Partial Order Reduction on Concurrent Probabilistic Programs*

Pedro R. D'Argenio† Peter Niebert

Laboratoire d'Informatique Fondamentale, CMI, Université de Provence,
39, Rue Joliot Curie, 13453 Marseille Cedex 13, France
E-mail: dargenio@cmi.univ-mrs.fr, Peter.Niebert@lif.univ-mrs.fr

Abstract

Partial order reduction has been used to alleviate the state explosion problem in model checkers for nondeterministic systems. The method relies on exploring only a fragment of the full state space of a program that is enough to assess the validity of a property. In this paper, we discuss partial order reduction for probabilistic programs represented as Markov decision processes. The technique we propose preserves probabilistic quantification of reachability properties and is based on partial order reduction techniques for (non probabilistic) branching *temporal logic. We also show that techniques for (non probabilistic) linear temporal logics are not correct for probabilistic reachability and that in turn our method is not sufficient for probabilistic CTL. We conjecture that our reduction technique also preserves maximum and minimum probabilities of next-free LTL properties.*

1. Introduction

Partial order techniques have been successfully used to tackle the state explosion problem in model checking [20, 8, 16, 7, etc.]. They are based on the observation that the execution order of concurrent operations does not usually change the validity of a property. Hence, fixing a particular order of interleaving operations helps to reduce the number of states and transitions required to represent the program in the memory while preserving properties of interest.

The use of partial order techniques has been limited to the functional analysis of concurrent systems not including probabilities. Thus reduction algorithms for reachability analysis [20, 8], LTL and CTL model checking [16, 7] have been devised. We present, instead, *a partial order reduction method for the quantitative analysis of concurrent probabilistic programs.* (Concurrent probabilistic programs are syntactic sugar for Markov decision processes [17].)

Partial order techniques are based on the selection of a subset of the operations enabled in each state that are independent from (i.e. concurrent with) the other enabled operations. These last operations are removed (or more precisely, not included in the reduced graph). When the reduction is focused on the analysis of linear properties (e.g. reachability or LTL), any possible set of operations which is independent of the rest is candidate to belong to the reduced graph. Instead, branching time techniques require a particular selection of such sets since not all of them preserve the branching structure of the original graph. Probabilistic programs suffer of a similar problem due to the interplay between non-deterministic branches and probabilistic branches somehow comparable to the interplay between existential and universal path quantification in CTL [7] or the alternating choices of opponents in games [10]. We show that the usual constraints for LTL model checking (adapted to concurrent probabilistic programs) are not correct for quantitative reachability analysis and give a partial order reduction algorithm that responds to the same constraints of CTL model checking (adapted to probabilistic programs).

We prove that our algorithm preserves *probabilistic complete forward simulation*, a relation that strengthen probabilistic forward simulation [18] in order to ensure also preservation of deadlock states. Segala's execution correspondence theorem [18] states that probabilistic forward simulation preserves trace distribution. As a consequence, our algorithm preserves trace distribution and hence also *maximum probabilities of reachability properties.* A variant of the latter theorem for a complete setting (namely, a theorem stating that probabilistic complete forward simulation preserves complete (i.e. maximal) trace distribution) would imply that our reductions preserve general quantitative (next-free) LTL properties. Unfortunately, the authors have no knowledge of such a result. Nevertheless, we conjec-

* Supported by the European Community Project IST-2001-35304, AMETIST (http://ametist.cs.utwente.nl)

† CONICET Researcher. On leave from FaMAF, UNC.

Proceedings of the First International Conference on the Quantitative Evaluation of Systems (QEST'04)
0-7695-2185-1/04 $ 20.00 IEEE

ture its validity and consequently the scope of the current reduction scheme for the largest set of quantitative linear time properties one can expect.

Remark that the advantage of partial order reduction on quantitative model checking goes beyond state space representation. In order to calculate the probability of a property it is necessary to solve a linear optimization problem [2]. Since each state amounts to a new variable and each transition to a new inequality in the system, the reduction of the state space has a direct impact on the size of the optimisation problem.

This article presents the first result on partial order techniques for quantitative model checking, a discovery shared with [1] —published also in this volume— that independently proposed a similar solution.

Related work. Among the non-probabilistic approaches, our algorithm is closest to that for CTL* in [7]. Both algorithms share the same type of constraints to define ample sets and they are basically the same (up to the consideration of probabilistic branching).

[21] defined a probabilistic variant of Mazurkiewicz traces. Though we share a similar notion of independence (in fact their definition is close to Proposition 1 below), [21] focuses on the language rather than in the structure of the model. Next, [21] is confined to conflict-free probabilistic Petri nets which limits the type of systems one can model (comparing to the richer expressiveness of Markov decision processes). Finally, these Petri-nets rely on weights which can make dynamically change the probability of a transition.

Quantitative model checking algorithms and tools have already been devised in the context of concurrent probabilistic programs [2, 5, 4, 11, 13, etc.] Quantitative model checking for LTL formulas was proposed in [2, 5]. In particular, [5] presents an automata theoretic approach that reduces the property checking problem to quantitative reachability. This can be combined with the tool RAPTURE [11] which is aimed to the efficient calculation of quantitative reachability. Our result could be integrated within this framework.

Organisation of the paper. We start by introducing concurrent probabilistic programs and the associated notion of independence of operations, which adapts the standard notion [6] to a probabilistic setting (Section 2). Section 3 discusses the algorithm focusing on the four constraints that guides the state space exploration to construct a valid reduced program. The correctness of the algorithm is proved in Section 4. After discussing some issues about independence, we introduce complete forward simulation and devote the rest of the section to prove that any reduction that satisfy those four constraints preserves complete forward simulation equivalence. Section 5 concludes the paper focusing on further research. We include there a conjecture of the application of our technique to LTL and a proof that it cannot be applied in general in a quantitative branching time setting.

2. Concurrent Probabilistic Programs

A *(discrete) probability distribution* on a set X is a function $\mu : X \to [0,1]$ such that $\sum_{x\in X} \mu(x) = 1$. Probability distributions range on $\mu, \mu', \mu_i, \nu, P, ...$ The *support* set of a distribution μ is the set $\text{supp}(\mu) = \{x \in X \mid \mu(x) > 0\}$. For $A \subseteq S$, $\mu(A) = \sum_{x\in A} \mu(x)$ (the overloading with $\mu(x)$ is common).

χ_A is the *characteristic* function on set A defined by $\chi_A(x) = \textbf{if } x \in A \textbf{ then } 1 \textbf{ else } 0$. If $A = \{a\}$ then we write χ_a. Notice that χ_a is the deterministic (or Dirac) distribution on a (but χ_A is *not* a distribution in general). We use the λ-notation to write unnamed functions: the term $\lambda\vec{x}.e$ represents the function f such that $f(\vec{x}) = e$, where e is an expression with variables on $\vec{x}$.

A *(concurrent) probabilistic program* is a structure $\mathsf{P} = (\mathsf{S}, \mathsf{ini}, \mathsf{L}, \mathsf{T})$ where S is a set of *states* (ranging on $s, r, q, s', ..$) containing the *initial state* ini; $\mathsf{L} : \mathsf{S} \to \wp(\mathsf{AP})$ is a *proposition labelling* that assigns to each state, a set of *atomic propositions* valid on it; and T is a set of *operations* (ranging on $\alpha, \beta, \alpha', ..$) s.t. every operation $\alpha \in \mathsf{T}$ has associated the following elements:

1. $n_\alpha > 0$ is the *arity* of α,
2. $t_\alpha : \mathsf{S} \to \mathsf{S}^{n_\alpha}$ is a partial function representing the *transition* defined by α s.t. $\forall s \in \mathsf{S} : t_\alpha(s)[i] = t_\alpha(s)[j] \implies i = j$, and
3. $p_\alpha \in (0,1]^{n_\alpha}$, with $\sum_{i=1}^{n_\alpha} p_\alpha[i] = 1$, is the *probability* associated to α.

We let $\mathsf{en}(s) = \{\alpha \in \mathsf{T} \mid s \in \text{dom}(t_\alpha)\}$, for all $s \in \mathsf{S}$. We write $s \xrightarrow{\alpha} \mu$ if $\alpha \in \mathsf{en}(s)$ and for all $s' \in \mathsf{S}$, $\mu(s') = \textbf{if } (\exists i : t_\alpha(s)[i] = s') \textbf{ then } p_\alpha[i] \textbf{ else } 0$.

Definition 1. A symmetric and reflexive relation $D \subseteq \mathsf{T} \times \mathsf{T}$ is a *dependency relation* if for all $(\alpha, \beta) \notin D$, and $s \in \mathsf{S}$ the following two conditions hold

1. $\alpha \in \mathsf{en}(s) \implies (\beta \in \mathsf{en}(s) \iff (\forall i : 1 \le i \le n_\alpha : \beta \in \mathsf{en}(t_\alpha(s)[i])))$
2. $\alpha, \beta \in \mathsf{en}(s) \implies \forall i, j : 1 \le i \le n_\alpha, 1 \le j \le n_\beta : t_\beta(t_\alpha(s)[i])[j] = t_\alpha(t_\beta(s)[j])[i]$.

The *independence relation* I is defined by $(\alpha, \beta) \in I \iff (\alpha, \beta) \notin D$.

The independence relation in the usual non-probabilistic setting [6] is related to the existence of "diamonds" like the one depicted on the left-hand side of Figure 1 (which represents the process $\alpha||\beta$). It is defined with two properties, so called *forward stability*

Proceedings of the First International Conference on the Quantitative Evaluation of Systems (QEST'04)
0-7695-2185-1/04 $ 20.00 IEEE

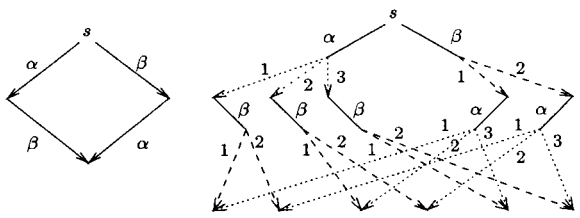

Figure 1. An illustration of independence

and *commutativity*. Forward stability states that independent operations do not disable one another. In a probabilistic setting this amounts to requiring that an operation remains enabled in *all* probabilistic branches of its independent counterpart. This is stated by property 1 in Definition 1. Commutativity states that exchanging the execution order of two independent operations from a given state leads to the same state. A probabilistic operation may lead to more than one state. The equivalent probabilistic property would then require that the execution order of two independent *probabilistic* operations from a given state leads to the same state *with equal probability*. Property 2 in Definition 1 ensures this, though in a structural manner. (See Proposition 1 for a pure semantic characterisation of independence). The right-hand side of Figure 1 shows a "probabilistic diamond" where both operations α and β have arity 2 (numbers on the arrow correspond to the indexes of the probabilistic branches).

Proposition 1 gives a semantic characterisation of independence.

Proposition 1. *For all* $(\alpha, \beta) \in I$ *and* $s \in \mathsf{S}$,

(a) if $s \xrightarrow{\alpha} P$ *then* $(s \xrightarrow{\beta} \iff P(\mathrm{dom}(t_\beta)) = 1)$ *(notice that* $s \in \mathrm{dom}(t_\beta)$ *iff* $\beta \in \mathsf{en}(s)$*); and*

(b) if $s \xrightarrow{\alpha} P_a$ *and* $s \xrightarrow{\beta} P_b$ *then for all* $s' \in \mathsf{S}$,

$$\sum_{(r_a,\beta,P'_b)\in(\to)} P_a(r_a) \cdot P'_b(s') = \sum_{(r_b,\alpha,P'_a)\in(\to)} P_b(r_b) \cdot P'_a(s')$$

3. Algorithm

The reduction algorithm simply prunes operations of the original probabilistic programs. It is hence vital that the pruned operations do not change the properties of the systems. Therefore we define the following notion of invisibility on probabilistic programs:

Definition 2. We say that an operation α is *invisible* if for all s, $\alpha \in \mathsf{en}(s)$, and i, $1 \leq i \leq n_\alpha$, $\mathsf{L}(s) = \mathsf{L}(t_\alpha(s)[i])$.

In other words: only operations that change the state properties of the systems can be observed.

Like usual algorithms for partial order reduction [8, 16, 7], this one is also based on a modified depth-first search algorithm. Rather than exploring through all enabled operations at the current state s, only operations in the so called *ample set* of s are used to generate the successors.

The algorithm generates a reduced probabilistic program $\mathsf{P}_{red} = (\mathsf{S}_{red}, \mathsf{ini}, \mathsf{L}, \mathsf{T}_{red})$ where $\mathsf{S}_{red} \subseteq \mathsf{S}$ is the subset of states reachable from ini only via operations in ample sets, and T_{red} contains the same operations as T only that $s \in \mathrm{dom}(t_\alpha)$ iff $\alpha \in \mathsf{amp}(s)$. For each state s, $\mathsf{amp}(s)$ is the ample set associated to s such that the following restrictions are satisfied:

C1. No operation $\alpha \in \mathsf{T} - \mathsf{amp}(s)$ that is dependent on an operation in $\mathsf{amp}(s)$ can be executed in the original graph before an operation from $\mathsf{amp}(s)$ is executed.

C2. For every cycle in the reduced graph, at least for one state s in the cycle $\mathsf{amp}(s) = \mathsf{en}(s)$, i.e. s is fully expanded.

C3. If $\mathsf{amp}(s) \neq \mathsf{en}(s)$ then every $\alpha \in \mathsf{amp}(s)$ is invisible.

C4. Either $\mathsf{amp}(s) = \mathsf{en}(s)$ or $|\mathsf{amp}(s)| = 1$.

Condition **C1** enforces that any finite sequence of operations leaving s that does not contain an operation from $\mathsf{amp}(s)$ can be extended with such an operation. This is the only condition that is concretely related to the notion of (in)dependence. Condition **C2** ensures that enabled independent operations are eventually taken. Notice that without **C2** there may exist a cycle on which operation α is enabled on every state s along the cycle but not in the ample set $\mathsf{amp}(s)$. Usually **C2** is strengthened in the algorithm to require that only the last explored state in the cycle is fully expanded. This allows a much simpler implementation. **C3** ensures that traces that are removed are observationally equivalent to those that remain.

Conditions **C1**, **C2**, and **C3** are sufficient to guarantee that the reduced program preserves validity of LTL properties in non-deterministic systems [16]. However they do not suffice to guarantee preservation of the measures on probabilistic programs. Take, for example, program P depicted in Figure 2.(a)[1]. There, the arity of α is 2 and the arity of all other operations is 1. Take $p_\alpha[1] = p_\alpha[2] = 0.5$. ☺ and ☹ are the only atomic propositions and they are valid on the indicated states. Therefore, α, β, and γ are invisible operations. According to conditions **C1**–**3** both $\{\alpha\}$ and $\{\beta, \gamma\}$ are good candidates to be $\mathsf{amp}(\mathsf{ini})$. Choosing $\mathsf{amp}\,\mathsf{ini} = \{\beta, \gamma\}$

1 Examples in Figures 2 and 4 are based on similar examples reported in [15] for non-probabilistic settings.

Proceedings of the First International Conference on the Quantitative Evaluation of Systems (QEST'04)
0-7695-2185-1/04 $ 20.00 IEEE

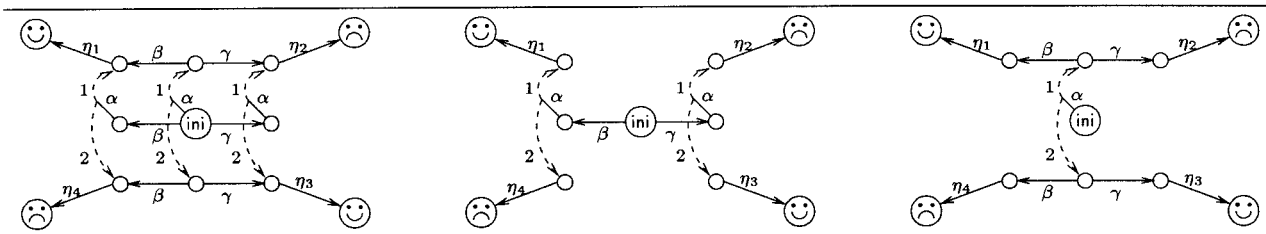

Figure 2. (a) Original program (b) Incorrect reduction (c) Correct reduction

leads to the reduced program P′ in Figure 2.(b). Notice however that it does not preserve the probability of reaching ☺. The maximum probability of reaching ☺ in P is 1 (by choosing first α and then β after branch 1 of α, or γ after branch 2 of α). It is instead .5 in P′ (independently of the choice between β and γ).

This effect has an analogy with that of branching time temporal logic. The interplay between nondeterminism and probabilism is comparable to that of existential and universal quantifications of branches. What we observed in Figure 2.(b) is an early optimising decision that will be jeopardized by a parallel component. Notice, instead, that choosing α first postpones the real nondeterministic decision between β and γ. As a consequence we adopt condition **C4** which ensures that the interplay between nondeterministic and probabilistic choices is preserved: either the unique nondeterministic choice is safe (and hence $|\mathsf{amp}(s)| = 1$) or all branching is preserved ($\mathsf{amp}(s) = \mathsf{en}(s)$). Figure 2.(c) shows a reduction that satisfies conditions **C1–4**.

The algorithm is a slight modification of the one presented in [7]. Conditions **C2–4** are explicitly considered in the algorithm. Instead, there are static techniques that allow for an efficient checking of **C1** [9].

4. Correctness of the Algorithm

In the following, we prove that any reduced program that satisfies conditions **C1–4** is complete forward simulation equivalent to the original graph. This is stated in Theorem 3 at the end of the section. Since the algorithm ensures the satisfaction of conditions **C1–4** by the output reduced program, its correctness is hence guaranteed.

The section is organised as follows. The first part, which extends until and including Lemma 2, discusses the behaviour of independence. We define the notion of *forming paths* which are executions that preserve conditions **C1**, **C3**, and **C4**, and have the ability to commute with independent operations. Next, we introduce complete forward simulation which is the aim of the proof. However the proof is focused on an auxiliary relation $\mathcal{R}$ that contains the necessary structure to link the simulation relation to the four conditions. In fact, forming paths are the building blocks of $\mathcal{R}$. $\mathcal{R}$ relates states in the original paths with (distributions on) forming paths that inevitably lead such a state (back) to some state belonging to the reduced graph.

$\mathcal{R}$ is constructed inductively using an auxiliary weak transition $\Rightarrow_\alpha$ which is guaranteed to exists due to condition **C2** (Lemma 4). After discussing the characteristics of $\Rightarrow_\alpha$ in Lemmas 3 to 5 and Theorem 1, we give the formal definition of $\mathcal{R}$ and discuss its structure (Lemma 6 and Theorem 2). Lemma 7 anticipates in $\mathcal{R}$ the simulation relation which is then defined (in terms of $\mathcal{R}$) and proved in Theorem 3.

A *path* in a program P is a finite sequence $s_0(\alpha_1,i_1)s_1(\alpha_2,i_2)s_2\cdots s_{n-1}(\alpha_n,i_n)s_n$ s.t. for all $0 \le k < n$, $\alpha_{k+1} \in \mathsf{en}(s_k)$ and $s_{k+1} = t_{\alpha_{k+1}}(s_k)[i_{k+1}]$. Notice that, because of determinism on operations[2], the path is uniquely determined by the sequence of operations $\rho = (\alpha_1,i_1)(\alpha_2,i_2)\cdots(\alpha_n,i_n)$ and its source state s_0. Then, we can write $s_0 \overset{\rho}{\rightsquigarrow} s_n$. Let $\alpha{\downarrow}\rho$ denote that a pair (α,i), with $1 \le i \le n_\alpha$, *occurs* in sequence ρ.

A set of operations A satisfies condition **C1** in state s if it satisfies **C1** when replacing $\mathsf{amp}(s)$ by A. That is, A is a "**C1**-candidate" to be an ample set on s. The following lemma states that if $\{a\}$ is a **C1**-candidate in s, it remains a **C1**-candidate after any other operation enabled in s was performed.

Lemma 1. *If $\alpha \in \mathsf{en}(s)$ such that $\{\alpha\}$ satisfies condition **C1** from s and $\alpha \neq \beta \in \mathsf{en}(s)$, then $\{\alpha\}$ satisfies **C1** from $t_\beta(s)[i]$ for all $1 \le i \le n_\beta$.*

Proof. By **C1**, $(\alpha,\beta) \in I$. By definition of independence $\alpha \in \mathsf{en}(t_\beta(s)[i])$ for all $1 \le i \le n_\beta$. Any path that starts with $t_\beta(s)[j]$ (for some j) that invalidates **C1** can be extended by affixing $s(\beta,j)t_\beta(s)[j]$ in front of it. This contradicts the fact that $\{a\}$ satisfy condition **C1** from s. □

2 Probabilistic programs are *non deterministic* in the sense that more than one operation can be enabled in one state, but *deterministic on operations* because different enabled operations cannot share the same name.

Proceedings of the First International Conference on the Quantitative Evaluation of Systems (QEST'04)
0-7695-2185-1/04 $ 20.00 IEEE

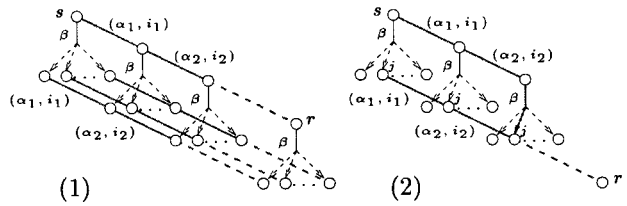

Figure 3. Illustration of Lemma 2

A *forming path* is a path $s_0(\alpha_1, i_1)s_1(\alpha_2, i_2)s_2 \cdots s_n$ s.t. for all k, $0 < k \leq n$, α_k is invisible and $\{\alpha_k\}$ satisfies condition **C1** from state s_{k-1}.

Notice that a forming path does not only ensure condition **C1**, but also **C3** (α_k must be invisible) and **C4** (it goes only through singleton set).

Forming paths enjoy the property that they can be replicated after an independent operation is performed. This is stated in Lemma 2.1 below. Lemma 2.2 is the variation on which the performed operation is also performed in the forming path. The result of Lemma 2 is illustrated in Figure 3.

Lemma 2. *If $s \overset{\rho}{\rightsquigarrow} r$, with ρ defining a forming path, and $\beta \in \mathsf{en}(s)$, then*

1. *$\neg(\beta\downarrow\rho)$ implies $\beta \in \mathsf{en}(r)$ and for all i, $1 \leq i \leq n_\beta$, $t_\beta(s)[i] \overset{\rho}{\rightsquigarrow} t_\beta(r)[i]$ with ρ defining a forming path.*
2. *$\beta\downarrow\rho$ implies there exist ρ_1, ρ_2, and j, $1 \leq j \leq n_\beta$, such that $\neg(\beta\downarrow\rho_1)$, $\rho = \rho_1(\beta, j)\rho_2$, and $t_\beta(s)[j] \overset{\rho_1\rho_2}{\rightsquigarrow} r$ with $\rho_1\rho_2$ defining a forming path.*

Proof. 1. By induction on the length of ρ. For the base case, $\rho = \epsilon$. Then $s = r$ and the lemma holds.

So, suppose $\rho = \rho'(\alpha, j)$. Hence, $\beta \neq \alpha$ and $\neg(\beta\downarrow\rho')$. Then, there is r' s.t. $s \overset{\rho'}{\rightsquigarrow} r'$ and $t_\alpha(r')[j] = r$. Clearly, ρ' also defines a forming path from s. By induction, $\beta \in \mathsf{en}(r')$ and for all i, $1 \leq i \leq n_\beta$, $t_\beta(s)[i] \overset{\rho'}{\rightsquigarrow} t_\beta(r')[i]$ with ρ' defining a forming path.

Because $\{\alpha\}$ satisfies **C1** from r', using Lemma 1, $\{\alpha\}$ satisfies **C1** from $t_\beta(r')[i]$. Moreover, $(\alpha, \beta) \in I$. Hence $\beta \in \mathsf{en}(r)$ (since $t_\alpha(r')[j] = r$) and for all i, $1 \leq i \leq n_\beta$, $t_\alpha(t_\beta(r')[i])[j] = t_\beta(r)[i]$. As a consequence, $t_\beta(s)[i] \overset{\rho}{\rightsquigarrow} t_\beta(r)[i]$ with ρ defining a forming path.

2. Follows as a corollary of the previous case. □

Lemmas 1 and 2 as well as the notion of forming path are borrowed and adapted from [7].

The correctness proof of the algorithm is granted by finding a (probabilistic weak) complete forward simulation. We first define weak transitions. Next, we define complete forward simulation and give an intuition behind it. Finally, we introduce and dive into the correctness proof.

$$\frac{r \xrightarrow{\alpha} \mu}{r \overset{\alpha}{\Longrightarrow} \mu} \qquad \frac{r \xrightarrow{\beta} \mu \quad \beta \text{ is invisible} \quad \forall r' \in \mathrm{supp}(\mu) : r' \overset{\alpha}{\Longrightarrow} \nu_{r'}}{r \overset{\alpha}{\Longrightarrow} \sum_{r' \in \mathrm{supp}(\mu)} \mu(r') \cdot \nu_{r'}}$$

Table 1. Probabilistic weak transition

Table 1 introduce probabilistic weak transitions. (The definition style is borrowed from [19].) So $r \overset{\alpha}{\Longrightarrow} \mu$ if there is a "probabilistic tree" with root on the starting state r, such that μ is the probability of reaching its "leaves" (i.e. the end states) and the last transitions in this tree are labelled with α. Then $r \overset{\hat{\alpha}}{\Longrightarrow} \mu$ iff (α is invisible and $\mu = \chi_r$) or $r \overset{\alpha}{\Longrightarrow} \mu$.

The definition of simulation relies on *weight functions* [14, 12] that take care of appropriately matching the probabilistic transitions. Given a relation $R \subseteq X \times Y$ define $\sqsubseteq_R \subseteq \mathsf{Dist}(X) \times \mathsf{Dist}(Y)$ by: $\mu_X \sqsubseteq_R \mu_Y$ iff there is a *weight function* $w : X \times Y \to [0, 1]$ s.t.

1. $\forall x \in X : \sum_{y \in Y} w(x, y) = \mu_X(x)$,
2. $\forall y \in Y : \sum_{x \in X} w(x, y) = \mu_Y(y)$, and
3. $\forall (x, y) \in X \times Y : w(x, y) > 0 \implies \langle x, y \rangle \in R$.

Definition 3. $\mathcal{S}$ is a *complete forward simulation* if for all $\langle s, Q \rangle \in \mathcal{S}$, the following *transfer properties* hold:

1. $s \xrightarrow{\alpha} \mu_1$ implies
   (a) for all $r \in \mathrm{supp}(Q)$, $\exists \mu_r : r \overset{\hat{\alpha}}{\Longrightarrow} \mu_r$, and
   (b) exists $\mu_2' \in \mathsf{Dist}(\mathsf{Dist}(\mathsf{S}_{red}))$ with $\mu_1 \sqsubseteq_{\mathcal{S}} \mu_2'$,
   s.t. $\sum_{r \in \mathrm{supp}(Q)} Q(r) \cdot \mu_r = \sum_{Q' \in \mathrm{supp}(\mu_2')} \mu_2'(Q') \cdot Q'$.
2. if $r \in \mathrm{supp}(Q)$ and $r \xrightarrow{\alpha} \mu$ for some α and μ, then $s \xrightarrow{\beta} \nu$ for some β and ν.

We say that two concurrent probabilistic programs P and P' with initial states ini and ini', respectively, are *complete forward simulation equivalent* if there are two simulation relations $\mathcal{S}$ and $\mathcal{S}'$ such that $\langle \mathsf{ini}, \chi_{\mathsf{ini}'} \rangle \in \mathcal{S}$ and $\langle \mathsf{ini}', \chi_{\mathsf{ini}} \rangle \in \mathcal{S}'$.

This simulation is strictly stronger than the probabilistic forward simulation of [18] in two senses: (i) we require completeness, i.e. the simulating process stops only if the simulated one does, and (ii) weak transitions are not convex combination of non deterministic transitions. Instead it is incomparable to the probabilistic simulation of [12] since (i) probabilistic simulation does not covers completeness, and (ii) it is a relation from states to states rather than from states to distribution of states.

Forward simulation allows to implement one single probabilistic operation by several probabilistic operations. Consider for instance the program P depicted in Figure 4 (left) together with a reduced program

Proceedings of the First International Conference on the Quantitative Evaluation of Systems (QEST'04)
0-7695-2185-1/04 $ 20.00 IEEE

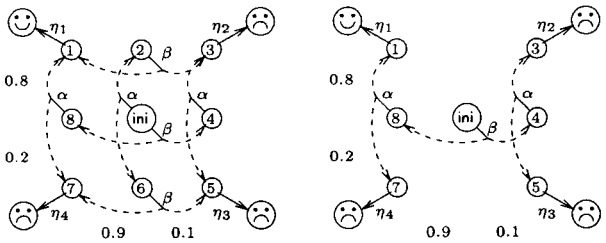

Figure 4. A simple reduction example

$$\textbf{W1}\quad \frac{\alpha \in \mathsf{amp}(r)}{(\rho,r) \Rightarrow_\alpha \chi_{(\rho,r)}}$$

$$\textbf{W2}\quad \frac{\beta \neq \alpha \in \mathsf{en}(r) \neq \mathsf{amp}(r) = \{\beta\} \quad \forall j : 1 \leq j \leq n_\beta : (\rho(\beta,j), t_\beta(r)[j]) \Rightarrow_\alpha \mu_j}{(\rho,r) \Rightarrow_\alpha \sum_{j=1}^{n_\beta} p_\beta[j] \cdot \mu_j}$$

Table 2. Auxiliary weak transition (to define $\mathcal{R}$)

P_{red} (right) (where α and β are invisible operations). Operation α from ini in P has no direct match in P_{red}. Our choice is, however, to let P_{red} execute both β and α and relate states in P with distributions on states P_{red}. This leads to pairs $\langle 2, P_2\rangle$ and $\langle 6, P_6\rangle$ where $P_2(1) = 0.9$ and $P_2(3) = 0.1$, and $P_6(7) = 0.9$ and $P_6(5) = 0.1$. When P executes operation β from, e.g., state 2, P_{red} does not perform any operation, but "offers" states 1 and 3 with probabilities 0.9 and 0.1, respectively (which match to states 1 and 3 in P, resp.). Hence, P_2 and P_6 record probabilistic choices that have already been resolved in P_{red} but not yet in P. (It is like an accounting of the probabilities that P "owes" to P_{red}.) Therefore, a forward simulation is a subset of $\mathsf{S} \times \mathsf{Dist}(\mathsf{S})$.

Like in [7], forming paths are in the back bone of our proof. Rather than giving directly a forward simulation between P and P_{red} (for any P), we give a relation $\mathcal{R} \subseteq \mathsf{S} \times \mathsf{Dist}((\mathsf{T} \times \mathbb{N})^* \times \mathsf{S})$ between states in P, and distributions on pairs of paths (in P) and states in P_{red}. In fact whenever $\langle s, P\rangle$ is an element on such relation, and (ρ, r) is in the support set of P, $s \overset{\rho}{\rightsquigarrow} r$ with ρ defining a forming path. Moreover, $P(\rho, r)$ is the probability of executing ρ from s. In our example, we would rather have the pairs $\langle 2, P_2'\rangle$ and $\langle 6, P_6'\rangle$ (instead of $\langle 2, P_2\rangle$ and $\langle 6, P_6\rangle$) where $P_2'((\beta,1),1) = 0.9$, $P_2'((\beta,2),3) = 0.1$, $P_6'((\beta,1),7) = 0.9$, and $P_6'((\beta,2),5) = 0.1$ (where the indexes next to β are the obvious ones). From $\mathcal{R}$, we then construct the forward simulation $\mathcal{S}$ by abstracting the forming paths.

The benefits of defining $\mathcal{R}$ is that it records not only the probabilities "owed" by P but also the operation sequence that P "owes" P_{red} (which need not be "paid" in the same order as long as independence permits.)

The construction of $\mathcal{R}$ is inductive with the transfer property of the forward simulation in mind. So let $(\rho, r) \Rightarrow_\alpha \mu$ be defined inductively by rules in Table 2 (ρ', r') is in the support of μ if ρ' extends ρ with a forming path in the reduced program and $\mathsf{amp}(r') = \{\alpha\}$. Thus $(\rho, r) \Rightarrow_\alpha \mu$ is a weak probabilistic transition that surely leads r to pairs containing states for which $\{\alpha\}$ is an ample set and recording the way that each state is reached. Lemmas 3, 4, and 5, and Theorem 1 ensure these properties.

Lemma 3. *If $(\rho_1, r_1) \Rightarrow_\alpha \mu$ and $(\rho_2, r_2) \in \mathrm{supp}(\mu)$ then there exists ρ_3 s.t. $\rho_2 = \rho_1\rho_3$ and $r_1 \overset{\rho_3}{\rightsquigarrow} r_2$ with ρ_3 defining a forming path.*

Proof. By induction on the proof tree of $\Rightarrow_\alpha$. The base case applies to rule **W1** and is immediate. For the inductive case, suppose $(\rho_2, r_2) \in \mathrm{supp}(\sum_{j=1}^{n_\beta} p_\beta[j] \cdot \mu_j)$. Then, there exists i, $1 \leq i \leq n_\beta$, s.t. $(\rho_2, r_2) \in \mu_i$. Notice that i is unique. By induction, there is ρ_3 s.t. $\rho_2 = \rho_1(\beta, i)\rho_3$ and $t_\beta(r_1)[i] \overset{\rho_3}{\rightsquigarrow} r_2$ with ρ_3 defining a forming path. Take $\rho_3' = (\beta, i)\rho_3$. Then

1. $\rho_2 = \rho_1\rho_3'$, and
2. $r_1 \overset{\rho_3'}{\rightsquigarrow} r_2$ with ρ_3' defining a forming path, since $\{\beta\} = \mathsf{amp}(r_1)$ and therefore it is invisible and satisfies **C1** in r_1. □

Lemma 4. *If $\alpha \in \mathsf{en}(r)$ then, $\forall \rho : \exists \mu : (\rho, r) \Rightarrow_\alpha \mu$.*

Proof. Suppose this is not the case. From **W1** $\alpha \notin \mathsf{amp}(r)$. From **W2**, taking into account Lemma 1, we deduce that there must be an infinite path $r \equiv r_0(\alpha_1, i_1)r_1(\alpha_2, i_2)r_2 \cdots$ s.t. for all $i \geq 0$, $\alpha_{i+1} \neq \alpha \in \mathsf{en}(r_i) \neq \{\alpha_{i+1}\} = \mathsf{amp}(r_i)$ which contradicts **C2**. □

Lemma 5. *If $(\rho, r) \Rightarrow_\alpha \mu$ and $(\rho', r') \in \mathrm{supp}(\mu)$ then $\alpha \in \mathsf{amp}(r')$.*

Proof. Straightforward induction on the proof tree. □

The following theorem completes the idea that $\Rightarrow_\alpha$ is well defined. The proof is omitted here.

Theorem 1. *If $(\rho, r) \Rightarrow_\alpha \mu$ then (i) μ is a distribution, and (ii) if r is in the reduced program, for all $(\rho', r') \in \mathrm{supp}(\mu)$, r' is a state in the reduced program.*

$\mathcal{R} \subseteq \mathsf{S} \times \mathsf{Dist}((\mathsf{T} \times \mathbb{N})^* \times \mathsf{S})$ is defied to be the least relation satisfying rules in Table 3. Rule **R1** is obvious. **R2** performs the inductive construction. Whenever $\langle s, P\rangle \in \mathcal{R}$, an operation α enabled in s and a probabilistic branch i from such operation induce a new pair in $\mathcal{R}$ between the state reached from s after such probabilistic branch and a probability distribution P_P^i constructed from P as follows.

For every pair $(\rho, r) \in \mathrm{supp}(P)$ in which α does not appear in ρ, the enabled α's closest to r are

Proceedings of the First International Conference on the Quantitative Evaluation of Systems (QEST'04)
0-7695-2185-1/04 $ 20.00 IEEE

R1

$$\frac{s \text{ is in the reduced program}}{\langle s, \chi_{(\epsilon,s)} \rangle \in \mathcal{R}}$$

R2

$$\frac{\begin{array}{lll} \langle s, P \rangle \in \mathcal{R} & \alpha \in \mathsf{en}(s) & 1 \le i \le n_\alpha \qquad P'_P = \sum_{c \in \mathrm{supp}(P)} p^i_c \cdot \nu^i_c \\ \forall c \in \mathrm{supp}(P) : c = (\rho, r) : & \big(\neg(\alpha \downarrow \rho) \wedge c \Longrightarrow_\alpha \mu \wedge \nu^i_c = \mu \circ (\lambda(\rho', r').(\rho', t_\alpha(r')[i]))^{-1} \wedge p^i_c = P(c)\big) & \\ & \vee \big(\neg(\alpha \downarrow \rho_1) \wedge \rho = \rho_1(\alpha, i)\rho_2 \wedge \nu^i_c = \chi_{(\rho_1\rho_2, r)} \wedge p^i_c = \frac{P(c)}{p_\alpha[i]}\big) & \\ & \vee \big(\neg(\alpha \downarrow \rho_1) \wedge \rho = \rho_1(\alpha, j)\rho_2 \wedge j \neq i \wedge \nu^i_c = \lambda x.0 \wedge p^i_c = 0\big) & \end{array}}{\langle t_\alpha(s)[i], P'_P \rangle \in \mathcal{R}}$$

Table 3. Rules defining $\mathcal{R}$

found with the weak transition $(\rho, r) \Longrightarrow_\alpha \mu$. For pairs $(\rho', r') \in \mathrm{supp}(\mu)$, ρ' extends ρ until reaching an enabled $\alpha \in \mathsf{amp}(r')$ (ensured by Lemmas 3 and 5). As a consequence, operations in the extension are "owed" to P_{red}. Operation α through branch i is then realized from r' and pair $(\rho', t_\alpha(r')[i])$ added to P'_P (using function $(\lambda(\rho', r').(\rho', t_\alpha(r')[i]))^{-1}$) with the new "owed" probability $P(\rho, r) \cdot \mu(\rho', r')$ (which is the same as $(p^i_{(\rho,r)} \cdot \nu^i_{(\rho,r)})(\rho', t_\alpha(r')[i])$).

If instead (α, i) occurs in ρ before any other occurrence of α, then it is removed (it is "paid back" by the original program P) and the probability $p_\alpha[i]$ is discounted (the new probability is $\frac{P(\rho,r)}{p_\alpha[i]}$). Notice that in this case P_{red} does not perform any operation.

If the first occurrence of α in ρ was through a different probabilistic branch j, then (ρ, r) is discarded since it does not belong to the current run.

Lemma 6 and Theorem 2 ensure that $\mathcal{R}$ enjoys the properties previously described. In particular, Theorem 2 states that $\mathcal{R}$ is well defined.

Lemma 6. *For all $\langle s, P \rangle \in \mathcal{R}$ and $(\rho, r) \in \mathrm{supp}(P)$, $s \overset{\rho}{\rightsquigarrow} r$ with ρ defining a forming path.*

Proof. We proceed by induction on the proof tree of $(s, P) \in \mathcal{R}$ by case analysis on the proof rules.

Case R1. Obvious.

Case R2. Let $\langle t_\alpha(s)[i], P \rangle \in \mathcal{R}$ and $(\rho, r) \in \mathrm{supp}(P'_P)$. Then $(\rho, r) \in \mathrm{supp}(\nu^i_c)$ for some $c = (\rho', r') \in \mathrm{supp}(P)$. Two subcases arises

Subcase $\neg(\alpha \downarrow \rho')$. Then, $c \Longrightarrow_\alpha \mu$ and exists $(\rho, r'') \in \mathrm{supp}(\mu)$ s.t. $r = t_\alpha(r'')[i]$. By Lemma 3, exists ξ s.t. $\rho = \rho'\xi$ and $r' \overset{\xi}{\rightsquigarrow} r''$ with ξ defining a forming path. By induction, $s \overset{\rho'}{\rightsquigarrow} r'$ with ρ' defining a forming path. Then $s \overset{\rho}{\rightsquigarrow} r''$ and $\rho = \rho'\xi$ defines a forming path. By Lemma 2.1, $t_\alpha(s)[i] \overset{\rho}{\rightsquigarrow} t_\alpha(r'')[i]$ (i.e. $t_\alpha(s)[i] \overset{\rho}{\rightsquigarrow} r$) and ρ defines a forming path.

Subcase $\alpha \downarrow \rho'$. Then, $\rho' = \rho_1(\alpha, i)\rho_2$, $r = r'$ and $\rho = \rho_1\rho_2$ with $\neg(\alpha \downarrow \rho_1)$. By induction, $s \overset{\rho'}{\rightsquigarrow} r$ with ρ' defining a forming path. By Lemma 2.2, $t_\alpha(s)[i] \overset{\rho}{\rightsquigarrow} r$ and ρ defines a forming path. □

WT

$$\frac{\begin{array}{l} (\rho, r) \Longrightarrow_\alpha \mu \\ \forall c \in \mathrm{supp}(\mu) : \quad c = (\rho', r') \wedge \alpha \in \mathsf{amp}(r') : \\ \qquad \mu_c = p_\alpha \circ (\lambda i.(\rho', t_\alpha(r')[i]))^{-1} \end{array}}{(\rho, r) \overset{\alpha}{\Longrightarrow} \sum_{c \in \mathrm{supp}(\mu)} \mu(c) \cdot \mu_c}$$

Table 4. Auxiliary weak transition

Theorem 2. *If $\langle s, P \rangle \in \mathcal{R}$ then (i) P is a distribution, and (ii) if $(\rho, r) \in \mathrm{supp}(P)$, r is in the reduced program.*

Transition $(\rho, r) \Longrightarrow_\alpha \mu$ leads (ρ, r) to the "threshold" of the execution of α (see Lemma 5). Transition $(\rho, r) \overset{\alpha}{\Longrightarrow} \nu$ (defined in Table 4) has this only one step further in which α is executed. By Lemma 5, if $(\rho, r) \Longrightarrow_\alpha \mu$, there exists ν such that $(\rho, r) \overset{\alpha}{\Longrightarrow} \nu$.

Lemma 7 states the characteristics of $\mathcal{R}$ and announce the complete forward simulation relation which is later defined and proved in Theorem 3.

Lemma 7. *If $\langle s, P \rangle \in \mathcal{R}$ then*

1. $s \xrightarrow{\alpha} \mu_1$ implies

(a) for all $c \in \mathrm{supp}(P)$, $c = (\rho, r)$, either

- *$\exists \mu_c : c \overset{\alpha}{\Longrightarrow} \mu_c$, or*
- *$\exists \rho_1, \rho_2, i : \neg(\alpha \downarrow \rho_1) \wedge \rho = \rho_1(\alpha, i)\rho_2 : \mu_c = \chi_{(\rho_1\rho_2, r)}$, and*

(b) exists $\mu'_2 \in \mathsf{Dist}(\mathsf{Dist}((\mathsf{T} \times I\!N)^* \times \mathsf{S}_{red}))$ with $\mu_1 \sqsubseteq_{\mathcal{R}} \mu'_2$,

s.t. $\sum_{c \in \mathrm{supp}(P)} P(c) \cdot \mu_c = \sum_{P' \in \mathrm{supp}(\mu'_2)} \mu'_2(P') \cdot P'$.

2. if $(\rho, r) \in \mathrm{supp}(P)$ and $r \xrightarrow{\alpha} \mu$ for some α and μ, then $s \xrightarrow{\beta} \nu$ for some β and ν.

Proof. Let $\langle s, P \rangle \in \mathcal{R}$ and $s \xrightarrow{\alpha} \mu_1$. Then, by Lemma 6, for all $c \in \mathrm{supp}(P)$, $c = (\rho, r)$, $s \overset{\rho}{\rightsquigarrow} r$ with ρ defining a forming path.

Since $\alpha \in \mathsf{en}(s)$, one of the following two cases holds:

Proceedings of the First International Conference on the Quantitative Evaluation of Systems (QEST'04)
0-7695-2185-1/04 $ 20.00 IEEE

1. $\neg(\alpha{\downarrow}\rho)$ and hence $\alpha \in \mathsf{en}(r)$ by Lemma 2.1. By Lemma 4, there is μ'_c s.t. $c \Longrightarrow_\alpha \mu'_c$. Therefore, for all $(\rho', r') \in \mathrm{supp}(\mu'_c)$, $\alpha \in \mathsf{amp}(r')$ by Lemma 5, and hence, by **WT**, $c \stackrel{\alpha}{\Longrightarrow} \mu_c$ with

$$\mu_c = \sum_{(\rho',r')\in \mathrm{supp}(\mu'_c)} \mu'_c((\rho', r')) \cdot (p_\alpha \circ (\lambda i.(\rho', t_\alpha(r')[i]))^{-1}) \tag{1}$$

2. $\alpha{\downarrow}\rho$ from which there are ρ_1, ρ_2, and i, s.t. $\neg(\alpha{\downarrow}\rho_1)$, $\rho = \rho_1(\alpha, i)\rho_2$, and

$$\mu_c = \chi_{(\rho_1\rho_2, r)}. \tag{2}$$

Define μ'_2 by $\mu'_2(P^i_P) = p_\alpha[i]$ for all i, $1 \le i \le n_\alpha$, and $\mu'_2(Q) = 0$ otherwise. Let w be defined by $w(t_\alpha(s)[i], P^i_P) = p_\alpha[i]$ for all i, $1 \le i \le n_\alpha$, and $w(x, y) = 0$ otherwise. It is easy to check that w is a weight function which ensures that $\mu_1 \sqsubseteq_{\mathcal{R}} \mu'_2$.

Now, we calculate

$$\sum_{P'\in \mathrm{supp}(\mu'_2)} \mu'_2(P') \cdot P' = \sum_{i=1}^{n_\alpha} p_\alpha[i] \cdot P^i_P \qquad \text{(Def. of } \mu'_2\text{)}$$

$$= \sum_{i=1}^{n_\alpha} p_\alpha[i] \cdot \sum_{c\in \mathrm{supp}(P)} p^i_c \cdot \nu^i_c \qquad \text{(Def. of } P^i_P\text{)}$$

$$\left[\begin{array}{l}\text{Define } A = \{(\rho', r') \in \mathrm{supp}(P) \mid \neg(\alpha{\downarrow}\rho')\}, \text{ and for all} \\ i,\ 1 \le i \le n_\alpha,\ B_i = \{(\rho', r') \in \mathrm{supp}(P) \mid \exists \rho_1, \rho_2 : \\ \neg(\alpha{\downarrow}\rho_1) \wedge \rho' = \rho_1(\alpha, i)\rho_2\}.\end{array}\right]$$

$$= \sum_{i=1}^{n_\alpha} p_\alpha[i] \cdot \Big(\sum_{c\in A} P(c) \cdot (\mu'_c \circ (\lambda(\rho', r').(\rho', t_\alpha(r')[i]))^{-1}) + \sum_{\substack{c\in B_i,\ \neg(\alpha{\downarrow}\rho_1) \\ c=(\rho_1(\alpha,k)\rho_2, r')}} \frac{P(c)}{p_\alpha[i]} \cdot \chi_{(\rho_1\rho_2, r')} \Big)$$

(Def. of p^i_c and ν^i_c)

$$= \sum_{c\in A} P(c) \cdot \sum_{i=1}^{n_\alpha} p_\alpha[i] \cdot \sum_{(\rho',r')\in \mathrm{supp}(\mu'_c)} \mu'_c(\rho', r') \cdot \chi_{(\rho', t_\alpha(r')[i])} + \sum_{i=1}^{n_\alpha} \sum_{\substack{c\in B_i,\ \neg(\alpha{\downarrow}\rho_1) \\ c=(\rho_1(\alpha,k)\rho_2, r')}} P(c) \cdot \chi_{(\rho_1\rho_2, r')} \qquad \text{(Arithmetics)}$$

$$= \sum_{c\in A} P(c) \cdot \sum_{(\rho',r')\in \mathrm{supp}(\mu'_c)} \mu'_c(\rho', r') \cdot \sum_{i=1}^{n_\alpha} p_\alpha[i] \cdot \chi_{(\rho', t_\alpha(r')[i])} + \sum_{\substack{c\in (\mathrm{supp}(P)-A),\ \neg(\alpha{\downarrow}\rho_1) \\ c=(\rho_1(\alpha,k)\rho_2, r')}} P(c) \cdot \sum_{i=1}^{n_\alpha} \chi_{B_i}(c) \cdot \chi_{(\rho_1\rho_2, r')}$$

(Arithmetics and definition of $\chi_{B_i}(c)$)

$$\left[\begin{array}{l}\text{Notice that } c \in (\mathrm{supp}(P) - A) \text{ implies } c \in B_i \text{ for some } i. \\ \text{Moreover, sets } B_i,\ 1 \le i \le n_\alpha, \text{ are pairwise disjoint. As} \\ \text{a consequence, if } c \in (\mathrm{supp}(P) - A), \sum_{i=1}^{n_\alpha} \chi_{B_i}(c) = 1.\end{array}\right]$$

$$= \sum_{c\in A} P(c) \cdot \sum_{(\rho',r')\in \mathrm{supp}(\mu'_c)} \mu'_c(\rho', r') \cdot (p_\alpha \circ (\lambda i.(\rho', t_\alpha(r')[i]))^{-1}) + \sum_{\substack{c\in (\mathrm{supp}(P)-A),\ \neg(\alpha{\downarrow}\rho_1) \\ c=(\rho_1(\alpha,k)\rho_2, r')}} P(c) \cdot \chi_{(\rho_1\rho_2, r')}$$

(Arithmetics and previous observation)

$$= \sum_{c\in A} P(c) \cdot \mu_c + \sum_{c\in (\mathrm{supp}(P)-A)} P(c) \cdot \mu_c$$

(Properties of μ_c, see (1) and (2))

$$= \sum_{c\in \mathrm{supp}(P)} P(c) \cdot \mu_c$$

This proves the item 1 in the theorem. For the second item, by Lemma 6, ensures that for all $\langle s, P\rangle \in \mathcal{R}$ and $(\rho, r) \in \mathrm{supp}(P)$, $s \stackrel{\rho}{\rightsquigarrow} r$. If $\rho \neq \epsilon$, $s \stackrel{\beta}{\longrightarrow} \nu$ for some ν provided β is the first operation in ρ. If $\rho = \epsilon$, $s = r$. In any case, item 2 holds. □

Lemma 8 relates the weak transition for pairs of paths and states defined in Table 2, and the weak transition for states defined in Table 1. The proof (which we omit) uses induction on the proof tree of $\Longrightarrow_\alpha$.

Lemma 8. *If $(\rho, r) \stackrel{\alpha}{\Longrightarrow} \mu$ in a (reduced) program P then $r \stackrel{\alpha}{\Longrightarrow} \mu \circ (\lambda(\rho', r').r')^{-1}$ in P_{red}.*

Voilá! The main theorem:

Theorem 3. *P and P_{red} are complete forward simulation equivalent.*

Proof. Let $\mathcal{S} = \{(s, P \circ (\lambda(\rho, r).r)^{-1}) \mid (s, P) \in \mathcal{R}\}$. Since $\langle \mathsf{ini}, \chi_{(\epsilon,\mathsf{ini})}\rangle \in \mathcal{R}$, $\langle \mathsf{ini}, \chi_{\mathsf{ini}}\rangle \in \mathcal{S}$. In the following, we show that $\mathcal{S}$ is a complete forward simulation.

Let $\langle s, Q\rangle \in \mathcal{S}$ and $s \stackrel{\alpha}{\longrightarrow} \mu_1$. Then there is $\langle s, P\rangle \in \mathcal{R}$ with $Q = P \circ (\lambda(\rho, r).r)^{-1})$. By Lemma 7.1,

(a) for all $c \in \mathrm{supp}(P)$, $c = (\rho, r)$, either
   i. $\exists \nu_c : c \stackrel{\alpha}{\Longrightarrow} \nu_c$, or
   ii. $\exists \rho_1, \rho_2, i : \neg(\alpha{\downarrow}\rho_1) \wedge \rho = \rho_1(\alpha, i)\rho_2 : \nu_c = \chi_{(\rho_1\rho_2, r)}$, and

(b) exists $\nu'_2 \in \mathsf{Dist}(\mathsf{Dist}((\mathsf{T} \times \mathbb{N})^* \times \mathsf{S}_{red}))$ with $\mu_1 \sqsubseteq_{\mathcal{R}} \nu'_2$,

s.t. $\sum_{c\in \mathrm{supp}(P)} P(c) \cdot \nu_c = \sum_{P'\in \mathrm{supp}(\nu'_2)} \nu'_2(P') \cdot P'$.

Let $r \in \mathrm{supp}(Q)$. Then there is ρ such that $(\rho, r) \in P$. If (a)i is the case for (ρ, r), then $r \stackrel{\widehat{\alpha}}{\Longrightarrow} \mu_r$ with $\mu_r = \nu_{(\rho,r)} \circ (\lambda(\rho', r').r')^{-1}$ by Lemma 8. If (a)ii is the case, α must be invisible since ρ induces a forming path. Hence $r \stackrel{\widehat{\alpha}}{\Longrightarrow} \mu_r$ with $\mu_r = \chi_r = \nu_{(\rho,r)} \circ (\lambda(\rho', r').r')^{-1}$.

Let μ'_2 be defined for all $Q' \in \mathsf{Dist}(\mathsf{S})$ by

$$\mu'_2(Q') = \textstyle\sum_{Q'=P'\circ(\lambda(\rho',r').r')^{-1}} \nu'_2(P').$$

If w is the weight function that determines $\mu_1 \sqsubseteq_{\mathcal{R}} \nu'_2$, let w' be defined by $w'(s', Q') = \sum_{Q'=P'\circ(\lambda(\rho',r').r')^{-1}} w(s', P')$. Notice that w' enjoys the properties of a weight function for $\mathcal{S}$:

Proceedings of the First International Conference on the Quantitative Evaluation of Systems (QEST'04)
0-7695-2185-1/04 $ 20.00 IEEE

1. Let $s' \in \mathsf{S}$ then

$$\sum_{Q' \in \mathsf{Dist}(\mathsf{S})} w'(s', Q') = \sum_{Q' \in \mathsf{Dist}(\mathsf{S})} \sum_{Q' = P' \circ (\lambda(\rho', r').r')^{-1}} w(s', P') = \sum_{P' \in \mathsf{Dist}((\mathsf{T} \times \mathbb{N})^* \times \mathsf{S})} w(s', P') = \mu_1(s')$$

2. Let $Q' \in \mathsf{Dist}(\mathsf{S})$ then

$$\sum_{s' \in \mathsf{S}} w'(s', Q') = \sum_{s' \in \mathsf{S}} \sum_{Q' = P' \circ (\lambda(\rho', r').r')^{-1}} w(s', P') = \sum_{Q' = P' \circ (\lambda(\rho', r').r')^{-1}} \sum_{s' \in \mathsf{S}} w(s', P') = \sum_{Q' = P' \circ (\lambda(\rho', r').r')^{-1}} \nu_2'(P') = \mu_2'(Q')$$

3. Let $w'(s', Q') > 0$. Then, $w(s', P') > 0$ for some P' such that $Q' = P' \circ (\lambda(\rho', r').r')^{-1}$. Since $\langle s', P' \rangle \in \mathcal{R}$ because w is a weight function, $\langle s', Q' \rangle \in \mathcal{S}$ by definition of $\mathcal{S}$.

As a consequence $\mu_1 \sqsubseteq_{\mathcal{S}} \mu_2'$. Finally, we calculate:

$$\begin{aligned}
&\textstyle\sum_{r \in \mathsf{supp}(Q)} Q(r) \cdot \mu_r \\
&= \textstyle\sum_{r \in \mathsf{supp}(Q)} (P \circ (\lambda(\rho, r).r)^{-1}))(r) \cdot (\nu_{(\rho,r)} \circ (\lambda(\rho', r').r')^{-1}) && \text{(Def. of } Q \text{ and } \mu_r\text{)} \\
&= \left(\textstyle\sum_{(\rho,r) \in \mathsf{supp}(P)} P(\rho, r) \cdot \nu_{(\rho,r)}\right) \circ (\lambda(\rho', r').r')^{-1} \\
&= \left(\textstyle\sum_{P' \in \mathsf{supp}(\nu_2')} \nu_2'(P') \cdot P'\right) \circ (\lambda(\rho', r').r')^{-1} && \text{(Def. of } \mathcal{R}\text{)} \\
&= \textstyle\sum_{P' \in \mathsf{supp}(\nu_2')} \nu_2'(P') \cdot (P' \circ (\lambda(\rho', r').r')^{-1}) \\
&= \textstyle\sum_{Q' \in \mathsf{supp}(\mu_2')} \sum_{Q' = P' \circ (\lambda(\rho', r').r')^{-1})} \nu_2'(P') \cdot (P' \circ (\lambda(\rho', r').r')^{-1}) \\
&= \textstyle\sum_{Q' \in \mathsf{supp}(\mu_2')} \mu_2'(Q') \cdot Q' && \text{(Def. of } \mu_2'\text{)}
\end{aligned}$$

This proves the first transfer property of Definition 3. Since $\langle s, Q \rangle \in \mathcal{S}$ implies that $\langle s, P \rangle \in \mathcal{R}$ and $Q = P \circ (\lambda(\rho, r).r)^{-1}$, the second transfer property follows immediately by Lemma 7.2. This shows that $\mathcal{S}$ is a complete forward simulation from P to P_{red}.

Since P_{red} is a reduction of P, there is an obvious injective homomorphism from P_{red} to P. Such a morphism defines the other complete forward simulation (the proof is routine). □

A concurrent probabilistic program P determines a trace (a standard trace, not a Mazurkiewicz trace) by collecting the sequence of sets of atomic propositions $\mathsf{L}(s)$ of each state s that is traversed by the execution, but only up to visibility (i.e. consecutive equal sets are omitted). A *trace distribution* is a probability measure μ on traces. Trace distributions can be retrieved from P by resolving the nondeterminism, e.g. by using adversaries [22]. Therefore, a probabilistic program P defines a set $TD(\mathsf{P})$ of distributions on traces. (A formal definition of trace distribution can be found in [18].)

Since complete forward simulation is strictly stronger than forward simulation which in turns preserves trace distributions [18], we have:

Corollary 1. *For all concurrent probabilistic program* P, $TD(\mathsf{P}) = TD(\mathsf{P}_{red})$.

As a consequence the reduction technique proposed in this paper preserves the maximum probability of reaching a particular property.

5. Further Discussions

Probabilistic linear temporal logics. De Alfaro showed that quantitative LTL model checking can be reduced to a quantitative reachability problem using an automata theoretic approach [5]. This enables the use of our technique for LTL model checking. However, states to be reached are those enclosed within so called strongly connected stable sets (SCSS) [5]. Unfortunately, finding SCSS may require to construct first the complete graph (in contrast to obtaining the reduced graph directly from the model specification). While the reduction may still be useful to produce smaller linear optimisation problems, this make our approach less appealing.

Instead, it would be desirable to avoid the construction of the complete graph as a whole and to directly obtain the reduced one from the syntactic specification of the model as it is done in non-probabilistic partial order reduction. For this, however, it would be necessary that the reduction algorithm preserves the probabilistic measures of LTL properties.

LTL formulas are only measurable (that is, they only make sense) in complete trace distributions. (Weak or stuttering) complete trace distribution equivalence preserves the maximum and minimum probabilities of next-free LTL properties (i.e., LTL formulas not including the next operator). Corollary 1, which states that reduction preserves (non-complete) trace distributions, is a consequence of Segala's execution correspondence theorem [18, Ch. 8]. This theorem states that probabilistic forward simulation preserves probabilistic executions (but not necessarily *complete* executions), and therefore trace distributions. Because of Theorem 3, a variant of the execution correspondence theorem in a complete setting would ensure that our reduction preserves *complete* trace distribution and hence quantitative next-free LTL properties. Unfortunately, the authors are not aware of such a theorem. Nevertheless, we conjecture its validity and consequently the preservation of next-free LTL properties by the reduction.

Probabilistic branching time analysis. The algorithm we propose *does not suffice* to model check probabilistic

Proceedings of the First International Conference on the Quantitative Evaluation of Systems (QEST'04)
0-7695-2185-1/04 $ 20.00 IEEE

branching time temporal logics. In fact, it cannot be used even to check the subset of PCTL without existential modalities nor negations except in front of atomic propositions.: Formula $\mathsf{AF}_{\geq.9}\ \mathsf{AF}_{\geq.8}$ ☺ is not satisfied by the program in Figure 4 (left) but it does by the respective reduced program in Figure 4, right.

The obvious question is then: which is (are) the extra condition(s) required for partial order reduction on model checking branching time probabilistic temporal logics?

About implementations. We have not yet worked in an actual implementation of the reduction algorithm. However we expect to obtain reductions comparable to those for branching time logics (see [7]).

Notice that the approach in this paper is orthogonal to the refinement techniques of [3, 4] for stepwise approximation of quantitative reachability properties as well as with bisimulation quotienting. Therefore, partial order reduction can be incorporated in RAPTURE [11] as follows. First apply partial order reduction to obtain the reduced program, since (i) it relies on the syntactic structure (to check **C1**), and (ii) it on-the-fly constructs the reduced program. Next, refinement techniques can be applied in the smaller program to successively approximate the actual probability value of the property.

Conclusion. We presented a partial order reduction algorithm for concurrent probabilistic programs. We stated the correctness of the algorithm by exposing a complete forward simulation relation from the original program to the reduce one. As a consequence, reduction preserves maximum probabilities of reachability properties.

However, Theorem 3 presents a result significantly stronger than the needed to ensure upper bounds for quantitative reachability. We do hope this result can also ensure preservation of maximum and minimum probabilities of next-free LTL properties.

A final question that arises after our result is how could it be applied to other quantitative analysis such as steady state analysis, and to which extend it can also be used on the analysis of Markov reward decision processes [17].

References

[1] C. Baier, M. Größer, and F. Ciesinski. Partial order reduction for probabilistic systems. In *Proc. of QEST 2004*. IEEE Press, 2004. (This volume).

[2] A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In *Proc. of 15th FST&TCS, LNCS 1026*, pp. 499–513. Springer, 1995.

[3] P. D'Argenio, B. Jeannet, H. Jensen, and K. Larsen. Reachability analysis of probabilistic systems by successive refinements. In *Proc. of PAPM/PROBMIV 2001, LNCS 2165*, pp. 39–56. Springer, 2001.

[4] P. D'Argenio, B. Jeannet, H. Jensen, and K. Larsen. Reduction and refinement strategies for probabilistic analysis. In *Proc. of PAPM/PROBMIV 2002, LNCS 2399*. Springer, 2002.

[5] L. de Alfaro. Temporal logics for the specification of performance and reliability. In *Proc. of STACS'97, LNCS 1200*, pp. 165–176. Springer, 1997.

[6] V. Diekert and G. Rozenberg, eds. *The Book of Traces*. World Scientific, 1995.

[7] R. Gerth, R. Kuiper, D. Peled, and W. Penczek. A partial order approach to branching time logic model checking. *Inf' & Comp.*, 150(2):132–152, 1999.

[8] P. Godefroid. *Partial-Order Methods for the Verification of Concurrent Systems, LNCS 1032*. Springer, 1996.

[9] G. Holzmann and D. Peled. An improvement in formal verification. In D. Hogrefe and S. Leue, eds., *Proc. of FORTE'94*, pp. 197–211. North-Holland, 1994.

[10] M. Huhn, P. Niebert, and H. Wehrheim. Partial order reduction for bisimulation checking. In *Proc. of 18th FST&TCS, LNCS 1530*, pp. 271–282. Springer, 1995.

[11] B. Jeannet, P. D'Argenio, and K. Larsen. RAPTURE: A tool for verifying Markov decision processes. In *Proc. of Tools Day at CONCUR 2002*, 2002. Web page: `www.irisa.fr/prive/bjeannet/prob/prob.html`.

[12] B. Jonsson and K. Larsen. Specification and refinement of probailistic processes. In *Proc. of* 6^{th} *LICS*, pp. 266–277. IEEE Press, 1991.

[13] M. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic symbolic model checker. In *Proc. of TOOLS'2002, LNCS 2324*. Springer, 2002. Web page: `www.cs.bham.ac.uk/~dxp/prism/`.

[14] K. Larsen and A. Skou. Bisimulation through probabilistic testing. *Inf' & Comp.*, 94:1–28, 1991.

[15] P. Niebert and W. Penczek. On connection of partial order logics and partial order reduction methods. Technical report, Tech. Univ. of Eindhoven, 1995.

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

[17] M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley, 1994.

[18] R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, MIT, 1995.

[19] M. Stoelinga. *Alea jacta est: Verification of Probabilistic, Real-time and Parametric Systems*. PhD thesis, University of Nijmegen, 2002.

[20] A. Valmari. Stubborn sets for reduced state space generation. In *Proc. of the 10th Int. Conf. on App. and Theory of Petri Nets, LNCS 483*, pp. 491–515. Springer, 1989.

[21] D. Varacca and M. Nielsen. Probabilistic Petri nets and mazurkiewicz equivalence, 2003. Unpublished draft.

[22] M. Vardi. Automatic verification of probabilistic concurrent finite state programs. In *Proc. of* 26^{th} *FOCS*, pp. 327–338. IEEE Press, 1985.

Proceedings of the First International Conference on the Quantitative Evaluation of Systems (QEST'04)
0-7695-2185-1/04 $ 20.00 IEEE