Norm Functions for Probabilistic Bisimulations with Delays

CHRISTEL BAIER[1], MARIËLLE STOELINGA[2]

[a]Institut für Informatik I, Universität Bonn
Römerstraße 164, D-53117 Bonn, Germany
baier@cs.uni-bonn.de

[2]Computing Science Institute, University of Nijmegen
P.O.Box 9010, 6500 GL Nijmegen,The Netherlands
marielle@cs.kun.nl

Abstract. In this paper, we consider action-labelled systems with non-deterministic and probabilistic choice. Using the concept of norm functions [GV98], we introduce two types of bisimulations that allow for delays when simulating a transition. The so obtained equivalences (called *(strict) normed bisimulation equivalence*) are strictly between strong and weak bisimulation equivalence à la [LS89, SL94, SL95]. Using a suitable modification of the prominent splitter/partitioning technique [KS83, PT87], we present polynomial-time algorithms that constructs the quotient space of the (strict) normed bisimulation equivalence classes. Moreover, we briefly discuss other aspects such as the soundness for establishing linear time properties and compositiality.

1 Introduction

Probabilistic aspects play a crucial role for a quantitative analysis of various types of parallel systems, such as systems that are designed on the basis of a randomized algorithms or computer systems with unreliable components. In the former case, probabilities can be used to specify the frequencies of the possible outcomes of an explicit probabilistic choice ("tossing a fair coin"); in the latter case, probabilities might express failure rates. In the literature, several extensions of labelled transition systems (LTSs) are proposed to reason about probabilistic phenomena. On the one hand, there is a wide range of models that are based on discrete-time Markov chains (we will refer to them as *fully probabilistic systems*) which allow for probabilistic (but not for non-deterministic) choice and can serve as operational models for processes of a calculus with synchronous parallel composition operator [GJS90, vGSST90] or probabilistic merge operators [BBS92, dAHK98]. Other models are based on Markov decision processes (MDPs) [Put94] that allow for both probabilistic and non-deterministic branching. These can be used for modelling probabilistic systems with asynchronous parallelism [Var85, HJ90, Han91, Seg95a, BK97] where the non-determinism is used to describe the *interleaving* of the subprocesses. Moreover, as observed by several authors [JHY94, JY95, Seg95a], the non-determinism can also be used to represent *under-specification* (that will be partly or totally resolved in further refinement steps) or *incomplete information* about the environment.

Due to the combination of non-determinism and probabilism, the design and analysis of such systems (with both types of choices) can be hard. Like for any kind of computer systems, the use of implementation relations (which compare two systems; thus yielding a formal definition of when a program $\mathcal{P}$ implements correctly another one $\mathcal{Q}$) have turned out to be useful for the design and the system analysis. In this paper, we shrink our attention to the equivalences that yield a notion of *process equality*. There are several highly desirable conditions that any reasonable process equivalence $\approx$ should fulfill, including e.g. the soundness for establishing quantitativ linear time properties and congruence properties w.r.t. certain composition operators of a process calculus (such as parallel composition) For mechanised purposes, the development of methods that support the proof of the equivalence of two processes (i.e. deductive or algorithmic techniques to show $\mathcal{P} \approx \mathcal{P}'$) is a further crucial aspect. In particular, the algorithmic methods are of great importance for automatic verification tools that take as their input a system $\mathcal{P}$ and its specification $\mathcal{P}'$ and returns the answer "yes" or "no" depending on whether or not $\mathcal{P}$ correctly implements $\mathcal{P}'$. Moreover, algorithms for computing the quotient space yield an abstraction technique which is highly relevant for the system analysis. For this, one replaces the states by their equivalence classes and then establishes the desired properties for the quotient space $S/\approx$ rather than the original state space S. Especially when we deal with *weak equivalences* (that abstract from internal computations) the switch from the original system S to the quotient space $S/\approx$ might lead to a much smaller equivalent system; and hence can be viewed as a technique to combat the state explosion problem.

Several (strong and weak) equivalences for various types of probabilistic systems have been proposed in the literature. They range over the full linear and branching time spectrum and are extensions of the corresponding relations on LTSs. While in the fully probabilistic setting, the equivalences are studied under several aspects (compositionality, axiomatization, decidability, logical characterizations, etc.), see e.g. [JS90, CC91, JL91, HT92, LS92, Ch93, BH97], the treatment of equivalences for probabilistic systems with non-determinism is less well-understood. Most of the standard relations that have proven to be useful in the non-probabilistic setting have been extended for the probabilistic case; see e.g. [Seg95b] for a trace-based relation, [YL92, JY95] for testing equivalences and [LS89, HJ90, Han91, SL94, Yi94, SL95, Seg95b, SV99, St99] for several types of (bi-)simulations. However, due to the combination of non-determinism and probabilism, the definitions are more complicate than the corresponding notions for non-probabilistic or fully probabilistic systems. Even though some important issues (like compositionality and axiomatization) have been addressed in the above mentioned literature, research on algorithmic methods to decide the equivalence of two systems or to compute the quotient space are rare. For strong bisimulation [LS89] and strong simulation [SL94], polynomial-time algorithms have been presented in [BEM99]. To the best of our knowledge, the forthcoming work [PSL99] is the first attempt to formulate an algorithmic method that deals with a weak equivalence for probabilistic processes with non-determinism. We are not aware of any complexity (or even decidability) result for weak bisimulation à la [SL94, SL95] or any linear time relation on probabilistic systems with non-determinism, e.g. trace distribution equivalence [Seg95b].[1]

Our contribution: We deal with probabilistic systems with non-determinism and action labels modelled by a probabilistic extension of LTSs where the (action-labelled) transitions are augmented with probabilities for the possible target states. Our model essentially agrees with the *simple probabilistic automata* of [SL94, Seg95a]). Our main contribution is the presentation of novel notions of bisimulation equivalence which (in some sense) are insensitive with respect to internal transitions. More precisely, our equivalences are conservative extensions of *delay bisimulation* equivalence [Wei89, vGl93] which relies on the assumption that the simulation of a step of a process $\mathcal{P}$ by another process $\mathcal{P}'$ might happen with a certain delay (i.e. after a sequence of internal transitions). The formal definition of our equivalences is provided by a probabilistic variant of *norm functions* in the style of [GV98]. Intuitively, the norm functions specify bounds for the delays (i.e. the number of internal transitions that might be performed before a "proper" transition of a process $\mathcal{P}$ is simulated by a corresponding transition of an equivalent process $\mathcal{P}'$). In the probabilistic setting where the combination of internal transitions leads to a tree rather than a linear chain, the

[1] As (non-probabilistic) LTSs are special instances of probabilistic systems with non-determinism and the trace distribution preorder à la Segala is a conservative extension of usual trace containment, the PSPACE-complete for non-probabilistic systems [KS83] yields the PSPACE-hardness for the trace distribution relation à la [Seg95b].

norm functions yield conditions on the length of the paths in the trees corresponding to a "delayed transition". Using a modification of the traditional splitter/partioning technique [KS83, PT87], we present polynomial time algorithms for computing the quotient spaces. Moreover, we briefly discuss some other aspects (compositionality w.r.t. parallel composition and preservation of linear time properties).

Organization of the paper: Section 2 introduces our model for probabilistic labelled transition systems and explains related notions. The definitions of norm functions and normed bisimulations are presented in Section 3. In Section 4, we present our algorithm for computing the bisimulation equivalence classes. Section 5 concludes the paper.

Because of space restrictions, we present our main results without proofs. We refer the interested reader to the technical report [BS99] where the proofs and other details (including results about various types of bisimulations and simulations) can be found.

2 Probabilistic labelled transition systems

In (ordinary) LTSs, the transitions $s \xrightarrow{a} t$ specify the possibility that the system in state s moves via the action a to state t. In this paper, we deal with a probabilistic variant of LTSs where any transition is augmented with a probabilistic choice for the possible target states (rather than a unique target state t as it is the case in LTSs). That is, in the probabilistic setting, the transitions are of the form $s \xrightarrow{a} \mu$ where s is the starting state, a an action label and μ a distribution on the state space which specifies the probabilities $\mu(t)$ for any possible successor state t. Non-determinism is present in our model since we allow several (possibly equally action-labelled) outgoing transitions of a state s.

Notation 1. Let S be a finite set. A *distribution* on S is a function $\mu : S \to [0,1]$ such that $\sum_{s \in S} \mu(s) = 1$. $Supp(\mu)$ denotes the *support* of μ, i.e. the set of elements $s \in S$ with $\mu(s) > 0$. If $\emptyset \neq A \subseteq S$ then $\mu[A] = \sum_{s \in A} \mu(s)$. Moreover, we put $\mu[\emptyset] = 0$. For $s \in S$, μ_s^1 denotes the unique distribution on S with $\mu_s^1(s) = 1$. $Distr(S)$ denotes the collection of all distributions on S. If R is an equivalence relation on S then we write S/R to denote quotient space (i.e. the set of equivalence classes) of S with respect to R. The induced equivalence $\equiv_R$ on $Distr(S)$ is given by $\mu \equiv_R \mu'$ iff $\mu[A] = \mu'[A]$ for all $A \in S/R$. We write $[\mu]_R$ to denote the equivalence class $\{\mu' : \mu \equiv_R \mu\}$ of μ with respect to $\equiv_R$. With abuse of notations, we often write $[s]_R$ to denote the equivalence class of an element $s \in S$ with respect to R as well as to denote the equivalence class $[\mu_s^1]_R = \{\mu' \in Distr(S) : Supp(\mu) \subseteq [s]_R\}$ of μ_s^1 with respect to the induced equivalence $\equiv_R$ on $Distr(S)$. ■

Definition 1. A *probabilistic labelled transition system* (PLTS) is a tuple $(S, Act, \longrightarrow)$ where S is a finite set of states, Act a finite set of actions and $\longrightarrow \subseteq S \times Act \times Distr(S)$ a transition relation such that for any state s, the set $Steps(s) = \{(a, \mu) : s \xrightarrow{a} \mu\}$ is finite.[2] We define $Steps_a(s) = \{\mu : s \xrightarrow{a} \mu\}$. A state s is called *terminal* iff there is no possible next step in s, i.e. if $Steps(s) = \emptyset$. A *probabilistic program* is a tuple $\mathcal{P} = (S, Act, \longrightarrow, s_{init})$ consisting of a PLTS $(S, Act, \longrightarrow)$ and an initial state $s_{init} \in S$. ■

In what follows, we assume that Act contains a special symbol τ that denotes any internal (invisible) activity. We refer to τ as the *internal* action. All other actions are called *visible*.

We depict PLTSs as follows. We use circles for the states. Thick lines stand for the outgoing transitions from a state. The thick line corresponding to a transition $s \xrightarrow{a} \mu$ is directed and ends in a small circle that represents the probabilistic choice. For transitions of the form $s \xrightarrow{a} \mu_t^1$ we often write $s \xrightarrow{a} t$ and represent them in the pictures by an arrow from state s to state t.

[2] Any finite (non-probabilistic) LTS $(S, Act, \longrightarrow)$ (where $\longrightarrow \subseteq S \times Act \times S$) can be viewed as a PLTS. For this, we identify any transition $s \xrightarrow{a} t$ with its probabilistic counterpart $s \xrightarrow{a} \mu_t^1$.

Example 1. We consider a simple communication protocol consisting of a sender (that produces certain messages and tries to submit the messages along an unreliable medium) and a receiver (that acknowledges the receipt and consumes the received messages). The failure rate of the medium is 1%; more presisely, with probability 1/100 the medium looses the messages in which case the sender retries to submit the message.[3] Figure 1 shows a PLTS for the protocol where we use the following four states. In state s_{init}, the sender produces a message and passes the message to the medium which leads to the state s_{del}: where the medium tries to deliver the message (via an internal action). When the message is delivered correctly, the state s_{ok} is reached. In state s_{ok}, the sender and the receiver can work in parallel (simultaneously): the sender may produce the next message while the receiver may consume the last message. [4] ■

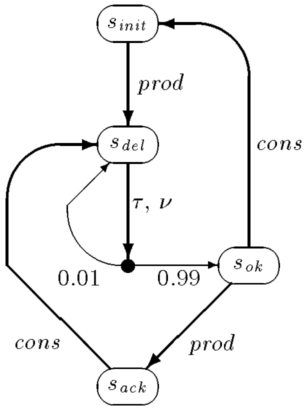


Fig. 1. The simple communication protocol

The executions of a PLTS are given by the paths in the underlying directed graph. They arise through the resolution of both the non-deterministic and probabilistic choices. Formally, a *path* is a finite or infinite alternating sequence $\gamma = s_0 a_1 \mu_1 s_1 a_2 \mu_2 s_2 \ldots a_l \mu_l s_l \ldots$ where $l \geq 0$, $s_0, s_1, \ldots \in S$, $s_{i-1} \xrightarrow{a_i} \mu_i$, $s_i \in Supp(\mu_i)$. For a finite path, we require that it ends up in a terminal state.

Typically, one assumes that the resolution of the non-deterministic choices are not under the control of the system itself. The entity that resolves the non-determinism (the "environment") can be formalized by a *scheduler* [Var85] (also called *adversary* [Seg95a] or *policy* in the theory of MDPs [Put94]). Given a scheduler A, the system behaviour under A can be

[3] For simplicity, we assume that both the sender and the receiver work with mailing boxes that cannot hold more than one message at any time. Thus, if the sender has produced a message m then the next message cannot be produced before m is delivered correctly; similarly, the medium cannot be activiated as long as there is an unread message in the mailing box of the receiver (i.e. as long as the acknowledgement for the last message is not yet arrived).

[4] The parallelism in state s_{ok} is described by interleaving, i.e. the non-deterministic choice that decides which process performs the next step: either the sender produces the next message or the receiver consumes the last message (in which case the system loops back to the initial state s_{init}). In the former case (where we move from s_{ok} to s_{ack}) the sender has already produced the next message while there is still an unread message in the mailing box of the receiver. Thus, the only possible step in s_{ack} is the one where the receiver consumes the message and acknowledges the receipt.

described by a (possibly infinite) Markov chain which yields a Borel field and probability measure on the paths that can be obtained by A. The details are not of importance for this paper and are omitted here. They can be found e.g. in the above mentioned references.

3 Normed bisimulation

In ordinary LTSs, the several types of bisimulations (e.g. strong, weak branching or delay bisimulation [Mil80, Par81, Mil89, vGW89, Wei89, vGl93]) establish a correspondence between the states and their stepwise behaviour. Intuitively, they identify those states s and s' where any outgoing transition from s can be simulated by s' and vice versa. Most types of bisimulation equvalences on a LTS $(S, Act, \longrightarrow)$ can be characterized as the coarsest equivalence R on the state space S such that

(Bis) If $(s,s') \in R$, $C \in S/R$ and $s \xrightarrow{a} C$ then $s' \in Pre^*(a,C)$.

Here, we write $s \xrightarrow{a} C$ if $s \xrightarrow{a} t$ for some $t \in C$. $Pre^*(a,C)$ denotes a certain *predecessor predicate*. Intuitively, $s' \in Pre^*(a,C)$ asserts that s' can "simulate" the transition $s \xrightarrow{a} C$. The formal definition of $Pre^*(a,C)$ depends on the concrete type of equivalence. E.g., *strong bisimulation* is obtained by using the predicate $Pre^{str}(a,C) = \{s' : s' \xrightarrow{a} C\}$ while *delay bisimulation* equivalence [Wei89, vGl93] focusses on the idea that the simulation of a transition $s \xrightarrow{a} t$ might happen with a certain delay (i.e. after a finite number of internal moves) and uses the predicates $Pre^{del}(\cdot)$ which can be characterized as the least subsets of S satisfying the following three conditions.[5]

(D0) $C \subseteq Pre^{del}(\tau, C)$

(D1) If $s \xrightarrow{a} C$ then $s \in Pre^{del}(a,C)$.

(D2) If $s \xrightarrow{\tau} t$ and $t \in Pre^{del}(a,C)$ then $s \in Pre^{del}(a,C)$.

[LS89] presented an elegant reformulation of strong bisimulation for a variant of PLTSs which takes the probabilistic effect of the transitions into account. Formally, strong bisimulation equivalence $\approx_{sbis}$ in a PLTSs is the coarsest equivalence R on the state space S such that for all $(s,s') \in R$ and transitions $s \xrightarrow{a} \mu$ there is a transition $s' \xrightarrow{a} \mu'$ where μ and μ' return the same probabilities for all equivalence classes under R (i.e. $\mu \equiv_R \mu'$, cf. Notation 1). [SL94, SL95] presented notions of weak and branching bisimulations for PLTSs. All these notions of bisimulation equivalences on a PLTS $(S, Act, \longrightarrow)$ can be characterized as the coarsest equivalence R on S such that

(PBis) If $(s,s') \in R$, $M \in Distr(S)/\equiv_R$ and $s \xrightarrow{a} M$ then $s' \in Pre^*(a,M)$.

Here, $s \xrightarrow{a} M$ iff $s \xrightarrow{a} \mu$ for some $\mu \in M$. E.g., strong bisimulation equivalence is given by (PBis) using the predecessor predicate $Pre^{str}(a,M) = \{s' : s' \xrightarrow{a} M\}$.

In this section, we propose novel notions of bisimulation equivalence for PLTSs which are conservative extensions of delay bisimulation equivalence [Wei89, vGl93]. Intuitively, two states s, s' are identified iff any transition $s \xrightarrow{a} \mu$ can be simulated by s' by first performing finitely many internal moves and then performing an a-labelled transition for which the outcome of the associated probabilistic choice agrees with μ. Thus, we aim at an appropriate definition of the predecessor predicate $Pre^{del}(a,M)$ where $s' \in Pre^{del}(a,M)$ states the possibility for s' to perform the action a (possibly with a certain delay) such that the associated distribution μ' of the a-labelled transition belongs to $M = [\mu]_R$.[6] Conditions

[5] Thus, the delay predecessor predicate is given by $Pre^{del}(a,C) = \{s' : s' \stackrel{\tau^* a}{\longrightarrow} C\}$ for any visible action a and $Pre^{del}(\tau,C) = \{s' : s' \stackrel{\tau^*}{\longrightarrow} C\}$. Here, we assume the obvious definition of the transition relation $\longrightarrow \subseteq S \times (Act \setminus \{\tau\})^* \times 2^S$ and use standard notations for regular expressions.

[6] This informal explanation assumes that $s \xrightarrow{a} \mu$ is a "proper" transition, i.e. either $a \neq \tau$ or $\mu_s^1 \notin M$. Transitions of the form $s \xrightarrow{\tau} \mu$ where all possible target states $t \in Supp(\mu)$ are equivalent to s can be viewed as "silent moves" and are not taken account when dealing with equivalences that abstract from internal computations.

(D0) and (D1) for $Pre^{del}(a, C)$ can easily be lifted to the probabilistic case (see conditions (BD0) and (BD1) below).

(BD0) If $\mu_s^1 \in M$ then $s \in Pre^{str}(\tau, M)$.

(BD1) If $s \xrightarrow{a} M$ then $s \in Pre^{str}(a, M)$.

To adapt condition (D2) for the probabilistic setting, we have two possibilities depending on whether or not we allow for unbounded delays. For the simpler case, we require *bounded delays* which leads to condition (BD2).

(BD2) If $s \xrightarrow{\tau} \nu$ and $Supp(\nu) \subseteq Pre^{str}(a, M)$ then $s \in Pre^{str}(a, M)$.

The resulting bisimulation equivalence only abstracts from the combination of finitely many internal moves (corresponding to a bounded delay) but cannot involve the effect of infinite τ-paths (*unbounded delays*). In the simple communication protocol of Example 1 (Figure 1), one might argue that the states s_{del} and s_{ok} have the same observable behaviour as s_{del} moves via τ-transitions to s_{ok} with probability 1. To formalize the effect of infinite τ-loops, we use the concept of *norm functions* which was introduced in [GV98] to reason about simulation-like relations in non-probabilistic systems. We slightly depart from the notations of [GV98] and define norm functions in LTSs as partial functions with three arguments (a state s, an action label a and a set C of target states) and whose range are the natural numbers. If the value $norm(s, a, C)$ is defined then $s \in Pre^{del}(a, C)$ in which case there is a τ^*-labelled path of length $\leq norm(s, a, C)$ from s to a state t where either $t \xrightarrow{a} C$ or $a = \tau$ and $t \in C$. If $s \notin Pre^{del}(a, C)$ then $norm(s, a, C)$ is undefined (denoted $norm(s, a, C) = \bot$). The formal definition that characterize a norm function for a LTS arise by "refining" the above mentioned three conditions for $Pre^{del}(a, C)$ in the sense that we involve the length of a delayed transition. Formally, norm functions in LTSs are partial functions satisfying the following three conditions.

(N0) $norm(s, a, C) = 0$ implies $a = \tau$ and $s \in C$

(N1) $norm(s, a, C) = 1$ implies $s \xrightarrow{a} C$

(N2) If $norm(s, a, C) \geq 2$ then there is a transition $s \xrightarrow{\tau} t$ where $norm(t, a, C) < norm(s, a, C)$.

To adapt these three conditions to the probabilistic setting, we deal with a set $M \subseteq Distr(S)$ as the third argument of a norm function. The modifications of (N0) and (N1) are straightforward. In (N2) we require that $norm(s, a, M) \geq 2$ implies the existence of a transition $s \xrightarrow{\tau} \nu$ satisfying a certain condition. When we aim at bounded delays then we deal with the constraint $norm(t, a, M) < norm(s, a, M)$ for *all* $t \in Supp(\nu)$. To reason about unbounded delays, we require that $norm(t, a, M)$ is defined for all $t \in Supp(\nu)$ and that $norm(t, a, M) < norm(s, a, M)$ for *some* $t \in Supp(\nu)$.[7]

Definition 2. A *norm function* for a PLTS $(S, Act, \longrightarrow)$ is a partial function $norm : S \times Act \times 2^{Distr(S)} \rightarrow I\!N$ which satisfies the following conditions.

(PN0) $norm(s, a, M) = 0$ implies $a = \tau$ and $\mu_s^1 \in M$.

(PN1) $norm(s, a, M) = 1$ implies $s \xrightarrow{a} M$ (i.e. $s \xrightarrow{a} \mu$ for some $\mu \in M$).

(PN2) If $norm(s, a, M) \geq 2$ then there is a transition $s \xrightarrow{\tau} \nu$ where

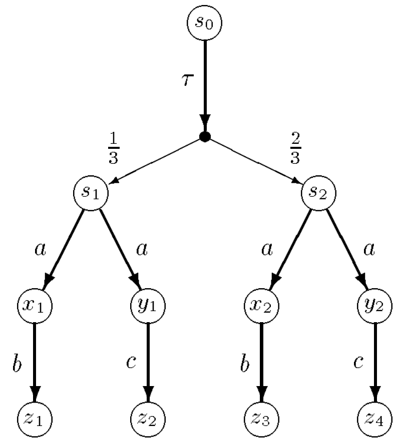
  (i) $norm(t, a, M) \neq \bot$ for all $t \in Supp(\nu)$

  (ii) $norm(t, a, M) < norm(s, a, M)$ for some $t \in Supp(\nu)$

$norm$ is *strict* iff, in (ii), $norm(t, a, M) < norm(s, a, M)$ for all $t \in Supp(\nu)$. ■

[7] These two conditions about $Supp(\nu)$ guarantee the existence of a scheduler where, for any state s for which $norm(s, a, M)$ is defined, almost all paths starting in s lead via τ's to a state t where $norm(t, a, M) \in \{0, 1\}$. Thus, in this scheduler, with probability 1, s performs finitely many τ's followed by a transitions $t \xrightarrow{a} \mu'$ where $\mu' \in M$. However, for this scheduler, there might be no upper bound for the number of τ's that will be performed before the action a.

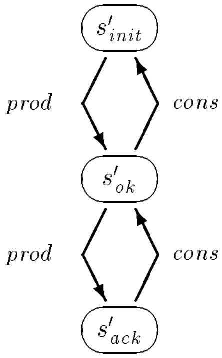
Example 3. Consider the system on the right. Let M be the set of distributions μ that return probability 1 for the x-states (i.e. $M = \{\mu : \mu(x_1) + \mu(x_2) = 1\}$). Then, $\mu^1_{x_1}$, $\mu^1_{x_2} \in M$ and $s_1 \xrightarrow{a} M$, $s_2 \xrightarrow{a} M$. Thus, the partial function *norm* with $norm(s_0, a, M) = 2$, $norm(s_1, a, M) = norm(s_2, a, M) = 1$ and $norm(\cdot) = \bot$ in all other cases is a strict norm function. ■



We now define (strict) normed bisimulation equivalence as the coarsest equivalence R on the state space S such that for all equivalent states s and s' and proper transitions $s \xrightarrow{a} \mu$ the existence of a matching delayed transition from s' is ensured by a (strict) norm function.

Definition 4. Let $(S, Act, \longrightarrow)$ be a PLTS and R an equivalence on S. R is called a *(strict) normed bisimulation* iff there exists a (strict) norm function *norm* such that for all $a \in Act$ and $M \in Distr(S)/\equiv_R$: if $(s, s') \in R$ and $s \xrightarrow{a} M$ then $norm(s', a, M) \neq \bot$. Two states s and s' are called *(strictly) normed bisimilar* (denoted $s \approx_n s'$ resp. $s \approx_{sn} s'$) iff there exists a (strict) normed bisimulation R such that $(s, s') \in R$. The equivalences $\approx_n$ and $\approx_{sn}$ are adapted for probabilistic programs in the obvious way.[8] ■

Example 5. It is easy to see that the states s_0, s_1 and s_2 in Example 3 are strictly normed bisimilar. For the simple communication protocol of Figure 1 and the smallest equivalence relation R that identifies s_{del} and s_{ok}, there is a norm function with $norm(s_{ok}, \tau, [\nu]_R,) = 0$ and $norm(s_{del}, cons, [s_{init}]_R) = 2$ but no strict norm function. Thus, $s_{ok} \approx_n s_{del}$ but $s_{ok} \not\approx_{sn} s_{del}$. The quotient system that we get when we identify the states by their normed bisimulation equivalence classes can be viewed as a failure-free specification (see the picture on the right). ■



$\approx_n$ and $\approx_{sn}$ can be characterized by condition (PBis) with suitable defined predecessor predicates. The unbounded delay predecessor predicate $Pre^{\omega}(a, M)$ is the set of states s where $norm(s, a, M) \neq \bot$ for some norm function *norm*. The bounded predecessor predicate $Pre^{str}(a, M)$ is the set of states s such that $norm(s, a, M) \neq \bot$ for some strict norm function *norm*.[9] Then, $Pre^{str}(a, M)$ is the least set satisfying the three conditions (BD0), (BD1), (BD2).

In what follws, we simply write $Pre^{del}(a, M)$ to denote $Pre^{str}(a, M)$ or $Pre^{\omega}(a, M)$ depending on whether we deal with strict normed bisimulation or normed bisimulation equivalence. It is easy to see that (strict) normed bisimulation equivalence meets the general

[8] Recall that a probabilistic program is a PLTS with an initial state (Def. 1). Let $\mathcal{P}_i$, $i = 1, 2$, be probabilistic programs with initial states s_1 and s_2 respectively. We define $\mathcal{P}_1 \approx_* \mathcal{P}_2$ iff $s_1 \approx_* s_2$ where s_1 and s_2 are viewed as states in the composed system $\mathcal{P}_1 \uplus \mathcal{P}_2$ which arises from the disjoint union of the state spaces of $\mathcal{P}_1$ and $\mathcal{P}_2$.

[9] It should be noticed that for a LTS, viewed as a PLTS, the unbounded and bounded predecessor predicates coincide. More precisely, $Pre^{del}(a, C) = Pre^{\omega}(a, M_C) = Pre^{str}(a, M_C)$ for any set C of states and $M_C = \{\mu^1_t : t \in C\}$.

charactization of bisimulation equivalences in PLTSs via condition (PBis). More precisely, (strict) normed bisimulation equivalence is the coarsest equivalence R on S such that

$$\text{If } (s,s') \in R,\ M \in Distr(S)/\equiv_R \text{ and } s \xrightarrow{a} M \text{ then } s' \in Pre^{del}(a,M).$$

(Strict) normed bisimulation equivalence lies strictly between strong ($\approx_{sbis}$) and weak ($\approx_{wbis}$) bisimulation equivalence à la [LS89, SL95], i.e. $\approx_{sbis} \subset \approx_{sn} \subset \approx_n \subset \approx_{wbis}$.[10] The simple communication protocol and its failure free specification are examples that demonstrate the difference between strict and (non-strict) normed bisimulation equivalence.

Without presenting the details, we briefly sketch how trace-based properties can be defined in the probabilistic setting and in which sense normed bisimilarity (and hence, also strict normed bisimilarity) is *sound for establishing linear time properties*. In the non-probabilistic case, the linear time behaviour of a concurrent system (described by a LTS) can be formalized by means of the *traces* (sequences of visible actions). Linear time properties can be specified by a set T of traces which represents the "allowed" behaviours. Thus, a non-probabilistic program $\mathcal{P}$ satisfies the property specified by T iff all its paths yield a trace of T. This ensures, that independent on the resolution of the non-deterministic choices (even in a worst case scenario), the program behaviour meets the specification. In the probabilistic setting, we deal with *quantitativ linear time properties* that we formalize by pairs $\langle T, p\rangle$ consisting of a certain set T of traces together with a lower bound p for the acceptable probabilities. A probabilistic program $\mathcal{P}$ satisfies $\langle T, p\rangle$ (written $\mathcal{P} \models \langle T, p\rangle$) iff the probabilities for the given linear time properties T are "sufficiently large" (at least p), independent on how the non-determinism is resolved.[11] For an example, the communication protocol of Example 1 satisfies the property $\langle T, 1\rangle$ where T is the set of all infinite traces where both actions *prod* and *cons* occur infinitely often. This asserts that, under all schedulers, in almost all execution paths, infinitely many messages are produced and consumed. It can be shown that, if $\mathcal{P}$ and $\mathcal{P}'$ are divergence free probabilistic programs with the same action set Act and $\mathcal{P} \approx_n \mathcal{P}'$ then $\mathcal{P}$ and $\mathcal{P}'$ satisfy exactly the same quantitativ linear time properties. (Divergence freedom means that, under all schedulers, the probability for the paths where almost all actions are τ's is 0.)

We briefly discuss the *compositionality* of normed bisimulation equivalence with respect to parallel composition with *CSP*-style communication [Seg95a].[12] Given two probabilistic programs $\mathcal{P}_1$ and $\mathcal{P}_2$ with state spaces S_1 and S_2 respectively and the same action set Act, the global states in the parallel composition $\mathcal{P}_1 \| \mathcal{P}_2$ are pairs $s_1 \| s_2$ consisting of local states s_i of each of the components $\mathcal{P}_i$. We assume a fixed set $Comm \subseteq Act \setminus \{\tau\}$ of *communication actions* on which the two probabilistic programs have to synchronize. The probabilistic choices associated with the synchronous execution of the transitions $s_1 \xrightarrow{c} \mu_1$ and $s_2 \xrightarrow{c} \mu_2$ (where $c \in Comm$) of $\mathcal{P}_1$ and $\mathcal{P}_2$ are resolved independently. Thus, in the global states $s_1 \| s_2$, the parallel composition $\mathcal{P}_1 \| \mathcal{P}_2$ has the transition $s_1 \| s_2 \xrightarrow{c} \mu_1 \| \mu_2$ where the distribution $\mu_1 \| \mu_2$ is given by $\mu_1 \| \mu_2 (s_1 \| s_2) = \mu_1(s_1) \cdot \mu_2(s_2)$. Transitions $s_i \xrightarrow{a} \mu_i$ with other action labels (a visible action $a \in Act \setminus Comm$ or the internal action $a = \tau$)

[10] In [SL94], the "strict" variant of weak bisimulation (that we denote by $\approx_{swbis}$) is introduced while the journal version [SL95] treats the variant where the effect of infinite τ-paths is involved (that we denote by $\approx_{wbis}$). Then, $\approx_{sn}$ is finer than $\approx_{swbis}$ while $\approx_{swbis}$ and $\approx_n$ are incomparable.

[11] The formal definition of the satisfaction relation $\models$ requires that, for any scheduler A for $\mathcal{P}$ the probability measure for the paths in A where the induced trace belongs to T is at least p. This requires an additional assumption about T that ensures the measurability of the corresponding set of paths.

[12] It can be shown that also other composition operators (such as prefixing, restriction, hiding or parallelism with *CCS*-style communication on complementary actions) preserve (strict) normed bisimulation equivalence. However, we cannot expect a compositionality result with respect to the standard *CCS*-like non-deterministic choice operator $+$ since already for non-probabilistic programs, weak bisimulation equivalence is not preserved by $+$ [Mil89].

are autonomous moves of the components $\mathcal{P}_i$. They are represented by *interleaving*, i.e. the active component $\mathcal{P}_i$ performs the action a and changes its local state according to the distribution μ_i while the local state of the other component does not change. Formally, $\mathcal{P}_1 \| \mathcal{P}_2 = (S_1 \times S_2, Act, \longrightarrow, s^1_{init} \| s^2_{init})$ where s^i_{init} is the initial state of $\mathcal{P}_i$ and where the transition relation $\longrightarrow \subseteq (S_1 \times S_2) \times Act \times (S_1 \times S_2)$ is given by the following two rules

$$\frac{s_1 \xrightarrow{a} \mu_1,\ a \in Act \setminus Comm}{s_1 \| s_2 \xrightarrow{a} \mu_1 \| \mu^1_{s_2}} \qquad \frac{s_1 \xrightarrow{c} \mu_1,\ s_2 \xrightarrow{c} \mu_2,\ c \in Comm}{s_1 \| s_2 \xrightarrow{c} \mu_1 \| \mu_2}$$

and the symmetric rule for the first one. Let R_i be (strict) normed bisimulations for the underlying PLTS of $\mathcal{P}_i$ with (strict) norm functions $norm_i$ $i = 1, 2$. We regard the partial function $norm$ which given by the following two conditions.[13] If $a \in Act \setminus Comm$ then

$$norm(s_1 \| s_2, a, M_1 \| M_2) \ = \ \begin{cases} norm_1(s_1, a, M_1) : \text{if } \mu^1_{s_2} \in M_2 \\ norm_2(s_2, a, M_2) : \text{if } \mu^1_{s_1} \in M_1 \end{cases}$$

and $norm(s_1 \| s_2, c, M_1 \| M_2) = norm_1(s_1, c, M_1) + norm_2(s_2, c, M_2) - 1$ for $c \in Comm$. Then, $norm$ is a (strict) norm function for the underlying PLTS of $\mathcal{P}_1 \| \mathcal{P}_2$ which turns $R = \{(s_1 \| s_2, s'_1 \| s'_2) : (s_i, s'_i) \in R_i\}$ into a (strict) normed bisimulation. Thus:

Proposition 6. *$\mathcal{P}_1 \approx_n \mathcal{P}'_1$, $\mathcal{P}_2 \approx_n \mathcal{P}'_2$ implies $\mathcal{P}_1 \| \mathcal{P}_2 \approx_n \mathcal{P}'_1 \| \mathcal{P}'_2$ and the corresponding result for $\approx_{sn}$.*

4 Decidability

In this section, we present an algorithm that computes the (strict) normed bisimulation equivalence classes in polynomial time and space. The main idea of our algorithm is a modification of the prominent splitter/partitioning technique [KS83, PT87] (which is sketched in Figure 2) that was proposed for computing the strong bisimulation equivalence classes in a non-probabilistic transition system. The basic idea is to start with the trivial partition $\mathcal{X} = \{S\}$ of the state space S and then successively refine $\mathcal{X}$ by splitting the blocks B of $\mathcal{X}$ into subblocks according to a refinement operator $Refine(\chi, a, C)$ that depends on a *splitter*, i.e. an action/block pair $\langle a, C \rangle$. More precisely, $Refine(\mathcal{X}, a, C)$ devides each block $B \in \mathcal{X}$ into the subblocks $B \cap Pre^{str}(a, C)$ (the set of B-states that are a-predecessors of C) and its complement $B \setminus Pre^{str}(a, C)$. [14] Using an appropriate organization of the splitters (resp. splitter candidates), this method can be implemented in time $\mathcal{O}(m \log n)$ where n is the number of states and m the number of transitions (i.e. the size of $\longrightarrow$) [PT87]. The above

$\mathcal{X} := \{S\}$;
While $\mathcal{X}$ can be refined do
    choose some splitter $\langle a, C \rangle$ of $\mathcal{X}$ and put $\mathcal{X} := Refine(\mathcal{X}, a, C)$;
Return $\mathcal{X}$.

Fig. 2. Schema for computing the bisimulation equivalence classes in LTSs

[13] We write $M_1 \| M_2$ for the set $\{\mu_1 \| \mu_2 : \mu_i \in M_i\}$ and assume that $norm(\ldots) = \bot$ in all cases that are not covered by the two conditions.

[14] $Refine(\mathcal{X}, a, C)$ yields the partition $\bigcup_{B \in \mathcal{X}} Refine(B, a, C)$ where $Refine(B, a, C) = \{B \cap Pre^{str}(a, C), B \setminus Pre^{str}(a, C)\} \setminus \{\emptyset\}$.

sketched splitter/partitioning technique can easily be modified to compute several other types of bisimulation equivalence classes, such as the strong [HT92] or weak [BH97] bisimulation equivalence classes in fully probabilistic systems, but fails for strong (and hence also for normed) bisimulation in PLTSs when action/block pairs are used as splitters [BEM99].

In the remainder of this section, we explain how the splitter/partitioning technique can be modified to get a polynomial-time algorithm for computing the (strict) normed bisimulation equivalence classes in a PLTS.

Notation 2. We fix a PLTS $(S, Act, \longrightarrow)$. Let $M_a = \bigcup_{s \in S} Steps_a(s)$. For Z to be a finite set, we write $|Z|$ to denote the number of elements in Z. Let $n = |S|$ the number of states, $m = |\longrightarrow|$ the total number of transitions and $m_\tau = |M_\tau|$ the number of τ-transitions. When we analyze the complexity of our algorithm we assume that Act does not contain redundant actions, i.e. we require that $M_a \neq \emptyset$ for all actions a. ■

We use similar ideas as suggested in [BEM99] where an algorithm for computing the strong bisimulation equivalence classes of a PLTS in time $\mathcal{O}(mn(\log m + \log n))$ is presented. The key idea is based on the abservation that the current state partition has be refined according to splitters of the form $\langle a, M \rangle$ where a is an action and M a subset of M_a. That is, we successively refine the current state partition χ according to the refinement operator

$$Refine(\chi, a, M) \;=\; \bigcup_{B \in \chi} \; Refine(B, a, M)$$

where $Refine(B, a, M) = \{B \cap Pre^{del}(a, M), B \setminus Pre^{del}(a, M)\} \setminus \{\emptyset\}$.

Notation 3. A *step partition* is a set $\mathcal{M}$ consisting of pairs $\langle a, M \rangle$ where $M \subseteq M_a$ and such that, for any action a, $\{M : \langle a, M \rangle \in \mathcal{M}\}$ is a partition of M_a. We refer to the pairs $\langle a, M \rangle$ as *step classes*. Given a state partition $\mathcal{X}$, the induced step partition $\mathcal{M}_\mathcal{X}$ consists of the step classes $\langle a, M \rangle$ where $M \in M_a / \equiv_\mathcal{X}$ and $\mu \equiv_\mathcal{X} \mu'$ iff $\mu[C] = \mu'[C]$ for all $C \in \mathcal{X}$. ■

The rough ideas behind our algorithm are sketched in Figure 3. To keep book about the splitter candidates $\langle a, M \rangle$ we use a step partition $\mathcal{M}$ (that agrees with $\mathcal{M}_\mathcal{X}$ after any iteration) and a set *Splitters* (e.g. organized as a queue) which contains the step classes that will serve as splitter candidates. Initially, *Splitters* consists of the "trivial" step classes $\langle a, M_a \rangle$. In each iteration, we first refine the state partition $\mathcal{X}$ according to a step class $\langle a, M \rangle \in$ *Splitters* which yields the new state partition $\mathcal{X}_{new} = Refine(\mathcal{X}, a, M)$. Then, we adjust $\mathcal{M}$ to $\mathcal{X}_{new}$, i.e. calculate $\mathcal{M}_{new} = \mathcal{M}_{\mathcal{X}_{new}}$. All new step classes $\langle b, N' \rangle \in \mathcal{M}_{new} \setminus \mathcal{M}$ are viewed as splitter candidates and are inserted into *Splitters*. To derive $\mathcal{M}_{new}$ from $\mathcal{M}$ we have to replace any step class $\langle b, N \rangle$ in $\mathcal{M}$ by the step classes $\langle b, N' \rangle$ where $N' \in N / \equiv_\mathcal{X}$. At the beginning of any iteration we have $\mathcal{M} = \mathcal{M}_\mathcal{X}$. Thus, for $\langle b, N \rangle \in \mathcal{M}$ and ν, $\nu' \in N$ we have $\nu[B] = \nu'[B]$ for all $B \in \mathcal{X}$. Let $B \in \mathcal{X}$ and $B' = B \cap Pre^{del}(a, M)$, $B'' = B \setminus B'$ and $C' \in \{B', B''\}$, $\langle b, N \rangle \in \mathcal{M}$ and ν, $\nu' \in N$. Then, $\nu[C'] = \nu'[C']$ iff $\nu[B'] = \nu'[B']$ and $\nu[B''] = \nu'[B'']$. These observations motivate the use of a set *NewBlocks* which contains only those blocks $C' \in \mathcal{X}_{new}$

$\mathcal{X} \;:=\; \{S\}$; While $\mathcal{X}$ can be refined do   choose some step class $\langle a, M \rangle$ of $\mathcal{M}_\mathcal{X}$ and put $\mathcal{X} := Refine(\mathcal{X}, a, M)$; Return $\mathcal{X}$.

Fig. 3. Schema for computing the bisimulation equivalence classes in PLTSs

that are relevant for the computation of $\mathcal{M}_{new}$. More precisely, for any block $B \in \mathcal{X}$ where $|Refine(B, a, M)| = 2$, we choose a block $C'_B \in Refine(B, a, M)$ such that $|C'_B| \leq |B|/2$ and define $NewBlocks = \{C'_B : B \in \mathcal{X}, |Refine(B, a, M)| = 2\}$. Then, $\mathcal{M}_{new}$ can be derived from $\mathcal{M}$ by replacing any $\langle b, N\rangle$ in $\mathcal{M}$ by the step classes in $Split(\langle b, N\rangle, NewBlocks)$ which we compute as follows.

$\mathcal{N} := \{\langle b, N\rangle\}$;
For all $C' \in NewBlocks$ do
    replace any $\langle b, N'\rangle$ in $\mathcal{N}$ by $Split(\langle b, N'\rangle, C')$;
$Split(\langle b, N\rangle, NewBlocks) := \mathcal{N}$.

The operator $Split(\langle b, N'\rangle, C')$ for a step class $\langle b, N'\rangle$ according to a block C' devides $\langle b, N'\rangle$ into the step classes $\langle b, N'_1\rangle, \ldots, \langle b, N'_r\rangle$ where $N'_1, \ldots, N'_r$ is the splitting of N' according to the probabilities for C'.[15]

These ideas lead to the algorithm sketched in Figure 4. We now discuss some details and

```
X := {S}; M := {⟨a, M_a⟩ : a ∈ Act}; Splitters := M;
While Splitters ≠ ∅ do
  (1) choose some step class ⟨a, M⟩ of Splitters and remove ⟨a, M⟩ from Splitters;
  (2) (* Computation of X_new := Refine(X, a, M) *)
        P := Pre^del(a, M);
        X_new := ∅; NewBlocks := ∅
        For all B ∈ X do
          B' := B ∩ P; B'' := B \ B';
          X_new := X_new ∪ {B', B''} \ {∅};
          If ∅ ≠ B' ≠ B then
            If |B'| ≤ |B''| then C' := B' else C' := B'';
            NewBlocks := NewBlocks ∪ {C'};
  (3) (* Computation of M_new := M_{X_new} *)
        M_new := ∅;
        For all ⟨b, N⟩ ∈ M do
          N := Split(⟨b, N⟩, NewBlocks);
          M_new := M_new ∪ N;
          If |N| ≥ 2 then Splitters := Splitters ∪ N;
  (4) X := X_new; M := M_new;
Return X.
```

Fig. 4. Algorithm for the (strict) normed bisimulation equivalence classes

the complexity of our algorithm where we shrink our attention to the time complexity and just observe that with appropriate data structures, our algorithm can be implemented in space $\mathcal{O}(mn)$. We first state the main theorem.

[15] $Split(\langle b, N\rangle, NewBlocks)$ returns the set of step classes $\langle b, N'\rangle$ where $N' \in N/\equiv$ and $\nu \equiv \nu'$ iff $\nu[C'] = \nu'[C']$ for all $C' \in NewBlocks$. As $N \in M_b/\equiv_{\mathcal{X}}$ yields that $\equiv$ and $\equiv_{\mathcal{X}_{new}}$ coincide, we get $\mathcal{M}_{\mathcal{X}_{new}} = \bigcup_{\langle b,N\rangle \in \mathcal{M}} Split(\langle b, N\rangle, NewBlocks)$ which yields the correctness of our method.

Theorem 7. *The (strict) normed bisimulation equivalence classes can be computed in time* $\mathcal{O}(mn(\log m + \log n) + m_\tau n^2)$ *and space* $\mathcal{O}(mn)$.

The remainder of this section is concerned with the proof of Theorem 7. It follows from Proposition 8 (which states that all refinement operations for the state partition $\mathcal{X}$ require $\mathcal{O}(m_\tau n^2)$ time) and Proposition 11 (which shows that $\mathcal{O}(mn(\log m + \log n))$ is an upper bound for the time complexity for the splitting operations of the step partition).

We put $\mathcal{X}_0 = \{S\}$ and write $\mathcal{X}_i$ to denote the state partition $\mathcal{X}$ after the i-th iteration. Similarly, we use the notations $\mathcal{M}_i$, $Splitters_i$ and $NewBlocks_i$ with the obvious meaning. Let $AllSplitters = \bigcup_{i \geq 0} Splitters$ the set of all step classes $\langle a, M \rangle$ that once serve as splitters for the state partition $\mathcal{X}$ and let $AllNewBlocks = \bigcup_i NewBlocks_i$ the set of all blocks C' that once are used in a splitting operation $Split(\cdot, C')$. Using set-theoretic arguments, we get:

(i) $|AllSplitters| \leq |\mathcal{M}_0 \cup \mathcal{M}_1 \cup \ldots| \leq 2(m-1)$

(ii) $|AllNewBlocks| \leq |\mathcal{X}_0 \cup \mathcal{X}_1 \cup \ldots| \leq 2(n-1)$

(iii) $\sum_{C' \in AllNewBlocks} |C'| \leq n \log n$.

Proposition 8. *The refinement operations* $Refine(\mathcal{X}, a, M)$ *in step (2) of the algorithm in Figure 4 can be implemented in time* $\mathcal{O}(m_\tau n^2)$ *(where we range over all iterations).*

Proof. Clearly, given the predecessor predicate $Pre^{del}(a, M)$ for a fixed step class $\langle a, M \rangle$, the refinement operator $Refine(\mathcal{X}, a, M)$ can be performed in time $\mathcal{O}(n)$ when appropriate data structures are used. Combing (i) and the following Lemmatas 9 and 10 we get the desired bound for the time complexity. ■

Lemma 9. $Pre^{str}(a, M)$ *can be computed in time* $\mathcal{O}(m_\tau n)$.

Proof. $Pre^{str}(a, M)$ is the least subset of S satisfying the conditions (BD0), (BD1) and (BD2). The standard iterative method for computing the least fixed point of a monotonic set-valued operator leads to the following method for computing $Pre^{str}(a, M)$. We consider the directed graph $G^{str}(a, M) = (V, E)$ with the vertex set V is $S \cup M_\tau$ where the edge set E $= \{(\nu, s) \in M_\tau \times S : s \xrightarrow{\tau} \nu\} \cup \{(s, \nu) \in S \times M_\tau : s \in Supp(\nu)\}$. We assume a representation of $G_b^{del}(a, M)$ by adjacency lists and write $E(\cdot)$ to denote the adjacency list of $(\cdot)$. We use the algorithm shown in Figure 5 to compute the set $Pre^{str}(a, M)$. For any distribution $\nu \in N$, we use a counter $c(\nu)$ for the number of states $t \in Supp(\nu)$ where the condition $t \in Pre^{str}(a, M)$ is not yet verified. N_0 collects all distributions ν where $c(\nu) = 0$, i.e. $Supp(\nu) \subseteq Pre^{str}(a, M)$. Hence, if $\nu_0 \in N_0$ and $s \xrightarrow{\tau} \nu_0$ then we may insert s into $Pre^{str}(a, M)$. It is clear that this method can be implemented in time $\mathcal{O}(m_\tau n)$. ■

Lemma 10. $Pre^{\omega}(a, M)$ *can be computed in time* $\mathcal{O}(m_\tau n)$.

Proof. To compute $Pre^{\omega}(a, M)$ we suggest a graph-theoretical method which is based on the following observation. $Pre^{\omega}(a, M)$ is the least subset of S which contains $Pre^{\leq 1}(a, M) = Pre^0(a, M) \cup Pre^1(a, M)$ (where $Pre^0(\tau, M) = \{s : \mu_s^1 \in M\}$, $Pre^0(a, M) = \emptyset$ if a is visible and $Pre^1(a, M) = \{s : s \xrightarrow{a} M\}$) and satisfies the following condition. Whenever $C \subseteq S$ and there is a function $C \to Distr(S)$, $s \mapsto \nu_s$, such that, for all $s \in C$:

- $s \xrightarrow{\tau} \nu_s$ and $Supp(\nu_s) \subseteq C \cup Pre^{\omega}(a, M)$
- there is a finite path $s = s_0 \overset{\tau,\nu_0}{\leadsto} s_1 \overset{\tau,\nu_1}{\leadsto} \ldots \overset{\tau,\nu_l}{\leadsto} s_l \overset{\tau,\nu_l}{\leadsto} t$ where $s_1, \ldots, s_l \in C$ and $\nu_i = \nu_{s_i}$, $i = 0, 1, \ldots, l$, and $t \in Pre^{\omega}(a, M)$

then $C \subseteq Pre^{\omega}(a, M)$. On the basis of this characterization, we compute $Pre^{\omega}(a, M)$ as follows. We start with $Pre^{\omega}(a, M) = Pre^{\leq 1}(a, M)$ Then, we successively add all states of a set C satisfying the above condition which can be reformulated by means of the strongly connected components (SCCs) in a certain directed graph. We consider the directed graph $G_{ub}^{del}(a, M) = (V, E)$ where the vertex set V is given by $V = \{(s, \nu) : \nu \in$

Compute the adjacency lists $E(\cdot)$ of the graph $G_b^{del}(a, M)$;

If $a = \tau$ then $Pre^{str}(a, M) := \{s : \mu_s^1 \in M\}$ else $Pre^{str}(a, M) := \emptyset$;

$N_0 := \emptyset$ and $c(\nu) := |Supp(\nu)|$ for all $\nu \in N$;

For all $s \in S$ where $s \xrightarrow{a} \mu$ for some $\mu \in M$ do:

  $Pre^{str}(a, M) := Pre^{str}(a, M) \cup \{s\}$;

  For all $\nu \in E(s)$ do

    $c(\nu) := c(\nu) - 1$;

    If $c(\nu) = 0$ then $N_0 := N_0 \cup \{\nu\}$;

While $N_0 \neq \emptyset$ do

  choose some $\nu_0 \in N_0$ and put $N_0 := N_0 \setminus \{\nu_0\}$;

  For all $s \in E(\nu_0) \setminus Pre^{str}(a, M)$ do

    $Pre^{str}(a, M) := Pre^{str}(a, M) \cup \{s\}$;

    For all $\nu \in E(s) \setminus N_0$ do

      $c(\nu) := c(\nu) - 1$;

      If $c(\nu) = 0$ then $N_0 := N_0 \cup \{\nu\}$;

Return $Pre^{str}(a, M)$.

Fig. 5. Algorithm for computing $Pre^{str}(a, M)$

$Steps_\tau(s), s \notin Pre^{\leq 1}(a, M)\} \cup Pre^{\leq 1}(a, M)$ and the edge set is $E = \{\langle (s, \nu), (s', \nu')\rangle : s \in Supp(\nu')\} \cup \{\langle u, (s', \nu')\rangle : u \in Pre^{\leq 1}(a, M) \cap Supp(\nu')\}$. First we compute the SCCs of $G = G_{ub}^{del}(a, M)$ and a topological sorting $C_1, \ldots, C_r$ on them.[16] The singleton sets $\{u\}$ where $u \in Pre^{\leq 1}(a, M)$ are bottom SCCs (BSCCs) in G. Thus, we may assume that there is some h such that $Pre^{\leq 1}(a, M) = C_1 \cup \ldots \cup C_h$ and $C_i \subseteq V \setminus Pre^{\leq 1}(a, M)$, $i = h+1, \ldots, r$. We start with $Pre^{\omega}(a, M) = Pre^{\leq 1}(a, M)$. For $i = h+1, \ldots, r$, if C_i is not a BSCC (i.e. $\{j : C_j \rightarrow_G C_i\} \neq \emptyset$) and all states of a predecessor SCC C_j of C_i belong to $Pre^{\omega}(a, M)$ then we insert the states of C_i into $Pre^{\omega}(a, M)$. Clearly, $G = G_{ub}^{del}(a, M)$ has $\mathcal{O}(m_\tau + n)$ vertices and $\mathcal{O}(m_\tau n)$ edges. Hence, this method can be implemented in time and space $\mathcal{O}(m_\tau n)$. ■

Proposition 11. *Ranging over all iterations, the computation of the step partitions $\mathcal{M}_{new}$ in step (3) of Figure 4 takes $\mathcal{O}(mn(\log m + \log n))$ time.*

Proof. For calculating $Split(\langle b, N'\rangle, C')$ we may apply a technique (similar to the one suggested in [BEM99]) which generates an ordered balanced tree (e.g. AVL tree) by successively inserting the values $\nu[C']$, $\nu \in N'$, possibly creating new nodes and performing the necessary rebalancing steps. Any node v in this tree is labelled by a key value $v.key$ (which is one of the probabilities $\nu[C']$ for one or more $\nu \in N'$) and a subset $v.distr$ of N'. Then, $Split(\langle b, N'\rangle, C')$ is the set of pairs $\langle b, v.distr\rangle$ where v is a node in the final tree. The construction of the tree causes the cost $\mathcal{O}(|N'| \log |N'|)$ as for any $\nu \in N'$ we traverse a tree of height $\leq \log |N'|$. For fixed ν and C', the computation of the values $\nu[C']$ can be implemented in time $\mathcal{O}(|C'|)$. Thus, for any call of the procedure $Split(\langle b, N'\rangle, C')$ the time spended for computing the values $\nu[C']$ is $\mathcal{O}(|N'| \cdot |C'|)$ where we range over all $\nu \in N'$. Summing up

[16] This means an ordering $C_1, \ldots, C_r$ of the SCCs of G such that $C_i \rightarrow_G C_j$ implies $i \geq j$. Here, $\rightarrow_G$ denotes the induced edge relation on the SCCs, i.e. $C_i \rightarrow_G C_j$ iff there is a edge from a vertex in C_i to a vertex in C_j.

over all step classes $\langle b, N\rangle$ in the current step partition $\mathcal{M}$, any $C' \in \mathit{AllNewBlocks}$ causes the cost $\mathcal{O}(m \log m + m|C'|)$. (ii) and (iii) yield the time complexity $\mathcal{O}(mn(\log m + \log n))$ for all $\mathit{Split}(\cdot)$ operations together. ■

5 Conclusion

We introduced two notions of bisimulation equivalence in probabilistic systems (with non-determinism) that abstract from internal computations. We presented polynomial-time algorithms that compute the quotient spaces and briefly discussed other important issues (soundness for establishing linear time properties and compositionality). Thus, our notion of bisimulation equivalence yields an alternative to the weak and branching bisimulations introduced by Segala & Lynch [SL94, SL95]. Even though the equivalences à la [SL94, SL95] are the natural probabilistic counterpart to weak/branching bisimulation equivalence in non-probabilistic systems [Mil80, Par81, vGW89], their definitions are rather complicate and the decidability is still an open problem. We argue that the definitions of our equivalences – which rely on the rather intuitive concept of norm functions à la [GV98] – are comparatively simple. Moreover, the use of norm functions in the definition of our equivalences allows for a characterization of the equivalence classes by means of graph-theoretical criteria which served as basis for our algorithm that computes the equivalence classes. In particular, the characterization of the delay predecessor predicates that we used in the proofs of Lemmatas 9 and 10 can easily be rewritten as terms of the relational mu-calculus. It would be interesting if our ideas can be combined with the techniques of [BCM$^+$90, EFT93] for computing the bisimulation equivalence in LTSs with a BDD-based model checking algorithm for the relational mu-calculus seems to get a symbolic technique that might combat the state explosion problem for PLTSs.

In this paper (where we mainly treated the issue of decidability) we shrinked our attention to finite systems. However, norm functions and the derived notions of bisimulations can also be defined for infinite systems.[17] We believe that, as in the non-probabilistic case, in many applications, it is quite simple to "guess" a norm function and then to check (e.g. by hand) whether it fulfills the necessary conditions. Further on, the concept of norm functions can also serve as basis for simulation preorders that abstracts from internal moves and is computable in finite systems. Further details about normed simulations can be found in [BS99] and the forthcoming work [St99].

Acknowledgements: The authors would like to thank Frits Vaandrager and Holger Hermanns for many helpful comments.

References

[dAHK98] P. d'Argenio, H. Hermanns, J. Katoen: On Generative Parallel Composition, Proc. PROBMIV'98, ENTCS, Vol. 21, 1999.

[BBS92] J. Baeten, J. Bergstra, S. Smolka: Axiomatizing Probabilistic Processes: ACP with Generative Probabilities, Proc. CONCUR'92, LNCS 630, pp 472-485, 1992. The full version has appeared in *Information and Computation*, Vol. 122, pp 234-255, 1995.

[BEM99] C. Baier, B. Engelen, M. Majster-Cederbaum: Deciding Bisimilarity and Similarity for Probabilistic Systems, to appear in *Journal of Computer and System Sciences*, 1999. A preliminary version by the first author with the title "Polynomial Time Algorithms for

[17] For our purposes, it was sufficient to consider the natural numbers as range of the norm functions. The framework of [GV98] also covers infinite, possibly uncountable, state spaces and allows for arbitrary well-founded sets as range of norm functions. We argue that these ideas can also be used to handle PLTSs of arbitrary size.

Testing Probabilistic Bisimulation and Simulation" can be found in Proc. CAV'96, LNCS 1102, pp 38-49, 1996.

[BH97] C. Baier, H. Hermanns: Weak Bisimulation for Fully Probabilistic Processes, Proc. CAV'97, LNCS 1254, pp 119-130, 1997.

[BK97] C. Baier, M. Kwiatkowska: Domain Equations for Probabilistic Processes, Proc. EXPRESS'97, ENTCS, Vol. 7 1997. The full version will appear in *Mathematical Structures in Computer Science*, 1999.

[BS99] C. Baier, M. Stoelinga: Probabilistic Bisimulation and Simulation: Decidability, Complexity and Compositionality, Techn. Report, University Nijmegen, 1999.

[BS87] T. Bolognesi, S. Smolka: Fundamental Results for the Verification of Observational Equivalence: a Survey, Proc. Protocol Specification, Testing and Verification, Elsevier Science Publishers, IFIP, pp 165-179, 1987.

[BCM+90] J. Burch, E. Clarke, K. McMillan, D. Dill, L. Hwang: Symbolic Model Checking: 10^{20} States and Beyond, Proc. LICS'90, pp 428-439, 1990. The full version with the same title has appeared in *Information and Computation*, Vol. 98 (2), pp 142-170, 1992.

[Ch93] L. Christoff: Specification and Verification Methods for Probabilistic Processes, Ph. D. Thesis, Uppsala University, 1993.

[CC91] L. Christoff, I. Christoff: Efficient Algorithms for Verification of Equivalences for Probabilistic Processes, Proc. CAV'91, LNCS 575, pp 310-321,1991.

[CLR96] T. Cormen, C. Leiserson, R. Rivest: Introduction to Algorithms, McGraw Hill, 1996.

[EFT93] R. Enders, T. Filkorn, D. Taubner: Generating BDDs for Symbolic Model checking in CCS, *Distributed Computing*, Vol. 6, pp 155-164, 1993.

[GJS90] A. Giacalone, C. Jou, S. Smolka: Algebraic Reasoning for Probabilistic Concurrent Systems, Proc. IFIP TC2 Working Conference on Programming Concepts and Methods, 1990.

[vGl93] R. 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, 1993.

[vGSST90] R. van Glabbeek, S. Smolka, B. Steffen, C. Tofts: Reactive, Generative, and Stratified Models for Probabilistic Processes, Proc. LICS'90, pp 130-141, 1990. A revised version with the same title by the authors R. van Glabbeek, S. Smolka, B. Steffen has appeared in *Information and Computation*, Vol. 121, pp 59-80, 1995.

[vGW89] R. van Glabbeek, W. Weijland: Branching Time and Abstraction in Bisimulation Semantics, *Information Processing*, Vol. 89, pp 613-618, 1989. The full version with the same title has appeared in *Journal of the ACM*, Vol. 43(3), pp 555-600,1996.

[GV98] D. Griffioen, F. Vaandrager: Normed Simulations, Proc. CAV'98, LNCS 1427, pp 332-344, 1998.

[Han91] H. Hansson: Time and Probability in Formal Design of Distributed Systems, Ph.D.Thesis, Uppsala University, 1991; published in *Real-Time Safety Ciritical Systems*, Vol. 1, Elsevier, 1994.

[HJ90] H. Hansson, B. Jonsson: A Calculus for Communicating Systems with Time and Probabilities, Proc. IEEE Real-Time Systems Symposium, IEEE Computer Society Press, pp 278-287, 1990.

[HT92] T. Huynh, L. Tian: On some Equivalence Relations for Probabilistic Processes, Fundamenta Informaticae, Vol. 17, pp 211-234, 1992.

[JHY94] B. Jonsson, C. Ho-Stuart, W. Yi: Testing and Refinement for Nondeterministic and Probabilistic Processes, Proc. FTRTFT'94, LNCS 863, pp 418-430, 1994.

[JL91] B. Jonsson, K.G. Larsen: Specification and Refinement of Probabilistic Processes, Proc. LICS'91, pp 266-277, 1991.

[JY95] B. Jonsson, W. Yi: Compositional Testing Preorders for Probabilistic Processes, Proc. LICS'95, pp 431-443, 1995.

[JS90] C. Jou, S. Smolka: Equivalences, Congruences and Complete Axiomatizations for Probabilistic Processes, Proc. CONCUR'90, LNCS 458, pp 367-383, 1990.

[KS83] P. Kannelakis, S. Smolka: CCS Expressions, Finite State Processes and Three Problems of Equivalence, Proc. 2nd ACM Symposium on the Principles of Distributed Computing, pp 228-240, 1983. The full version has appeared in *Information and Computation*, Vol. 86, pp 43-68, 1990.

[LS89] K. Larsen, A. Skou: Bisimulation through Probabilistic Testing, Proc. POPL'89, 1989. The full version with the same title has appeared in *Information and Computation*, Vol. 94, pp 1-28, 1991.

[LS92] K. Larsen, A. Skou: Compositional Verification of Probabilistic Processes, Proc. CONCUR'92, LNCS 630, pp 456-471, 1992.

[Mil80] R. Milner: A Calculus of Communicating Systems, LNCS 92, 1980.

[Mil89] R. Milner: Communication and Concurrency, Prentice Hall, 1989.

[PT87] R. Paige, R. Tarjan: Three Partition Refinement Algorithms, *SIAM Journal of Computing*, Vol. 16, No. 6, pp 973-989, 1987.

[Par81] D. Park: Concurrency and Automata on Infinite Sequences, Proc. 5th GI Conference, LNCS 104, pp 167-183, 1981.

[PSL99] A. Philippou, O. Sokolsky, I. Lee: Weak Bisimulation for Probabilistic Systems, submitted for publication.

[Put94] M. Puterman: Markov Decision Processes, John Wiley and Sons, 1994.

[Seg95a] R. Segala: Modeling and Verification of Randomized Distributed Real-Time Systems, Ph.D.Thesis, Massachusetts Institute of Technology, 1995.

[Seg95b] R. Segala: A Compositional Trace-Based Semantics for Probabilistic Automata, Proc. CONCUR'95, LNCS 962, pp 234-248, 1995.

[SL94] R. Segala, N. Lynch: Probabilistic Simulations for Probabilistic Processes, Proc. CONCUR'94, LNCS 836, pp 481-496, 1994.

[SL95] R. Segala, N. Lynch: Probabilistic Simulations for Probabilistic Processes, *Nordic Journal of Computing*, Vol. 2 (2), pp 250-273, 1995.

[SV99] M. Stoelinga, F. Vaandrager: Root Contention in IEEE 1394, Proc. ARTS'99, LNCS 1601, pp 53-74, 1999.

[St99] M. Stoelinga: Hypernorms for Probabilistic Systems, in preparation, 1999.

[Wei89] W. Weijland: Synchrony and Asynchrony in Process Algebra, Ph.D.Thesis, Univesity of Amsterdam, 1989.

[Var85] M. Vardi: Automatic Verification of Probabilistic Concurrent Finite-State Programs, Proc. FOCS'85, pp 327-338, 1985.

[YL92] W. Yi, K. Larsen: Testing Probabilistic and Nondeterministic Processes, Proc. Protocol, Specification, Testing, Verification XII, pp 47-61, 1992.

[Yi94] W. Yi: Algebraic Reasoning for Real-Time Probabilistic Processes with Uncertain Information, Proc. FTRTFT'94, LNCS 863, pp 680-693, 1994.