rho)}{norm(s,\rho)-\mathbf{P}(s,\tau,A)} &= x_s \cdot \left(1 + \frac{\mathbf{P}(s,\tau,A)}{norm(s,\rho)-\mathbf{P}(s,\tau,A)}\right) \\
= \frac{1}{norm(s,\rho)-\mathbf{P}(s,\tau,A)} &\cdot \left(\rho(\alpha)\cdot\mathbf{P}(s,\alpha,C) \;+\; \sum_{t\in S}\mathbf{P}(s,\tau,t)\cdot x_t\right).
\end{aligned}
$$

Hence,

$$
x_s \;=\; \frac{\rho(\alpha)}{norm(s,\rho)} \cdot \mathbf{P}(s,\alpha,C) \;+\; \frac{1}{norm(s,\rho)} \cdot \sum_{t\in S} \mathbf{P}(s,\tau,t)\cdot x_t.
$$

This yields $x_s = N(s,\alpha,\rho,C)$ for all $s \in A$. ■

Theorem A.27 $\approx$ *is finer than* $\equiv_0$.

Proof: follows by Lemma A.26 and induction on the length of the probabilistic traces. ■

References

[1] S. Abramsky, A. Jung: Domain Theory, In S. Abramsky, D.M. Gabbay and T.S.E. Maibaum (ed.), *Handbook of Logic in Computer Science*, Vol. 3, Clarendon Press, pp 1-168, 1994.

[2] A. Aho, J. Hopcroft, J. Ullman: The Design and Analysis of of Computer Algorithms, Addison-Wesley Publishing Company, 1974.

[3] S. Andova. Process algebra with probabilistic choice. In J.-P. Katoen, editor, *Formal Methods for Real-Time and Probabilistic Systems*, LNCS 1601, pages 111–129, 1999.

[4] A. Aziz, V. Singhal, F. Balarin, R. Brayton, A. Sangiovanni-Vincentelli: It usually works: The Temporal Logic of Stochastic Systems, Proc. CAV'95, LNCS 939, pp 155-165, 1995.

[5] J. Baeten, J. Bergstra, S. Smolka: Axiomatizing Probabilistic Processes: ACP with Generative Probabilities, Information and Computation, Vol. 122, pp 234-255, 1995.

[6] C. Baier: Polynomial Time Algorithms for Testing Probabilistic Bisimulation and Simulation, Proc. CAV'96, LNCS 1102, pp 38-49, 1996.

[7] C. Baier, H. Hermanns: Weak Bisimulation for Fully Probabilistic Processes, Proc. CAV'97, LNCS 1254, pp. 119-130, 1997.

[8] C. Baier: On algorithmic verification methods for probabilistic systems. Habilitation thesis, Univ. Mannheim, 1998.

[9] C. Baier, M. Stoelinga: Probabilistic Bisimulation and Simulation: Decidability, Complexity and Compositionality, in preparation.

[10] J. Bergstra, J. Klop: Process Algebra for Synchronous Communication, Information and Computation, Vol. 60, pp 109-137, 1984.

[11] G. Berry, G. Gonthier: The ESTEREL Synchronous Programming Language: Design, Semantics, Implementation, Science of Computer Programming, Vol. 19, 1992.

[12] A. Bianco, L. de Alfaro: Model Checking of Probabilistic and Nondeterministic Systems, Proc. FST&TCS, LNCS 1026, pp 499-513, 1995.

[13] S. Campos: A Quantitative Approach to the Formal Verification of Real-Time Systems, Ph.D.Thesis, Carnegie Mellon University, 1996.

[14] G. Chehaivbar, H. Garavel, N. Tawbi, F. Zulian: Specification and Verification of the Powerscale Bus Arbitration Protocol: An Industrial Experiment with LOTOS, Formal Description Techniques IX, pp 435-450, Chapmann&Hall, 1996.

[15] I. Christoff: Testing Equivalences and Fully Abstract Models for Probabilistic Processes: Proc. CONCUR'90, LNCS 458, pp 126-140, 1990.

[16] I. Christoff: Testing Equivalences for Probabilistic Processes, Ph. D. Thesis, Department of Computer Science, Uppsala University, 1990.

[17] L. Christoff, I. Christoff: Efficient Algorithms for Verification of Equivalences for Probabilistic Processes, Proc. CAV'91, LNCS 575, pp 310-321, 1991.

[18] E. Clarke, O. Grumberg, and D. Long: Model Checking and Abstraction, ACM Transactions on Programming Languages and Systems, Vol. 16, pp 1512-1542, 1994.

[19] R. Cleaveland, S. Smolka, and A. Zwarico: Testing Preorders for Probabilistic Processes, Proc. ICALP 1992, LNCS 623, pp 708-719, 1992.

[20] D. Coppersmith, S. Winograd: Matrix Multiplication via Arithmetic Progressions, Proc. 19th ACM Symposium on Theory of Computing, pp 1-6, 1987.

[21] C. Courcoubetis, M. Yannakakis: Verifying Temporal Properties of Finite-State Probabilistic Programs, Proc. FOCS'88, pp 338-345, 1988.

[22] C. Courcoubetis, M. Yannakakis: The Complexity of Probabilistic Verification, Journal of the ACM, Vol. 42, No. 4, pp 857-907, 1995.

[23] P.R. D'Argenio, H. Hermanns, and J.-P. Katoen. On asynchronous generative parallel composition. Proc. PROBMIV 98, Electronic Notes in Theoretical Computer Science, Vol. 21, 1999.

[24] M. Fang, C. Ho-Stuart, H. Zedan: Specification of Real-Time Probabilistic Behaviour, Proc. Protocol, Specification, Testing and Verification, IFIP, Elsevier Science Publishers, pp 143-157, 1993.

[25] A. Giacalone, C. Jou, S. Smolka: Algebraic Reasoning for Probabilistic Concurrent Systems, Proc. IFIP TC2 Working Conference on Programming Concepts and Methods, 1990.

[26] R.J. van Glabbeek: The Linear Time – Branching Time Spectrum II: The semantics of sequential systems with silent moves (Extended Abstract). Proc. CONCUR '93, LNCS 715, pp 66–81. Springer, 1993.

[27] R.J. van Glabbeek, S. Smolka, B. Steffen: Reactive, Generative, and Stratified Models for Probabilistic processes, Information and Computation, Vol. 121, pp 59-80, 1995.

[28] R.J. van Glabbeek, W. Weijland: Branching Time and Abstraction in Bisimulation Semantics, Journal of the ACM 43(3), pp 555-600,1996.

[29] C. Gregorio-Rodriguez, L. Llana-Diaz, M. Núñez, and P. Paloa-Gostanza. In M. Bertran and T. Rus, editors, *Transformation-Based Reactive Systems Development*, LNCS 1231, pages 353–367. Springer-Verlag, 1997.

[30] J.F. Groote, F. Vaandrager: An Efficient Algorithm for Branching Bisimulation and Stuttering Equivalence, Proc. ICALP'90, LNCS 443, pp 626-638, 1990.

[31] N. Halbwachs, F. Lagnier and C. Ratel. Programming and verifying critical systems by means of the synchronous data-flow programming language Lustre, IEEE Transactions on Software Engineering, 1992.

[32] H. Hansson: Time and Probability in Formal Design of Distributed Systems, Ph.D.Thesis, Uppsala University, 1991.

[33] H. Hansson, B. Jonsson: A Logic for Reasoning about Time and Probability, Formal Aspects of Computing, Vol. 6, pp 512-535, 1994.

[34] S. Hart, M. Sharir: Probabilistic Temporal Logic for Finite and Bounded Models, Proc. 16th ACM Symposium on Theory of Computing, pp 1-13, 1984.

[35] J. Hartog, E. de Vink: Mixing up Nondeterminism and Probability: a Preliminary Report, Proc. PROBMIV'98, Electronic Notes in Theoretical Computer Science, Vol. 21, 1999.

[36] V. Hartonas-Garmhausen: Probabilistic Symbolic Model Checking with Engineering Models and Applications, Ph.D.Thesis, Carnegie Mellon University, 1998.

[37] J. He, A. McIver, K. Seidel: Probabilistic Models for the Guarded Command Language, *Science of Computer Programming*, Vol. 28, No. 2/3, pp 171-192, 1997.

[38] M. Hennessy: Algebraic Theory of Processes, MIT Press, Boston, Mass., 1988.

[39] C.A.R. Hoare: Communicating Sequential Processes, Prentice Hall, 1985.

[40] ISO. *LOTOS : A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour*, Standard IS 8807, TC97/SC21, 1989.

[41] B. Jonsson, K.G. Larsen: Specification and Refinement of Probabilistic Processes, Proc. LICS'91, pp 266-277, 1991.

[42] B. Jonsson, W. Yi: Compositional Testing Preorders for Probabilistic Processes, Proc. LICS'95, pp 431-443, 1995.

[43] C.C. Jou, S. Smolka, Equivalences, Congruences and Complete Axiomatizations for Probabilistic Processes, Proc. CONCUR'90, LNCS 458, pp 367-383, 1990.

[44] B. Jonsson, W. Yi: Compositional Testing Preorders for Probabilistic Processes, Proc. LICS'95, pp 431-443, 1995.

[45] P. Kanellakis, S. Smolka: CCS Expressions, Finite State Processes, and Three Problems of Equivalence, Information and Computation, Vol. 86, pp 43-68, 1990.

[46] J.G. Kemeny and J.L. Snell. *Finite Markov Chains*. Springer, 1976.

[47] M. Kwiatkowska, G. Norman: A Fully Abstract Metric-Space Denotational Semantics for Reactive Probabilistic Processes, Proc. COMPROX'98, *Electronic Notes in Theoretical Computer Science*, Vol. 13, 1998.

[48] K.G. Larsen, A. Skou: Bisimulation through Probabilistic Testing, Information and Computation, Vol. 94, pp 1-28, 1991.

[49] K.G. Larsen, A. Skou: Compositional Verification of Probabilistic Processes, Proc. CONCUR'92, LNCS 630, pp 456-471, 1992.

[50] G. Lowe: Probabilities and Priorities in Timed CSP, Ph.D.Thesis, Oxford University, 1993.

[51] R. Milner: A Calculus of Communicating Systems, LNCS 92, 1980.

[52] R. Milner: Calculi for Synchrony and Asynchrony, Theoretical Computer Science, Vol. 25, pp 269-310, 1983.

[53] R. Milner: Communication and Concurrency, Prentice Hall, 1989.

[54] C. Morgan, A. McIver, K. Seidel, J. Sanders: Refinement-Oriented Probability for *CSP*, Techn. Report TR-12-94, Oxford University, to appear in *Formal Aspects of Computing*.

[55] R. de Nicola, M. Hennessy: Testing Equivalences for Processes, Theoretical Computer Science, Vol. 34, pp 83-133, 1983.

[56] M. Núñez, D. de Frutos: Testing Semantics for Probabilistic LOTOS, Proc. Formal Description Techniques VIII, pp 365-380, Chapman&Hall, 1995.

[57] R. Paige, R. Tarjan: Three Partition Refinement Algorithms, SIAM Journal of Computing, Vol. 16, No. 6, pp 973-989, 1987.

[58] D. Park: Concurrency and Automata on Infinite Sequences, Proc. 5th GI Conference, LNCS 104, pp 167-183, 1981.

[59] A. Philippou, O. Sokolsky: Weak Bisimulation for Probabilistic Processes, 1998.

[60] A. Pnueli, L. Zuck: Verification of Multiprocess Probabilistic Protocols, Distributed Computing, Vol. 1, No. 1, pp 53-72, 1986.

[61] A. Pnueli, L. Zuck: Probabilistic Verification, Information and Computation, Vol. 103, pp 1-29, 1993.

[62] G.D. Plotkin. A Structured Approach to Operational Semantics. Technical Report DAIMI FM-19, Computer Science Department, Aarhus University, 1981.

[63] R. Segala: Modeling and Verification of Randomized Distributed Real-Time Systems, Ph.D.Thesis, Massachusetts Institute of Technology, 1995.

[64] R. Segala, N. Lynch: Probabilistic Simulations for Probabilistic Processes, Proc. CONCUR'94, LNCS 836, pp 481-496, 1994.

[65] R. Segala, N. Lynch: Probabilistic Simulations for Probabilistic Processes, Nordic Journal of Computing, Vol. 2 (2), pp 250-273, 1995.

[66] K. Seidel: Probabilistic Communicating Processes, Theoretical Computer Science, Vol. 152, pp 219-249, 1995.

[67] R. Sisto, L. Ciminiera, A. Valenzo: Probabilistic Characterization of Algebraic Protocol Specifications, Proc. 12th International Conference on Distributed Computing Systems, pp 260-268, IEEE Comp. Soc. Press, 1992.

[68] S. Smolka, B. Steffen: Priority as Extremal Probability, Proc. CONCUR'90, LNCS 458, pp 456-466, 1990.

[69] S. Smolka, B. Steffen: Priority as Extremal Probability, Formal Aspects of Computing, Vol. 8, pp 585-606, 1996.

[70] C. Tofts: Processes with Probabilities, Priorities, and Time, Formal Aspects of Computing, Vol. 6, No. 5, 1994.

[71] M. Vardi: Automatic Verification of Probabilistic Concurrent Finite-State Programs, Proc. FOCS'85, pp 327-338, 1985.

[72] M. Vardi, P. Wolper: An Automata-Theoretic Approach to Automatic Program Verification, Proc. LICS'86, pp 332-344, 1986.

[73] S. Yuen, R. Cleaveland, Z. Dayar, S. Smolka: Fully Abstract Characterizations of Testing Preorders for probabilistic Processes, Proc. CONCUR 94, LNCS 836, pp 497-512, 1994.

[74] W. Yi, K.G. Larsen: Testing Probabilistic and Nondeterministic Processes, Proc. Protocol, Specification, Testing, Verification XII, pp 47-61, 1992.