CHAPTER 5
Probabilistic Bisimulation and Simulation

Abstract We consider probabilistic systems with nondeterminism and study several simulation and bisimulation relations on them. The purpose of this chapter is twofold. First, we give an overview of several types of *strong* and *weak (bi–)simulations* known in the literature. Second, we introduce a novel type of (bi–)simulation, called *delay (bi–)simulation* and show that it can be characterized by means of *norm functions* [GV98]. We discuss the connection between the several (bi–)simulation relations that result from the use of randomized versus deterministic adversaries and combining finitely versus infinitely many internal steps. While for the weak (bi-)simulations, the decidability is still an open problem, we present algorithms to decide the delay (bi-)simulation relations that run in polynomial time and space.

5.1 Introduction

Correctness proofs for concurrent systems are often based on simulation and bisimulation relations. These have been developed for and applied to various types of systems [LV95, AL91, RE98], including timed systems [LV96b] and probabilistic systems [LS91, HJ94, SL95].

Simulations and bisimulation relations are — for each type of systems mentioned — relations on the state space of the system that compare the stepwise behavior of related states. As we will formally define later in this chapter, *bisimulation relations* are equivalences on the state space of a system such that two related states exhibit the same stepwise behavior. In other words, states that are related via a bisimulation relation can be regarded as equivalent. *Simulation relations* are preorders $\sqsubseteq$ on the state space such that if $s \sqsubseteq t$, then t can mimic all stepwise behavior of s. The converse need not be true, so t may perform steps that cannot be mimicked by s. If we have $s \sqsubseteq t$, then we can say that s behaves as good as t. One can also use (bi–)simulation to compare systems, rather than states, by defining that two systems are (bi–)simulation if their start states are so.

Different notions of (bi–)simulation exist, depending on what one considers as the system's stepwise behavior. Typically, one distinguishes between strong and weak (bi–)simulation. Weak (bi–)simulation abstracts from internal computations, whereas strong (bi–)simulation takes all steps of the system into account, including internal ones.

This chapter focuses on probabilistic systems. We extensively study various simulation and bisimulation relations for these systems. We formulate the theory in a minor variation of the framework presented in Chapter 2. As before, the systems incorporate both probabilistic and nondeterministic choice. We consider several known (bi–)simulations from the literature for these systems, namely, strong, strong combined and weak (bi–)simulation. Moreover, we introduce a novel type of relations, which we call *delay (bi–)simulation*. We define two variants of them, *finitary* and ω–(bi–)simulation.

There are several ways in which simulation techniques can be used in verification proofs. We discuss the two most important ones. Firstly, simulation relations have proven to be very successful in showing that a lower level description of a system correctly implements a higher level specification of it. Chapter 2 proposed $\sqsubseteq_{\mathrm{PTD}}$ as an implementation relation for probabilistic labeled transition systems. That is, a probabilistic system $\mathcal{A}$ correctly implements another one $\mathcal{B}$ if and only if $\mathcal{A} \sqsubseteq_{\mathrm{PTD}} \mathcal{B}$. However, a direct proof showing $\mathcal{A} \sqsubseteq_{\mathrm{PTD}} \mathcal{B}$ is often very complex because it involves reasoning about the *global* behavior of two systems $\mathcal{A}$ and $\mathcal{B}$ in terms of trace distributions. Simulation relations, instead, involve reasoning about the *local* behavior of a system, that is, in terms of states and transitions. Establishing $\mathcal{A} \sqsubseteq \mathcal{B}$ is therefore often far more easy than establishing $\mathcal{A} \sqsubseteq_{\mathrm{PTD}} \mathcal{B}$. And since the former implies the latter, simulation techniques are sound for proving $\mathcal{A} \sqsubseteq_{\mathrm{PTD}} \mathcal{B}$ and often reduce the verification effort drastically.

Secondly, bisimulation equivalences $\approx$ can be used to reduce a system to a smaller one. To do so, one replaces each state in a system S by its equivalence class in $\approx$, yielding a new system $S/\!\approx$. Then one can verify $S/\!\approx$ instead of S, provided that the bisimulation $\approx$ preserves the properties one verifies. This is a gain, because $S/\!\approx$ is often significantly smaller that S, especially for weak bisimulation relations. Moreover, bisimulation relations do preserve many properties of interest, so this reduction technique can be used in many cases.

To be applicable in verification, the relations have to meet certain properties. Firstly, as we have already seen, the use of simulation relations depends on their soundness for establishing $\sqsubseteq_{\mathrm{PTD}}$. Similarly, we have seen that the use of bisimulations depends on their preservation of the properties one wishes to verify. We have also seen that the use of (bi–)simulation is based on the fact that they are relatively easy to establish. This can be done with deductive methods to be carried out by hand, but preferably, a (bi–)simulation is decidable. This opens the way to automatic verification, which is both faster and more reliable than manual proofs. Finally, we want the (bi–)simulations to support two important techniques for keeping a verification process manageable. That is, they should allow for *modular verification*, in which one derives the correctness of the whole system from the correctness of its components. Moreover, they should allow for *stepwise verification*, in which one derives the correctness of a system via several intermediate systems. Technically, this means that a (bi–)simulation relation should be transitive and substitutive (with respect to parallel composition). The former is trivial when we consider states, but not when comparing systems.

At the end of this chapter, we will discuss to what extend the relations we study meet the properties that make them useful in verification.

Related work Apart for bisimulation and simulation relations, several other types of preorders and equivalences for probabilistic systems have been proposed in the literature. Most of the standard relations defined for nonprobabilistic systems have been extended to the probabilistic ones: see e.g. [Seg95a] for a trace–based relation, [YL92, JY01] for testing equivalences and [LS91, HJ90, Han91, Yi94, SL95, Seg95a, SV99b] for several types of (bi–)simulations. In the fully probabilistic setting, that is, for systems with probabilistic choice only, the equivalences are studied under several aspects (compositionality, axiomatization, decidability, logical characterizations, etc.), see e.g. [JS90, CC92, BJ91, HT92, KL92, Chr90b, BH97].

On the other hand, the treatment of equivalences and preorders for probabilistic systems is less well–understood. Due to the combination of nondeterminism and probability, the definitions are more complicated than the corresponding notions for nonprobabilistic 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 less common. For strong bisimulation [LS91] and strong simulation [SL95], polynomial-time algorithms have been presented in [BEM00]. To the best of our knowledge, the work [PSL98] is the first attempt to formulate an algorithmic method that deals with a weak equivalence for probabilistic processes with nondeterminism. Decidability of weak bisimulation à la [SL95] was an open problem for a long time; it was conjectured to be decidable in exponential time. Segala [SC01] has proposed a polynomial time algorithm that decides weak bisimulation. A publication of these results is currently in preparation. We are not aware of any complexity (or even decidability) result for any linear time relation on probabilistic systems with nondeterminism, e.g. trace distribution equivalence [Seg95a]. Obviously, since the trace distribution preorder à la Segala is a conservative extension of usual trace inclusion, the PSPACE–completeness result for trace inclusion of nonprobabilistic systems [KS90] yields a PSPACE–hardness for the trace distribution preorder.

Our contribution: We present a comprehensive study of several notions of strong, combined strong and weak (bi–)simulation in the probabilistic setting. We simplify existing definitions and present several important results known from the literature, such as compositionality and decidability, in a single framework.

The main contribution of our work is, however, the introduction of novel notions of finitary and ω (bi–)simulation. These relations are (in some sense) insensitive to internal transitions. Moreover, they are conservative extensions of *delay bisimulation* equivalence [Wei89, Gla93] for nonprobabilistic systems. This equivalence relies on the assumption that the simulation of a step of a system $\mathcal{A}$ by another process $\mathcal{B}$ might happen with a certain delay (i.e. after a sequence of internal transitions). Our notions of delay bisimulation can be characterized by a probabilistic variant of *norm functions* in the style of [GV98]. Intuitively, the norm functions specify bounds on the number of internal transitions that can be used by a step of $\mathcal{B}$ when simulating a step of $\mathcal{A}$. In the probabilistic setting, where the combination of internal transitions leads to a tree rather than a linear chain, the norm functions yield conditions on the length of the paths in the trees corresponding to a "delay transition". Using a modification of the traditional splitter/partitioning technique [KS90, PT87], we present polynomial time algorithms for computing the quotient spaces. We briefly discuss some other aspects (compositionality w.r.t. parallel composition and preservation of linear time properties).

Organization of the chapter: In Section 5.2, we present some notations concerning relations, partitions and distributions that we use throughout the chapter. Section 5.3 introduces the model for probabilistic systems which we consider and its behavior. Section 5.4 recalls the definition of strong bisimulation [LS91] and the several variants of weak and branching bisimulations [SL95, Seg95b]. The novel delay (bi–)simulation relations and their characterization by norm functions are presented in Section 5.5. Section 5.6 treats several important properties, including compositionality and the interconnection between the (bi–)simulation relations. Section 5.7 summarizes the results on the decidability of the various relations that have been established in the literature and presents polynomial-time algorithms for computing the delay (bi–)simulation relations. In the conclusion (Section 5.8), we summarize the results and present some conclusions.

5.2 Preliminaries

This section briefly summarizes some standard notations that are used throughout this chapter.

The domain of a partial function: Let X, Y be sets. We denote the set of partial functions from X to Y by $X \hookrightarrow Y$. For a partial function $f : X \hookrightarrow Y$, let $\mathrm{Dom}(f)$ denote the *domain* of f, that is the set of all $x \in X$ such that $f(x)$ is defined.

Sequences: Let X be a set. We denote the set of finite sequences over X by X^*, the set of infinite sequences over X by X^ω and the set $X^* \cup X^\omega$ by X^∞. The empty sequence is written by ε. We denote the prefix relation on X^∞ by $\sqsubseteq^X$ and the proper prefix relation by $\sqsubset^X$. When X is clear from the context, we simply write $\sqsubseteq$ and $\sqsubset$.

Equivalences: If R is an equivalence relation on a set X, then X/R denotes the quotient space of X with respect to R and $[x]_R$ denotes the equivalence class of x with respect to R.

Partitions: Let X be a nonempty finite set and x, $x' \in X$. A *partition* of X is a set $\chi = \{X_1, \ldots, X_l\}$ consisting of pairwise disjoint, nonempty subsets of X such that $X_1 \cup \ldots \cup X_l = X$. The elements X_i of χ are called *blocks* of χ and $[x]_\chi$ denotes the unique block X_i of χ that contains x.

Clearly, there is a one–to–one correspondence between the equivalences on X and the partitions of X. For any equivalence relation R on X, the quotient space X/R is a partition of X. Vice versa, each partition χ induces an equivalence R_χ by identifying those elements that are contained in the same block of χ. A partition χ of X is called *finer* than a partition χ' of X (in which case we also say that χ' is *coarser* than χ) iff R_χ is finer than $R_{\chi'}$, which means that $R_{\chi'} \subseteq R_\chi$, or equivalently, any block in χ' can be written as a union of blocks in χ. The partition χ is called *strictly finer* than χ' (or χ' *strictly coarser* than χ) if χ is finer than but not equal to χ'.

Distributions: Let X be a nonempty set. A *(probability) distribution* on X is a function $\mu : X \to [0, 1]$ such that $\mu(x) > 0$ for countably[1] many elements $x \in X$ and $\sum_{x \in X} \mu(x) = 1$. Let $supp(\mu)$ denote the *support* of μ, i.e. the set of elements $x \in X$ with $\mu(x) > 0$. If $\emptyset \neq A \subseteq X$ then $\mu[A] = \sum_{x \in A} \mu(x)$. Moreover, we put $\mu[\emptyset] = 0$. For $x \in X$, μ^1_x denotes the unique distribution on X with $\mu(x) = 1$, which is called the *Dirac distribution* over x. If $(p_i)_{i \in I}$ is a countable family of real numbers $p_i \in [0, 1]$ such that $\sum_{i \in I} p_i = 1$ and $(\mu_i)_{i \in I}$ a family of distributions on X, then the distribution given by $x \mapsto \sum_{i \in I} p_i \cdot \mu_i(x)$ is called a *convex combination* of the distributions μ_i, $i \in I$. We denote the collection of all distributions on X by $Distr(X)$.

The equivalence $\equiv_R$: If R is an equivalence on X, then the induced equivalence $\equiv_R$ on $Distr(X)$ is given by: $\mu \equiv_R \mu'$ iff $\mu[C] = \mu'[C]$ for all $C \in X/R$.

[1] In our definition, the countable sets subsume the finite sets.

Weight functions: Let R be a binary relation on X and $\mu, \mu' \in \mathit{Distr}(X)$. A *weight function* for (μ, μ') with respect to R is a function $wgt : X \times X \to [0, 1]$ such that $wgt(x, x') > 0$ implies $(x, x') \in R$ and, for all $x, x' \in X$:

$$\sum_{y' \in X} wgt(x, y') \;=\; \mu(x), \qquad \sum_{y \in X} wgt(y, x') \;=\; \mu'(x').$$

Intuitively, a weight function shows how we can distribute the probability $\mu(x)$ among the related states x' in such a way that $\mu'(x')$ equals the total amount of probability gets distributed this way. In fact, the function wgt is a probability distribution on $X \times X$ such that the probability to pick an element (x, x') in R is one, the probability to pick an element whose first component is x equals $\mu(x)$ and the probability to pick an element whose second component is x' equals $\mu'(x')$.

The relation $\sqsubseteq_R$ and the set $\mu\!\uparrow_R$: We write $\mu \sqsubseteq_R \mu'$ iff there exists a weight function for (μ, μ') with respect to R. We define $\mu\!\uparrow_R = \{\mu' \in \mathit{Distr}(X) : \mu \sqsubseteq_R \mu'\}$. We call $\mu\!\uparrow_R$ the *cone* above μ w.r.t. $\sqsubseteq_R$.

The following proposition collects some elementary properties of the relations $\equiv_R$ and $\sqsubseteq_R$.

Proposition 5.2.1 *Let μ, μ', $\mu'' \in \mathit{Distr}(X)$ and R, R_1, R_2 binary relations on X.*

1. *If R is an equivalence relation, then $\mu \equiv_R \mu' \iff \mu \sqsubseteq_R \mu'$.*
2. *If $\mu \sqsubseteq_{R_1} \mu'$ and $\mu' \sqsubseteq_{R_2} \mu''$, then $\mu \sqsubseteq_{R_1 \circ R_2} \mu''$.*[2]
3. *If $\mu \sqsubseteq_R \mu'$, then $\mu' \sqsubseteq_{R^{-1}} \mu$.*[3]
4. *Let $\mu = \sum_{i \in I} p_i \cdot \mu_i$ be a convex combination over the distributions μ_i. If $\mu_i \sqsubseteq_R \mu'$ for all $i \in I$, then $\mu \sqsubseteq_R \mu'$.*
5. *If $R_1 \subseteq R_2$, then $\mu \sqsubseteq_{R_1} \mu' \implies \mu \sqsubseteq_{R_2} \mu'$.*

PROOF:

1. ($\implies$) Let $\mu \equiv_R \mu'$. Define a function wgt for (μ, μ') with respect to R as follows. If $(x, x') \notin R$ then put $wgt(x, x') = 0$. If $(x, x') \in R$ then put

$$wgt(x, x') \;=\; \frac{\mu(x) \cdot \mu'(x')}{\mu[[x]_R]}.$$

   Using that $\mu[[x]_R] = \mu'[[x]_R]$ for each x, one easily shows that wgt is indeed a weight function for (μ, μ').

[2]Recall that $R_1 \circ R_2$ consists of all pairs (x, x'') where $(x, x') \in R_1$ and $(x', x'') \in R_2$ for some $x' \in X$.
[3]Recall that R^{-1} consists of all pairs (x', x) where $(x, x') \in R$.

($\Longleftarrow$) Assume $\mu \sqsubseteq_R \mu'$. Let wgt be a weight function for (μ, μ'). Since $wgt(x, x') = 0$ for $(x, x') \notin R$, we have $\mu(x) = \sum_{y' \in C} wgt(x, y')$ and $\mu'(x') = \sum_{y \in C} wgt(y, x')$ for all x, x' and C with $x, x' \in C$. From this, we get

$$\begin{aligned}\mu[C] = \sum_{x \in C} \mu(x) \ &= \ \sum_{x \in C} \sum_{x' \in C} \ wgt(x, x') \ = \ \sum_{x' \in C} \sum_{x \in C} \ wgt(x, x') \\ &= \ \sum_{x' \in C} \ \mu'(x') = \mu'[C].\end{aligned}$$

Thus, $\mu \equiv_R \mu'$.

2. Let wgt_1 and wgt_2 be weight functions for (μ, μ') with respect to R_1 and for (μ', μ'') with respect to R_2 respectively. It is easy to see that

$$wgt(x, x'') \ = \ \sum_{x' \in supp(\mu')} \frac{wgt_1(x, x') \cdot wgt_2(x', x'')}{\mu'(x')}$$

is a weight function for (μ, μ'') with respect to $R_1 \circ R_2$.

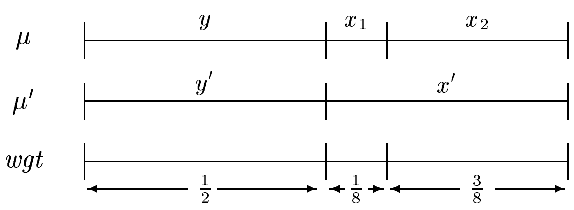
3–5. Easy verification.

□

Example 5.2.2 Let $X = \{x_1, x_2, x', y, y'\}$ and let R be the smallest equivalence relation on X which identifies the elements x', x_1 and x_2 and the elements y and y' (i.e. $X/R = \{C_x, C_y\}$ where $C_x = \{x_1, x_2, x'\}$ and $C_y = \{y, y'\}$). Let μ and μ' be the distributions with

$$\mu'(x') = \mu'(y') = \tfrac{1}{2} \ \text{ and } \ \mu(x_1) = \tfrac{1}{8}, \mu(x_2) = \tfrac{3}{8}, \mu(y) = \tfrac{1}{2} \ .$$

Then, $\mu[C_x] = \mu(x_1) + \mu(x_2) + \mu(x') = \frac{1}{2} = \mu'(x_1) + \mu'(x_2) + \mu'(x') = \mu'[C_x]$ and $\mu[C_y] = \mu(y) + \mu(y') = \frac{1}{2} = \mu'(y) + \mu'(y') = \mu'[C_y]$. Thus, $\mu \equiv_R \mu'$ and therefore also $\mu \sqsubseteq_R \mu'$. A weight function for (μ, μ') with respect to R is given by $wgt(y, y') = 1/2$, $wgt(x_1, x') = 1/8$, $wgt(x_2, x') = 3/8$ (and $wgt(\cdot) = 0$ in all other cases), see the picture below.



5.3 Probabilistic Systems

This section introduces the probabilistic systems we study and their behavior. The basic ideas behind this model have been explained in Chapter 2. This chapter treats more technical details, although several notions are repeated.

Throughout the paper, we deal with a fixed set of action symbols including a special action symbol τ that stands for any internal activity, i.e. any activity that is not observable by the environment.

Notation 5.3.1 (The action set Act_τ) Let Act_τ be a finite set of actions containing a special action τ (called the *internal* action). The other actions are called *visible*. We write $Act = Act_\tau \setminus \{\tau\}$. We define an embedding $\widehat{\ }: Act_\tau \to Act_\varepsilon$, by $\widehat{\tau} = \varepsilon$ and $\widehat{a} = a$ for any visible action $a \in Act$.

Definition 5.3.2 (Probabilistic system) A *probabilistic system* $\mathcal{S}$ is a pair (S, Steps) where S is a finite[4] set of states and

$$\mathit{Steps} : S \to 2^{Act_\tau \times \mathit{Distr}(S)}$$

a function which assigns to each state $s \in S$ a finite set $\mathit{Steps}(s)$ of pairs (a, μ) consisting of an action $a \in Act_\tau$ and a distribution μ on S. We define $\mathit{Steps}_a(s) = \{\mu : (a, \mu) \in \mathit{Steps}(s)\}$. The function Steps induces a transition relation $\longrightarrow \subseteq S \times Act_\tau \times \mathit{Distr}(S)$ which is defined by $s \xrightarrow{a} \mu$ iff $(a, \mu) \in \mathit{Steps}(s)$. A state s is called *terminal* iff there is no possible next step in s, i.e. if $\mathit{Steps}(s) = \emptyset$. We often write $s \xrightarrow{a} t$ for $s \xrightarrow{a} \mu_t^1$.

The function Steps represents the nondeterministic alternatives in each state: each transition $s \xrightarrow{a} \mu$ represents the possibility to move from state s with an a–action to a next state t, where the probability to reach t is given by $\mu(t)$.

Remark 5.3.3 ((Non-probabilistic) labeled transition systems) Any finite (nonprobabilistic) labeled transition system $(S, \longrightarrow)$ (where $\longrightarrow \subseteq S \times Act_\tau \times S$) can be identified with the probabilistic system (S, Steps) where $\mathit{Steps}(s) = \{(a, \mu_t^1) : s \xrightarrow{a} t\}$.

Remark 5.3.4 (Probabilistic automata) A *probabilistic automaton* is a probabilistic system equipped with an initial state, i.e. a tuple $\mathcal{A} = (S, \mathit{Steps}, s_{init})$ consisting of a probabilistic system (S, Steps) and an initial state $s_{init} \in S$. Sections 5.4 and 5.5 define several relations on the state space S of a probabilistic system. These can easily be adapted to relations on probabilistic automata by defining that two probabilistic automata $\mathcal{A}_1$ and $\mathcal{A}_2$ are related if and only if their initial states are related when we view them as states in disjoint union $\mathcal{A}_1 \uplus \mathcal{A}_2$[5].

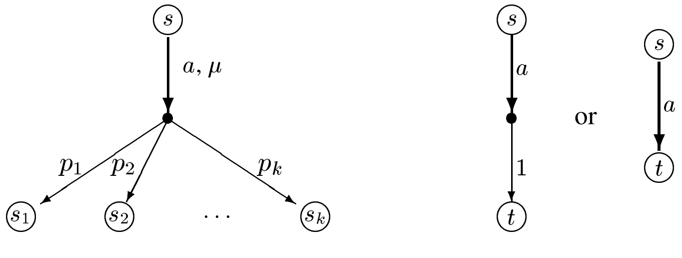


Figure 5.1: The transitions $s \xrightarrow{a} \mu$ and $s \xrightarrow{a} \mu_t^1$

We depict probabilistic systems 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 $s \xrightarrow{a} \mu$

[4]For algorithmic purposes, we restrict our attention to finite probabilistic systems. Most of the results on adversaries and (bi-)simulation relations also hold for infinite systems.

[5]For $\mathcal{A}_i = (S_i, \mathit{Steps}_i, s_i)$, the disjoint union $\mathcal{A}_1 \uplus \mathcal{A}_2$ is the system (S, Steps). Here $S = S_1 \uplus S_2$ is the disjoint union of S_1 and S_2 and $\mathit{Steps} = \mathit{Steps}_1 \cup \mathit{Steps}_2$, where we identify each element in $(a, \mu) \in \mathit{Steps}_i$ with the element $(a, \mu') \in Act_\tau \times \mathit{Distr}(S)$ given by $\mu'(s) = \mu(s)$ if $s \in S_i$ and $\mu'(s) = 0$ if $s \notin S_i$.

is directed and ends in a small circle, which represents the probabilistic choice but which is not a state in the system. The picture on the left of Figure 5.1 stands for a transition $s \xrightarrow{a} \mu$ where $supp(\mu) = \{s_1, \dots, s_k\}$ and $\mu(s_i) = p_i > 0$, $i = 1, \dots, k$. Sometimes we omit the distribution and just write a rather than a, μ. Transitions of the form $s \xrightarrow{a} \mu$ are depicted as shown in the pictures on the right. In our examples, we often depict the unfolding of a probabilistic system into a tree-like structure and duplicate certain states. Moreover, we often omit the outgoing transitions of certain states if they are not important.

Example 5.3.5 [Simple communication system] Figure 5.2 shows a probabilistic system that describes a toy communication system consisting of a sender (which *produces* certain messages and tries to submit the messages via an unreliable medium) and a receiver (which acknowledges the receipt and *consumes* the received messages). The failure rate of the medium is 1%; more precisely, with probability $1/100$ the medium looses the messages in which case the sender resends the message. 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 has been delivered correctly; similarly, the medium cannot be activated 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 has not arrived yet). We use the following four states:

- s_{init}: the state where the sender produces a message and passes the message to the medium.
- s_{del}: the state in which the medium tries to deliver the message (via an internal action).
- s_{ok}: the state reached when the message is delivered correctly.
- s_{ack}: the state in which the receiver consumes the message (i.e. reads and processes the message and acknowledges the receipt).

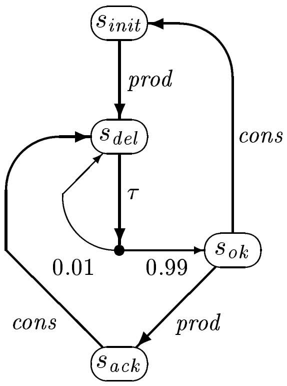


Figure 5.2: The simple communication system

5.3.1 Paths

The executions of probabilistic systems are given by the paths in the underlying nonprobabilistic LTS. Paths in probabilistic systems arise by resolving both the nondeterministic and probabilistic choices. We distinguish between finite paths (representing execution fragments) and fullpaths (representing complete executions). The latter are either finite paths that end up in a terminal state or infinite paths.

Definition 5.3.6 (paths) A *path* of a probabilistic system $\mathcal{S} = (S, Steps)$ is an alternating, finite sequence

$$\sigma \;=\; s_0 \stackrel{a_1,\mu_1}{\leadsto} s_1 \stackrel{a_2,\mu_2}{\leadsto} s_2 \stackrel{a_3,\mu_3}{\leadsto} \ldots \stackrel{a_l,\mu_l}{\leadsto} s_l$$

or an alternating, infinite sequence

$$\sigma \;=\; s_0 \stackrel{a_1,\mu_1}{\leadsto} s_1 \stackrel{a_2,\mu_2}{\leadsto} s_2 \stackrel{a_3,\mu_3}{\leadsto} \ldots$$

where $s_i \in S$, $s_i \xrightarrow{a_{i+1}} \mu_{i+1}$, $s_{i+1} \in supp(\mu_{i+1})$. We often omit the distributions μ_i and just write $\ldots s_{i-1} \stackrel{a_i}{\leadsto} s_i \ldots$. We define the following notations for paths.

- Let $length(\sigma) = l$ if σ is finite and $length(\sigma) = \infty$ otherwise. Thus, the length of a path denotes its number actions.
- Let $first(()\sigma) = s_0$ and, for finite paths, $last(\sigma) = s_l$,
- We can retrieve respectively the i^{th} state, step and action by defining $state(\sigma, i) = s_i$, $Step(\sigma, i) = (a_i, \mu_i)$ and $act(\sigma, i) = a_i$,
- $prefix(\sigma, i)$ denotes the prefix of length i, i.e.

  $$prefix(\sigma, i) \;=\; s_0 \stackrel{a_1,\mu_1}{\leadsto} s_1 \stackrel{a_2,\mu_2}{\leadsto} s_2 \stackrel{a_3,\mu_3}{\leadsto} \ldots \stackrel{a_i,\mu_i}{\leadsto} s_i, \quad \text{if } i < length(\sigma),$$

  and otherwise we put $prefix(\sigma, i) = \sigma$.
- Let $word(\sigma) = a_1 a_2 \ldots$. be the finite or infinite word over Act_τ associated with σ.

Definition 5.3.7 (Fullpaths) A *fullpath* of a probabilistic system $\mathcal{S}$ is either an infinite path or a finite path σ, where $last(\sigma)$ is a terminal state. Let $Path^{\mathcal{S}}_{ful}$ denote the set of all fullpaths in $\mathcal{S}$ and $Path^{\mathcal{S}}_{ful}(s)$ the set of fullpaths starting in s (i.e. with $first(()\pi) = s$). Similarly, $Path^{\mathcal{S}}_{*}$ denotes the set of all finite paths in $\mathcal{S}$ and $Path^{\mathcal{S}}_{*}(s)$ the set of finite paths in $\mathcal{S}$ starting in s.

Example 5.3.8 Consider the probabilistic system in Figure 5.3. This system contains infinitely many finite paths and one infinite path starting in s. The paths starting in s are given by

$$\begin{array}{ll}
s, & s \stackrel{\tau,\nu}{\leadsto} s \stackrel{\tau,\nu}{\leadsto} s \ldots \stackrel{\tau,\nu}{\leadsto} s \stackrel{\tau,\nu}{\leadsto} s' \\
s \stackrel{a,\rho}{\leadsto} t & s \stackrel{\tau,\nu}{\leadsto} s \stackrel{\tau,\nu}{\leadsto} s \ldots \stackrel{\tau,\nu}{\leadsto} s \stackrel{\tau,\nu}{\leadsto} s' \stackrel{a,\mu}{\leadsto} u \\
s \stackrel{\tau,\nu}{\leadsto} s \stackrel{\tau,\nu}{\leadsto} s \ldots \stackrel{\tau,\nu}{\leadsto} s & s \stackrel{\tau,\nu}{\leadsto} s \stackrel{\tau,\nu}{\leadsto} s \stackrel{\tau,\nu}{\leadsto} s \ldots.
\end{array}$$

The only fullpaths starting in s are $s \stackrel{a,\rho}{\leadsto} t$ and $s \stackrel{\tau,\nu}{\leadsto} s \stackrel{\tau,\nu}{\leadsto} s \stackrel{\tau,\nu}{\leadsto} s \ldots$.

5.3.2 Adversaries

Adversaries (also called *schedulers* [Var85] or *policies* in the theory of MDPs) are entities that resolve the nondeterministic choices. Being in a state of the system, an adversary determines the next step to be taken. This step may depend on the history, i.e. the path leading to the current state. Moreover, the adversary may choose randomly which of the possible next steps is chosen. Thus, given a finite path π ending in a state s and each transition $s \xrightarrow{a} \mu$, a randomized partial adversary A yields a probability $A(\pi)(a, \mu)$, specifying the probability with which A schedules $s \xrightarrow{a} \mu$. The remaining value $A(\sigma)(\perp)$ is the probability for an interruption.

We define several special classes of adversaries. *Deterministic* adversaries are adversaries that schedule a unique next step with probability one; *total* adversaries, or just adversaries, schedule a step whenever possible, i.e. only choose $\perp$ in a terminal state of the system. Furthermore, we define the classes of *memoryless*, *finitary* and *almost finitary* adversaries.

Definition 5.3.9 ((Randomized) partial adversary) A *(randomized) partial adversary* of a probabilistic system $\mathcal{S}$ is a function

$$A : Path^* \to \ Distr\,(Act_\tau \times Distr(S) \ \cup \ \{\perp\})$$

such that, for each finite path σ: if $A(\sigma)(a, \mu) > 0$ then $(a, \mu) \in Steps(last(\sigma))$.

Definition 5.3.10 (Deterministic and total adversary) A partial adversary D is called *deterministic* if for all $\sigma \in Path_*^D$, $D(\sigma)$ yields a Dirac distribution. As before, we often write $D(\sigma) = x$ for $D(\sigma) = \mu_x^1$. A *total adversary* is a partial adversary A where $A(\sigma)(\perp) = 0$ for all finite paths σ where $last(\sigma)$ is nonterminal. Let $RAdv_\perp$ denote the set of all randomized partial adversaries, $RAdv$ the set of all randomized adversaries, $DAdv_\perp$ the set of all deterministic partial adversaries and finally $DAdv$ the set of all deterministic adversaries.

Since we deal mainly with partial adversaries, we will often leave out the adjective partial and simply speak of adversaries rather than of partial adversaries.

Definition 5.3.11 (Memoryless adversary) A partial adversary A is called *memoryless* if it depends only on the last state, i.e. $last(\sigma) = last(\sigma') \implies A(\sigma) = A(\sigma')$. Therefore, we write $A(last(\sigma))$ rather than of $A(\sigma)$ if A is memoryless.

Example 5.3.12 Consider Figure 5.3. We illustrate the definitions above by three adversaries.

- The function D on finite paths given by $D(s) = (a, \rho)$ and $D(\sigma) = \perp$ for $\sigma \neq s$ is a deterministic partial adversary. It is not total because $D(s') = \perp$, whereas s' is not terminal.
- Furthermore, define the function A by $A(s)(a, \rho) = \frac{1}{3}$, $A(s)(\tau, \nu) = \frac{1}{3}$, $A(s)(\perp) = \frac{1}{3}$ and $A(\sigma) = \perp$ for $\sigma \neq s$. Then A is a randomized partial adversary.
- Define the deterministic total adversary D_ω by

$$D_\omega(\sigma) = \begin{cases} (\tau, \nu) & \text{if } last(\sigma) = s, \\ (a, \mu) & \text{if } last(\sigma) = s', \\ \perp & \text{otherwise.} \end{cases}$$

Informally, D_ω keeps on scheduling the τ-transition until the τ-loop is exited. This eventually happens with probability 1.

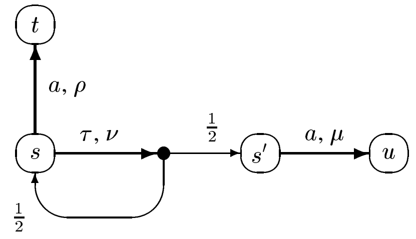


Figure 5.3: Paths in a probabilistic system

5.3.3 Paths in Adversaries

Apart from paths in a system, we can also identify paths in a partial adversary A. Informally, the latter are those paths in a system that are scheduled by A with a positive probability. Thus, these describe the execution sequences which are obtained when the nondeterministic choices are resolved according to A (and the probabilistic choices according to the transitions scheduled). The fullpaths in A are the fullpaths of the system. The maximal paths in A represent "completed" behavior in A and contain the infinite paths and the paths in which A can interrupt the execution (with a positive probability).

Definition 5.3.13 (Paths in partial adversaries) Let A be a partial adversary. A *path in* A is a finite or infinite path

$$\sigma \;=\; s_0 \overset{a_1,\mu_1}{\rightsquigarrow} s_1 \overset{a_2,\mu_2}{\rightsquigarrow} \ldots$$

with $A(\mathit{prefix}(\sigma,i))(\mathit{Step}(\sigma,i+1)) > 0$ for $0 \leq i < \mathit{length}(\sigma)$. A *fullpath* in A is a fullpath of the system that is also path in A. The *maximal paths in* A are the infinite paths in A and the finite paths σ in A where $A(\sigma)(\perp) > 0$. $\mathit{Path}_*^A(s)$ denotes the set of all finite paths σ in A where $\mathit{first}(()\sigma) = s$. Similarly, we define $\mathit{Path}_{max}^A(s)$ (resp. $\mathit{Path}_{*,max}^A(s)$ or $\mathit{Path}_{ful}^A(s)$) as the set of maximal paths (resp. finite maximal paths or fullpaths) in A that start in s.

Note that all fullpaths in A are maximal paths in A, but not conversely. If A is partial, then $A(\sigma)(\perp) > 0$ is possible even if $\mathit{last}(\sigma)$ is nonterminal. Thus, maximality of a finite path σ in a partial adversary does not imply that σ is a fullpath.

Notation 5.3.14 (The generated words $\mathit{Words}(s, A)$) Let A be a randomized partial adversary and $s \in S$. We define $\mathit{Words}(s,A) \;=\; \Big\{ \mathit{word}(\sigma) : \sigma \in \mathit{Path}_{*,max}^A(s) \Big\}$.

Thus, $\mathit{Words}(s, A)$ is the set of all finite words in Act_τ^* which can be generated by a finite maximal path in A that starts in s. These correspond to the action sequences that occur in a finite path (starting in s) which is obtained when the nondeterministic choices are resolved according to A.

Example 5.3.15 Consider the adversaries in Example 5.3.12 again. Their paths are the following.

- The deterministic adversary D contains two paths starting in s: s and $s \overset{a,\rho}{\leadsto} t$. The latter is a maximal fullpath in D, the former is neither maximal nor a fullpath. The only word in $\mathit{Words}(s, D)$ is a.
- The paths in A starting in s are s, $s \overset{a,\rho}{\leadsto} t$, $s \overset{\varepsilon,\nu}{\leadsto} s$, $s \overset{\tau,\nu}{\leadsto} s'$. Each of these is maximal, but $s \overset{a,\rho}{\leadsto} t$ is the only fullpath. We have $\mathit{Words}(s, A) = \{\varepsilon, \tau, a\}$.
- The adversary D_ω contains infinitely many paths starting in s: the paths $s \overset{\tau,\nu}{\leadsto} s \overset{\tau,\nu}{\leadsto} s \ldots s$ and $s \overset{\tau,\nu}{\leadsto} s \overset{\tau,\nu}{\leadsto} s \ldots s \overset{\tau,\nu}{\leadsto} s'$ and the maximal fullpaths $s \overset{\tau,\nu}{\leadsto} s \overset{\tau,\nu}{\leadsto} s \ldots$ and $s \overset{\tau,\nu}{\leadsto} s \overset{\tau,\nu}{\leadsto} s \ldots s \overset{\tau,\nu}{\leadsto} s' \overset{a,\mu}{\leadsto} u$. As only the latter path is finite and maximal, we have $\mathit{Words}(s, D_\omega) = \tau^* a$.

5.3.4 The Probabilistic Behavior of an Adversary

Each adversary defines a probability space on the set of paths.

Notation 5.3.16 (The transition probabilities $\mathbf{P}^A(\cdot)$) Let A be a partial adversary. Define the function $\mathbf{Q}^A : \mathit{Path}_*^A \to [0, 1]$ inductively by

$$\mathbf{Q}^A(s) = 1 \text{ and } \mathbf{Q}^A(\sigma \overset{a,\mu}{\leadsto} t) = \mathbf{Q}^A(\sigma) \cdot A(\sigma)(a, \mu) \cdot \mu(t).$$

Moreover, let $\mathbf{P}^A(\sigma) = \mathbf{Q}^A(\sigma) \cdot A(\sigma)(\perp)$. For s, $t \in S$, $C \subseteq S$, $W \subseteq \mathit{Act}_\tau^*$ and $\alpha = a_1 a_2 \ldots a_l \in \mathit{Act}_\tau^*$ define

$$\begin{aligned}
\mathbf{P}^A(s, \alpha, t) &= \sum_{\sigma \in \mathit{Path}^A_{\mathit{finmax}}(s,\alpha,t)} \mathbf{P}^A(\sigma), \\
\mathbf{P}^A(s, \alpha, C) &= \sum_{t \in C} \mathbf{P}^A(s, \alpha, t) \\
\mathbf{P}^A(s, t) &= \sum_{\sigma \in \mathit{Path}^A_{\mathit{finmax}}(s,t)} \mathbf{P}^A(\sigma), \\
\mathbf{P}^A(s, C) &= \sum_{t \in C} \mathbf{P}^A(s, t) \\
\mathbf{P}^A(s, W, C) &= \sum_{\alpha \in W} \sum_{t \in C} \mathbf{P}^A(s, \alpha, t)
\end{aligned}$$

where $\mathit{Path}^A_{\mathit{finmax}}(s, \alpha, t) = \{\sigma \in \mathit{Path}^A_{\mathit{finmax}}(s) : \mathit{last}(\sigma) = t, \mathit{word}(\sigma) = \alpha\}$ and $\mathit{Path}^A_{\mathit{finmax}}(s, t) = \{\sigma \in \mathit{Path}^A_{\mathit{finmax}}(s) : \mathit{last}(\sigma) = t\}$.

Note that $\mathbf{Q}^D(s_0 \overset{a_1,\mu_1}{\leadsto} s_1 \ldots \overset{a_n,\mu_n}{\leadsto} s_n) = \mu_1(s_1) \cdot \ldots \cdot \mu_l(s_l)$ for deterministic adversaries D and that $\mathbf{P}^A(s, t) = \sum_{\alpha \in \mathit{Act}_\tau^*} \mathbf{P}^A(s, \alpha, t)$ for all adversaries A.

The value $\mathbf{Q}^A(\sigma)$ for a finite path σ is the probability for an execution to start with the prefix σ when the nondeterministic choices are resolved according to A. $\mathbf{P}^A(\sigma)$ stands for the probability for σ to be a maximal finite path in A. The *transition probabilities* $\mathbf{P}^A(s, \alpha, t)$ denote the probability for state s to move to state t via a finite maximal path in A labeled by the word $\alpha \in \mathit{Act}_\tau^*$. $\mathbf{P}^A(s, t)$ denotes the probability to move from s to t in A via any finite maximal path in A.

Example 5.3.17 Consider the adversaries in Example 5.3.12 once more. We compute their transition probabilities.

- For the partial adversary D, we have $\mathbf{P}^D(s) = 0$, $\mathbf{P}^D(s \overset{a,\rho}{\leadsto} t) = 1$ and $\mathbf{P}^D(s,t) = 1$.
- For A we have $\mathbf{P}^A(s) = \frac{1}{3}$, $\mathbf{P}^A(s \overset{a,\rho}{\leadsto} t) = \frac{1}{3}$, $\mathbf{P}^A(s \overset{\tau,\nu}{\leadsto} s) = \frac{1}{6}$, $\mathbf{P}^A(s \overset{\tau,\nu}{\leadsto} s') = \frac{1}{6}$. Hence $\mathbf{P}^A(s,\tau^*,\{s,s'\}) = \mathbf{P}^A(s) + \mathbf{P}^A(s \overset{\tau,\nu}{\leadsto} s) + \mathbf{P}^A(s \overset{\tau,\nu}{\leadsto} s') = \frac{2}{3}$.
- For the paths in D_ω, for $l \in \mathbb{N}$ let σ_l denote the path of the form $s \overset{\tau,\nu}{\leadsto} s \overset{\tau,\nu}{\leadsto} s \ldots \overset{\tau,\nu}{\leadsto} s$ of length l, σ'_{l+1} the path $\sigma_l \overset{\tau,\nu}{\leadsto} s'$ and σ^a_{l+1} the path $\sigma_l \overset{\tau,\nu}{\leadsto} s' \overset{a,\mu}{\leadsto} u$. Then, for $l \leq 1$,

$$\begin{aligned}
&\mathbf{Q}^{\mathcal{D}_\omega}(\sigma'_l) = \mathbf{Q}^{\mathcal{D}_\omega}(\sigma_l) = \mathbf{P}^{\mathcal{D}_\omega}(\sigma^a_l) = \tfrac{1}{2^l},\\
&\mathbf{P}^{D_\omega}(s,\tau^*,\{s,s'\}) = 0,\\
&\mathbf{P}^{D_\omega}(s,s) = \mathbf{P}^{D_\omega}(s,s') = 0,\\
&\mathbf{P}^{D_\omega}(s,\tau^* a,u) = \mathbf{P}^{D_\omega}(s,u) = 1.
\end{aligned}$$

As in Chapter 2, we can associate a probability space to each adversary.

Notation 5.3.18 (The probability measure $Prob^A$) Let $A \in RAdv_\perp$ and $s \in S$. If $\sigma \in Path^{*A}(s)$ then the *basic cone* of σ in A is given by

$$\sigma{\uparrow}^A = \{\pi \in Path^A_{max}(s) : \sigma \sqsubseteq \pi\}.$$

Let $\mathcal{F}^A(s)$ denote the smallest sigma-field on $Path^A_{max}(s)$ that contains the basic cones $\sigma{\uparrow}^A$ and let $Prob^A$ denote the unique probability measure on $\mathcal{F}^A(s)$, where the probability measure of the basic cones is

$$Prob^A\left(\sigma{\uparrow}^A\right) = \mathbf{Q}^A(\sigma).$$

Then $(Path^A_{max}(s), \mathcal{F}_A(s), \mathbf{Q}^A)$ is a probability space for each $s \in S$.

5.3.5 Finitary and Almost Finitary Adversaries

Definition 5.3.19 ((Almost) finitary randomized partial adversaries) A randomized partial adversary A is called *finitary* iff all paths in A are finite, i.e. if $Path^A(s) = Path^A_{fin}(s)$ for all states s. We call A *almost finitary* iff almost all paths in A are finite, i.e. if $\mathbf{P}^A(s,S) = 1$.[6]. We denote the sets of finitary and almost finitary randomized adversaries by $RAdv^{fin}_\perp$ and $RAdv^\omega_\perp$ respectively.

Obviously, any finitary partial adversary is almost finitary. Finitary partial adversaries are those that do not have infinite fullpaths. In other words, the tree associated with them is finite. They describe the probabilistic effect of the combination of *finitely many* transitions. Almost finitary partial adversaries are those where the probability measure of the infinite fullpaths in them is 0 (but infinite fullpaths in them might exist).

Remark 5.3.20 For each almost finitary randomized partial adversary A and state s, the function $t \mapsto \mathbf{P}^A(s,t)$ is a probability distribution on S. Similarly, $(\alpha,t) \mapsto \mathbf{P}^A(s,\alpha,t)$ is a probability distribution on $Act^*_\tau \times S$.

[6] Note that it is sufficient that all (almost all, resp.) *maximal* paths in A are finite.

Example 5.3.21 First, we give an informal example to illustrate the difference between finitary and almost finitary adversaries. Flipping a coin three times in a row corresponds to a finitary adversary, because all maximal paths have length 3. Flipping a coin until heads come up corresponds to an almost finitary adversary, because the probability on termination (i.e. on a finite path) is one. It is not finitary because the infinite sequence in which only heads come up is a maximal path in the corresponding adversary.

For a more formal example, based on the same ideas, reconsider the system of Figure 5.3. Define the partial deterministic adversaries D_l $(l \in \mathbb{N})$ by

$$D_l(\sigma) = \begin{cases} (\tau, \nu) & \text{if } \mathit{last}(\sigma) = s \wedge \mathit{length}(\sigma) \leq l, \\ (a, \mu) & \text{if } \mathit{last}(\sigma) = s', \\ \perp & \text{otherwise.} \end{cases}$$

Note that D_l is not memoryless. For each l, the adversary D_l is finitary and interrupts the execution after the τ loop has been taken l times or more. This yields the transition probability

$$\mathbf{P}^{D_l}(s,s) = \left(\tfrac{1}{2}\right)^l, \mathbf{P}^{D_l}(s,u) = 1 - \left(\tfrac{1}{2}\right)^l, \mathbf{P}^{D_l}(s,s') = \mathbf{P}^{D_l}(s,t) = 0.$$

The adversary D_ω from Example 5.3.12 is almost finitary but not finitary as it yields the infinite path $s \overset{\tau}{\rightsquigarrow} s \overset{\tau}{\rightsquigarrow} s \overset{\tau}{\rightsquigarrow} \ldots$. As we have seen, the transition probabilities are

$$\mathbf{P}^{D_\omega}(s,u) = 1, \quad \mathbf{P}^{D_\omega}(s,s) = \mathbf{P}^{D_\omega}(s,s') = 0.$$

These cannot be obtained by any finitary partial adversary.

5.3.6 Properties of Adversaries

The following proposition states that each partial adversary A can be written as a pointwise limit of finitary adversaries A_l. We believe that, by defining a suitable metric on adversaries, one can also write A as a uniform limit in this metric.

Proposition 5.3.22 *Let A be partial randomized (deterministic) adversary.*

1. *Then there are finitary randomized (deterministic) partial adversaries A_l such that $A = \lim_{l \to \infty} A_l$, where $\lim$ denotes the pointwise limit of the functions A_l.*
2. *If $A = \lim_{l \to \infty} A_l$ is a pointwise limit, then $\mathbf{Q}^A(\sigma) = \lim_{l \to \infty} \mathbf{Q}^{A_l}(\sigma)$.*

PROOF:

1. Take $A_l(\sigma) = A(\sigma)$ if $\mathit{length}(\sigma) \leq l$ and $A_l(\sigma)(\perp) = 1$ otherwise.
2. Easy verification.

□

The following proposition shows that each finite randomized partial adversary can be written as a convex combination of deterministic partial adversaries. The proof given below crucially depends on the fact that the systems we consider are finite. The proof immediately carries over to finitely branching systems, but not to systems with an infinite branching structure.

Notation 5.3.23 (The set $\mathcal{D}(s, A)$) Let A be a finite partial randomized adversary and let $s \in S$. $\mathcal{D}(s, A)$ denotes the set of deterministic partial adversaries D where $Path^{D}_{max}(s) \subseteq Path^{A}_{max}(s)$.[7]

Proposition 5.3.24 *Let $A \in RAdv^{fin}_{\perp}$ and $s \in S$.*

1. *Then, there is a finite family $(D_i)_{i \in I}$ of deterministic finitary partial adversaries D_i and a family $(p_i)_{i \in I}$ of real numbers $p_i \in [0, 1]$ such that $\sum_{i \in I} p_i = 1$ and*

$$A(\sigma)(a, \mu) \;=\; \sum_{i \in I} p_i \cdot D_i(\sigma)(a, \mu), \quad \textit{for all } \sigma \in Path^{A}(s),\ a,\ \mu.$$

2. *If $A = \sum_{i \in I} p_i \cdot A_i$, for adversaries $(A_i)_{i \in I}$, real numbers $(p_i)_{i \in I}$, $p_i \in [0, 1]$ with $\sum_{i \in I} p_i = 1$ then*

$$\mathbf{Q}^{A}(\sigma) \;=\; \sum_{i \in I} p_i \cdot \mathbf{Q}^{A_i}(\sigma).$$

PROOF:

1. Let A be a finitary randomized partial adversary and $s \in S$. Since A is finitary, the set $\mathcal{D}(s, A)$ is finite and all $D \in \mathcal{D}(s, A)$ are finitary.

   The idea in the proof is as follows. Each D in $\mathcal{D}(s, A)$ can be seen as an adversary within A because, among the steps that A schedules with a positive probability, D schedules one with probability one. If we multiply all the probabilities that A assigns to steps taken in D (yielding a value $p(D, A)$), then A is obtained by selecting the adversary D with probability $p(D, A)$, as we show below.

   Let

$$p(D, A) = \prod_{\sigma \in Path^{A}(s)} A(\sigma)(D(\sigma)),$$

   where, as before, we write $D(\sigma) = (a, \mu)$ for $D(\sigma)(a, \mu) = 1$. Then

$$\sum_{D \in \mathcal{D}(s,A)} p(D, A) \;=\; \sum_{D \in \mathcal{D}(s,A)} \prod_{\sigma \in Path^{A}(s)} A(\sigma)(D(\sigma)) \;=\; 1,$$

   The last step holds because $\mathcal{D}(s, A)$ is finite and $\sum_{x} A(\sigma)(x) = 1$ for all σ. For the same reason we have for all ρ, a, μ that

$$\begin{aligned}
&\sum_{D \in \mathcal{D}(s,A)} p(D, A) \cdot D(\rho)(a, \mu) \;=\\
&\sum_{D \in \mathcal{D}(s,A)} \prod_{\sigma \in Path^{A}(s)} A(\sigma)(D(\sigma)) \cdot D(\rho)(a, \mu) \;=\\
&\sum_{D \in \mathcal{D}(s,A), D(\rho) = (a,\mu)} \prod_{\sigma \in Path^{A}(s)} A(\rho)(a, \mu) \cdot 1 \;=\\
&A(\rho)(a, \mu).
\end{aligned}$$

[7] Thus, $D \in \mathcal{D}(s, A)$ iff, for all $\sigma \in Path^{D}_{fin}(s)$, $D(\sigma) = x$ implies $A(\sigma)(x) > 0$.

2. By induction on the length of σ.

   - If $\sigma = s$ has length 0, then the statement follows immediately.
   - Let $\sigma = \sigma' \overset{a,\mu}{\leadsto} t$, then

$$\begin{aligned}\mathbf{Q}^A(\sigma' \overset{a,\mu}{\leadsto} t) &= \mathbf{Q}^A(\sigma') \cdot A(\sigma')(a,\mu) \cdot \mu(t) = \\ &= \sum_{i \in I} p_i \cdot \mathbf{Q}^{A_i}(\sigma') \cdot \sum_{j \in I} p_j \cdot A_j(\sigma')(a,\mu) \cdot \mu(t) \;= \\ &= \sum_{i \in I} p_i \cdot \mathbf{Q}^{A_i}(\sigma') \cdot A_i(\sigma')(a,\mu) \cdot \mu(t) \;= \\ &= \sum_{i \in I} p_i \cdot \mathbf{Q}^{A_i}(\sigma' \overset{a,\mu}{\leadsto} t).\end{aligned}$$

□

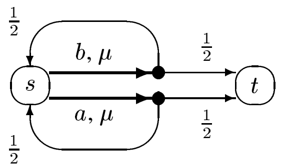


Figure 5.4: Almost finitary adversaries need not be a convex combination of deterministic ones.

The following example shows that the result of Proposition 5.3.24 cannot be generalized immediately to almost finitary partial adversaries. This should not be too much of a surprise because, even for finite systems, there are uncountably many maximal paths in an almost finitary adversary, whereas there are only countably many paths in a finitary adversary. Hence, there are countably many maximal paths in a convex combination of finitary adversaries. In fact, the natural generalization of Proposition 5.3.24 to almost finitary adversaries would be to define a measure space over the set of adversaries and to express an arbitrary almost finitary adversary as an integral over deterministic ones. The investigation whether it is indeed possible to generalize Proposition 5.3.24 along these line is a topic for future research.

Example 5.3.25 Consider the system in Figure 5.4 and define a randomized partial adversary A which in state s schedules the a and the b–transition each with probability $\frac{1}{2}$, i.e. if $last(\sigma) = s$ then $A(\sigma)(a,\mu) = A(\sigma)(b,\mu) = \frac{1}{2}$ and $A(\sigma)(x) = \perp$ otherwise.

It is not difficult to see that A is almost finitary: the probability of ending up in t after exactly k steps is $\frac{1}{2^k}$.

Every deterministic partial adversary D of this system can be identified uniquely by its longest maximal path σ, which is the longest maximal path in D. We write D_σ for the deterministic partial adversary whose longest maximal path is σ. Note that, $\mathbf{Q}^{D_\rho}(\sigma) > 0$ iff $s \neq \sigma \sqsubseteq \rho$ and in that case $\mathbf{Q}^{D_\rho}(\sigma) = \frac{1}{2^{length(\sigma)}}$.

Now assume that A is a convex combination of deterministic partial adversaries as in Proposition 5.3.24 [8]. This means that there is a countable set C of (in)finite maximal path and a family $\{p_\rho\}_{\rho \in C}$ of real numbers in $[0,1]$ such that $\sum_{\rho \in C} p_\rho = 1$ and

$$A(\sigma)(a,\mu) \;=\; \sum_{\rho \in C} p_\rho \cdot D_\rho(\sigma)(a,\mu) \quad \text{for all } \sigma \in Act_\tau^* \text{ and } t \in S.$$

Let σ be a maximal path of length $n > 0$. Then $\mathbf{Q}^A(\sigma) \;=\; \frac{2}{4^n}$ and hence

$$\sum_{\rho \in C} p_\rho \cdot \mathbf{Q}^{D_\rho}(\sigma) = \sum_{\rho \in C, \sigma \sqsubseteq \rho} p_\rho \cdot \mathbf{Q}^{D_\rho}(\sigma) = \sum_{\rho \in C, \sigma \sqsubseteq \rho} p_\rho \cdot \frac{1}{2^n} = \frac{1}{4^n}.$$

Therefore, we have for all $n > 0$ and all σ of length n, $\sum_{\rho \in C, \sigma \sqsubseteq \rho} p_\rho = \frac{1}{2^n}$.

$$\sum_{\rho \in C, length(\rho) \geq n} p_\rho = \sum_{\sigma \in \{a,b\}^n} \sum_{\sigma \sqsubseteq \rho} p_\rho = \sum_{\sigma \in \{a,b\}^n} \frac{1}{2^n} = 2^n \cdot \frac{1}{2^n} = 1 = \sum_{\rho \in C} p_\rho.$$

This means that $p_\rho = 0$ for all $\rho \in C$ with length $< n$ and, since this holds for all $n > 0$, we have $p_\rho = 0$ for all finite ρ. Now we claim that $p_\sigma = 0$ for infinite sequences σ. Let $\sigma \in \{a,b\}^\infty$ and consider $\sigma = \mathit{Prefix}(\sigma, n)$. Then for all $n > 0$

$$\frac{1}{2^n} = \sum_{\rho \in C, \sigma \sqsubseteq \rho} p_\rho \cdot \frac{1}{2^n} > p_\sigma.$$

Thus, $p_\sigma = 0$ and A is not the convex combination of countably many deterministic partial adversaries.

5.4 Strong and Weak (Bi–)simulation

This section recalls several notions of simulation and bisimulation for probabilistic systems that appear in the literature. More specifically, we treat strong, combined strong and weak (bi–)simulation. These relations have been introduced by Larsen & Skou [LS91], Hansson & Jonsson [HJ94] and Segala & Lynch [SL95] and they are the natural extensions of the corresponding notions for nonprobabilistic labeled transition systems [Mil80, Par81, Mil89, GW89]. Although few of the results in this section are new, we believe that we have contributed by simplifying existing definitions.

Both for nonprobabilistic and for probabilistic systems, simulations and bisimulations are relations on the state space of the system that compare the stepwise behavior of the states. *Bisimulation relations* are equivalences on the state space such that related states exhibit the same stepwise behavior. *Simulation relations* are preorders on the state space (i.e. transitive and reflexive) such that if a state s is related to a state t, then t can mimic all behavior of s. The converse is however not required: t may perform steps that cannot be exhibited by s.

Different notions of (bi–)simulation exist, depending on what is meant by "exhibiting the same stepwise behavior." The basic scheme for (bi–)simulation in the nonprobabilistic case is as follows. A relation R is a simulation iff

[8] In this proposition we only needed the convex combination of finitely many partial adversaries. In general a convex combination can be an infinite sum.

$$\text{If } sRs' \wedge s \xrightarrow{a} t \text{ then there is a state } t' \text{ such that } s' \overset{a}{\dashrightarrow} t' \wedge tRt'.$$

A bisimulation is a simulation relation that is an equivalence. Here, $s \overset{a}{\dashrightarrow} t$ refers to some kind of step from s to t whose visible behavior is a or, if the relation abstracts from internal steps, $\widehat{a}$. The different types of (bi–)simulations are obtained by different choices for $\dashrightarrow$. If we have $s \xrightarrow{a} t$ and $s' \overset{a}{\dashrightarrow} t'$ with sRs' and tRt', then we say that the transition $s \xrightarrow{a} t$ is *matched*, *simulated* or *mimicked* by the step $s' \overset{a}{\dashrightarrow} t'$.

The situation in the probabilistic case is similar. However, we cannot use the relation R to compare the target of two transitions, because these are now probability distributions instead of states. Therefore, we lift R to the set of probability distributions, yielding the relation $\sqsubseteq_R$. In the sequel, we explain why $\sqsubseteq_R$ is a reasonable choice for comparing the target distributions. Thus, the scheme becomes:

$$\text{If } sRs' \wedge s \xrightarrow{a} \mu \text{ then there is a distribution } \mu' \text{ such that } s' \overset{a}{\dashrightarrow} \mu' \wedge \mu \sqsubseteq_R \mu'.$$

Again, we get the different types of (bi–)simulations by taking different instances of $\dashrightarrow$. The ideas behind the variants considered in this chapter are the following.

- *Strong (bi–)simulation* does not abstract from internal transitions. That is, it treats a τ–transition as any other transition. It is obtained by taking the transition relation $\rightarrow$ for $\dashrightarrow$. In other words, strong (bi–)simulation requires each transition to be matched with a single step.

- *Strong combined (bi–)simulation* does not abstract from internal transitions either. But unlike strong simulation relations, combined strong bisimulations allow for a single step to be mimicked by a convex combination of steps. This is formalized by the combined transition relation $\rightarrowtail$.

- *Weak (bi–)simulation* does abstract from internal computation. This means that one a–transition can be mimicked by a combination of τ transitions and one a–transition. This is formalized by the weak transition relation $\Longrightarrow$, whose definition depends on schedulers. These tell which τ and a–transitions are taken. The various types of schedulers (randomized, deterministic, finitary, ω) give rise to different types of weak transitions and, hence, different types of weak (bi–)simulation (notably randomized, deterministic, finitary, ω (bi–)simulation).

In Section 5.5.2 the notion of delay bisimulation is introduced.

- *Delay (bi–)simulation* also abstracts from internal computation, but in a more limited way than weak (bi–)simulation. What we gain is a polynomial algorithm. Delay (bi–)simulation also depends on schedulers and, here too, different kinds of schedulers give rise to different notions of delay (bi–)simulation.

Another well–known relation from the literature is branching bisimulation. This relation is strictly finer than weak bisimulation and takes into account the branching structure of a system. We do not treat it here, but it can be introduced along the exactly same lines as weak bisimulation. The only difference is that the adversaries have to take into account the branching structure. Different types of schedulers give rise to different branching bisimulations. Moreover, one can introduce delay variants of branching bisimulation. We conjecture that these can be decided in polynomial time and space by similar algorithms as the ones introduced in Section 5.7.4.

5.4.1 Strong (Bi–)simulation

Strong (bi–)simulation, introduced by [LS91], requires that each step leaving from a state s can be matched by step from a related state.

Definition 5.4.1 (Strong (bi–)simulation) A relation $R \subseteq S \times S$ is a *strong simulation* iff for all $(s, s') \in R$:

$$\text{If } s \xrightarrow{a} \mu \text{ then there is a transition } s' \xrightarrow{a} \mu' \text{ with } \mu \sqsubseteq_R \mu'.$$

A *strong bisimulation* is a strong simulation which is an equivalence. We say that the state s_2 *strongly simulates* s_1 and write $s_1 \preceq_{ssim} s_2$ iff there exists a strong simulation R such that $(s_1, s_2) \in R$. If there exists a bisimulation containing (s_1, s_2), then s_1 and s_2 are called *strongly bisimilar*, denoted by $s_1 \approx_{sbis} s_2$.

The definition of strong bisimulation can be understood as follows. First, recall that $\equiv_R$ and $\sqsubseteq_R$ coincide for equivalences (see part (a) of Proposition 5.2.1). Therefore, we may replace $\sqsubseteq_R$ with $\equiv_R$ in the definition of a strong *bi*simulation. As bisimilar states are interchangeable, it does not matter which state within the same bisimulation equivalence class is reached. Therefore, we are interested in the probability to reach an equivalence class, rather than in the probability to reach an individual state. Thus, when we match two transitions in a bisimulation relation, we require that they reach each equivalence class with the same probability. This is exactly the requirement $\mu \equiv_R \mu'$. Before explaining the simulation relations, we give an example of a strong bisimulation relation.

Example 5.4.2 The states s and s' in the system in Figure 5.5 are strongly bisimilar and so are y and y' and so are x_1, x_2 and x'. Indeed μ and μ' assign exactly the same probabilities to each of the equivalence classes $\{s, s'\}$, $\{z\}$, $\{y, y'\}$ and $\{x_1, x_2, x\}$.

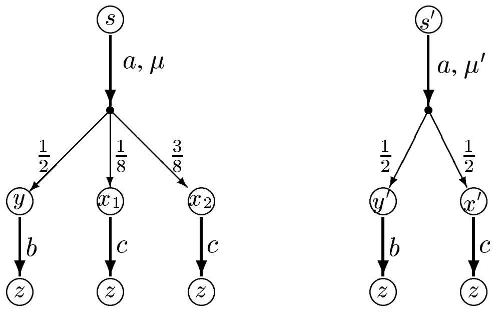


Figure 5.5: $s \approx_{sbis} s'$

The situation for strong simulation is similar, except that now we cannot compare the target probability distributions of two transitions by comparing the probabilities they assign to the equivalence classes. Instead, $\mu \sqsubseteq_R \mu'$ is established by a weight function. This function quantifies to which extend the probability $\mu(x)$ contributes to the probability $\mu'(x')$ of a related state.

Example 5.4.3 Now, consider the system in Figure 5.6. The relation $R = \{(s, s'), (t, t'), (u, u'), (u, t'), (v, v')\}$ is a strong simulation. For instance, the state u is related to the states t' and u'. We can distribute $\mu(u) = \frac{2}{3}$ over t' and u' by having $wgt(t,t') = 1/3$, $wgt(u,u') = 1/2$. Moreover, take $wgt(u,t') = 1/6$ and $wgt(\cdot,\cdot) = 0$ in the other cases. This shows $\mu \sqsubseteq_R \mu'$.

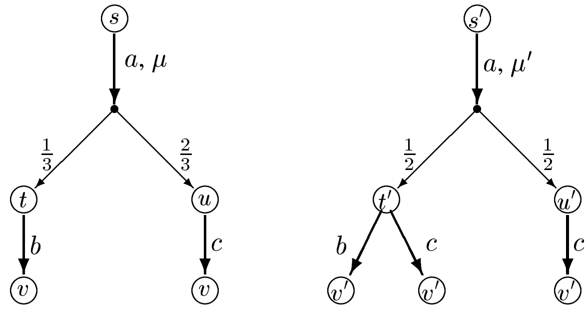


Figure 5.6: $s \preceq_{ssim} s'$

We refer the reader to Section 5.6 for some basic properties of strong bisimulation relations.

5.4.2 Strong Combined (Bi–)simulation

In [Seg95b], Segala argues that strong (bi–)simulation is too strong in certain cases. He argues that it is more natural to allow an a–transition to be matched by a convex combination of a–transitions, rather than by a single a–transition. This leads to a notion which we call *combined bisimulation*.

Definition 5.4.4 We write $s \overset{a}{\rightarrowtail} \mu$ iff there is a countable family of transitions $s \xrightarrow{a} \mu_i$, such that μ is a convex combination of the distributions μ_i.

Definition 5.4.5 (Strong combined (bi–)simulation) A *strong combined simulation* is a relation R on S such that for all $(s, s') \in R$:

If $s \xrightarrow{a} \mu$ then there is a transition $s' \overset{a}{\rightarrowtail} \mu'$ with $\mu \sqsubseteq_R \mu'$.

A *strong combined bisimulation* is a strong combined simulation which is an equivalence. We write $s_1 \preceq_{scsim} s_2$ ($s_1 \approx_{scbis} s_2$) iff there exists a strong combined (bi–)simulation which contains (s_1, s_2).

Simulation relations are often used to establish that one state (or system) correctly implements another one. The example below shows that, unlike strong simulations, strong *combined* simulations allow a nondeterministic choice to be implemented by a probabilistic choice.

Example 5.4.6 Consider the systems in Figure 5.7. The relation $\{(s_1, t_1), (s_2, t_2), (s_3, t_3)\}$ is a strong combined simulation, but not a strong simulation. The transition $t_1 \overset{a}{\rightarrowtail} \nu$ with $\nu(t_2) = \nu(t_3) = \frac{1}{2}$ is obtained as a convex combination of the steps $t_1 \xrightarrow{a} \{t_2 \mapsto 1\}$ and $t_1 \xrightarrow{a} \{t_3 \mapsto 1\}$ with $\nu = \frac{1}{2} \cdot \{t_2 \mapsto 1\} + \frac{1}{2} \cdot \{t_3 \mapsto 1\}$.

Similar systems show the difference between strong bisimulation and strong combined bisimulation: in Figure 5.7, we have $t_1 \approx_{scbis} u_1$ but $t_1 \not\approx_{sbis} u_1$.

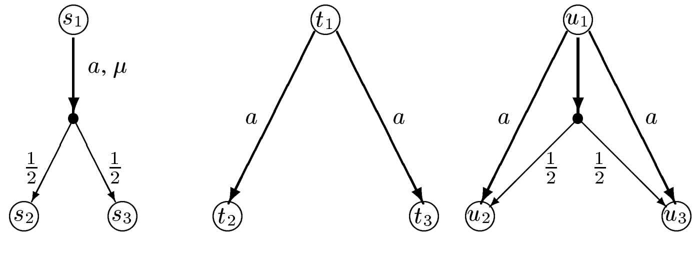


Figure 5.7: Simulating a probabilistic choice by a nondeterministic one.

5.4.3 Weak (Bi–)simulation

Unlike strong (bi–)simulation and strong combined (bi–)simulation, weak (bi–)simulation abstracts from internal computation. This means that τ–transitions, being invisible for the environment, may always be used to mimic another step. More precisely, a τ–transition leaving from a state s may be mimicked by any number (zero or more) of τ transitions in a related state. An a–transition in s may be matched by first taking some (zero or more) τ transitions, then an a–transition and some τ–transitions again.

In the nonprobabilistic setting, a weak simulation relation R is required to satisfy for all s, s' such that sRs'

$$\text{if } s \xrightarrow{a} u \text{ then there is some } u' \text{ with } s' \stackrel{a}{\Longrightarrow} u' \text{ and } s'Ru'.$$

A bisimulation is a simulation that is an equivalence [Mil89]. Here, the double arrow relation $\Longrightarrow \; \subseteq S \times Act_\varepsilon \times S$ is defined via the transitive reflexive closure of the internal steps. Thus, $s \stackrel{\varepsilon}{\Longrightarrow} u$ iff $s \xrightarrow{\tau} \ldots \xrightarrow{\tau} u$ and $s \stackrel{a}{\Longrightarrow} u$ iff $s \xrightarrow{\tau} \ldots \xrightarrow{\tau} v \xrightarrow{a} t \xrightarrow{\tau} \ldots \xrightarrow{\tau} u$ for any visible action a. Note that $s \stackrel{\widehat{a}}{\Longrightarrow} t$ iff there is an execution starting at s, leading to t, whose word is an element of $\tau^*\widehat{a}\tau^*$.

For probabilistic systems, the situation is similar, but more complex. The definitions of weak (bi-)simulations are adapted for the probabilistic setting using suitable modifications of the relation $\Longrightarrow$ [SL95, Seg95b]. A weak transition $s \stackrel{\widehat{a}}{\Longrightarrow} \mu$ corresponds to an execution over the words $\tau^*\widehat{a}\tau^*$. Since executions for probabilistic systems are formalized as adversaries, $s \stackrel{\widehat{a}}{\Longrightarrow} \mu$ corresponds to an adversary A with $Words(s, A) \subseteq \tau^*\widehat{a}\tau^*$ such that μ equals the associated distribution $\mathbf{P}^A(s, \cdot)$. The use of randomized versus deterministic and almost finitary versus finitary adversaries yields four different notions of weak transitions, and hence, four notions of weak (bi–)simulation. However, we will see that the use of deterministic adversaries induces a notion of (bi–)simulation with completely different properties than the other (bi–)simulations. For instance, the corresponding (bi–)simulations are not transitive.

Notation 5.4.7 [The partial adversary classes $\mathcal{C}[s, W]$] Let $\mathcal{C}$ be a class of partial adversaries, let $s \in S$ and $W \subseteq Act_\tau^\infty$ be a set of finite and infinite words. We define $\mathcal{C}[s, W] = \{A \in \mathcal{C} : Words(s, A) \subseteq W\}$.

In the sequel, we consider adversaries over the set words $\tau^* a \tau^*$ and, in particular, we consider the sets

- $RAdv_\perp^{fin}[s, \tau^*\widehat{a}\tau^*]$ consisting of finitary, randomized, partial adversaries starting in s over the words $\tau^*\widehat{a}\tau^*$,

- $DAdv_{\perp}^{fin}[s, \tau^*\widehat{a}\tau^*]$ consisting of finitary, deterministic, partial adversaries starting in s over the words $\tau^*\widehat{a}\tau^*$,

- $RAdv_{\perp}^{\omega}[s, \tau^*\widehat{a}\tau^*]$ consisting of almost finitary, randomized, partial adversaries starting in s over the words $\tau^*\widehat{a}\tau^*$,

- $DAdv_{\perp}^{\omega}[s, \tau^*\widehat{a}\tau^*]$ consisting of almost finitary, deterministic, partial adversaries starting in s over the words $\tau^*\widehat{a}\tau^*$.

For a visible action a, the adversaries in $RAdv_{\perp}^{fin}[s, \tau^*\widehat{a}\tau^*]$ and $DAdv_{\perp}^{fin}[s, \tau^*\widehat{a}\tau^*]$ correspond to trees in which each path is finite and in which each path to a leaf contains exactly one a–transition and zero or more τ–transitions. The adversaries in $RAdv_{\perp}^{\omega}[s, \tau^*\widehat{a}\tau^*]$ and $DAdv_{\perp}^{\omega}[s, \tau^*\widehat{a}\tau^*]$ are trees that might have infinite paths, but the total probability measure on these infinite paths is zero. Each finite path to a leaf contains exactly one a–transition and zero or more τ–transitions. The infinite paths contain at most one a–transition and infinitely many τ–transitions. The latter is a consequence of an adversary being almost finitary, it does not hold for arbitrary adversaries over the words $\tau^*\widehat{a}\tau^*$.

Recall that for an adversary A, the probability to reach a state t from a state s via a maximal path in A is given by $\mathbf{P}^A(s, t)$. Thus, $\mathbf{P}^A(s, t)$ is the probability to end up in t in the execution generated by A, which is exactly what we need when defining the weak transition relation $\Longrightarrow$ for probabilistic systems.

Each class $\mathcal{C}$ of adversaries yields a notion of weak transition $\Longrightarrow^{\mathcal{C}}$ defined by

$$s \stackrel{\widehat{a}}{\Longrightarrow}{}^{\mathcal{C}} \mu \text{ iff there exists } A \in \mathcal{C}[s, \tau^*\widehat{a}\tau^*] \text{ with } \mu = \mathbf{P}^A(s, \cdot).$$

Thus, the classes defined above yield four different types of weak transitions.

Definition 5.4.8 Let $s \in S$, $a \in Act_\tau$ and $\mu \in Distr(S)$. Define

$$s \stackrel{\widehat{a}}{\Longrightarrow}{}^{d,fin} \mu \text{ iff there exists } A \in DAdv_{\perp}^{fin}[s, \tau^*\widehat{a}\tau^*] \text{ with } \mu = \mathbf{P}^A(s, \cdot).$$

If A is an element of $DAdv_{\perp}^{fin}[s, \tau^*\widehat{a}\tau^*]$ with $\mu = \mathbf{P}^A(s, \cdot)$, we say that A *establishes* the transition $s \stackrel{\widehat{a}}{\Longrightarrow}{}^{d,fin} \mu$. The weak relations $\stackrel{\widehat{a}}{\Longrightarrow}{}^{r,fin}$, $\stackrel{\widehat{a}}{\Longrightarrow}{}^{\omega,d}$ and $\stackrel{\widehat{a}}{\Longrightarrow}{}^{\omega,r}$ are defined by taking the corresponding classes of adversaries.

It is clear that $s \stackrel{\widehat{a}}{\Longrightarrow}{}^{fin,d} \mu$ implies both $s \stackrel{\widehat{a}}{\Longrightarrow}{}^{fin,r} \mu$ and $s \stackrel{\widehat{a}}{\Longrightarrow}{}^{\omega,d} \mu$. The latter two are incomparable and each imply $s \stackrel{\widehat{a}}{\Longrightarrow}{}^{\omega,r} \mu$. The following example shows that every weak transition extends the relation $\rightarrow$ and that the randomized variants extend $\rightarrowtail$. Moreover, the example shows that, even though the systems we consider are finite, the number of finitary, weak transitions can be infinite.

Example 5.4.9

- The deterministic adversary $D_\perp$ with $D_\perp(\sigma) = \perp$ for all σ yields the trivial weak transition $s \stackrel{\varepsilon}{\Longrightarrow} s$.

- If $s \stackrel{a}{\rightarrow} \mu$ then $s \stackrel{\widehat{a}}{\Longrightarrow}{}^{fin,d} \mu$. This is established by the finitary, deterministic adversary D with $D(s) = (a, \mu)$ and $D(\sigma) = \perp$ if $\sigma \neq s$. (cf. the adversary D in Example 5.3.12 on page 94.)

- Assume $s \stackrel{a}{\rightarrowtail} \mu$. Let $s \stackrel{a}{\rightarrow} \mu_i$ be transitions such that $\sum_i p_i \cdot \mu_i = \mu$. Then the finitary, randomized adversary A with $A(s)(a, \mu_i) = p_i$ and $A(\sigma)(\perp) = 1$ in all other cases establishes that $s \stackrel{\widehat{a}}{\Longrightarrow}{}^{fin,r} \mu$.

- Now, consider the system in Figure 5.3 on page 95. The deterministic, almost finitary adversary D_ω (Example 5.3.12, page 94) establishes $s \overset{a}{\Longrightarrow}^{\omega,d} u$. As there is no finitary adversary A with $\mathbf{P}^A(s,\cdot) = \{u \mapsto 1\}$, we do not have $s \overset{a}{\Longrightarrow}^{\mathit{fin},d} u$.
- Let D' be the convex combination of the adversaries D and D_ω (both from Example 5.3.12 again) given by $D'(\sigma)(x) = \frac{1}{2}D_\omega(\sigma)(x) + \frac{1}{2}D(\sigma)(x)$. Then D' shows $s \overset{a}{\Longrightarrow}^{\omega,r} \rho'$, where $\rho'(t) = \frac{2}{3}$ and $\rho'(u) = \frac{1}{3}$. We clearly do not have $s \overset{a}{\Longrightarrow}^{\omega,d} \rho'$.
- The deterministic, finitary adversary D'_l given by

$$D'_l(\sigma) = \begin{cases} (\tau, \nu) & \text{if } \mathit{last}(\sigma) = s \wedge \mathit{length}(\sigma) \leq l, \\ (a, \rho) & \text{if } \mathit{last}(\sigma) = s \wedge \mathit{length}(\sigma) = l + 1, \\ (a, \mu) & \text{if } \mathit{last}(\sigma) = s', \\ \perp & \text{otherwise.} \end{cases}$$

  Thus, D'_l tries exiting the τ–loop l times and if it has not succeeded after l steps, it takes the a–transition leading to t. Then D'_l shows that $s \overset{a}{\Longrightarrow}^{d,\mathit{fin}} \mu_l$, where $\mu_l(t) = \frac{1}{2^{l+1}}$ and $\mu_l(u) = 1 - \frac{1}{2^{l+1}}$.
- For the simple communication system of Example 5.3.5 on page 92, we have that $s_{del} \overset{cons}{\Longrightarrow}^{\omega,d} s_{init}$ and $s_{del} \overset{prod}{\Longrightarrow}^{\omega,d} s_{ack}$.

Now, we can define the notions of (bi-)simulation induced by the weak transition relations. The ω-variants of randomized, weak (bi-)simulations coincide with those introduced by Segala & Lynch [SL95]. The deterministic and the finitary variants have not been defined in literature. We will see in Section 5.6 that deterministic variants lead to rather questionable relations.

Definition 5.4.10 ((bi-)simulation, cf. [SL95]) A relation $R \subseteq S \times S$ is called a *finitary, randomized, weak simulation* iff for all $(s, s') \in R$:

$$s \xrightarrow{a} \mu \text{ implies } s' \overset{\widehat{a}}{\Longrightarrow}^{r,\mathit{fin}} \mu' \text{ for some } \mu' \text{ with } \mu \sqsubseteq_R \mu'.$$

A *finitary, randomized, weak bisimulation* is a finitary, randomized, weak simulation R that is an equivalence. The finitary, randomized, weak simulation preorder $\preceq^{r,\mathit{fin}}_{wsim}$ is defined by $s \preceq^{r,\mathit{fin}}_{wsim} s'$ iff $(s, s') \in R$ for some finitary, randomized, weak simulation R. We define finitary, randomized, weak bisimulation equivalence by $s \approx^{r,\mathit{fin}}_{wbis} s'$ iff there exists a finitary, randomized, weak bisimulation R with $(s, s') \in R$. In a similar way, we define the finitary variants of the deterministic, weak (bi–)simulation relations $\preceq^{d,\mathit{fin}}_{wsim}$, and $\approx^{d,\mathit{fin}}_{wbis}$. The randomized and deterministic variants of ω-(bi-)simulation are denoted by $\preceq^{r,\omega}_{wsim}$, $\approx^{r,\omega}_{wsim}$ and $\preceq^{d,\omega}_{wsim}$, $\approx^{d,\omega}_{wsim}$ respectively.

Recall that the relations $\equiv_R$ and $\sqsubseteq_R$ coincide for equivalences and that we may therefore replace $\sqsubseteq_R$ with $\equiv_R$ when dealing with bisimulations.

Example 5.4.11 Reconsider the system in Figure 5.3 on page 95. Let R be the smallest equivalence relation R identifying s and s' and identifying t and u. Then R is a weak, deterministic, finitary bisimulation. The step $s \xrightarrow{\tau} \nu$ in s is matched by the step $s' \overset{\tau}{\Longrightarrow}^{\mathit{fin},d} s'$ in s', since we have $\nu \equiv_R \{s' \mapsto 1\}$. The step $s \xrightarrow{a} \rho$ is matched by $s' \xrightarrow{a} \mu$ and vice versa, since $\mu \equiv_R \rho$. Hence $s \approx^{d,\mathit{fin}} s'$.

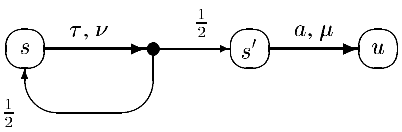


Figure 5.8: $s \approx^{d,\omega} s'$, but $s \not\approx^{d,fin} s'$

Example 5.4.12 Now consider the system in Fig 5.8. Let R be the smallest equivalence relation identifying s and s'. Then R is a deterministic, almost finitary, weak bisimulation. The step $s \xrightarrow{\tau} \nu$ is matched by $s' \stackrel{\tau}{\Longrightarrow}^{\omega,d} s'$ as before and the step $s' \xrightarrow{a} \mu$ is matched by $s \stackrel{a}{\Longrightarrow}^{\omega,d} \mu$. Therefore $s \approx^{d,\omega} s'$. Since we do not have $s \stackrel{a}{\Longrightarrow}^{fin,d} u$, we have $s \not\approx^{d,fin} s'$.

Example 5.4.13 Consider the system in Figure 5.9. We assume that u_i and u_i' are bisimilar. Then the relation $R = \{(s, s'), (u_1, u_1'), (u_2, u_2'), (u_3, u_3'), (u_4, u_4')\}$ is a weak, deterministic, finitary simulation. The transition $s \xrightarrow{a} \mu$ is matched by $s' \stackrel{a}{\Longrightarrow}^{fin,d} \nu'$, where $\nu'(u_i') = \frac{1}{4}$ for $i = 1, 2, 3, 4$ and clearly $\mu \sqsubseteq_R \nu'$. This illustrates the fact that weak bisimulation takes the cumulative probabilistic effect of several transitions into account.

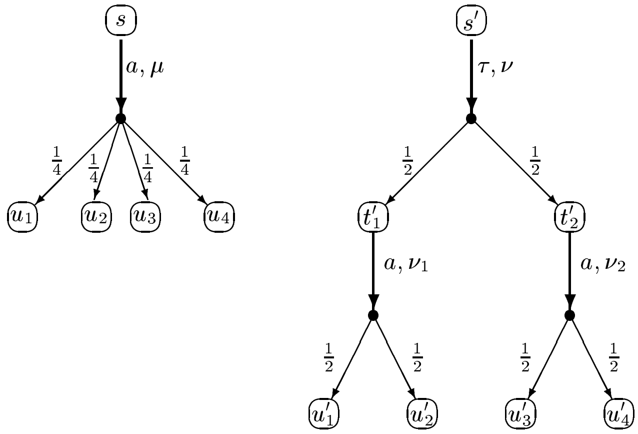


Figure 5.9: $s \preceq^{d,fin} s'$

5.4.4 Axiomatization

This section provides an alternative characterization of the finitary weak transition relations from Section 5.4.3 by means of axioms.

The nonprobabilistic, weak transition relation $\Longrightarrow$ is the smallest relation satisfying the following conditions (TR) (which yields the *trivial weak transition*), (PR) (which yields *proper transitions*), (WL) and (WR) (which assert that *weak* transitions can be combined with τ-transitions on the *left* and *right* respectively.

(TR) $s \stackrel{\varepsilon}{\Longrightarrow} s$.

(PR) If $s \stackrel{a}{\longrightarrow} u$ then $s \stackrel{\widehat{a}}{\Longrightarrow} u$.

(TR) $s \stackrel{\widehat{\varepsilon}}{\Longrightarrow} \mu_s^1$

(PR) If $s \stackrel{a}{\longrightarrow} \mu$ then $s \stackrel{\widehat{a}}{\Longrightarrow} \mu$

(CC) If $s \stackrel{\widehat{a}}{\Longrightarrow} \mu_i$, $i = 1, \ldots, l$, then

$$s \stackrel{\widehat{a}}{\Longrightarrow} \mu \text{ for any convex combination } \mu \text{ of } \mu_1, \ldots, \mu_l$$

(WL) If $s \stackrel{\tau}{\to} \nu$ and $\forall t \in supp(\nu)[s \stackrel{a}{\Longrightarrow} \mu_t]$, then $s \stackrel{\widehat{a}}{\Longrightarrow} \sum_{t \in supp(\nu)} \nu(t) \cdot \mu_t$

(WR) If $s \stackrel{\widehat{a}}{\Longrightarrow} \mu'$ and $\forall t \in supp(\mu')[\nu_t = \mu_t^1 \vee t \stackrel{\tau}{\longrightarrow} \nu_t]$, then

$$s \stackrel{\widehat{a}}{\Longrightarrow} \sum_{t \in supp(\mu')} \mu'(t) \cdot \nu_t.$$

Figure 5.10: Axioms and rules for the finitary, weak transitions

(WL) $s \stackrel{\widehat{a}}{\Longrightarrow} u$ if there exist a transition $s \stackrel{\tau}{\Longrightarrow} t$ and a weak transition $t \stackrel{\widehat{a}}{\Longrightarrow} u$.

(WR) $s \stackrel{\widehat{a}}{\Longrightarrow} u$ if there exist a weak transition $s \stackrel{\widehat{a}}{\Longrightarrow} t$ and a transition $t \stackrel{\tau}{\Longrightarrow} u$.

Figure 5.10 shows how these rules can be lifted to the probabilistic setting, yielding the *finitary* variants.

Proposition 5.4.14 *The finitary weak transition relations can be characterized as the smallest subsets of $S \times Act_\varepsilon \times Distr(S)$ satisfying the following axioms and rules of Figure 5.10.*

- *for the finitary, deterministic, weak transitions:* (TR), (PR), (WL), (WR).
- *for the finitary, randomized, weak transitions:* (TR), (PR), (CC), (WL), (WR).

PROOF: (sketch) On the one hand, we have to prove that every finitary, deterministic, weak transition can be derived from the rules (TR), (PR), (WL) and (WR). This is a routine proof by induction on $\max\{length(\sigma) \mid \sigma \in Path^A_{*,max}(s)\}$, where A is a partial adversary that establishes $s \stackrel{a}{\Longrightarrow} \mu$. Since A is finitary, Königs Lemma yields that this maximum is finite. For the randomized, finitary, weak transition relations, one uses the result for the deterministic variants together with Proposition 5.3.24.

On the other hand, we have to prove that each transition that can be derived from the rules (TR), (PR), (WL) and (WR) (and from the rules (TR), (PR), (WL), (WR) and (CC), respectively) is a finitary, deterministic (randomized, respectively) transition. This is a routine proof by induction on structure of the derivation that establishes the transition. □

Example 5.4.15 Consider Figure 5.7. In Example 5.4.6, we have seen that $t_1 \stackrel{a}{\rightarrowtail} \nu$ with $\nu(t_2) = \nu(t_3) = \frac{1}{2}$. Then we also have $t_1 \stackrel{a}{\Longrightarrow}^{fin,r} \nu$. The latter is obtained by the following derivation.

$$\dfrac{\dfrac{t_1 \stackrel{a}{\to} \{t_2 \mapsto 1\}}{t_1 \stackrel{a}{\Longrightarrow} \{t_2 \mapsto 1\}}\,(\mathrm{PR}) \qquad \dfrac{t_1 \stackrel{a}{\to} \{t_3 \mapsto 1\}}{t_1 \stackrel{a}{\Longrightarrow} \{t_3 \mapsto 1\}}\,(\mathrm{PR})}{t_1 \stackrel{a}{\Longrightarrow} \frac{1}{2} \cdot \{t_2 \mapsto 1\} + \frac{1}{2} \cdot \{t_3 \mapsto 1\} = \nu}\,(\mathrm{CC})$$

Example 5.4.16 Consider Figure 5.9. Example 5.4.13 has shown that $s' \stackrel{a}{\Longrightarrow}^{fin,d} \nu'$, where $\nu'(u'_i) = \frac{1}{4}$ for $i = 1, 2, 3, 4$. This is also obtained by the following derivation.

$$\dfrac{s \xrightarrow{a} \nu \quad supp(\nu) = \{t'_1, t'_2\} \quad \dfrac{t'_1 \xrightarrow{a} \nu_1}{t'_1 \stackrel{a}{\Longrightarrow} \nu_1}\,(\mathrm{PR}) \quad \dfrac{t'_1 \xrightarrow{a} \nu_2}{t'_1 \stackrel{a}{\Longrightarrow} \nu_2}\,(\mathrm{PR})}{s' \stackrel{a}{\Longrightarrow} \nu(t_1) \cdot \nu_1 + \nu(t_1) \cdot \nu_2 = \frac{1}{2} \cdot \nu_1 + \frac{1}{2} \cdot \nu_2 = \nu'}\,(\mathrm{WL})$$

5.5 Delay Simulation and Delay Bisimulation

5.5.1 Delay (Bi–)simulation

This section introduces the new notions of *delay simulation* and *delay bisimulation*. These refine the (randomized) weak (bi–)simulation relations, but still abstract from internal moves in a certain way. We follow the lines of Section 5.4.3 and base the delay (bi–)simulation relations on *delay transitions*, which in their turn depend on certain classes of adversaries. As in Section 5.4.3, we consider finitary, almost finitary, randomized and deterministic variants. Unlike for weak transitions, the use of randomized and deterministic schedulers gives rise to the same notion of delay transition. The use of finitary and almost finitary adversaries, however, still yields two different notions of delay (bi–)simulation.

There are two ways in which the delay (bi–)simulations are more restrictive than the weak (bi–)simulations. Firstly, delay transitions are established by adversaries over the words in $\tau^*\widehat{a}$ rather than over words in $\tau^*\widehat{a}\tau^*$. Secondly, for a visible action a, a delay simulation relation R requires each step $s \xrightarrow{a} \mu$ to be matched by a delay transition $s \stackrel{a}{=\!=\!\Rightarrow} \rho$. The latter is established by an adversary A which can only schedule a step (a, ν) if $\mu \sqsubseteq_R \nu$. In order words, $A(\sigma)(a, \nu) > 0$ implies $\nu \in \mu\!\uparrow_R$. For the matching of τ–transitions $s \xrightarrow{\tau} \mu$, similar conditions are required.

Definition 5.5.1 [The partial adversary classes $\mathcal{C}[s, W, M]$] Let $\mathcal{C}$ be a class of partial adversaries, let $s \in S$, let $W \subseteq Act_\tau^\infty$ be a set of finite and infinite words, and let M be a set of distributions. Then set $\mathcal{C}[s, W, M]$ is defined as the set of adversaries A such that

- $A \in \mathcal{C}$,
- $Words(s, A) \subseteq W$ and
- for all maximal paths $\sigma = s_0 \stackrel{a_1,\mu_1}{\leadsto} s_1 \dots \stackrel{a_n,\mu_n}{\leadsto} s_n$ in A, we have $\{s_0 \mapsto 1\} \in M$ if $n = 0$ and $\mu_n \in M$ if $n > 0$.

In the sequel, we consider adversaries over the set of words $\tau^* a$. In particular, we work with the sets of adversaries $RAdv^{fin}[s, \tau^*\widehat{a}, \mu\uparrow_R]$, $DAdv^{fin}[s, \tau^*\widehat{a}, \mu\uparrow_R]$, $RAdv^{\omega}[s, \tau^*\widehat{a}, \mu\uparrow_R]$ and $DAdv^{\omega}[s, \tau^*\widehat{a}, \mu\uparrow_R]$. For a visible action a, the sets $RAdv^{fin}_\perp[s, \tau^*\widehat{a}, \mu\uparrow_R]$ and $DAdv^{fin}_\perp[s, \tau^*\widehat{a}, \mu\uparrow_R]$ correspond to finite trees whose paths are labeled by elements from $\tau^* a$. The adversaries in $RAdv^{\omega}_\perp[s, \tau^*\widehat{a}, \mu\uparrow_R]$ and $DAdv^{\omega}_\perp[s, \tau^*\widehat{a}, \mu\uparrow_R]$ might have infinite paths, which are then labeled by τ–actions, but the probability measure on the set of infinite paths is zero. The finite maximal paths in these adversaries are labeled by elements from $\tau^* a$, as before. As a consequence, each adversary in one of the classes above reaches some of the states s' with an outgoing a–transition $s' \xrightarrow{a} \nu$ such that $\mu \sqsubseteq_R \nu$ with probability one. The proof of the following proposition is routine.

Proposition 5.5.2 *For all partial adversaries A in $RAdv_{\perp}^{\omega}[s, \tau^*\widehat{a}, \mu\uparrow_R]$, the distribution $x \mapsto \mathbf{P}^A(s,x)$ is an element in $\mu\uparrow_R$.*

This result holds in particular for adversaries in $RAdv_{\perp}^{fin}[s, \tau^*\widehat{a}, \mu\uparrow_R]$, $DAdv_{\perp}^{fin}[s, \tau^*\widehat{a}, \mu\uparrow_R]$ and $DAdv_{\perp}^{\omega}[s, \tau^*\widehat{a}, \mu\uparrow_R]$, because these sets are all subsets of $RAdv_{\perp}^{\omega}[s, \tau^*\widehat{a}, \mu\uparrow_R]$. Now, we can define the several variants of delay transition and delay (bi–)simulation in the same way as we defined the weak transition and (bi–)simulation relations.

Definition 5.5.3 (Delay transitions) Define the deterministic, finitary delay transition relation $\Longrightarrow^{fin,d}$ as follows. For $s \in S$, $a \in Act_\tau$, $\mu \in Distr(S)$ and $M \subseteq Distr(S)$,

1. define $s \overset{\widehat{a}}{\Longrightarrow}{}^{fin,d}_M \mu$ iff there exists $A \in DAdv_{\perp}^{fin}[s, \tau^*\widehat{a}, M]$ with $\mu = \mathbf{P}^A(s,\cdot)$ and
2. define $s \overset{\widehat{a}}{\Longrightarrow}{}^{fin,d} M$ iff there exists $\mu \in M$ with $s \overset{\widehat{a}}{\Longrightarrow}{}^{fin,d}_M \mu$.

The delay transitions relations $\Longrightarrow^{fin,r}_M$, $\Longrightarrow^{\omega,d}_M$, $\Longrightarrow^{\omega,r}_M$, $\Longrightarrow^{fin,r} M$, $\Longrightarrow^{\omega,d} M$ and $\Longrightarrow^{\omega,r} M$ are defined by taking the corresponding classes of adversaries.

The following proposition is an immediate consequence of Proposition 5.5.2.

Proposition 5.5.4 *For all s, a and μ*

$$s \overset{\widehat{a}}{\Longrightarrow}{}^{fin,d} \mu\uparrow_R \text{ iff } DAdv_{\perp}^{fin}[s, \tau^*\widehat{a}, \mu\uparrow_R] \neq \emptyset.$$

For the relations $\overset{\widehat{a}}{\Longrightarrow}{}^{fin,r}\mu\uparrow_R$, $\overset{\widehat{a}}{\Longrightarrow}{}^{\omega,d}\mu\uparrow_R$ and $\overset{\widehat{a}}{\Longrightarrow}{}^{\omega,r}\mu\uparrow_R$, we have similar results.

The following proposition states that the use of deterministic and randomized adversaries gives rise to the same notion of delay transitions if we consider target sets of the form $\mu\uparrow_R$. Therefore, we will omit the subscripts r and d in the delay transitions reaching such sets.

Proposition 5.5.5 *For all s, a and μ we have*

1. $s \overset{\widehat{a}}{\Longrightarrow}{}^{\omega,d} \mu\uparrow_R$ *iff* $s \overset{\widehat{a}}{\Longrightarrow}{}^{\omega,r} \mu\uparrow_R$.
2. $s \overset{\widehat{a}}{\Longrightarrow}{}^{fin,d} \mu\uparrow_R$ *iff* $s \overset{\widehat{a}}{\Longrightarrow}{}^{fin,r} \mu\uparrow_R$.

PROOF:

1. The "only-if" part is clear. For the "if"-part, we use Proposition 5.5.4. It is easy to see that if $A \in RAdv_{\perp}^{fin}[s, \tau^*\widehat{a}, \mu\uparrow_R]$, then every D in $\mathcal{D}(s,A)$ (where $\mathcal{D}(s,A)$ is as in Notation 5.3.23) is an element of $DAdv_{\perp}^{fin}[s, \tau^*\widehat{a}, \mu\uparrow_R]$.
2. Similarly.

□

The following proposition, which is an immediate consequence of the definitions, is used in the correctness arguments for the decidability algorithms for the delay (bi–)simulation relations.

Proposition 5.5.6 *For all s, a and M we have*

1. $s \overset{\widehat{a}}{\Longrightarrow}{}^{\omega,d} M$ *iff* $s \overset{\widehat{a}}{\Longrightarrow}{}^{\omega,d} M \cap \bigcup_{a,t} Steps_a(t)$.

2. $s \stackrel{\widehat{a}}{=\!=\!\Rightarrow}^{\mathit{fin},d} M$ *iff* $s \stackrel{\widehat{a}}{=\!=\!\Rightarrow}^{\mathit{fin},d} M \cap \bigcup_{a,t} \mathit{Steps}_a(t)$.

Example 5.5.7 Consider the systems in Figure 5.11. Let R be the smallest equivalence over S that identifies the elements s, t and t'.

- For the leftmost system, we have that $s \stackrel{\widehat{a}}{=\!=\!\Rightarrow}^{\mathit{fin}} \nu\!\uparrow_R$. Note that $\nu = \nu'$ and therefore $\nu \equiv_R \nu'$. This delay transition is established by the deterministic scheduler that in each state takes the unique outgoing transition with probability one.
- For the rightmost system in the same figure, we do not have $s \stackrel{\widehat{a}}{=\!=\!\Rightarrow}^{\mathit{fin}} \rho\uparrow_R$ because $\mathit{RAdv}^{\mathit{fin}}[s, \tau^* a, \rho\!\uparrow_R] = \emptyset$. Each scheduler over $\tau^* a$ has to schedule the ρ'–transition with a positive probability, but $\rho \not\equiv_R \rho'$.

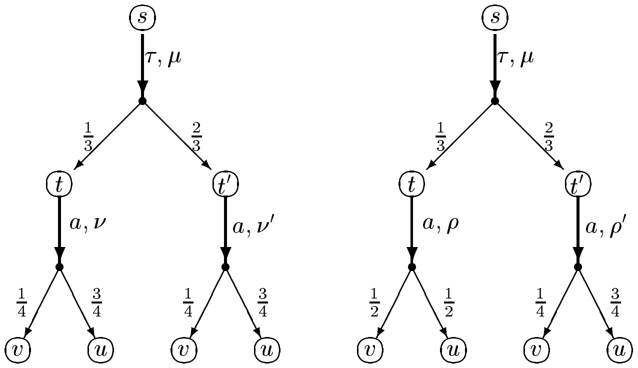


Figure 5.11: Delay transitions

Example 5.5.8 Reconsider the system in Figure 5.3 on page 95. Let R be the smallest equivalence relation identifying s and s' and identifying t and u. Since $\mu \equiv_R \rho$, the adversary D from Example 5.3.12 on page 94 is an element of $\mathit{DAdv}^{\mathit{fin}}[s, a, \mu\uparrow_R]$. Hence, D establishes that $s \stackrel{a}{=\!=\!\Rightarrow}^{\mathit{fin}} \mu\uparrow_R$. Similarly, we have that $s' \stackrel{a}{=\!=\!\Rightarrow}^{\mathit{fin}} \rho\uparrow_R$. Note that the step $s' \stackrel{\varepsilon}{=\!=\!\Rightarrow}^{\mathit{fin}} \{s' \mapsto 1\}$ is also a finitary delay transition.

Example 5.5.9 Now consider the system in Fig 5.8 on page 108. Let R be the smallest equivalence relation identifying s and s'. The adversary D_ω from Example 5.3.12 (page 94) is an element of $\mathit{DAdv}^{\omega}[s, a, \mu\uparrow_R]$ and hence establishes that $s \stackrel{a}{=\!=\!\Rightarrow}^{\omega} \mu\uparrow_R$. Note that $\mathit{DAdv}^{\mathit{fin}}[s, a, \mu\!\uparrow_R]$ is empty, so we do not have $s \stackrel{a}{=\!=\!\Rightarrow}^{\mathit{fin}} \mu\!\uparrow_R$.

Example 5.5.10 Consider the system in Figure 5.9 on page 108. Let $R = \{(s, s'), (u_1, u_1'), (u_2, u_2'), (u_3, u_3'), (u_4, u_4')\}$. Example 5.4.13 (page 5.4.13) has shown that $s' \stackrel{a}{\Longrightarrow}^{\mathit{fin},d} \nu'\uparrow_R$ for ν' with $\nu'(u_i') = \frac{1}{4}$. Since $\nu' \not\sqsubseteq_R \nu_1$, we have $\mathit{DAdv}^{\mathit{fin}}[s, a, \nu'\uparrow_R] = \emptyset$ and hence $s' \not\stackrel{a}{=\!=\!\Rightarrow}^{\mathit{fin}} \nu'\uparrow_R$. (For the same reason, we do not have $s' \stackrel{a}{=\!=\!\Rightarrow}^{\omega} \nu'\!\uparrow_R$ either.) This illustrates the fact that, unlike for weak transitions, the target distribution of a delayed transition cannot be obtained by "multiplying probabilities along the maximal paths."

Definition 5.5.11 (Delay (bi–)simulation) A binary relation R on S is called a *delay ω–simulation* iff for all $(s, s') \in R$ and $\mu \in \mathit{Distr}(S)$

$$\text{if } s \xrightarrow{a} \mu \text{ then } s' \stackrel{\widehat{a}}{\Longrightarrow}^{\omega} \mu\uparrow_R.$$

A *delay ω–bisimulation* is an equivalence which is a delay ω–simulation. We write $s \preceq^{\omega}_{\mathit{del}} s'$ iff there exists a delay ω–simulation R such that $(s, s') \in R$. In that case, we say that s' *delay ω–simulates* s. We write $s \approx^{\omega}_{\mathit{del}} s'$ iff there exists a delay ω–bisimulation containing (s, s'). In that case s and s' are said to be *delay ω–bisimilar*. Now the finitary delay simulation relation $\approx^{\mathit{fin}}_{\mathit{del}}$ and the finitary delay bisimulation equivalence $\preceq^{\mathit{fin}}_{\mathit{del}}$ are defined in a similar way.

Example 5.5.12 Reconsider the system in Figure 5.3 on page 95. Let R be the smallest equivalence relation R identifying s and s' and identifying t and u. Then R is a finitary delay bisimulation. The step $s \xrightarrow{\tau} \nu$ in s is matched by the step $s' \stackrel{\varepsilon}{\Longrightarrow}^{\mathit{fin}} \{s' \mapsto 1\}$. The step $s' \xrightarrow{a} \mu$ is matched by $s \stackrel{a}{\Longrightarrow}^{\mathit{fin}} \mu\uparrow_R$ and the step $s \xrightarrow{a} \rho$ is matched by $s' \stackrel{a}{\Longrightarrow}^{\mathit{fin}} \rho\uparrow_R$. Thus, $s \approx^{\mathit{fin}}_{\mathit{del}} s'$.

Example 5.5.13 Now consider the system in Fig 5.8 on page 108. Let R be the smallest equivalence relation R identifying s and s'. We claim that R is an almost finitary, delay bisimulation. The step $s \xrightarrow{\tau} \nu$ is matched by $s' \stackrel{\varepsilon}{\Longrightarrow}^{\omega} \nu\uparrow_R$ as before and the step $s' \xrightarrow{a} \mu$ is matched by $s \stackrel{a}{\Longrightarrow}^{\omega} \mu\uparrow_R$. Therefore, $s \approx^{\omega}_{\mathit{del}} s'$. Since $s \;\not\stackrel{a}{\Longrightarrow}^{\mathit{fin}} \mu\uparrow_R$, we do not have $s \approx^{\mathit{fin}}_{\mathit{del}} s'$.

Example 5.5.14 Consider the system in Figure 5.9. Then the relation $R = \{(s, s'), (u_1, u_1'), (u_2, u_2'), (u_3, u_3'), (u_4, u_4')\}$ is not a delay simulation, because the transition $s \xrightarrow{a} \mu$ cannot be matched in the state s', since there is no outgoing delay transition labeled by a. Recall from Example 5.4.13 (page 108) that s' does weakly simulate s.

5.5.2 Alternative Characterization with Norm Functions

We now show that the several types of the delay relations can be characterized by probabilistic variants of *norm functions*, introduced by Griffioen & Vaandrager [GV98]. Our formulation differs from [GV98] in two ways. Firstly, we work with norm functions whose codomain are the natural numbers, whereas in [GV98], the range of a norm function can be any well–ordered set. Moreover, we adapt the formulation such that we characterize the delay variants, whereas [GV98] treats branching (bi-)simulations. As remarked before, it is not difficult to adapt the theory presented here for branching bisimulation.

In the nonprobabilistic case, a norm function for a binary relation R is a partial function $\mathit{norm} : S \times \mathit{Act}_\tau \times S \times S \hookrightarrow \mathbb{N}$ such that[9]

1. If $s \xrightarrow{a} u$ and $(s, s') \in R$, then $(s, a, u, s') \in \mathit{dom}(\mathit{norm})$,
2. If $(s, a, u, s') \in \mathit{dom}(\mathit{norm})$, then $s \xrightarrow{a} u$ and the following holds.
   - If $\mathit{norm}(s \xrightarrow{a} u, s') = 0$, then $a = \tau$ and $(u, s') \in R$,
   - if $\mathit{norm}(s \xrightarrow{a} u, s') = 1$, then there is a transition $s' \xrightarrow{a} u'$ where $(u, u') \in R$,
   - otherwise, there is a transition $s' \xrightarrow{\tau} t'$ such that

$$(s, a, u, t') \in \mathit{dom}(\mathit{norm}) \text{ and } \mathit{norm}(s \xrightarrow{a} u, t') < \mathit{norm}(s \xrightarrow{a} u, s').$$

[9]Recall that $\mathit{dom}(\mathit{norm})$ denotes the domain of norm (Section 5.2). We write $\mathit{norm}(s \xrightarrow{a} \mu, s')$ rather than $\mathit{norm}(s, a, \mu, s')$.

Intuitively, the value $norm(s \xrightarrow{a} u, s')$ is an upper bound for the number of τ transitions that are needed to establish a weak transition $s' \overset{a}{\Longrightarrow} u'$ such that $(u, u') \in R$. More precisely, if $(s, s') \in R$ and $norm(s \xrightarrow{a} u, s') = n$, then there is a path $s' = s_0 \xrightarrow{\tau} s_1 \xrightarrow{\tau} s_2 \dots s_n \xrightarrow{a} t'$ which consists of exactly n τ–transitions and ends with an a–transition such that $(s', t') \in R$. Thus, this path shows that $s' \overset{a}{\Longrightarrow} t'$ for some t' with $(s', t') \in R$. Using this property one can show that an equivalence R on the state space S is a weak bisimulation iff there is a norm function for R with

$$dom(norm) = \{(s, a, u, s') : s \xrightarrow{a} u \ \wedge \ (s, s') \in R\}.$$

Thus, norm functions provide an alternative characterization of weak[10] bisimulations.

For probabilistic systems, there are two alternative ways to generalize the third clause in condition (2) above. Firstly, given a transition $s \xrightarrow{a} \mu$ and a pair $(s, s') \in R$, we may require the existence of a transition $s' \xrightarrow{\tau} \nu'$ such that *for all states* $t' \in supp(\nu')$ the value $norm(s \xrightarrow{a} \mu, t')$ is defined and strictly less than $norm(s \xrightarrow{a} \mu, s')$. Second, we may use a weaker condition and require the existence of a transition $s' \xrightarrow{\tau} \nu'$ such that for all states $t' \in supp(\nu')$ the value $norm(s \xrightarrow{a} \mu, t')$ is defined and *for some state* $t' \in supp(\nu')$ $norm(s \xrightarrow{a} \mu, t')$ is strictly less than $norm(s \xrightarrow{a} \mu, s')$. We will see that norm functions of the former type provide a characterization of the finitary delay transitions, whereas the latter characterize the ω-variants.

Definition 5.5.15 (Norm functions) Let R be a binary relation on S. A *norm function* for R is partial function

$$norm : S \times Act_\tau \times Distr(S) \times S \hookrightarrow \mathbb{N}$$

which satisfies the following conditions.

1. If $s \xrightarrow{a} \mu$ and $(s, s') \in R$ then $(s, a, \mu, s') \in dom(norm)$.

2. If $(s, a, \mu, s') \in dom(norm)$ then $s \xrightarrow{a} u$ and one of the following conditions holds.

   - If $norm(s \xrightarrow{a} \mu, s') = 0$ then $a = \tau$ and $\mu \sqsubseteq_R \mu^1_{s'}$. [11]
   - If $norm(s \xrightarrow{a} \mu, s') = 1$ then there exists a transition $s' \xrightarrow{a} \mu'$ with $\mu \sqsubseteq_R \mu'$.
   - If $norm(s \xrightarrow{a} \mu, s') > 1$ then there exists a transition $s' \xrightarrow{\tau} \nu'$ such that

$$\forall t' \in supp(\nu')[(s, a, \mu, t') \in dom(norm)] \qquad \text{(WN)}$$

$$\exists t' \in supp(\nu')[norm(s \xrightarrow{a} \mu, t') < norm(s \xrightarrow{a} \mu, s')] \qquad \text{(N}\exists\text{)}$$

A norm function $norm$ for R is called *strict* iff condition (N$\exists$) is replaced by the following stronger condition (N$\forall$).

$$\forall t' \in supp(\nu')[norm(s \xrightarrow{a} \mu, t') < norm(s \xrightarrow{a} \mu, s')]. \qquad \text{(N}\forall\text{)}$$

[10] If we add the condition $(s, t') \in R$ in the third clause of (2), then we get a characterization of branching bisimulation.

[11] The condition $\mu \sqsubseteq_R \mu^1_{s'}$ is equivalent to the requirement that $supp(\mu) \subseteq \{t \in S : (t, s') \in R\}$.

Example 5.5.16 Consider the leftmost system in Figure 5.11. Let $R = \{(t, s), (t', s), (u, u), (v, v)\}$. A strict norm function for R is given by

$$norm(t \xrightarrow{a} \nu, s) = 2$$
$$norm(t' \xrightarrow{a} \nu', s) = 2$$
$$norm(t \xrightarrow{a} \nu, t) = 1$$
$$norm(t' \xrightarrow{a} \nu, t') = 1$$

and undefined for all other elements. The values for the function $x \mapsto norm(s' \xrightarrow{a} \nu, x)$ are depicted in Figure 5.12.

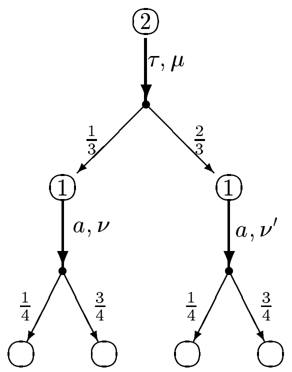


Figure 5.12: The function $x \mapsto norm(s \xrightarrow{a} \nu, x)$

Example 5.5.17 Reconsider the system in Figure 5.3 on page 95. Let R be the the smallest equivalence relation R identifying s and s' and identifying t and u. Then a strict norm function for R is given by

$$norm(s' \xrightarrow{a} \rho, s) = 1$$
$$norm(s' \xrightarrow{a} \rho, s') = 1$$
$$norm(s' \xrightarrow{a} \mu, s) = 1$$
$$norm(s' \xrightarrow{a} \mu, s') = 1$$
$$norm(s \xrightarrow{\tau} \nu, s) = 1$$
$$norm(s \xrightarrow{\tau} \nu, s') = 0$$

and undefined for all other elements. The values for the function $x \mapsto norm(s' \xrightarrow{a} \mu, x)$ are depicted in Figure 5.13.

Example 5.5.18 Now consider the system in Fig 5.8. Let R be the the smallest equivalence relation R identifying s and s'. A norm for R is given by

$$norm(s' \xrightarrow{a} \mu, s) = 2$$
$$norm(s' \xrightarrow{a} \mu, s') = 1$$
$$norm(s \xrightarrow{\tau} \nu, s) = 1$$
$$norm(s \xrightarrow{\tau} \nu, s') = 0$$

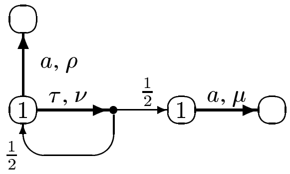


Figure 5.13: The function $x \mapsto norm(s' \xrightarrow{a} \mu, x)$

and undefined for all other elements. The values for the function $x \mapsto norm(s' \xrightarrow{a} \mu, x)$ are depicted in Figure 5.14.

Note that there is no strict norm function for R.

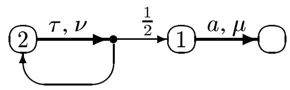


Figure 5.14: The function $x \mapsto norm(s' \xrightarrow{a} \mu, x)$

The reader may have noticed that the examples above correspond to Examples 5.5.8, 5.5.9 and 5.5.10, which showed the existence of delay transitions. This is not a coincidence. The following two propositions provide a connection between norm functions and the delay transition relations.

Proposition 5.5.19 *Let R be a binary relation on S and norm a norm function for R. Then, we have:*

1. *If $(s, a, \mu, s') \in dom(norm)$ then $s' \overset{\widehat{a}}{\Longrightarrow}^{\omega} \mu\!\uparrow_R$.*
2. *If norm is strict and $(s, a, \mu, s') \in dom(norm)$ then $s' \overset{\widehat{a}}{\Longrightarrow}^{fin} \mu\!\uparrow_R$.*

PROOF: Assume that $(s, a, \mu, s') \in dom(norm)$ then $s \xrightarrow{a} \mu$ is a transition. We define a deterministic partial adversary D in $DAdv^{\omega}[s', \tau^* a, \mu\!\uparrow_R]$ and hence $s \overset{\widehat{a}}{\Longrightarrow}^{\omega} \mu\!\uparrow_R$.

First, define an auxiliary function $B : S \to Act_\tau \times Distr(S) \cup \{\perp\}$ as follows.

- If $(s, a, \mu, s') \notin dom(norm)$, put $B(s') = \perp$,

and for $(s, a, \mu, s') \in dom(norm)$, define $B(s')$ by

- if $norm(s \xrightarrow{a} \mu, s') = 0$, define $B(s') = \perp$,
- if $norm(s \xrightarrow{a} \mu, s') = 1$, choose $s' \xrightarrow{a} \mu'$, where $\mu \sqsubseteq_R \mu'$ and put $B(s') = (a, \mu')$,
- if $norm(s \xrightarrow{a} \mu, s') > 1$, then choose a transition $s' \xrightarrow{\tau} \nu'$ such that conditions (WN) and (N∃) of Definition 5.5.15 are fulfilled and put $B(s') = (\tau, \nu')$. For strict normed functions we require condition (N∀) instead of (N∃).

Let $\Sigma = \{\sigma \in Path(s') \mid word(\sigma) \in \tau^*\}$. Now define the adversary D by

$$D(\sigma) = \begin{cases} B(last(\sigma)) & \text{if } \sigma \in \Sigma, \\ \perp & \text{otherwise.} \end{cases}$$

Below, we show that D is almost finitary and then it is easy to see that $D \in DAdv^{\omega}[s', \widehat{a}, \mu{\uparrow}_R]$. Then Proposition 5.5.4 implies $s' \stackrel{\widehat{a}}{\Longrightarrow}{}^{\omega} \mu{\uparrow}_R$ and we are done for statement 1. For part 2., we will prove below that, if the function $norm$ is strict, then $D \in DAdv^{fin}[s', \widehat{a}, \mu{\uparrow}_R]$ and then we are also done by Proposition 5.5.4.

By the results of [Bai98] (or using the results of [BK98]) it follows that D is almost finitary. This can be seen as follows. The adversary D induces a finite Markov chain $(U, \mathbf{Q})$ with state space

$$U = \{u \in S : (s \xrightarrow{a} \mu, u) \in dom(norm)\}$$

and the transition probabilities $\mathbf{Q}$ given by

$$\mathbf{Q}(u, u') = \begin{cases} \nu(u') & \text{if } norm(s \xrightarrow{a} \mu, u) \geq 2 \text{ and } B(u) = (\tau, \nu), \\ 1 & \text{if } norm(s \xrightarrow{a} \mu, u) \leq 1 \text{ and } u = u', \\ 0 & \text{otherwise.} \end{cases}$$

The absorbing states in this Markov chain are given by the set $U_0 = \{u \mid norm(s \xrightarrow{a} \mu, u) \in \{0, 1\}\}$. Let $Reach(u)$ denote the set of states that are reachable from u. Clearly, $Reach(u) \cap U_0 \neq \emptyset$ for all $u \in U$. Let $\mathbf{Q}^*(s', u')$ denote the probability for state s' to reach the state $u' \in U_0$ in this Markov chain. Then, an elementary result in the theory of Markov chains yields that the probability to reach a state in U_0 is 1, no matter from which state $u \in U$ one starts. It is not difficult to see that $\mathbf{P}^D(s, s') = \mathbf{Q}^*(s, s')$. As a consequence,

$$\mathbf{P}^D(s', S) = \sum_{u' \in U_0} \mathbf{Q}^*(s', u') = 1,$$

in other words, D is almost finitary.

Now, assume that $norm$ is strict. Then, the adversary D defined above is finitary. This can be seen as follows. For any path $s' \stackrel{\tau}{\rightsquigarrow} s'_1 \stackrel{\tau}{\rightsquigarrow} \ldots \stackrel{\tau}{\rightsquigarrow} s'_l$ in D, we have

$$norm(s \xrightarrow{a} \mu, s') > norm(s \xrightarrow{a} \mu, s'_1) > \ldots > norm(s \xrightarrow{a} \mu, s'_l).$$

It then follows that any maximal path in D is finite and, therefore, D is finitary. □

Thus, the existence of a norm function for R implies that R is a delay ω-simulation. Similarly, if R has a strict norm function then R is a finitary delay simulation. The next proposition shows the converse, i.e. that any delay ω–simulation has a norm function and any delay finitary simulation has a strict norm function.

Proposition 5.5.20 *Let R be a binary relation on S.*

1. *If R is ω–simulation, then there exists a norm function $norm$ such that $(s, a, \mu, s') \in dom(norm)$ whenever $(s, s') \in R$ and $s \xrightarrow{a} \mu$.*

2. *If R is finitary simulation, then there exists a strict norm function $norm$ such that $(s, a, \mu, s') \in dom(norm)$ whenever $(s, s') \in R$ and $s \xrightarrow{a} \mu$.*

PROOF: Let R be a binary subset of S, $a \in Act_\tau$ and $\rho \in Distr(S)$. We define $\mathcal{U}[s, a, \rho, R]$ as the set of deterministic almost finitary partial adversaries D that induces a delay ω–delay transition from s to $\rho{\uparrow}_R$. We define

$$n_R(s, a, \rho) = \min \{ \ell(s, D) : D \in \mathcal{U}[s, a, \rho, R] \}$$

where $\min \emptyset = \infty$ and $\ell(s, D) = \min \left\{ \mathit{length}(\sigma) : \sigma \in \mathit{Path}^D_{\mathit{fmax}}(s) \right\}$ is the length of a shortest finite maximal path in D with starting state s. If $n_R(s', a, \rho) < \infty$ then we put

$$\mathit{norm}_R \left(s \xrightarrow{a} \mu, s' \right) = n_R(s', a, \mu).$$

It is easy to see that norm_R is a norm function for R provided that R is a delay ω-simulation. For part (b), we replace $\mathcal{U}[s, a, \rho, R]$ by $\mathcal{U}^{\mathit{fin}}[s, a, \rho, R]$, which is the set of all partial adversaries $D \in \mathcal{U}[s, a, \rho, R]$ that are finitary. Moreover, we replace $\ell(s, D)$ by $\ell^{\mathit{fin}}(s, D) = \max \left\{ \mathit{length}(\sigma) : \sigma \in \mathit{Path}^D_{\mathit{fmax}}(s) \right\}$. (Since D is finitary, this maximum is finite.) Hence, we get that norm_R is strict if R is a delay simulation. □

5.5.3 Axiomatization

The finitary delay transition can be characterized by axioms and rules in a similar way as the finitary weak transitions (Section 5.4.4). We adapt the rules (TR) and (PR) by requiring that they yield distributions in the class M.

Note that there are no counterparts to the rules (WR) and (CC). The counterpart of (WR) is missing because delay transitions do not combine τ-transitions on the right and the rule (CC) may be skipped since Proposition 5.5.5 states that the choice between deterministic and randomized partial adversaries is irrelevant for the delay transitions.

Proposition 5.5.21 *Consider the axioms and rules of Figure 5.15. The finitary delayed transition relation can be characterized as the smallest subsets of $S \times Act_\varepsilon \times Distr(S)$ satisfying the axioms and rules* (DelTR)*,* (DelPR) *and* (DelWL).

PROOF: By induction on the structure of the derivation, one easily shows that every transition that can be derived from the axioms is indeed a finitary delay transition. Conversely, to show that every delay transition can be derived from the axioms, we use an argument similar to the proof of Proposition 5.5.20. For $s \overset{a}{=\!=\!\Rightarrow} M$, define $\mathcal{U}^{\mathit{fin}}[s, a, M]$ as the set of deterministic finitary partial adversaries D that induce a deterministic delay transition from s to M and where $D(s) \neq \bot$. For every state s and every finitary deterministic adversary D, let $\ell(s, D) = \max\{\mathit{length}(\sigma) : \sigma \in \mathit{Path}^D_{\mathit{fmax}}(s)\}$. Since D is finitary, this maximum is finite. By induction on $\ell(s, D)$, it can be shown that for all deterministic adversaries D that establish $s \overset{a}{=\!=\!\Rightarrow} M$ we also have a derivation from the axioms. □

Example 5.5.22 Reconsider the leftmost system in Figure 5.11. Let R be the smallest equivalence over S that identifies the elements s, t and t'. Example 5.5.7 has shown that $s \overset{\widehat{a}}{=\!=\!\Rightarrow}{}^{\mathit{fin}} \nu \uparrow_R$. This can also be obtained by the following derivation, in which the final step is justified by the axiom (DelPR).

$$\dfrac{s \xrightarrow{\tau} \mu \;\; \mathit{supp}(\mu) = \{t, t'\} \quad \dfrac{t \xrightarrow{a} \nu \uparrow_R}{t \overset{a}{=\!=\!\Rightarrow}{}^{\mathit{fin}} \nu \uparrow_R}\,(\mathrm{DelPR}) \quad \dfrac{t' \xrightarrow{a} \nu' \uparrow_R = \nu \uparrow_R}{t' \overset{a}{=\!=\!\Rightarrow}{}^{\mathit{fin}} \nu \uparrow_R}}{s \overset{a}{=\!=\!\Rightarrow}{}^{\mathit{fin}} \nu \uparrow_R}\,(\mathrm{DelWL})$$

(DelTR) If $\mu_s^1 \in M$ then $s \stackrel{\varepsilon}{\Longrightarrow} M$.

(DelPR) If $s \stackrel{a}{\longrightarrow} \mu$ and $\mu \in M$ then $s \stackrel{\widehat{a}}{\Longrightarrow} M$.

(DelWL) If $s \stackrel{\tau}{\longrightarrow} \nu$ and $t \stackrel{a}{\Longrightarrow} M$ for all $t \in supp(\nu)$, then $s \stackrel{\widehat{a}}{\Longrightarrow} M$.

Figure 5.15: Axioms and rules for the finitary delay transitions

5.6 Basic Properties

This section lists several basic properties for simulation and bisimulation relations defined in previous sections. We focus on those properties that are important for the decidability algorithms in Section 5.7.

An important conclusion from this section is that the deterministic weak (bi–)simulation relations have properties that make them unsuitable for practical verification. All the other (bi–)simulations enjoy properties that are natural generalizations of the nonprobabilistic setting. In particular, this holds for the strong (bi–)simulation. Since strong (bi–)simulation can be considered as the deterministic variant of strong combined (bi–)simulation, the effects of determinism versus randomization differ in the strong and the weak case.

To avoid tedious repetitions, the results in this section have – quite arbitrarily – been formulated for ω–delay (bi–)simulation. However, the corresponding results hold and are also proven for the following variants.

- finitary delay (bi–)simulation,
- randomized, finitary (bi–)simulation,
- randomized, ω (bi–)simulation,
- strong (bi–)simulation,
- strong combined (bi–)simulation.

That is, in the propositions in this section, one can substitute the ω–delay (bi–)simulation with any of the variants listed above, provided that one also takes the corresponding transition relations. We provide several counter examples, showing that the results do not hold for the deterministic, weak ω and finitary (bi–)simulation.

In the nonprobabilistic case, it is well–known that one can replace the transition relation $\rightarrow$ in the definition of weak (bi–)simulation, by the weak transition relation $\Longrightarrow$. This characterization also holds in the probabilistic case when we deal with the randomized variants of simulations, but fails for the deterministic variants. This result plays an important role in the proofs for several other results. Note that, for the strong simulation relation, the proposition is exactly the same as the definition.

Proposition 5.6.1 *Let R be a binary relation (an equivalence relation, respectively) on S. Then R is a delay ω–simulation relation (a delay ω–bisimulation) if and only if for all $(s, s') \in R$ and all $\mu \in \mathrm{Distr}(S)$*

$$s \stackrel{a}{\Longrightarrow}^{\omega} \mu{\uparrow_R} \ \textit{implies}\ s' \stackrel{\widehat{a}}{\Longrightarrow}^{\omega} \mu{\uparrow_R} .$$

Remark 5.6.2 The corresponding result of Proposition 5.6.1 does not hold for the deterministic (bi-)simulation relations. For example, consider the states s, s' of Figure 5.16 and the smallest equivalence R that identifies the states s, s', t and u. Then s and s' are weakly, deterministically bisimilar. However, the weak, deterministic transition $s \stackrel{a}{\Longrightarrow}^{\mathit{fin},d} \mu$, where $\mu(x) = 1/3$, $\mu(y) = 2/3$, cannot be matched by a weak, deterministic transition leaving from s'.

The following result is essential for the correctness of the decidability procedures based on partition refinement.

Proposition 5.6.3 *1. The relation $\preceq^{\omega}_{del}$ is a preorder.*

2. The relation $\preceq^{\omega}_{del}$ is the coarsest delay ω–simulation relation.

PROOF: As in the nonprobabilistic case, using Proposition 5.6.1. □

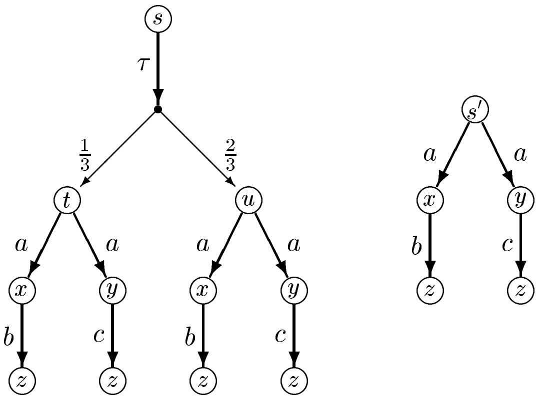


Figure 5.16: $s \approx^{d,\mathit{fin}} s'$ and $s \stackrel{a}{\Longrightarrow}^{\mathit{fin},d} \mu$, but not $s' \stackrel{a}{\Longrightarrow}^{\mathit{fin},d} \mu$.

Proposition 5.6.4 *1. The relation $\approx^{\omega}_{del}$ is an equivalence.*

2. The relation $\approx^{\omega}_{del}$ is the coarsest delay ω–bisimulation relation.

PROOF: As in the nonprobabilistic case, using Proposition 5.6.1. □

The following example shows that the deterministic simulations are not transitive.

Example 5.6.5 Consider the system in Figure 5.17. We assume that s_2 is not simulated by t_3, s_3 not by s_2, t_2 not by u_3, t_3 not by u_2, due to transitions not shown in the picture.

Let

$$R = \{(s_1, t_1), (s_2, t_2), (s_3, t_3), (s_4, t_4), (s_5, t_5)\}$$
$$T = \{(t_1, u_1), (t_1', u_1), (t_1'', u_1), (t_2, u_2), (t_3, u_3), (t_4, u_4), (t_5, u_5)\}.$$

Then R shows that $s_1 \preceq^{d}_{wbis} t_1$ and T establishes $t_1 \preceq^{d}_{wbis} u_1$. Due to the absence of randomized adversaries, we *do not* have $s_1 \preceq^{d}_{wbis} u_1$.

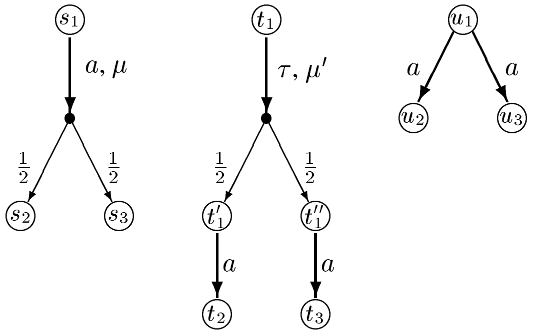


Figure 5.17: $s_1 \preceq^d_{wbis} t_1$, $t_1 \preceq^d_{wbis} u_1$, $s_1 \not\preceq^d_{wbis} u_1$ and $s_1 \not\preceq_{dwbis} u_1$.

The following proposition states that our notions of probabilistic (bi–)simulation are conservative extensions of (bi–)simulation on nonprobabilistic systems.

Proposition 5.6.6 *When applied to a nonprobabilistic labeled transition system $(S, \longrightarrow)$, the notions of strong and strong combined (bi–)simulation coincide with the classical notions of strong (bi–)simulation on nonprobabilistic systems. Similarly, the notion of delay (bi–)simulation coincides with the classical notion of delay (bi–)simulation on nonprobabilistic systems. Finally, the notion of weak (bi–)simulation coincides with the classical notion of delay weak (bi–)simulation on nonprobabilistic systems [Mil80].*

Proposition 5.6.7 *The deterministic weak (bi-)simulation relations are strictly coarser than the randomized ones, the finitary (bi-)simulation relations are strictly coarser than the ω–ones. The strong (bi-)simulation simulation relations are strictly finer than the strong combined ones. The delay weak (bi-)simulation variants are strictly between the strong and the weak (bi-)simulation. The strong combined (bi-)simulation variants and the delay variants are incomparable.*

Proposition 5.6.8 (cf. [Seg95b]) *The $\preceq^\omega_{del}$ is a precongruence and the $\approx^\omega_{del}$ is a congruence with respect to the parallel composition operator.*

The following theorem states the soundness of the simulations for trace distribution inclusion $\sqsubseteq_{\mathrm{TD}}$ and of the bisimulations for trace distribution equivalence $\equiv_{\mathrm{TD}}$. Recall that $\sqsubseteq_{\mathrm{TD}}$ and $\equiv_{\mathrm{TD}}$ were defined in Definition 2.4.1 on page 49 and that the (bi–)simulation relations can be lifted to PAs via their start states.

Theorem 5.6.9 1. *If $\mathcal{A} \preceq^\omega_{del} \mathcal{B}$, then $\mathcal{A} \sqsubseteq_{\mathrm{TD}} \mathcal{B}$.*

2. *If $\mathcal{A} \approx^\omega_{del} \mathcal{B}$, then $\mathcal{A} \equiv_{\mathrm{TD}} \mathcal{B}$.*

PROOF: (sketch) For the proof of part (1) of this theorem, we refer to Segala [Seg95b]. Since any of the simulation relations considered in this chapter is finer than the weak, randomized ω–simulations and since the latter are proven to be sound for trace distribution inclusion, all the relations from this chapter are also sound for it. Then the proof of part (2) follows immediately. □

5.7 Decidability Algorithms for Simulation and Bisimulation Relations

In the previous sections, we treated the notions of strong, strong combined, weak and delayed (bi–)simulation. This section is concerned with the decidability and complexity results for these (bi–)simulation relations.

A decidability algorithm for a bisimulation relation $\approx$ is an algorithm that takes a system $(S, \rightarrow)$ and two states s and s' in S and yields whether or not $s \approx s'$. The algorithms in this section for deciding bisimulations are all based on a so–called *partition refinement technique*. This means that the algorithm obtains the partition $S/\approx$ by starting from the trivial partition $\{S\}$ and successively refining it into finer partitions until the result $S/\approx$ is obtained. Then one can check whether two states are bisimilar by verifying whether they are in the same block in $S/\approx$. As observed by [Her99], computing the entire quotient space just for deciding the bisimilarity of two states may seem too much work, but is in fact not, because bisimilarity intrinsically involves all states of the system.

The algorithms for deciding simulation preorders work similarly. These compute the pairs contained in a simulation relation $\sqsubseteq$ by starting with the trivial preorder S^2 and successively remove pairs which are not in the simulation preorder until the relation $\sqsubseteq$ is obtained.

In this section, we will firstly summarize the main existing algorithmic methods for computing the (bi–)simulation relations, both for the probabilistic and the nonprobabilistic case. We concentrate on those aspects relevant for the algorithms to compute the delay (bi–)simulations. We treat polynomial time algorithms for deciding strong and weak (bi–)simulation in the nonprobabilistic case and for strong (bi–)simulation in the probabilistic case. To the best of our knowledge, there are no published results on the decidability of strong combined and weak (bi–)simulation relations from Section 5.4, but these are strongly conjectured to be decidable in polynomial time [SC01].

Most of this section is devoted to the algorithms for deciding the delay simulation and bisimulation relations defined in Section 5.5. All these algorithms run in polynomial time and space.

In the sequel, we assume a finite probabilistic system $(S, Steps)$ with n states and m transitions (i.e. $n = |S|$ and $m = \sum_{s \in S} |Steps(s)|$).

5.7.1 Decidability of Strong (Bi–)simulation

The algorithms to compute the strong (bi–)simulations are the basis for the algorithms computing several other bisimulation relations.

Strong bisimulation The main idea for computing the strong bisimulation equivalence classes in nonprobabilistic systems is the use of a *partitioning technique* as proposed by Kanellakis & Smolka [KS90] (and improved by Paige & Tarjan [PT87]). This technique is sketched in Figure 5.18: We 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, until splitting is not possible anymore. In that case, the partition $\mathcal{X}$ equals the quotient space $S/\approx_{sbis}$ and we are done.

The refinement operator depends on a *splitter* of $\mathcal{X}$. Intuitively, a splitter is a pair $\langle a, A \rangle$ consisting of an action a and a block $A \in \mathcal{X}$ that prevents the induced equivalence $R_{\mathcal{X}}$ to fulfill the condition of a strong bisimulation. That is, a splitter is a pair $\langle a, A \rangle$ such that

Computing the bisimulation equivalence classes

Input: a finite (nonprobabilistic or probabilistic) system with state space S

Output: $S/\approx_{bis}$ for a (strong or weak) bisimulation equivalence $\approx_{bis}$

Method:

While there exists a splitter candidate sc of $\mathcal{X}$ do
$\mathcal{X} := \mathit{Refine}(\mathcal{X}, sc)$;
Return $\mathcal{X}$.

Figure 5.18: Schema for computing the bisimulation equivalence classes

there are states s, $s' \in S$ that belong to the same block C of $\mathcal{X}$), but where $s \xrightarrow{a} A$ while $s \stackrel{a}{\not\longrightarrow} A$. s' cannot move via a to a state of A[12]. Then we know that s and s' are not bisimilar and therefore we split the block C into two subblocks, $C_1 = \{s \in C \mid s \xrightarrow{a} A\}$ and $C_2 = \{s \in C \mid s \stackrel{a}{\not\longrightarrow} A\}$. More precisely, if $Pre(a, A) = \{s \in S \mid s \xrightarrow{a} A\}$ denotes the set of a-predecessors of A, then $C_1 = C \cap Pre(a, A)$ and $C_2 = C \setminus Pre(a, A)$. Thus, define

$$\mathit{Refine}(C, a, A) \;=\; \{C \cap Pre(a, A), C \setminus Pre(a, A)\} \setminus \{\emptyset\}.$$

and

$$\mathit{Refine}(\mathcal{X}, a, A) \;=\; \bigcup_{C \in \mathcal{X}} Refine(C, a, A)$$

Note that, if the *Refine*–operator is applied, it is not checked whether an action/block pair $\langle a, A\rangle$ is indeed a splitter. This can be done because if $\langle a, A\rangle$ is not a splitter, then we have $\mathit{Refine}(\chi, a, C) = \chi$ and therefore splitting is harmless. It follows now Proposition 5.6.4 that $\mathcal{X} = S/\approx_{sbis}$ iff no more splitters are available. Using an efficient organization of the splitter candidates, this method can be implemented in time $\mathcal{O}(m \log n)$ [PT87] (see also [Fer90]).

Strong bisimulation in probabilistic systems As mentioned by Huynh & Tian [HT92], the splitter/partitioning technique can easily be modified to fully probabilistic systems and to reactive systems. The complexity for these systems is $\mathcal{O}(nk \log n)$, where k is the number of nonzero entries in the transition probability matrix.

In [BEM00], a modification of the partitioning technique is given to compute the strong bisimulation equivalence classes in time $\mathcal{O}(nm(\log m + \log n))$ and space $\mathcal{O}(nm)$. The main idea is no longer to split with respect to action/block pairs but w.r.t pairs $\langle a, M\rangle$, where a is an action and M an equivalence class of $\equiv_\chi$. We will also use this idea in our algorithms.

Strong simulation Henzinger, Henzinger & Kopke [HHK95] present an algorithm for computing the strong simulation preorder in nonprobabilistic systems that has time complexity $\mathcal{O}(nm)$. The schema of this method is sketched in Figure 5.19.

[12] Here, we write $s \xrightarrow{a} A$ iff $s \xrightarrow{a} u$ for some $u \in A$.

Computing the simulation preorder

Input: a finite (probabilistic or nonprobabilistic) system with state space S

Output: the simulation preorder $\preceq_{sim}$ on S

Method:

$R := S \times S$;
While there exists $(s,s') \in R$ with $s \not\sqsubseteq_R s'$ do $R := R \setminus \{(s,s')\}$;
Return R.

Figure 5.19: General schema for computing the simulation preorder

We start with the trivial preorder $R = S \times S$ and then successively remove those pairs (s,s') from R, iff $s \not\sqsubseteq_R s'$, in other words, if s has a transition that cannot be "simulated" by a transition of s'. More precisely, in the nonprobabilistic case, we have

$s \sqsubseteq_R s'$ iff for each transition $s \xrightarrow{a} t$ there is a transition $s' \xrightarrow{a} t'$ with $(t,t') \in R$.

In [Bai96, Bai98], it is shown that the schema of Figure 5.19 is also applicable for the strong simulation preorder $\preceq_{ssim}$ in probabilistic systems. Here, the relation $\sqsubseteq_R$ as a relation on the state space S of a probabilistic system is given by:

$s \sqsubseteq_R s'$ iff for each transition $s \xrightarrow{a} \mu$ there is a transition $s' \xrightarrow{a} \mu'$ with $\mu \sqsubseteq_R \mu'$.

The algorithm proposed in [Bai96, Bai98] requires polynomial time and space, namely, time $\mathcal{O}((mn^6 + m^2n^3)/\log n)$ and space $\mathcal{O}(mn + n^2)$. Another important insight in [Bai96], which we also need in our algorithms, is that the relation $\sqsubseteq_R$ can be decided via a maximal flow in a network.

Strong combined (bi–)simulation As far as the authors know, there are no algorithms for deciding the strong combined simulation and bisimulation from Section 5.4.2. The main problem here is that the relation $\rightarrowtail$ can be infinite, even for finite systems. These, however, could be solved by the methods proposed by Segala [SC01].

5.7.2 Decidability of Weak (Bi–)simulation

The key to the algorithms for deciding weak bisimilarity is that the weak bisimulation equivalence classes in a nonprobabilistic system $(S, \longrightarrow)$ agree with the strong bisimulation equivalence classes in the system $(S, \Longrightarrow)$. In the latter system, the action set is Act_ε and the transition relation $\Longrightarrow \subseteq S \times Act_\varepsilon \times S$ has been given in Section 5.4.3. Thus, for computing the weak bisimulation equivalence classes in nonprobabilistic systems, one first calculates the transitive closure of the τ-labeled transitions from which the weak transition relation $\Longrightarrow$ can be derived. Then, the above mentioned splitter/partitioning technique can be applied to the system $(S, \Longrightarrow)$. The efficiency of this method crucially depends on the transitive closure operation. Using the method of [CW87], one gets an $\mathcal{O}(n^{2.3})$ algorithm for computing the weak transition relation $\Longrightarrow$ and the weak bisimulation equivalence classes. For further details see [BS87].

As for weak bisimulation equivalence, the weak simulation preorder in $(S, \longrightarrow)$ agrees with the strong simulation preorder in the system $(S, \Longrightarrow)$. Hence, the methods of [HHK95] combined with a method for computing the relation $\Longrightarrow$ yields an algorithm for computing the strong simulation preorder in nonprobabilistic systems.

Weak (bi–)simulation in probabilistic systems Weak (bi–)simulation equivalence in a probabilistic system (S, Steps) is the same as strong bisimulation equivalence in the probabilistic system $(S, \widehat{\mathit{Steps}})$. In the latter, we deal with the action set $\mathit{Act}_\varepsilon = \mathit{Act} \cup \{\varepsilon\}$ and $\widehat{\mathit{Steps}}(s) = \{(a, \mu) : s \stackrel{a}{\Longrightarrow} \mu\}$. However, an algorithmic reduction of the weak bisimulation problem to the strong bisimulation problem is not possible since the system $(S, \widehat{\mathit{Steps}})$ might be infinite. As far as the authors know, the question of whether (any type of) weak or branching bisimulation equivalence is decidable for probabilistic systems in our sense is still open. The main problem with weak and branching bisimulation in probabilistic systems is that the number of weak transitions might be infinite.

To the best of our knowledge, there are no published results on the decidability of (any type of) weak (bi–)simulation from Section 5.4. The main problem here is that the number of weak transitions might be infinite. Hence, we cannot reduce the weak (bi–)simulation problem in a system (S, Steps) to the strong (bi–)simulation problem in $(S, \widehat{\mathit{Steps}})$, where $\widehat{\mathit{Steps}}(s) = \{(a, \mu) : s \stackrel{a}{\Longrightarrow} \mu\}$, as we did for weak (bi–)simulation in nonprobabilistic systems. Segala [SC01] has proposed a solution for this, using similar techniques as in [SC01], yielding a polynomial time algorithm for deciding weak, randomized bisimulation. A publication of these results is currently in preparation.

On the other hand, there exist decidability results for several other notions of weak bisimulation for probabilistic systems.

In [BH97], the notions of weak and branching bisimulation are proposed for fully probabilistic systems. It is shown that weak and branching bisimulation equivalence coincide and that the splitter/partitioning technique is applicable to get an $\mathcal{O}(n^3)$ algorithm for computing the weak/branching bisimulation equivalence.

5.7.3 The delay predecessor predicates

The algorithm sketched in Figure 5.20 for deciding strong bisimulation depends on the operator *Refine*, which in its turn depends on a set $\mathit{Pre}(a, A)$ and also the algorithm for deciding simulation depends on this predicate. The decidability algorithms for delay (bi–)simulation depend on a similar predicate, namely $\mathit{Pre}^{str}(a, M)$ for the finitary delay variants and $\mathit{Pre}^{\omega}(a, M)$ for the ω delay variants. This section is concerned with algorithms for those two sets.

Definition 5.7.1 (Strict and ω-predecessors) The sets of strict and ω–predecessors of an action a and a set of distributions M are defined respectively by

$$\mathit{Pre}^{str}(a, M) = \{s \in S \mid s \stackrel{\widehat{a}}{=\!=\!\Rightarrow}{}^{\mathit{fin}} M\}$$
$$\mathit{Pre}^{\omega}(a, M) = \{s \in S \mid s \stackrel{\widehat{a}}{=\!=\!\Rightarrow}{}^{\omega} M\}$$

We present the algorithms for computing the predecessors for actions $a \neq \tau$ because this is technically simpler and we do not need the predecessors of the τ action for deciding the (bi–)simulation relations. However, it is easy to adapt the given algorithms in such a way that these can also handle τ's (with the same complexity).

The strict predecessors

The fixed point characterization of the finitary delay predicates from Section 5.4.4 yields an (inefficient) algorithm to compute the set $Pre^{str}(a, M)$. We start from $P = \{s \mid s \xrightarrow{a} M\}$ and whenever we find $t \xrightarrow{\tau} \nu$ with $supp(\nu) \subseteq P$ and $s \notin P$, we add s to P, until a fixed point has been reached. In that case, $P = Pre^{str}(a, M)$.

We improve this algorithm. We basically simplify the test whether $t \xrightarrow{\tau} \nu$ is a transition satisfying $supp(\nu) \subseteq P$ by using information obtained in previous tests. The idea is as follows. For each distribution ν occurring in a transition $t \xrightarrow{\tau} \nu$, we keep a counter $c(\nu)$ yielding the number of states $s \in supp(\nu)$ for which the condition $s \in P$ has not been verified yet. Thus, initially, $c(\nu)$ equals the number of elements in $supp(\nu)$ and each time we find $s \in supp(\nu) \cap P$, we decrease the counter with 1 until $c(\nu) = 0$. In that case, we know that all t with $t \xrightarrow{\tau} \nu$ may be inserted into P, because all elements in $supp(\nu)$ are so. Obviously, adding t to P may cause other counters to decrease.

For an efficient handling of the counters, we wish to have easy access to all the distributions that have a certain element s in their supports and to all the states that have an outgoing τ transition leading to a given distribution ν. For this, consider the directed graph $G^{str}(S, Steps) = (V, E)$ with

$$N = \bigcup_{s \in S} Steps_\tau(s).$$

the vertex set V is $S \cup N$ where the set $E \subseteq S \times N \cup N \times S$ of edges is defined by

$$E = \left\{(\nu, s) \in N \times S : s \xrightarrow{\tau} \nu\right\} \cup \{(s, \nu) \in S \times N : s \in supp(\nu)\}.$$

Thus, $E(s)$ yields the set of distributions whose support contains the state s and $E(\nu)$ yields the set of states with an outgoing τ–transition leading to ν. The variable N_0 in the algorithm collects all distributions ν with $c(\nu) = 0$. Thus, N_0 can be considered as a wait list of distributions that are waiting to be processed.

Now consider the following algorithm in Figure 5.20. We assume that the transition system $(S, Steps)$ and the set M are represented by adjacency lists.

Example 5.7.2 Consider the system in Figure 5.21. We show how the algorithm computes the set $Pre^{str}(a, M)$, where we assume transitions $u \xrightarrow{a} M$ and $u' \xrightarrow{a} M$. In this example, if we only mention variable if its value has changed. Initially, we have

$$\begin{array}{ll} E(s) = \emptyset & E(\nu_1) = \{s\} \\ E(t) = E(t') = \{\nu_1\} & E(\nu_2) = \{t\} \\ E(u) = \{\nu_2, \nu_3\} & E(\nu_3) = \{t'\} \\ E(u') = \{\nu_1, \nu_2, \nu_3\} & \end{array}$$

$Pre^{str}(a, M) = \emptyset$, $N_0 = \emptyset$, $c(\nu_1) = 3$ and $c(\nu_2) = c(\nu_3) = 2$. Executing the first for–loop with $s = u_1$ yields $Pre^{str}(a, M) = \{u_1\}$ and $c(\nu_2) = c(\nu_3) = 1$ and executing

Input: probabilistic system $(S, Steps)$, $a \in Act$ and a set of distributions $M \subseteq \bigcup_s Steps_a(s)$.

Compute the adjacency lists $E(\cdot)$ of the graph $G^{str}(S, Steps)$;

Put $P := \emptyset$, $N_0 := \emptyset$ 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:

  $P := P \cup \{s\}$;
  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 $s \in E(\nu_0) \setminus P$ do
    $P := P \cup \{s\}$;
    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 P.

Figure 5.20: Algorithm for computing $Pre^{str}(a, M)$

the same loop with $s = u_2$ yields $Pre^{str}(a, M) = \{u_1, u_2\}$ and $c(\nu_2) = c(\nu_3) = 0$ and $N_0 = \{\nu_2, \nu_3\}$. Then, for the second part of the algorithm, execution of the while–loop with $\nu_0 = \nu_2$ yields $Pre^{str}(a, M) = \{u_1, u_2, t\}$, $c(\nu_1) = 1$ and $N_0 = \{\nu_3\}$. Execution of the while–loop with $\nu_0 = \nu_3$ yields $Pre^{str}(a, M) = \{u_1, u_2, t, t'\}$, $c(\nu_1) = 0$ and $N_0 = \{\nu_1\}$. Finally, execution with $\nu_0 = \nu_3$ yields $Pre^{str}(a, M) = \{u_1, u_2, t, t', s\}$ and $N_0 = \emptyset$.

Proposition 5.7.3 *The algorithm in Figure 5.20 computes the set $Pre^{str}(a, M)$.*

Proposition 5.7.4 *The algorithm in Figure 5.20 runs in time and space $\mathcal{O}(nm)$.*

PROOF: The space complexity is clear. The time complexity follows easily from the following observations.

Building the graph $G^{str}(S, Steps)$ takes time $\mathcal{O}(nm)$. The first part (i.e. the first for–loop) of the algorithm clearly has complexity $\mathcal{O}(nm)$ because $|S| = n$ and $|E(s)| \leq m$ for each s.

For the second part of the algorithm, observe that each element ν_0 is inserted in the set N_0 only once: this follows from the fact that ν_0 is added to N_0 when its counter is set to value 0 and the fact that counters only decrease. Hence, for each $\nu \in N$ the while–loop

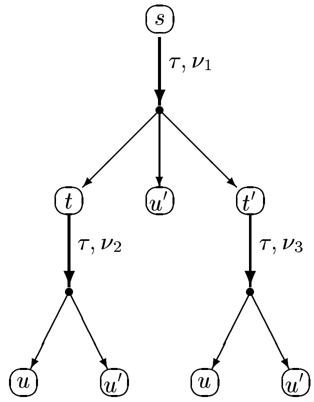


Figure 5.21: The computation of $Pre^{str}(a, M)$

in the algorithm is executed at most once. Then it is not difficult to see that, moreover, the combination of the while–loop and the outermost for–loop is executed at most once for each of the pairs (ν_0, s) such that $s \in E(\nu_0)$. Since there are at most m of those pairs the combination of those loops is executed at most m times. Since innermost for–loop is has complexity $\mathcal{O}(n)$, the entire algorithm runs in time $\mathcal{O}(nm)$. □

The ω-predecessors:

Our algorithm to compute $Pre^{\omega}(a, M)$ is based on the following characterization.

Proposition 5.7.5

$$s \stackrel{a}{=\!=\!\Rightarrow}^{\omega} \mu{\uparrow}_R \iff \mathbf{Q}^A(s, \tau^*, M) = 1, \textit{ for some adversary } A.$$

where $M = \{s' \mid s' \xrightarrow{a} \nu, \nu \in \mu{\uparrow}_R\}$.

Algorithms for computing the set $\{s \mid \mathbf{Q}^A(s, \tau^*, M), A \in DAdv^{\omega}_{\perp}\}$ can be found in [Var85]. They run in time $O(n^2m^2)$ and space $\mathcal{O}(nm)$. This implies the following.

Proposition 5.7.6 *The set* $Pre^{\omega}(a, M)$ *can be computed in time* $O(n^2m^2)$ *and space* $\mathcal{O}(nm)$.

5.7.4 Delay Bisimulation Equivalence

This section presents the algorithms for computing the delay bisimulation equivalences $\approx^{fin}_{del}$ and $\approx^{\omega}_{del}$; the algorithms for deciding the delay simulations preorders $\preceq^{fin}_{del}$ and $\preceq^{\omega}_{del}$ will be given in Section 5.7.5.

The algorithms in this section adapt the splitter/partitioning technique from Figure 5.18, which leads to the basic algorithm given in Figure 5.22. This algorithm is further refined in Figure 5.24. Operations appearing in these algorithms can be computed as given in Figures 5.25 and 5.26. We will first state the main result in this section, which is the polynomial time and space complexity of the algorithms for deciding $\preceq^{fin}_{del}$ and $\preceq^{\omega}_{del}$.

Theorem 5.7.7 *The delay bisimulation equivalence classes of the relation $\approx_{del}^{fin}$ can be computed in time $\mathcal{O}(nm(n+m^2))$ and space $\mathcal{O}(nm)$ and the classes of the relation $\approx_{del}^{\omega}$ can be computed in time $\mathcal{O}(nm(n+m^3))$ and space $\mathcal{O}(nm^2)$.*

In what follows, we briefly write $\approx_{del}$ to denote one of the delay bisimulation equivalences $\approx_{del}^{fin}$ and $\approx_{del}^{\omega}$; we write $\Longrightarrow$ for one of the transitions $\Longrightarrow^{fin}$ and $\Longrightarrow^{\omega}$ and $Pre(a,M)$ for one of the predecessor predicates $Pre^{str}(a,M)$ and $Pre^{\omega}(a,M)$.

As said before, the basic idea behind the computation of the delay equivalences is to adapt the partition refinement technique sketched in Figure 5.18, leading to the basic schema given in Figure 5.22: We start from the trivial partition $\chi = \{S\}$ and whenever we find a block C of χ, states $s, s' \in C$ an equivalence class M of $\equiv_\chi$ such that

$$s \stackrel{\widehat{a}}{\Longrightarrow} M \text{ but } s' \not\stackrel{\widehat{a}}{\Longrightarrow} M,$$

then we split C into two blocks, being $C \cap Pre(a,M)$ and $C \setminus Pre(a,M)$. Thus, as before, we define

$$\begin{aligned} Refine(C,a,M) &= \{C \cap Pre(a,M), C \setminus Pre(a,M)\} \setminus \{\emptyset\} \\ Refine(\chi,a,M) &= \bigcup_{C \in \chi} Refine(C,a,M). \end{aligned}$$

Recall that in the nonprobabilistic case, we dealt with $M \subseteq S$. A problem here is that the relation $\equiv_\chi$ is infinite. This can be solved by using Proposition 5.5.6, which basically states that $Pre(a,M) = Pre(a, M \cap \cup_t Steps_a(t))$. Thus, we can work with pairs $\langle a, M\rangle$ such that $a \in Act_\tau$ and $M \in \cup_t Steps_a(t)/\equiv_\chi$. Then M is a set of distributions that appear as targets of transitions in the system and that are equated by the relation $\equiv_\chi$. The pairs $\langle a, M\rangle$ are also called *step classes* induced by χ. The set of all step classes is called the *step partition* induced by χ and denoted by $\mathcal{M}_\chi$. The partition χ is in this context called a *state partition* and its blocks are called *state classes*.

For the obvious reasons of efficiency, the algorithm does not compute the step partition $\mathcal{M}_\chi$ all over again each time that χ is updated. Rather, we store $\mathcal{M}_\chi$ (in the variable $\mathcal{M}$) and update it accordingly when χ is updated. This leads to a partition refinement technique that works in two phases given in Figure 5.22. First, we update the state partition using the step classes from the latest step partition as splitters. Then, we update the step partition according to the state partition just obtained. Note that, initially, we have $\chi = \{S\}$ and $\mathcal{M}_{new} = \{\langle a, M_a\rangle \mid a \in Act_\tau\}$, where $M_a = \{\mu \mid s \xrightarrow{a} \mu\}$, which is indeed the step partition induced by $\{S\}$.

Updating $\mathcal{M}_{new}$ proceeds as follows. After updating, $\mathcal{M}$ should consist of pairs $\langle a, M\rangle$ such that for all blocks C of χ

$$\mu[C] = \mu'[C] \text{ for all } \langle a, M\rangle \in \mathcal{M}, \text{ and all distributions } \mu, \mu' \in M. \qquad (*)$$

This is achieved by replacing, for every block $C \in \chi$, each step class $\langle b, N\rangle \in \mathcal{M}$ by $\langle b, N_1\rangle$, $\langle b, N_2\rangle, \ldots,$ $\langle b, N_r\rangle$. Here, N_i collects all distributions ν in N that agree on the value $\nu[C]$. More precisely, $\{N_1, \ldots, N_r\}$ are the equivalence classes of the relation $\cong_C$ defined by $\nu_1 \cong_C \nu_2$ iff $\nu_1[C] = \nu_2[C]$. For $N/\cong_C = \{N_1, \ldots, N_r\}$ define

$$\begin{aligned} Split(N,C) &= \{N_1, \ldots, N_r\} \text{ and} \\ Split(\langle b, N\rangle, C) &= \{\langle b, N_1\rangle, \langle b, N_2\rangle, \ldots, \langle b, N_r\rangle\}. \end{aligned}$$

Thus, if we replace every step class $\langle b, N\rangle$ in $\mathcal{M}$ by $\mathit{Split}(\langle b, N\rangle, C)$, then $\mathcal{M}$ meets condition $(*)$ for the block C. That is, $\mathcal{M}$ is replaced by

$$\mathit{Split}(\mathcal{M}, C) \;=\; \bigcup_{\langle b,N\rangle \in \mathcal{M}} \mathit{Split}(\langle b, N\rangle, C).$$

Now, the procedure for updating $\mathcal{M}$ replaces $\mathit{Split}(\mathcal{M}, C)$ for every block C. It is obvious that we only have to consider blocks in $\chi_{new} \setminus \chi_{old}$ since for blocks C in $\chi_{old} \setminus \chi_{new}$, we have $\mathit{Split}(\mathcal{M}, C) = \mathcal{M}$.

$\chi_{new} := \{S\}$;

$\mathcal{M} := \{\langle a, M_a\rangle : a \in Act_\tau\}$ where $M_a = \bigcup_{s\in S}\ \mathit{Steps}_a(s)$;

Repeat

> $\chi_{old} := \chi_{new}$; $\chi_{new} := \emptyset$;
>
> (* Refine state partition w.r.t step partition*)
>
> for all $\langle a, M\rangle \in \mathcal{M}_{old}$ do $\chi_{new} := \mathit{Refine}(\chi_{new}, a, M)$ od;
>
> (* Refine step partition w.r.t state partition*)
>
> for all $C' \in \chi_{new} \setminus \chi_{old}$ do $\mathcal{M} := \mathit{Split}(\mathcal{M}, C')$ od;

until $\chi_{new} = \chi_{old}$;

Return χ_{new}.

Figure 5.22: The two–phased partitioning technique

We now describe how this two–phased partitioning algorithm can be implemented. We first give the skeleton in Figure 5.23; the most detailed algorithm is given in Figure 5.24.

The idea to merge the two refinement phases. So, rather than first performing the refine operator $\mathit{Refine}(C, a, M)$ for several blocks C and then performing the split operator $\mathit{Split}(\mathcal{M}, C')$ for several blocks C' as in Figure 5.22, the algorithm in Figure 5.23 works by performing $\mathit{Refine}(C, a, M)$ for a single block and immediately afterwards performing $\mathit{Split}(\mathcal{M}, C')$ for a block $C' \in \mathit{Refine}(C, a, M)$. Recall that, for a fixed splitter $\langle a, M\rangle$, the set $\mathit{Refine}(C, a, M)$ contains either one or two blocks. Moreover, if $\mathit{Refine}(C, a, M) = \{C_1, C_2\}$, we have that $\mathit{Split}(\langle b, N\rangle, C_1) = \mathit{Split}(\langle b, N\rangle, C_2)$ for all $\langle b, N\rangle \in \mathcal{M}$. This is because M contains step classes $\langle b, N\rangle$ such that $\nu_1[C] = \nu_2[C]$. Therefore, $\nu_1[C'] = \nu_2[C']$ iff $\nu_1[C \setminus C'] = \nu_2[C \setminus C']$. Finally, we remark that by definition $C_1 = C \setminus C_2$ and $C_2 = C \setminus C_1$, which immediately implies the desired result. Therefore, it suffices to perform $\mathit{Split}(\mathcal{M}, C')$ for only one block in C' in $\mathit{Refine}(C, a, M)$.

We further refine the algorithm in Figure 5.23, leading to the algorithm in Figure 5.24. Here, we organize the step partition $\mathcal{M}$ and the state partitions χ_{new} and χ_{old} by queues. We use the usual operations $head(Q)$ which returns first element of a queue Q and $tail(Q)$ removes the first element of Q (provided that Q is not empty) and $Add(Q, \alpha)$ which adds the element α at the end of Q. In the queue representing the step partition we use a special

$\chi_{new} := \{S\}$;

$\mathcal{M} := \{\langle a, M_a\rangle : a \in Act_\tau\}$ where $M_a = \bigcup_{s\in S} \mathit{Steps}_a(s)$;

Repeat

  $\chi_{old} := \chi_{new}; \chi_{new} := \emptyset$;

  for all $\langle a, M\rangle \in \mathcal{M}$

    for all $C \in \chi_{new}$ do

      $\chi_{new} := \chi_{new} \cup \mathit{Refine}(C, a, M)$;

      if $|\mathit{Refine}(C, a, M)| = 2$

      then choose an element C' from $\mathit{Refine}(C, a, M)$;

        $\mathcal{M} := \mathit{Split}(\mathcal{M}, C')$

      fi

    od

  od

until $\chi_{new} = \chi_{old}$;

Return χ_{new}.

Figure 5.23: The two–phased partitioning technique more detailed

symbol @, which separates “old” step classes from “new” ones. More precisely, if we think of the queue $\mathcal{M}$ as a list ordered from the left to the right

$$\langle a, M\rangle \ \ldots \ \langle c, L\rangle \ @ \ \ldots \ \langle b, N\rangle \ldots$$

then all step classes $\langle b, N\rangle$ behind (on the right of) @–symbol in $\mathcal{M}$ have already been chosen as splitter candidates in the second command of the repeat–loop (the assignment $\alpha := \mathit{head}(\mathcal{M})$), whereas the step class $\langle c, L\rangle$ has not yet served as splitter candidate. In particular, if we split $\mathcal{M}$ in step (3.3.2) of the algorithm, then all the step classes are new and hence @ is the last element of $\mathcal{M}$. On the other hand, if @ is the first element of the list then all step classes are “old” and therefore we have for all step classes $\langle b, N\rangle$ behind @ that $\nu_1[C] = \nu_2[C]$ for all $\nu_1, \nu_2 \in N$ and all blocks C in the current partition χ. Thus, when the algorithm terminates (and returns $\chi_{new} = \chi_{old}$) then in the last execution of the repeat–loop the step partition $\mathcal{M}$ has not changed and we have: if $\langle b, N\rangle$ is step class of $\mathcal{M}$ and $\nu_1, \nu_2 \in N$ then $\nu_1 \equiv_{R_\chi} \nu_2$.

In the initialization of $\mathcal{M}$, we assume a procedure as shown in Figure 5.25. Here, the assignment $\mathcal{M} := \emptyset$ stands for the operation which creates an empty queue. A more precise formulation of step (3.3.2) is given in Figure 5.26.

The splitting operator for the step classes: A naive procedure to calculate the operation $\mathit{Split}(N, C')$ is to compute the value $\nu[C']$ for each $\nu \in N$ and then to collect those distributions that agree on $\nu[C']$ in the same block.

Computing the delay bisimulation equivalence classes

Input: a probabilistic system $(S, Steps)$

Output: the set $S/\approx_{dbis}$ of delay bisimulation equivalence classes

Method:

$\chi_{new} := \{S\}$;
Create a queue $\mathcal{M}$ whose elements are the pairs

$$\langle a, M_a\rangle,\ a \in Act_\tau, \text{ where } M_a = \bigcup_{s\in S} Steps_a(s).$$

Insert the symbol @ at the end of $\mathcal{M}$.
Repeat
 $\chi_{old} := \chi_{new}$;
 $\alpha := head(\mathcal{M})$; $\mathcal{M} := tail(\mathcal{M})$; $Add(\mathcal{M}, \alpha)$;
 If $\alpha = \langle a, M\rangle$ is a step class then
 (1) $P := Pre(a, M)$;
 (2) $\chi_{new} := \emptyset$;
 (3) for all blocks $C \in \chi_{old}$ do
 (3.1) If $a = \tau$ and $M \cap \{\nu \in Distr(S) : supp(\nu) \subseteq C\} \neq \emptyset$
 then $C' := C$
 else $C' := C \cap P$;
 (3.2) If $C' = \emptyset$ or $C' = C$ then $\chi_{new} := \chi_{new} \cup \{C\}$;
 (3.3) If $C' \neq \emptyset$ and $C' \neq C$ then
 (3.3.1) $\chi_{new} := \chi_{new} \cup \{C', C \setminus C'\}$;
 (3.3.2) Replace any $\langle b, N\rangle$ in $\mathcal{M}$ by the step classes $Split(\langle b, N\rangle, C')$ and move @ to the end of $\mathcal{M}$;
 od
until $\chi_{new} = \chi_{old}$ and $\alpha = @$;
Return χ.

Figure 5.24: Computation of $\approx_{dbis}$

$\mathcal{M} := \emptyset$;

For all $a \in Act_\tau$ do $Add(\mathcal{M}, \langle a, M_a \rangle)$;

$Add(\mathcal{M}, @)$.

Figure 5.25: Initialization of $\mathcal{M}$

$\mathcal{M}_{new} := \emptyset$;

While $\mathcal{M} \neq \emptyset$ do

  $\beta := head(\mathcal{M})$;
  $\mathcal{M} := tail(\mathcal{M})$;
  If $\beta = \langle b, N \rangle$ is a step class then
    Compute the set $Split(N, C') = \{N'_1, \ldots, N'_r\}$;
    For all $i = 1, \ldots, r$ do $Add(\mathcal{M}_{new}, \langle b, N'_i \rangle)$;
  $Add(\mathcal{M}_{new}, @)$;

$\mathcal{M} := \mathcal{M}_{new}$.

Figure 5.26: Step (3.3.2): Splitting of the step partition

This procedure can be optimized using ordered trees, which makes the comparison of the distributions easier. We propose the following procedure for computing $Split(N, C')$. We construct a balanced ordered binary search tree (e.g. an AVL tree or Red/Black tree; see e.g. [CLR90]) as follows. The vertices v in this tree are the pairs $(v.prob, v.distr)$, where $v.prob \in [0, 1]$, $v.distr \subseteq N$ such that $v.distr \neq \emptyset$ and

$$v.distr = \{\nu \in N \mid \nu[C'] = v.prob\}$$

Note that there are no two different nodes v, w such that $v.prob = w.prob$. The nodes are ordered according to the value of $v.prob$, that is $v < w$ iff $w.prob < v.prob$. Let $N = \{\nu_1, \ldots, \nu_l\}$. We generate the tree by successively inserting the values $\nu_i[C']$, $i = 1, \ldots, l$. More precisely,

- We start with a tree consisting of its root v_0 where $v_0.distr = \{\nu_1\}$ and $v_0.prob = \nu_1[C']$.
- For $i = 2, \ldots, l$, we insert the value $\nu_i[B]$ into the tree (possibly creating a new node and performing the necessary rebalance operations) and the distribution ν_i into the set $v.distr$ where v is the node with $v.prob = \nu_i[C']$.

Now, the classes in $split(N, C')$ are exactly the sets $v.distr$, for v a node in the tree.

For fixed $\nu \in N$, the computation of $\nu[C']$ takes $\mathcal{O}(n)$ time. Thus, we get the values $\nu_i[C']$, $i = 1, \ldots, l$, in time $\mathcal{O}(ln) = \mathcal{O}(|N|n)$. The construction of the tree takes $\mathcal{O}(|N| \log |N|))$ time. Thus, in step (3.3.2), any step class $\langle b, N \rangle$ causes the cost $\mathcal{O}(|N|n + |N| \log |N|))$. Ranging over all step classes $\langle b, N \rangle$ in $\mathcal{M}$, we obtain the time complexity $\mathcal{O}(mn + m \log m))$ for step (3.3.2).

It is now obvious how to compute $Split(\langle b, M \rangle, C)$ in the same complexity.

Complexity: We now turn to the proof of Theorem 5.7.7 and compute the worst case complexity of the algorithm of Figure 5.24. The space complexity $\mathcal{O}(nm)$ and $\mathcal{O}(nm^2)$ is clear. We now discuss the time complexity. Therefore, we first observe that, for any set sequence $\chi_1, \chi_2, \ldots$ of partitions of some set S such that χ_{1+1} is finer than χ_i, we have that the number of blocks $C \in \chi_1 \cup \chi_2 \cup \ldots$ is at most $2 \cdot |S| - 1$. This can be seen as follows. We can see the sequence $\chi_1, \chi_2, \ldots$ as a tree, whose nodes are the blocks in the partitions and where we have and edge between $C \in \chi_k$ and $C' \in \chi_{k+1}$ iff $C \supseteq C'$. Now an easy combinatoric argument shows that the number of nodes in the tree is smaller than $2 \cdot |S| - 1$.

For this the finitary variants, we observe that the repeat–loop in Figure 5.24 is executed at most m^2 times. Thus, the computation of the predecessor predicates $Pre(a, M)$ in step (1) of the repeat–loop requires $\mathcal{O}(nm^3)$ time when we range over all iterations, because Section 5.7.3, we saw that the sets $Pre(a, M)$ can be obtained in time and space $\mathcal{O}(nm)$.

Step (3.3.2) is executed at most $\mathcal{O}(n)$ times. Any execution of step (3.3.2) calls the splitting procedure explained above which runs in time $\mathcal{O}(mn + m \log m)$. Thus, when we sum up over all executions of step (3.3.2) then we get the total cost $\mathcal{O}(n^2 m + nm \log m)$. This yields the time complexity

$$\mathcal{O}(nm^3 + n^2 m + nm \log m) \ = \ \mathcal{O}(nm(n + m^2))$$

for the whole algorithm. A similar argument yields the time complexity $\mathcal{O}(nm^2(n + m^2))$ for the ω–variants. For the weak variants $\approx_{dwbis}$ and $\approx^{\omega}_{dwbis}$, multiple computations of $Pre(a, M)$ (i.e. the set $Pre^{str}(a, M)$ or $Pre^{\omega}(a, M)$) in step (1) can be avoided since the weak transitions does not depend on the current state partition χ. For this, we shift the computation of $Pre(a, M)$ into step (3.3.2); i.e., whenever a new step class $\langle b, N' \rangle$ is created in step (3.3.2) then we may apply the corresponding $\mathcal{O}(nm)$ predecessor operator. As step (3.3.2) is executed at most $\mathcal{O}(n)$ times we get the time complexity $\mathcal{O}(n^2 m)$ for all predecessor operations together. For any of the step classes $\langle b, N \rangle$ in $\mathcal{M}$, we have to keep a record of the sets $Pre(b, N)$; e.g. by adding the component $Pre(b, N)$ to the step classes $\langle b, N \rangle$ in $\mathcal{M}$. Since at any moment $\mathcal{M}$ contains at most m step classes this additional component for the step classes does not change the space complexity. We get:

Theorem 5.7.8 *The delay weak finitary bisimulation equivalence classes can be computed in time $\mathcal{O}(nm(n + m))$ and space $\mathcal{O}(nm)$; the delay weak ω–bisimulation in time $\mathcal{O}(nm(n + m^2))$ and space $\mathcal{O}(nm^2)$.*

Remark 5.7.9 These complexity results holds weak and branching delay bisimulations. For the weak variants $\approx_{dwbis}$ and $\approx^{\omega}_{dwbis}$ our method can be modified to get an algorithm with lower time but higher space complexities. See Theorem 5.7.8.

5.7.5 The Delay Simulation Preorder

This section presents polynomial time and space algorithms that compute the for the delay simulation preorders $\preceq_{del}^{fin}$ and $\preceq_{del}^{\omega}$. These algorithms are based on the general schema of Figure 5.19. We first formulate the main result of this section.

Theorem 5.7.10 *The delay finitary simulation preorder $\preceq_{del}^{fin}$ can be computed in time complexity $\mathcal{O}(n^5 m^2/\log n)$ and space complexity $\mathcal{O}(nm + n^2 + m^2)$. The delay ω–simulation preorder $\preceq_{del}^{\omega}$ can be computed in time $\mathcal{O}(n^5 m^3/\log n + n^3 m^3)$ and space complexity $\mathcal{O}(nm^2 + n^2 + m^2)$.*

```
R := S × S;
Repeat
    R_new := R;
    For all (s, s') ∈ R do
        For all (a, μ) ∈ Steps(s) do
            sim := true;
            If (a ≠ τ ∨ μ ⋢_R μ¹_s') ∧ s' ∉ Pre(a, μ↑_R) then sim := false;
        If ¬sim then R_new := R_new \ {(s, s')};
until R = R_new;
Return R.
```

Figure 5.27: Basic idea for computing the delay simulation preorder

In the sequel, $\preceq_{del}$ denotes one of the delay simulation preorders $\preceq_{del}^{fin}$ or $\preceq_{del}^{\omega}$. As before, we briefly write $\Longrightarrow$ to denote one of the transitions $\Longrightarrow^{fin}$ and $\Longrightarrow^{\omega}$ and $Pre(a, M)$ for one of the predecessor predicates $Pre^{str}(a, M)$ and $Pre^{\omega}(a, M)$.

The remainder of this section is concerned with algorithms for computing the relation $\preceq_{del}$ with the time and space complexity given above. The basic idea of the probabilistic variant of the general schema in Figure 5.19 is sketched in Figure‘5.27.

A basic operation in the algorithm is the predicate $\sqsubseteq_R \subseteq S \times S$ defined by

$$s \sqsubseteq_R s' \text{ iff } s \xrightarrow{a} \mu \text{ implies } s' \stackrel{\widehat{a}}{\Longrightarrow} \mu\uparrow_R.$$

Then Proposition 5.6.3 implies that the relation $\preceq_{del}$ is the largest relation R such that $(s, s') \in R \implies s \sqsubseteq s'$. Hence, we can compute $\preceq_{del}$ as follows. We start with the trivial preorder $R = S \times S$. Then, we test whether $s \sqsubseteq_R s'$ for all pairs $(s, s') \in R$. If $s \sqsubseteq_R s'$ then we keep the pair (s, s') in R_{new}, otherwise, we remove it from R_{new}.

The test whether $s \sqsubseteq_R s'$ is simply done by checking for all outgoing transitions $s \xrightarrow{a} \mu$ of s whether there exists a matching delay transition $s' \stackrel{\widehat{a}}{\Longrightarrow} \mu\uparrow_R$; in other words, whether $s' \in Pre(a, \mu\uparrow_R)$. We treat the case $a = \tau$ and $\mu \sqsubseteq_R \{s' \mapsto 1\}$, in which case we do have a matching delay transition from s' separately, because this condition can be checked easily. [13] The problem with the set $Pre(a, \mu\uparrow_R)$ (for $a \neq \tau$ and $\mu \not\sqsubseteq_R \mu^1_{s'}$)is that the set $\mu\uparrow_R$ is infinite. As before, we use the fact that $Pre(a, \mu\uparrow_R) = Pre(a, \mu\uparrow_R \cap \cup_s Steps_a(s))$ to obtain a finite representation. The computation of $Pre(a, M)$ has been explained in Section 5.7.3.

The computation of $\mu\uparrow_R \cap \cup_s Steps_a(s)$ is done as follows. Let $Steps[a]$ denote the set $\cup_s Steps_a(s)$. Whenever we have a decreasing sequence $R_0 \supseteq R_1 \supseteq R_2 \supseteq \ldots$ of binary relations on the state space S then $\mu \not\sqsubseteq_{R_i} \mu'$ implies $\mu \not\sqsubseteq_{R_j} \mu'$ for all $j \geq i$. Thus, if we have detected that $\mu \not\sqsubseteq_R \mu'$ for a certain distribution $\mu' \in Steps[a]$ and the current relation R then we also know that $\mu \not\sqsubseteq_R \mu'$ in all further iterations. This observation motivates the idea to keep a record of the sets

$$Sim[a, \mu] \ = \ \{\mu' \in Steps[a] : \mu \sqsubseteq_R \mu'\}.$$

Then, to check whether $s' \in Pre(a, \mu\uparrow_R \cap [\xrightarrow{a}])$, we calculate the set $Pre(a, Sim[a, \mu])$ with the $\mathcal{O}(nm)$–method described in Section 5.7.4 and get:

$$s \sqsubseteq_R s' \text{ iff } s' \in Pre(s, Sim[a, \mu]) \text{ for all } s \xrightarrow{a} \mu \text{ where } a \neq \tau \text{ or } \mu \not\sqsubseteq_R \mu^1_{s'}.$$

Moreover, instead of the repeat–loop (where we distinguish between R_{new} and the relation R obtained by the previous iteration) we may do all modifications (that is, removements of certain pairs (s, s')) on the current relation R. This requires the recomputation of $Sim[a, \mu]$ whenever a pair (s, s') is removed from R; e.g. with the method sketched in Figure 5.28. Here, we make use of the observation that

$$\text{if } \mu \sqsubseteq_R \mu' \text{ and } s \notin supp(\mu) \text{ or } s' \notin supp(\mu') \text{ then } \mu \sqsubseteq_{R'} \mu' \text{ where } R' = R \setminus \{(s, s')\}.$$

For the test whether $\mu \sqsubseteq_R \mu'$ we may apply the network–based method mentioned in [BEM00].

For all $a \in Act_\tau$ and $\mu \in Steps[a]$ where $s \in supp(\mu)$ do

  $Sim_{new}[a, \mu] := \emptyset$;

  For all $\mu' \in Sim[a, \mu]$ where $s' \in supp(\mu')$ do

    If $\mu \sqsubseteq_R \mu'$ then $Sim_{new}[a, \mu] := Sim_{new}[a, \mu] \cup \{\mu'\}$;

  $Sim[a, \mu] := Sim_{new}[a, \mu]$;

Figure 5.28: Recomputation of $Sim[a, \mu]$ after the removement of (s, s') from R

These ideas lead to the algorithm shown in Figure 5.30. We organize R as a queue where the ordering in the initial queue is arbitrary.[14] Moreover, we use a special variable *last* which

[13] If one adds the distributions μ^1_s to $Steps_\tau(s)$ at the beginning of the algorithm then the τ–labeled transitions $s \xrightarrow{\tau} \mu$ with $\mu \sqsubseteq_R \mu^1_{s'}$ do not require a special treatment. These additional transitions does not change the simulation preorder and do not affect the complexity of our method. However, the test whether $\mu \sqsubseteq_R \mu^1_{s'}$ is possible without the (quite complicate) network–based methods. We just have to check whether $(t, s') \in R$ for all $t \in supp(t)$. This can be done in time $\mathcal{O}(n)$ while the computation of the maximum flow in $\mathcal{N}(\mu, \mu^1_{s'}, R)$ takes $\mathcal{O}(n^3/\log n)$ time.

[14] Of course, in the initialization of R the "trivial" pairs (s, s) can be ignored (since we always have $s \preceq_{dsim} s$). However, including the pairs (s, s) does not affect the asymptotic time or space complexity.

is either undefined (i.e. $last = \perp$) or an element (t_0, t_0') of R. Initially, $last$ is the last element of R. In the case where $last = (t_0, t_0')$ then none pair (t, t') in R that have been investigated after the last investigation of (t_0, t_0') was removed from R. (Here, by an *investigation* of an element (s, s') of R, we mean the execution of the steps (2)–(4) where (s, s') is the chosen element in step (2).) I.e. we have $t \sqsubseteq_R t'$ for all pairs (t, t') that are behind (on the right of) of (t_0, t_0') in R (see Figure 5.29). In step (3), we test whether $s \sqsubseteq_R s'$. The meaning of the

$$R: \quad \underbrace{(s,s')}_{head(R)} \quad \ldots \quad \underbrace{\underbrace{(t_0,t_0')}_{=last} \quad \ldots (t,t') \ldots}_{t \sqsubseteq_R t'}$$

Figure 5.29: R organized as a queue

boolean variable sim in step (4) is as before, i.e. it records whether or not $s \sqsubseteq_R s'$. Thus, when we get $s \sqsubseteq_R s'$ for the current pair (s, s') (i.e. the boolean variable sim is true in line (4)) then we reinsert (s, s') into R and distinguish two cases:

1. If $last = (s, s')$ then $t \sqsubseteq_R t'$ for simulation preorder $\preceq_{dsim}$ and our algorithms terminates (see the "then"–branch of step (4.2)).
2. If $(s, s') \neq last$ then we are in the situation of Figure 5.29 where $last = (t_0, t_0') \neq (s, s')$. We just move (s, s') to the end of R (via the command $Add(R, (s, s'))$ in step (4.1)) and investigate the next pair in R (see step (4.2.2) where we jump back to step (2)) but still deal with $last = (t_0, t_0')$.

If $s \not\sqsubseteq_R s'$ then we put $last = \perp$ (in step (4.3)) and "remove"(i.e. do not reinsert) the current pair (s, s') from R. In this case, the induced relations $\sqsubseteq_R$ on $Distr(S)$ and S change and the condition $t \sqsubseteq_R t'$ might be violated for some pair in R. Thus, we recompute $Sim[a, \mu]$ (in step (4.4)) and "wait" for the next pair (t_0, t_0') that will be not removed from R (see step (4.2.1)).

Whenever we start an investigation of the first pair (s, s') of R in step (2) then the set $Sim[a, \mu]$ contains exactly those distributions $\mu' \in Steps[a]$ where $\mu \sqsubseteq_R \mu'$. I.e. $Sim[a, \mu] = \mu \uparrow_R \cap Steps[a]$. This can be seen as follows. Initially, we deal with $Sim[a, \mu] = Steps[a]$ and $R = S \times S$. Thus, $\mu \sqsubseteq_R \mu'$ for all distributions μ, μ' on S. Whenever we remove a pair (s, s') from R (more precisely, we take (s, s') in step (2), obtain $s \not\sqsubseteq_R s'$ in step (3) but do not add (s, s') to the end of R) then we recalculate the sets $Sim[a, \mu]$ in step (4.4) (where we call a procedure as sketched in Figure 5.28).

Complexity: We first consider the space complexity. $\mathcal{O}(nm)$ space is needed for the representation of the probabilistic system itself. The representation of R and the networks $\mathcal{N}(\mu, \mu', R)$ require $\mathcal{O}(n^2)$ space while we need $\mathcal{O}(m^2)$ space for the sets $Sim[a, \mu]$. This yields the space complexity $\mathcal{O}(nm + n^2 + m^2)$ for the whole algorithm.

Next we discuss the time complexity. For the finitary case, we observe that steps (2)–(4) are executed at most n^2 times. Each execution of step (3) takes $\mathcal{O}(nm^2)$ time.[15] Ranging

[15] Here, we use the observation that the condition $\mu \not\sqsubseteq_R \mu^1_{s'}$ is equivalent to $(t, s') \notin R$ for some $t \in supp(\mu)$ which can be checked in linear time. To test whether $s' \in Pre(a, Sim[a, \mu])$ requires $\mathcal{O}(nm)$ time (see Proposition 5.7.6). Thus, given (s, s'), any transition $s \xrightarrow{a} \mu$ causes the cost $\mathcal{O}(nm)$. Ranging over all outgoing transitions from s, we get the time complexity $\mathcal{O}(nm^2)$ for step (3).

over all executions, step (3) requires $\mathcal{O}(n^3 m^2)$ time. In step (4), only the time complexity for the recomputation of $Sim[a,\mu]$ in step (4.4) is of interest. In each execution of step (4.4), we check whether $\mu \sqsubseteq_R \mu'$ for at most $\mathcal{O}(m^2)$ pairs (μ,μ'). Using the maximum flow algorithm proposed by [CHM90], we get the time complexity $\mathcal{O}(m^2 n^3/\log n)$ for each execution of step (4.4). Summing up over all pairs (s,s') that are removed from R we get the total cost $\mathcal{O}(m^2 n^5/\log n)$ for step (4.4). This yields the worst case time complexity $\mathcal{O}(n^5 m^2/\log n)$ for the whole computation of finitary delay bisimulation. For the ω variants we obtain $\mathcal{O}(n^5 m^2/\log n + nm^2)$.

Computing the delay simulation preorder

Input: a probabilistic system $(S, Steps)$

Output: the delay simulation preorder $\preceq_{dsim}$

Method:

(1) [Initialization]

(1.1) Create a queue R whose elements are the pairs $(s, s') \in S \times S$;

(1.2) Let $last$ be the last element of R;

(1.3) For any $a \in Act_\tau$ let $Steps[a] := \bigcup_{t \in S} Steps_a(t)$;

(1.4) For any $a \in Act_\tau$ and $\mu \in Steps[a]$ let $Sim[a, \mu] := Steps[a]$;

(2) $(s, s') := head(R)$;

$R := tail(R)$;

(3) $sim := true$; For all $(a, \mu) \in Steps(s)$ do

If $(a \neq \tau) \vee (\mu \not\sqsubseteq_R \mu^1_{s'}) \ \wedge \ s' \notin Pre(a, Sim[a, \mu])$ then $sim := false$;

(4) If sim then

(4.1) $Add(R, (s, s'))$;

(4.2) If $last = (s, s')$

then go to (5)

else $(*\ s \sqsubseteq_R s'$ and $last \neq (s, s')\ *)$

(4.2.1) If $last = \bot$ then $last := (s, s')$ fi;

(4.2.2) Go to (2)

fi

else $(*\ \neg sim$, i.e. $s \not\sqsubseteq_R s'\ *)$

(4.3) $last := \bot$

(4.4) For all $a \in Act_\tau$ and $\mu \in Steps[a]$ put

$$Sim[a, \mu] := \{\mu' \in Sim[a, \mu] : \mu \sqsubseteq_R \mu'\};$$

(4.5) go to (2);

fi

(5) Return R.

Figure 5.30: Algorithm for computing the delay simulation preorder

5.8 Conclusions

We have studied various notions of simulation and bisimulation for probabilistic systems. First we have recalled several variants of strong and weak (bi-)simulation, most of which are known from literature, and studied their properties. An important conclusion is that the weak bisimulations based on deterministic adversaries are not useful in verification, since they fail the desired soundness and compositionality results. The randomized variants, however, are sound for trace distribution inclusion and compositional w.r.t. parallel composition.

Moreover, we introduced two novel notions of (bi-)simulation equivalence that abstract from internal computations. We presented polynomial-time algorithms that compute the quotient spaces and proved the desired soundness and compositionality results. Thus, our notion of bisimulation equivalence could yield an alternative to the weak bisimulations of [SL95]. Although the equivalences à la [SL95] are the natural probabilistic counterpart to weak bisimulation equivalence in LTSs [Mil80, Par81, GW89], their definitions are rather complicated 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 from 5.7.1 can easily be rewritten as terms of the relational mu-calculus.

Of course, a lot of work still has to be done: the algorithms have to be implemented and case studies have to be carried out in order to estimate their usefulness. Furthermore, we believe that the complexity bounds we derived are rather coarse and serious gains can be obtained by adapting the algorithms.

Furthermore, norm functions and the derived notions of bisimulations can also be defined for infinite systems.[16] We believe that, as in the nonprobabilistic case, in many applications, it is quite simple to "guess" a norm function and then to check (e.g. by hand or by theorem proving) whether it fulfills the necessary conditions.

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

[16] 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.