CHAPTER 2
Probabilistic Automata

> Al–hoewel in spelen daer alleen het geval plaats heeft, de uytkomsten onseecker zijn, so heeft nochtans de kansse die yemand om te winnen of te verliesen, haere seeckere bepaling [...] maer hoe veel minder kans hij heeft om te winnen of te verliesen, dat is in selver seecker, en werde door reeckeningh uyt–gevonden."
>
> Christiaan Huygens, *Van Rekeningh in Spelen van Geluck* [Huy50]

Abstract This chapter provides an informal introduction to the probabilistic automaton (PA) model. This framework has been developed by Segala [Seg95b] and serves as a basis for subsequent chapters. We describe how distributed systems with discrete probabilities can be modeled and analyzed by means of PAs. We also point out how the basic concepts for the analysis of nonprobabilistic systems can be extended to probabilistic systems. In particular, we treat the parallel composition operator on PAs, the semantics of a PA as a set of trace distributions, an extension of the PA model with time and two simulation relations for PAs. Finally, we give an overview of various other state based models that are used for the analysis of probabilistic systems.

2.1 Introduction

Probabilistic Automata (PAs) constitute a mathematical framework for the specification and analysis of probabilistic systems. They have been developed by Segala [Seg95b, SL95, Seg95a] for the purpose of modeling and analyzing asynchronous, concurrent systems with discrete probabilistic choice in a formal and precise way. Examples of such systems are randomized, distributed algorithms, such as the randomized dining philosophers [LR81]; probabilistic communication protocols, such as the IEEE1394 Root Contention protocol [IEE96] and the Binary Exponential Back Off protocol (see [Tan81]); and fault tolerant systems, such as unreliable communication channels.

PAs are based on state transition systems and make a clear distinction between probabilistic and nondeterministic choice. We will extensively treat the differences and similarities between both types of choices. In fact, PAs can be considered as a combination of discrete time Markov chains and (nondeterministic) state machines. Moreover, PAs subsume the Markov decision processes. The PA framework does not provide any syntax. However, several syntaxes could be defined (and in fact have been defined) on top of it to facilitate the modeling of a system as a PA.

A PA has a well–defined semantics as a set of trace distributions. The parallel composition operator $\|$ allows one to construct a PA from several component PAs running in parallel and thus keeping system models modular. Properties of probabilistic systems that can be established formally using PAs include correctness and performance issues, such as: Is the

probability that an error occurs small enough (e.g. $< 10^{-9}$)? Is the average time between two failures large enough?

The aim of this chapter is to explain the basic ideas behind PAs and their behavior in an informal way. In our explanation, we stress the differences and similarities with nonprobabilistic automata.

Organization of the chapter

This chapter is organized as follows. Section 2.2 introduces the probabilistic automaton model, Section 2.3 treats the behavior (or semantics) of the model and Section 2.4 is concerned with simulation relations for PAs. In Section 2.6, we give several other models dealing with probabilistic choice. Finally, Section 2.7 presents a summary of the chapter.

2.2 The Probabilistic Automaton Model

Basically, a probabilistic automaton is just an ordinary automaton (also called *labeled transition system* or *state machine*) with the only difference that the target of a transition is a probabilistic choice over several next states. Before going into the details of this, we briefly explain the notion of a nonprobabilistic automaton, abbreviated NA.

2.2.1 Nonprobabilistic Automata

As we have already pointed out in Chapter 1, an NA is a structure consisting of *states* and *transitions*; the latter are also called *steps*. The states of an NA represent the states in the system that is modeled. One or more states are designated as *start states*, representing the initial configuration of the system. The transitions in the NA represent changes of the system states and are labeled by actions. Thus, the transition $s \xrightarrow{a} s'$ represents that, being in state s, the system can move via an a–action to another state s'. The action labels are partitioned into *internal* and *external* actions. The former represent internal computation steps of the automaton[1] and are not visible to the automaton's environment. We often use the symbol τ to denote an internal action. The external actions are visible for the automaton's environment and are used for interaction with it. This will be explained in more detail in Section 2.2.4.

Example 2.2.1 Consider the NA in Figure 2.1. It models a 1–place buffer used as a communication channel for the transmission of bits between two processes. The states ε, 0 and 1 represent respectively the channel being empty, the channel containing the bit 0 and the channel containing a 1. Initially, the channel is empty (denoted by the double circle in the picture). The transitions $\varepsilon \xrightarrow{snd(i)} i$ represent the sender process (not depicted here) sending the bit i to the channel. Similarly, the transitions $i \xrightarrow{rec(i)} \varepsilon$ represent the delivery of the bit at the receiver process. Notice that the transition labeled $snd(i)$ represents the sending of a bit by the sender, which is the receipt of a bit by the communication channel. Similarly, the receipt of a bit at the receiving process corresponds to the sending of a bit by the channel, which is modeled by the $rec(i)$–transitions.

It is natural that the actions $snd(0)$, $rec(0)$, $snd(1)$ and $rec(1)$ in this NA are external, since these are used for communication with the environment.

[1] Note that at this point, the notation differs from Chapter 1, where internal transitions were unlabeled. In the current model, each transition has a label.

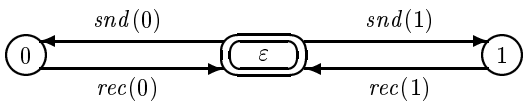


Figure 2.1: A channel automaton

Thus, the notion of an NA can be formalized as follows.

Definition 2.2.2 An NA $\mathcal{A}$ consists of four components:

1. A set $S_{\mathcal{A}}$ of *states*,
2. a nonempty set $S^0_{\mathcal{A}} \subseteq S_{\mathcal{A}}$ of *start states*,
3. An *action signature* $sig_{\mathcal{A}} = (V_{\mathcal{A}}, I_{\mathcal{A}})$, consisting of *external (visible)* and *internal actions* respectively. We require $V_{\mathcal{A}}$ and $I_{\mathcal{A}}$ to be disjoint and define the set of *actions* as $\mathrm{Act}_{\mathcal{A}} = V_{\mathcal{A}} \cup I_{\mathcal{A}}$.
4. a *transition relation* $\Delta_{\mathcal{A}} \subseteq S_{\mathcal{A}} \times \mathrm{Act}_{\mathcal{A}} \times S_{\mathcal{A}}$.

We write $s \xrightarrow{a}_{\mathcal{A}} s'$, or $s \xrightarrow{a} s'$ if $\mathcal{A}$ is clear from the context, for $(s, a, s') \in \Delta_{\mathcal{A}}$. Moreover, we say that the action a is *enabled* in s, if s has an outgoing transition labeled by a.

Several minor variations of this definition exist. Other definitions require, for instance, a unique start state rather than a set of start states or allow only a single internal action, rather than a set of these. In the I/O automaton model [LT89], external actions are divided into *input* and *output* actions. Input actions, not being under the control of the NA, are required to be enabled in any state. The basic concepts are similar for all the various definitions.

Nondeterminism

Nondeterministic choices can be specified in an automaton by having several transitions leaving from the same state. Nondeterminism is used when we wish to incorporate several potential system behaviors in a model. Hoare [Hoa85] phrases it as follows:

> There is nothing mysterious about nondeterminism, it arises from the deliberated decision to ignore the factors which influence the selection.

Nondeterministic choices are often divided into *external* and *internal* nondeterministic choices. External nondeterministic choices are choices that can be influenced by the environment. Since interaction with the environment is performed via external actions, external nondeterminism is incorporated by having several transitions with different labels leaving from the same state.

Internal nondeterministic choices are choices that are made by the automaton itself, independent of the environment. These are modeled by internal actions or by having several

transitions with the same labels leaving from the same state. In the literature, the word nondeterminism sometimes refers to what we call internal nondeterminism.

As pointed out by [Seg95b, Alf97], nondeterminism is essential for the modeling of the following phenomena.

Scheduling freedom When a system consists of several components running in parallel, we often do not want to make any assumptions on the relative speeds of the components, because we want the application to work no matter what these relative speeds are. Therefore, nondeterminism is essential to define the parallel composition operator (see Definition 2.2.14), where we model the choice of which automaton in the system takes the next step as an (internal or external) nondeterministic choice.

Implementation freedom Automata are often used to represent a specification. Good software engineering practice requires the specification to describes *what* the system should do, not *how* it should be implemented. Therefore, a specification usually leaves room for several alternative implementations. Since it does not matter for the correct functioning of the system which of the alternatives is implemented, such choices are also represented by (internal or external) nondeterminism.

External environment An automaton interacts with its environment via its external actions. When modeling a system, we do not wish to stipulate how the environment will behave. Therefore the possible interactions with the environment are modeled by (external) nondeterministic choices.

Incomplete information Sometimes it is not possible to obtain exact information about the system to be modeled. For instance, one might not know the exact duration of or — in probabilistic systems — the exact probability of an event, but only a lower and upper bound. In that case, one can incorporate all possible values by a nondeterministic choice. This is appropriate since we consider a system to be correct if it behaves as desired no matter how the nondeterministic choices are resolved.

Example 2.2.3 In the state ε, the channel NA in Figure 2.1 contains an external nondeterministic choice between the actions $snd(0)$ and $snd(1)$. This models a choice to be resolved by the environment (in this case a sender process) which decides which bit to sent.

Nondeterministic choices modeling implementation freedom, scheduling freedom, and incomplete information are given in Examples 2.2.9 and 2.2.15.

2.2.2 Probabilistic Automata

As said before, the only difference between a nonprobabilistic and a probabilistic automaton is that the target of a transition in the latter is no longer a single state, but is a probabilistic choice over several next states. For instance, a transition in a PA may reach one state with probability $\frac{1}{2}$ and another one with probability $\frac{1}{2}$ too. In this way, we can represent a coin flip and a dice roll, see Figure 2.2.

Thus, a transition in a PA relates a state and an action to a *probability distribution* over the set of states. A probability distribution over a set X is a function μ that assigns a probability in $[0, 1]$ to each element of X, such that the sum of the probabilities of all elements is 1. The reader is referred to Chapter 3 for a formal definition. Let $\mathrm{Distr}(X)$ denote the set of

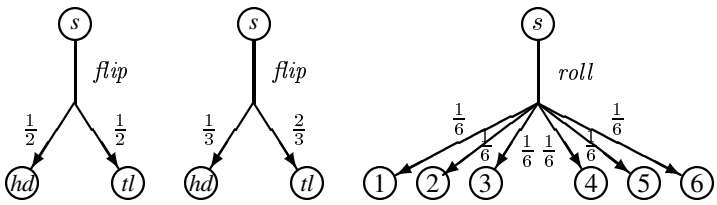


Figure 2.2: Transitions representing a fair coin flip, an unfair coin flip and a fair dice roll.

all probability distributions over X. The *support* of μ is the set $\{x \in X \mid \mu(x) > 0\}$ of elements that are assigned a positive probability. If $X = \{x_1, x_2, \ldots\}$, then we often write the probability distribution μ as $\{x_1 \mapsto \mu(x_1), x_2 \mapsto \mu(x_2) \ldots\}$ and we leave out the elements that have probability 0.

Example 2.2.4 Let the set of states be given by s, hd, tl, 1, 2, 3, 4, 5 and 6. The transitions in Figure 2.2 are respectively given by

$$s \xrightarrow{\mathit{flip}} \{hd \mapsto \tfrac{1}{2}, tl \mapsto \tfrac{1}{2}\},$$
$$s \xrightarrow{\mathit{flip}} \{hd \mapsto \tfrac{1}{3}, tl \mapsto \tfrac{2}{3}\} \text{ and}$$
$$s \xrightarrow{\mathit{roll}} \{1 \mapsto \tfrac{1}{6}, 2 \mapsto \tfrac{1}{6}, 3 \mapsto \tfrac{1}{6}, 4 \mapsto \tfrac{1}{6}, 5 \mapsto \tfrac{1}{6}, 6 \mapsto \tfrac{1}{6}\}.$$

Note that we have left out many elements with probability 0, for instance the state s is reached with probability 0 by each of the transitions above. Moreover, each of the three pictures in Figure 2.2 represents a single transition, where several arrows are needed to represent the probabilistic information.

The definition of a PA is now given as follows.

Definition 2.2.5 A *PA* $\mathcal{A}$ consists of four components:

1. A set $S_{\mathcal{A}}$ of *states*,
2. a nonempty set $S^0_{\mathcal{A}} \subseteq S_{\mathcal{A}}$ of *start states*,
3. An *action signature* $sig_{\mathcal{A}} = (V_{\mathcal{A}}, I_{\mathcal{A}})$, consisting of *external* and *internal actions* respectively. We require $V_{\mathcal{A}}$ and $I_{\mathcal{A}}$ to be disjoint and define the set of *actions* as $\mathrm{Act}_{\mathcal{A}} = V_{\mathcal{A}} \cup I_{\mathcal{A}}$.
4. a *transition relation* $\Delta_{\mathcal{A}} \subseteq S_{\mathcal{A}} \times \mathrm{Act}_{\mathcal{A}} \times \mathrm{Distr}(S_{\mathcal{A}})$.

Again, we write $s \xrightarrow{a}_{\mathcal{A}} \mu$ for $(s, a, \mu) \in \Delta_{\mathcal{A}}$. Furthermore, we simply write $s \xrightarrow{a}_{\mathcal{A}} s'$ for $s \xrightarrow{a}_{\mathcal{A}} \{s' \mapsto 1\}$.

Obviously, the definition of PAs gives rise to the same variations as the definition of NAs. Table 2.16 gives an overview of the variations used in subsequent chapters of this thesis. The basic concepts are the same for all variants.

Example 2.2.6 The PA in Figure 2.3 represents the same 1–place communication channel as the NA in Figure 2.1, except that now the channel is lossy: a bit sent to the channel is lost with a probability of $\frac{1}{100}$. By convention, the transitions $i \xrightarrow{rec(i)} \varepsilon$ reach the state ε with probability 1.

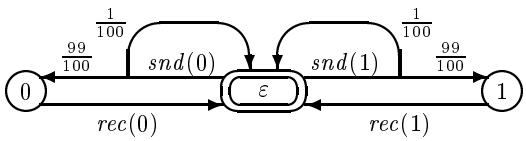


Figure 2.3: A lossy channel PA

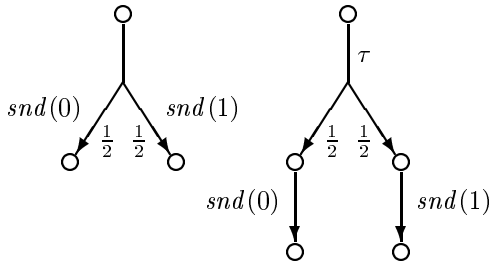


Figure 2.4: Modeling multi–labeled transitions in a PA

Remark 2.2.7 Note that each transition in a PA is labeled with a single action. The system depicted on the left in Figure 2.4 models a process sending the bits 0 and 1 each with probability $\frac{1}{2}$. However, this system is not a PA, because two actions appear in the same transition. Rather, a PA model of such a process is shown on the right of Figure 2.4.

There is, however, a multilabeled version of the PA model, see Section 2.2.5. That model is technically more complex and, in practice, the PA model is expressive enough.

Remark 2.2.8 The nonprobabilistic automata can be embedded in the probabilistic ones by viewing each transition $s \xrightarrow{a} s'$ in an NA as the transition $s \xrightarrow{a} \{s' \mapsto 1\}$ in a PA. Conversely, each PA $\mathcal{A}$ can be "deprobabilized," yielding an NA $\mathcal{A}^-$, by forgetting the specific probabilistic information and by only considering whether a state can be reached with a positive probability. That is, $s \xrightarrow{a} s'$ is a transition in $\mathcal{A}^-$ if and only if there is a transition $s \xrightarrow{a} \mu$ in $\mathcal{A}$ with $\mu(s') > 0$.

Probabilistic versus nondeterministic choice

One can specify nondeterministic choices in a PA in exactly the same way as in a NA, viz. by having internal transitions or by having several transitions leaving from the same state. Also the distinction between external and internal nondeterminism immediately carries over to PAs. Hence, the probabilistic choices are specified *within* the transitions of a PA and the nondeterministic choices *between* the transitions (leaving from the same state) of a PA.

Section 2.2.1 has pointed out the need for nondeterministic choices in automata, namely to model scheduling freedom, implementation freedom, the external environment and incomplete information. These arguments are still valid in the presence of probabilistic choice. In

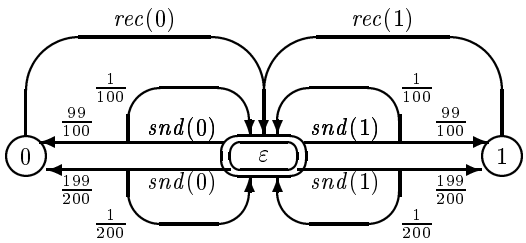


Figure 2.5: A lossy channel PA with partially unknown probabilities

particular, *nondeterminism cannot be replaced by probability* in these cases. As mentioned, nondeterminism is used if we deliberately decide not to specify how a certain choice is made, so in particular we do not want to specify a probability mechanism that governs the choice. Rather, we use a probabilistic choice if the event to be modeled has really all the characteristics of a probabilistic choice. For instance, the outcome of a coin flip, random choices in programming languages, and the arrivals of consumers in a shop.

Thus, probability and nondeterminism are two orthogonal and essential ingredients in the PA model.

An important difference between probabilistic and nondeterministic choice is that the former are governed by a probability mechanism, whereas the latter are completely free. Therefore, probabilistic choices fulfill the laws from probability theory, and in particular the law of large numbers. Informally, this law states that if the same random choice is made very often, the average number of times that a certain event occurs is approximately (or, more precisely, it converges to) its expected value. For instance, if we flip a fair coin one hundred times, it is very likely that about half of the outcomes is heads and the other half is tails. If, on the other hand, we make a nondeterministic choice between two events, then we cannot quantify the likelihood of the outcomes. In particular, we cannot say that each of the sequences is equally likely, because this refers to a probabilistic choice!

The following example illustrates the combination of nondeterministic choice and probabilistic choice.

Example 2.2.9 Like in Figure 2.3, the PA in Figure 2.5 also represents a faulty communication channel. The difference with the PA in Figure 2.3 is that, here, we do not know the probability that a bit is lost exactly. Depending on which transition is taken, it can be $\frac{1}{200}$ or $\frac{1}{100}$. However, we will see in Section 2.3 that this probability can in fact have any value between $\frac{1}{200}$ and $\frac{1}{100}$. Thus, the nondeterministic choice between the two $send(i)$ transitions is used here to represent incomplete information about exact probabilities.

This PA can also be considered as the specification of a system. Then the nondeterministic choice represents implementation freedom: A PA correctly implements the one in Figure 2.5 if the probability to lose a message is at most $\frac{1}{100}$ and at least $\frac{1}{200}$. (The latter might be a bit awkward in practice.)

For future use, define distributions μ_x^i and the transitions θ_x^i as follows. For $x = 100, 200$ and $i = 0, 1$ let

$$\theta_x^i = \varepsilon \xrightarrow{snd(i)} \mu_x^i, \qquad \mu_x^i = \{\varepsilon \mapsto \tfrac{1}{x}, i \mapsto \tfrac{x-1}{x}\}.$$

Thus, the superscripts denote the bit and the subscripts the probability involved.

Finally, we remark that the philosophical debate on the nature of nondeterminism choice and probability has not ended. In this thesis, we do not take a standpoint in that discussion and neither do we contribute to it. Rather, we take a practical point of view. The only assumption we rely on in this thesis that both types of choices are appropriate for describing and predicting certain phenomena. For more information on the philosophical issues arising in the area of probability theory, the reader is referred to [Coh89].

2.2.3 Timing

Timing can be incorporated in the PA model in a similar way as in the NA model (c.f. the "old fashioned recipe for time" [AL92]). A *probabilistic timed automaton (PTA)* is a PA with time passage actions. These are actions $d \in \mathbb{R}^{>0}$ that indicate the passage of d time units. While time elapses, no other actions take place and, in the PTA approach, time advances deterministically. So, in particular, no (internally) nondeterministic or probabilistic choices can be specified within time passage actions, see requirements 1 and 2 in the definition below. The third requirement below, Wang's Axiom [Yi90], requires that, while time advances, the state of the PTA is well–defined at each point in time and that, conversely, two subsequent time passage actions can be combined into a single one.

The PTA model presented here is somewhat simpler, but similar to the one in [Seg95b], which is based on a variation of Wang's axiom.

Definition 2.2.10 A PTA $\mathcal{A}$ is a PA enriched with a partition of $\mathrm{Act} \setminus \{\tau\}$ into a set of *discrete actions* Act_D and the set $\mathbb{R}^{>0}$ of positive real numbers or *time–passage actions*. We require[2] that, for all $s, s', s'' \in S_{\mathcal{A}}$ and $d, d' \in \mathbb{R}^{>0}$ with $d' < d$,

1. each transition labeled with a time–passage action leads to a distribution that chooses one element with probability 1,
2. (Time determinism) if $s \xrightarrow{d}_{\mathcal{A}} s'$ and $s \xrightarrow{d}_{\mathcal{A}} s''$ then $s' = s''$.
3. (Wang's Axiom) $s \xrightarrow{d}_{\mathcal{A}} s'$ iff $\exists s'' : s \xrightarrow{d'}_{\mathcal{A}} s''$ and $s'' \xrightarrow{d-d'}_{\mathcal{A}} s'$.

As PTAs are a special kind of PAs, we can use the notions defined for PAs also for PTAs.

By letting time pass deterministically, PTAs treat probabilistic choice, nondeterministic choice and time passage as orthogonal concepts, which leads to a technically clean model. Example 2.2.11 below shows that discrete probabilistic choices over time can be encoded in PTAs via internal actions. Nondeterministic choices over time can be encoded similarly: just replace the probabilistic choice in the example by a nondeterministic one. Thus, although we started from a deterministic view on time, nondeterminism and probabilistic choices over time sneak in via a back door. The advantage of the PTA approach is that we separate concerns by specifying one thing at the time: time passage or probabilistic/nondeterministic choice.

Example 2.2.11 We can use a PTA to model a system that decides with an internal probabilistic choice whether to wait a short period (duration one time unit) or a long period (two time units) before performing an a action. This PTA is partially given in Figure 2.6, where the second element of each state records the amount of time that has elapsed. Note that there are uncountably many transitions missing in the picture, for instance the transitions

[2] For simplicity the conditions here are slightly more restrictive than those in [LV96b].

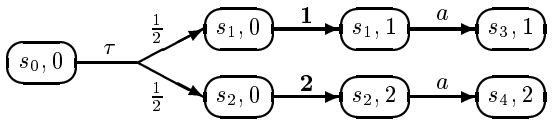


Figure 2.6: A part of a PTA

$(s_1, 0) \xrightarrow{\mathbf{0.2}} (s_1, 0.2)$ and $(s_1, 0.2) \xrightarrow{\mathbf{0.5}} (s_1, 0.7)$, see Wang's Axiom in the preceding definition. The full transition relation is given by

$$
\begin{array}{ll}
(s_0, 0) \xrightarrow{\tau} \{(s_1, 0) \mapsto \frac{1}{2}, (s_2, 0) \mapsto \frac{1}{2}\}, & \\
(s_1, d) \xrightarrow{d'} (s_1, d + d'), & \text{if } d + d' \leq 1, \\
(s_2, d) \xrightarrow{d'} (s_2, d + d'), & \text{if } d + d' \leq 2, \\
(s_1, 1) \xrightarrow{a} (s_3, 1), & \\
(s_2, 2) \xrightarrow{a} (s_4, 1). &
\end{array}
$$

Continuous distributions over time can of course not be encoded in a PTA, since such distributions cannot be modeled in the PA model anyhow. There are various other models combining time and probability, see Section 2.6, including models dealing with continuous distributions over time.

Moreover, there is a second model that extends PAs with nondeterministic timing. The automata introduced in [KNSS01] — also called PTAs — augment the classical timed automata [AD94] with discrete probabilistic choice. They allow timing constraints to be specified via real–valued clocks, as in Chapter 1. An example PTA model of this kind has been given in Example 1.2.4 on page 18.

2.2.4 Parallel Composition

The parallel composition operator $\|$ allows one to construct a PA from several component PAs. This makes system descriptions more understandable and enables component–wise design. The component PAs run in parallel and interact via their external actions. As before, the situation is similar to the nonprobabilistic case.

Consider a composite PA that is built from two component PAs. Then the state space of the composite PA consists of pairs (s, t), reflecting that the first component is in state s and the second in state t. If one of the components can take a step, then so can the composite PA, where *synchronization* on shared actions has to be taken into account. This means that whenever one component performs a transition involving a visible action a, the other one should do so simultaneously, provided that a is in its action set.

When synchronization occurs, both automata resolve their probabilistic choices independently, because the probability mechanisms used in different components are not supposed to influence each other. Thus, if the transitions $s_1 \xrightarrow{a} \mu_1$ and $s_2 \xrightarrow{a} \mu_2$ synchronize, then the state (s_1', s_2') is reached with probability $\mu_1(s_1') \cdot \mu_2(s_2')$.

No synchronization is required for transitions labeled by an internal action $a \in I$ nor for visible actions which are not shared (i.e. present in only one of the automata). In this case, one component takes a transition, while the other remains in its current state with probability one. For instance, if the first component takes the transition $s_1 \xrightarrow{\tau} \mu_1$ and the other one

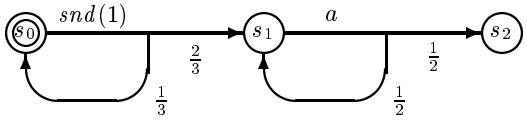


Figure 2.7: A sender PA

remains in the state s_2, then the probability to reach the state (s_1', s_2) by taking this transition equals $\mu_1(s_1')$ and the probability to reach a state (s_1', s_2') with $s_2' \neq s_2$ is zero.

To define the parallel composition, we first need two auxiliary definitions. The first one defines the probabilistic choice arising from two independent probabilistic choices. The second one ensures that we only take the parallel composition of PAs whose action signatures do not clash.

Notation 2.2.12 Let μ be a probability distribution on X and ν one on Y. Define the distribution $\mu \times \nu$ on $X \times Y$ by

$$(\mu \times \nu)(x, y) = \mu(x) \cdot \nu(y).$$

Definition 2.2.13 We say that two PAs $\mathcal{A}$ and $\mathcal{B}$ are *compatible* if $I_{\mathcal{A}} \cap \mathrm{Act}_{\mathcal{B}} = \mathrm{Act}_{\mathcal{A}} \cap I_{\mathcal{B}} = \emptyset$.

Definition 2.2.14 For two compatible PAs $\mathcal{A}$ and $\mathcal{B}$, the *parallel composition* is the probabilistic automaton $\mathcal{A} \parallel \mathcal{B}$ defined by:

1. $S_{\mathcal{A}\parallel\mathcal{B}} = S_{\mathcal{A}} \times S_{\mathcal{B}}$.
2. $S^0_{\mathcal{A}\parallel\mathcal{B}} = S^0_{\mathcal{A}} \times S^0_{\mathcal{B}}$.
3. $sig_{\mathcal{A}\parallel\mathcal{B}} = (V_{\mathcal{A}} \cup V_{\mathcal{B}}, I_{\mathcal{A}} \cup I_{\mathcal{B}})$.
4. $\Delta_{\mathcal{A}\parallel\mathcal{B}}$ is the set of transitions $(s_1, s_2) \xrightarrow{a} \mu_1 \times \mu_2$ such that at least one of the following requirements is met.
   - $a \in V_{\mathcal{A}} \cap V_{\mathcal{B}}$, $s_1 \xrightarrow{a} \mu_1 \in \Delta_{\mathcal{A}}$ and $s_2 \xrightarrow{a} \mu_2 \in \Delta_{\mathcal{B}}$.
   - $a \in \mathrm{Act}_{\mathcal{A}} \setminus \mathrm{Act}_{\mathcal{B}}$ or $a \in I_{\mathcal{A}}$, and $s_1 \xrightarrow{a} \mu_1 \in \Delta_{\mathcal{A}}$ and $\mu_2 = \{s_2 \mapsto 1\}$.
   - $a \in \mathrm{Act}_{\mathcal{B}} \setminus \mathrm{Act}_{\mathcal{A}}$ or $a \in I_{\mathcal{B}}$, and $s_2 \xrightarrow{a} \mu_2 \in \Delta_{\mathcal{B}}$ and $\mu_1 = \{s_1 \mapsto 1\}$.

Note that nondeterminism is essential in this definition.

Example 2.2.15 The system in Figure 2.7 represents a sender process. Its action set is $\{snd(0), snd(1), a\}$. Figure 2.8 shows the parallel composition process of this process and the channel process in Figure 2.3.

2.2.5 Other Automaton Models combining Nondeterminism and Discrete Probabilities

Below, we discuss two other automaton–based frameworks that combine discrete probabilistic and nondeterministic choice. We treat the GPA model and the alternating model, which are both equivalent to the PA model in some sense.

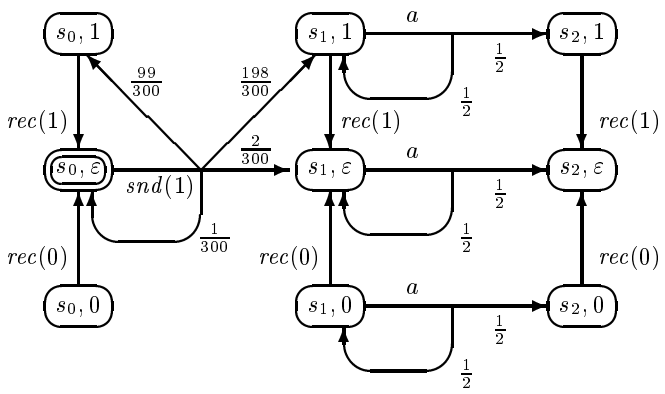


Figure 2.8: Parallel composition

General probabilistic automata

Segala [Seg95b] introduces a more general notion of probabilistic automata, which we call *general probabilistic automata* (GPAs)[3]. A GPA is the same as a PA, except that the transitions have the type $S \times \mathrm{Distr}(\mathrm{Act} \times S \cup \{\bot\})$. Thus, each transition chooses both the action and the next state probabilistically. Moreover, it can choose to deadlock ($\bot$) with some probability. In the latter case, no target state is reached. Figure 2.4 on page 34 shows a GPA that is not a PA.

Problems arise when defining the parallel composition operator for arbitrary GPAs. The trouble comes from synchronizing transitions that have some shared actions and some actions which are not shared (see [Seg95b], Section 4.3.3).

The problem can be solved by imposing the I/O distinction on GPAs (c.f. the remark below Definition 2.2.2). This distinction comes with the requirement that input actions are enabled in every state and occur only on transitions labeled by a single action. This approach is followed in [WSS97] and in [HP00]. Surprisingly, the latter reverses the role of input and output actions.

In our experience, many practical systems can be modeled conveniently with PAs (see Remark 2.2.7; deadlocks can be modeled by moving to a special deadlock state). Moreover, several notions have been worked out for the PA model only. Therefore, this thesis deals with PAs rather than with GPAs.

Alternating model

The alternating model, introduced by Hansson and Jonsson [Han94, HJ94], distinguishes between *probabilistic* and *nondeterministic states*. Nondeterministic states have zero or more outgoing transitions. These are labeled by actions and lead to a unique probabilistic state. Probabilistic states have one or more outgoing transitions. These are labeled by probabilities and specify a probability distribution over the nondeterministic states.

The alternating model and the PA model are isomorphic upto so-called strong bisimu-

[3] Segala uses the word PA for what we call GPA and says simple PA to what we call PA.

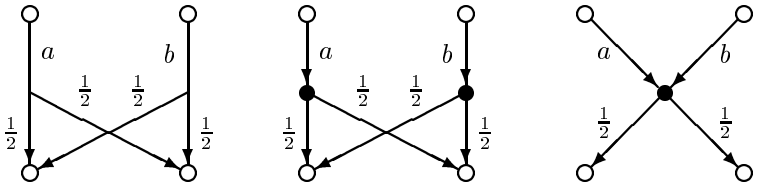


Figure 2.9: A PA model and two alternating models equivalent to it

lation [LS91]. (This is a kind of graph isomorphism on unfolded automata that takes into account the action labels, see Chapter 5 for details.) This means that all notions defined on PAs that respect strong bisimulation can be translated into the alternating model and vice versa.

In order to translate an alternating model into a PA, one removes all probabilistic states and contracts each ingoing transition of a probabilistic state with the probability distribution going out of that state, thus passing by the probabilistic state. Conversely, in order to translate a PA into an alternating model, one introduces an intermediate probabilistic state for each transition. The reason why this only yields an isomorphism upto bisimulation, rather than just an isomorphism, is illustrated in Figure 2.9.

2.3 The Behavior of Probabilistic Automata

This section defines the semantics of a PA as the set of its trace distributions. Each trace distribution is a probability space assigning a probability to certain sets of traces. The idea is that a trace distribution arises from resolving the nondeterministic choice first and by then abstracting from nonvisible elements, i.e. by removing the states and internal actions and leaving only the external actions. A difference with the nonprobabilistic case is that the theory of PAs allows nondeterministic choices to be resolved by probabilistic choices. As we will see, these are described by an adversary. Moreover, resolving the nondeterministic choices in a PA no longer yields a linear execution, as in an NA but can be considered as a tree–like structure.

We recall the definition of traces for NAs first.

2.3.1 Paths and Traces

The semantics of an NA is given by the set of its traces. Each trace represents one of the potential visible behaviors of the system and is obtained by the following steps. First, the nondeterministic choices in the NA are resolved. This yields an execution, i.e. a sequence of states and actions. Then the states and internal actions are removed from the execution. This yields a sequence of actions, called a *trace*.

Definition 2.3.1 1. An *path* (or *execution*) of an NA $\mathcal{A}$ is a possibly infinite sequence $\pi = s_0 a_1 s_1 a_2 \ldots$ where s_0 is an initial state of $\mathcal{A}$, s_i is a state of $\mathcal{A}$, a_i is an (internal or external) action of $\mathcal{A}$ and $s_i \xrightarrow{a_{i+1}}_{\mathcal{A}} s_{i+1}$ is a transition. Moreover, we require that if π is finite, then it ends in a state.

2. A *trace* is a finite or infinite sequence of external actions that is obtained from a path by omitting the states and internal actions. We denote the set of traces of $\mathcal{A}$ by $trace(\mathcal{A})$.

Example 2.3.2 The three sequences ε and $\langle \varepsilon\ snd(1)\ 1\ rec(1)\ \varepsilon\ snd(0)\ 0\rangle$ and $\langle \varepsilon\ snd(0)\ 0\ rec(0)\ \varepsilon\ snd(0)\ 0\ rec(0)\ldots\rangle$ are paths of the NA in Figure 2.1. Their traces are, respectively, the empty sequence ε, the sequence $\langle snd(1)\ rec(1)\ snd(0)\rangle$ and the sequence $\langle snd(0)\ rec(0)\ snd(0)\ rec(0)\ snd(0)\ldots\rangle$. (The brackets $\langle\rangle$ here have been inserted for the sake of readability, but have no meaning.)

We can also identify paths and traces in a PA.

Definition 2.3.3 1. A *probabilistic path* (abbreviated as *path*) of a PA $\mathcal{A}$ is an alternating, finite or infinite sequence

$$\pi \ = \ s_0 a_1 \mu_1 s_1 a_2 \mu_2 s_2 a_3 \mu_3 \ldots$$

where $s_0 \in S^0_{\mathcal{A}}$ $s_i \in S_{\mathcal{A}}$, $a_i \in \mathrm{Act}_{\mathcal{A}}$, $s_i \xrightarrow{a_{i+1}}_{\mathcal{A}} \mu_{i+1}$ and $\mu_{i+1}(s_{i+1}) > 0$. Moreover, if π is finite, then it has to end in a state. Let $last(\pi)$ denote the last state of a finite path, $|\pi| \in \mathbb{N} \cup \{\infty\}$ the number of actions occurring in π, $Path^*(\mathcal{A})$ the set of finite paths of $\mathcal{A}$ and $Path^*(s, \mathcal{A})$ the set of finite paths in $\mathcal{A}$ starting in state s.

2. A *trace* is a finite or infinite sequence of external actions that is obtained from a path by omitting the states, internal actions and distributions. Let $trace$ denote the function that assigns to each execution its trace.

Both the probabilistic and the nondeterministic choices have been resolved in a probabilistic path, since it reveals that the i^{th} transition taken is $s_{i-1} \xrightarrow{a_i} \mu_i$ and that the outcome of the i^{th} probabilistic choice μ_i is s_i. Moreover, note that each probabilistic path of $\mathcal{A}$ gives rise to a path of $\mathcal{A}^-$, by removing all the probability distributions.

Example 2.3.4 The sequence $\langle \varepsilon\ snd(1)\ \mu^1_{100}\ \varepsilon\ snd(1)\ \mu^1_{100}\ 1\rangle$ is a finite probabilistic path of the PA in Figure 2.5. Its trace is $\langle snd(1)\ snd(1)\rangle$.

In practical verifications, a lot of effort is usually spent on finding out which states in a system can be reached and which cannot. Thus, the following definition prepares for the case study in Chapter 7.

Definition 2.3.5 A state s in a NA or a PA $\mathcal{A}$ is *reachable* if there is a finite path ending in s. We denote the set of reachable states by $reach_{\mathcal{A}}$.

2.3.2 Trace Distributions

The semantics of a PA is given by the set of distributions over its traces, called *trace distributions*. Each trace distribution of a PA represents one of the potential visible behaviors of the system, just as a trace in an NA. As said before, a trace distribution is obtained by resolving the nondeterministic choices in the PA (replacing them by probabilistic choices) first and by then removing the states and the internal actions next.

More precisely, the way in which the nondeterministic choices in a PA are resolved is given by a so–called *randomized, partial adversary*. An adversary can be considered as the equivalent of an execution in an NA. To study the probabilistic behavior generated by an

adversary, a *probability space* is associated to each adversary. This assigns a probability to certain sets of paths in the PA. The trace distribution is then obtained by removing all states and internal actions from the associated probability space.

The forthcoming sections discuss each of the steps above in more detail.

Resolving nondeterministic choices in PAs

In order to understand what the behavior of a PA is like and how the nondeterministic choices in a PA are resolved, consider the channel in Figure 2.3 on page 34 again and recall that $snd(i)$ models the sending of a bit by a sender process, which corresponds to the receipt by the channel.

What can happen during the execution of this channel? Being in its start state ε, the channel may either receive a 0, a 1 or it might receive no bit at all. One of the fundamental aspects in the theory of PAs is that each of these possibilities may occur with a certain probability. Say that the probability on a 0 to arrive is q_0, on a 1 to arrive is q_1 and on no bit to arrive at all is $1 - q_0 - q_1$. Then the channel takes the transition $snd(0)$ with probability q_0. Similarly, it takes the transition $snd(1)$ with probability q_1 and remains in the state ε (forever) with $1 - q_0 - q_1$. In the latter case we say that the execution of the channel is interrupted.

Each choice for the values q_0 and q_1 in $[0, 1]$ yields a potential (and different) behavior of the channel. In this example, the probabilities naturally arise from a probabilistic environment (a sender process) that determines probabilistically whether to send a bit and which one. In general, we describe the resolution of the nondeterministic choices by an adversary.

Upon taking the transition that has been chosen probabilistically, the system determines its next state according to the target distribution of the transition chosen. In the example, the channel moves to the state 0 with probability $q_0 \cdot \frac{99}{100}$, to 1 with probability $q_1 \cdot \frac{99}{100}$ and stays in ε with the remaining probability mass $1 - q_0 \cdot \frac{99}{100} - q_1 \cdot \frac{99}{100}$. Here, we see that the probability to lose a bit is only determined exactly if we know how the nondeterminism in the system is resolved (i.e. if we know q_0 and q_1). Before resolving the nondeterministic choices, do not know the probability to lose a bit, we can only say that it is at most $\frac{1}{100}$.

After taking the transition, the procedure starts over in the new state: the channel makes a probabilistic choice between the outgoing transitions in the new state and an interruption. That is, in the states i, the channel has the choice between $rec(i)$ and an interruption; in the state ε there is a choice between $snd(0)$, $snd(1)$ and interruption. Obviously, these choices are not there if we are in ε as the result of an interruption. Moreover, when resolving the nondeterministic choice in ε, we do not have to take the same probabilities q_0 and q_1 as before: for instance, the environment may now send the bits with different probabilities. Moreover, the probabilities may be different depending on the bit that the channel previously received. Therefore the resolution of the nondeterminism can be history–dependent: it may not only depend on the current system state, but also on the path leading to that state.

Formally, the resolution of the nondeterministic choices in a PA is described by an *adversary* (also called *scheduler* or *policy*). In each state of the system, the adversary determines the next transition to be taken. The explanation above shows that we need adversaries that are

- *partial* i.e. may interrupt the execution at any time,
- *randomized* i.e. determine their choices randomly and
- *history–dependent* i.e. may base their choices not only on the current state, but also on

the path leading to that state.

This means that, given a finite path π ending in a state s, the adversary A schedules the transition $s \xrightarrow{a} \mu$ with probability $A(\pi)(a,\mu)$. The value $A(\pi)(\bot)$ is the probability on an interruption. We let our adversaries start from a fixed start state s_0[4].

Definition 2.3.6 Let s_0 be a start state of a PA $\mathcal{A}$. A *randomized, partial, history–dependent adversary* (or shortly an *adversary*) of $\mathcal{A}$ starting from s_0 is a function

$$A : Path^*(s_0, \mathcal{A}) \rightarrow \text{Distr}(\text{Act}_{\mathcal{A}} \times \text{Distr}(S_{\mathcal{A}}) \cup \{\bot\})$$

such that if $A(\pi)(a,\mu) > 0$ then $last(\pi) \xrightarrow{a}_{\mathcal{A}} \mu$.

Example 2.3.7 Reconsider the lossy communication channel from Example 2.2.9. Let A_1 be the adversary that schedules the transition θ^1_{100} (i.e. sends a 1) whenever the system is in the state ε. Furthermore, A_1 schedules the $rec(i)$–action whenever the system is in i. Then A_1 is defined by

$$\begin{aligned} &A_1(\pi)(snd(1), \mu^1_{100}) = 1, && \text{if } last(\pi) = \varepsilon \\ &A_1(\pi)(rec(i), \{\varepsilon \mapsto 1\}) = 1, && \text{if } last(\pi) = i \end{aligned}$$

and 0 in all other cases. Obviously, in the second clause, only the case $i = 1$ is relevant, because the bit 0 is never sent. Nevertheless, we require the adversary also to be defined on paths containing an $snd(0)$ action, since this is technically simpler. Later, we will see that such paths are assigned probability 0.

The adversary A_2 schedules the transitions θ^0_{100}, θ^0_{200}, θ^1_{100} and θ^1_{200} each with probability $\frac{1}{4}$ whenever the system is in state ε . The $rec(i)$ action is taken with probability one if the system is in the state i. Then A_2 is given by

$$\begin{aligned} &A_2(\pi)(snd(i), \mu^i_j) = \tfrac{1}{4}, && \text{if } last(\pi) = \varepsilon \\ &A_2(\pi)(rec(i), \{\varepsilon \mapsto 1\}) = 1, && \text{if } last(\pi) = i \end{aligned}$$

for $i = 0, 1$, $j = 100, 200$ and 0 in all other cases.

The adversary A_3 corresponds to scheduling the transition θ^1_{100} in state ε with probability $\frac{1}{3}$, the transition θ^1_{200} with probability $\frac{1}{3}$, the transitions θ^0_{100} and θ^0_{200} with probability 0 and to interrupt the execution with probability $\frac{1}{3}$. Also, in state i, the probability of interruption is $\frac{1}{3}$. This adversary is defined by

$$\begin{aligned} &A_3(\pi)(snd(1), \mu^1_{100}) = \tfrac{1}{3}, && \text{if } last(\pi) = \varepsilon \\ &A_3(\pi)(snd(1), \mu^1_{200}) = \tfrac{1}{3}, && \text{if } last(\pi) = \varepsilon \\ &A_3(\pi)(rec(i), \{\varepsilon \mapsto 1\}) = \tfrac{2}{3}, && \text{if } last(\pi) = i \\ &A_3(\pi)(\bot) = \tfrac{1}{3}, \end{aligned}$$

and 0 otherwise.

[4]Obviously, the forthcoming theory also applies to adversaries starting in non–start states, but we wish to consider the behavior generated from start states only.

Remark 2.3.8 An adversary A starting in a state s_0 can be described by a tree whose root is the state s_0, whose nodes are the finite paths in A and whose leaves are the sequences $\pi\perp$, where π is a path satisfying $A(\pi)(\perp) > 0$. The children of a node π are the finite paths $\pi a\mu t$, where $A(\pi)(a,\mu) > 0$ and $\mu(t) > 0$, and the sequence $\pi\perp$ if$A(\pi)(\perp) > 0$. The edge from π to $\pi a\mu t$ is labeled with the probability $A(\pi)(a,\mu) \cdot \mu(t)$ and the edge from π to $\pi\perp$ with the probability $A(\pi)(\perp)$. In fact, this tree is a cycle–free discrete time Markov chain.

By considering partial, history–dependent, randomized adversaries, the theory of PAs makes three fundamental choices for the behavior of PAs. We have motivated these choices by the channel NA already and below we give a more generic motivation of these decisions.

Partiality is already present in the nonprobabilistic case, where the execution of an NA may end in any state, even if there are transitions available in that state. Partiality is needed for compositionality results, both in the probabilistic and the nondeterministic case.

History dependence is also exactly the same as in the non–probabilistic case: each time the execution of an NA visits a certain state, it may take a different outgoing transition to leave that state.

Randomization has no counterpart in NAs. There are several arguments why we need randomized adversaries rather than deterministic ones.

- *Including all possible resolutions.* First of all, it is very natural to allow a nondeterministic choice to be resolved probabilistically. Nondeterminism is used if we do not wish to specify the factors that influence the choice. In particular, such factors can be probabilistic; there is no reason to assume that these are deterministic. Thus, randomized adversaries are needed to include all possible ways (including probabilistic ones) to resolve the nondeterministic choices.

- *Modeling probabilistic environments.* As we saw in the channel example, nondeterminism can model choices to be resolved by the environment. Since the environment may be probabilistic, randomized adversaries are needed to model the behavior of the PA in this environment.

- *Randomized algorithms: implementing a nondeterministic choice by a probabilistic choice*. The specification of a system often leaves room for several alternative implementations. Randomized algorithms usually implement their specifications by a probabilistic choice over those alternative implementations. By allowing randomized adversaries, the behavior of the randomized algorithm is included in the behavior of the specification, but this is not true when ranging over deterministic adversaries only. In Section 2.4 we will see that implementation relations for PAs are based on inclusion of external behavior. If we base the notion of behavior based on deterministic adversaries, then it is not possible to implement nondeterministic with randomized algorithms, unlike a notion of randomized adversaries [Alf97].

However, it has been proven [Alf99] that, if one is only interested in the minimal and maximal probability of a certain event, then it suffices to consider only deterministic adversaries.

The probability space associated to an adversary

Once the nondeterminism has been resolved by an adversary, we can study the probabilistic behavior of the system under this adversary. This is done via the associated probability space. The behavior generated by an adversary A is obtained by scheduling the transitions described by A and executing them until – possibly – A tells us to stop. The paths obtained in this way are the *maximal paths* in A, i.e. the infinite paths and the finite paths that have a positive probability on termination. Thus, the maximal paths represent the complete rather than partial behavior of A. Each maximal path can be assigned a probability. As we will see, the associated probability space assigns a probability to certain sets of maximal paths.

Throughout this section, let A be a randomized partial adversary for a PA $\mathcal{A}$.

Definition 2.3.9 A *path in* A is a finite or infinite path

$$\pi \;=\; s_0 a_1 \mu_1 s_1 a_2 \mu_2 \ldots$$

such that $A(s_0 a_1 \mu_1 s_1 \ldots a_i \mu_i)(a_{i+1}, \mu_{i+1}) > 0$ for all $0 \leq i < |\pi|$. The *maximal paths in* A are the infinite paths in A and the finite paths π in A where $A(\pi)(\bot) > 0$. Define $Path^{max}(A)$ as the set of maximal paths in A.

Note the difference between paths in a PA and those in an adversary. The former are a superset of the latter: compare Definitions 2.3.3 and 2.3.9.

For every finite path π, we can compute the probability $\mathbf{Q}^A(\pi)$ that a path generated by A starts with π. This probability is obtained by multiplying the probabilities that A actually schedules the transitions given by π with the probabilities that taking a transition actually yields the state specified by π. Note that the probability that the path generated by A is *exactly* π equals $\mathbf{Q}^A(\pi) \cdot A(\pi)(\bot)$.

Definition 2.3.10 Let $\mathcal{A}$ be a PA and let $s_0 \in S_{\mathcal{A}}$ be a state. Then we define the function $\mathbf{Q}^A : Path^*(s_0, \mathcal{A}) \to [0,1]$ inductively by

$$\mathbf{Q}^A(s_0) = 1 \text{ and } \mathbf{Q}^A(\pi a \mu t) = \mathbf{Q}^A(\pi) \cdot A(\pi)(a,\mu) \cdot \mu(t).$$

Example 2.3.11 Reconsider the adversaries A_1, A_2 and A_3 from the Example 2.3.7 The path $\pi = \langle \varepsilon, snd(1), \mu^1_{100}, \varepsilon, snd(1), \mu^1_{100}, 1\rangle$ is a path in A_1, A_2 and in A_3. It is assigned the following probabilities by the adversaries:

$$\mathbf{Q}^{A_1}(\pi) = 1 \cdot \tfrac{1}{100} \cdot 1 \cdot \tfrac{99}{100}, \quad \mathbf{Q}^{A_2}(\pi) = \tfrac{1}{4} \cdot \tfrac{1}{100} \cdot \tfrac{1}{4} \cdot \tfrac{99}{100} \quad \mathbf{Q}^{A_3}(\pi) = \tfrac{1}{3} \cdot \tfrac{1}{100} \cdot \tfrac{1}{3} \cdot \tfrac{99}{100}.$$

This path is maximal in A_3, but not in A_1 and A_2. Furthermore, the sequence $\langle \varepsilon,\ snd(0),\ \mu^0_{100}, \varepsilon\rangle$ is a path of the system in Figure 2.5. It is also a path of the adversary A_2, but not of the adversaries A_1 and A_3.

To study the probabilistic behavior generated by an adversary A, we associate a probability space to it. A probability space is a mathematical structure that assigns a probability to certain sets of (in this case) maximal paths such that the axioms of probability are respected (viz., the probability on the set of all events is 1; the probability on the complement of a set is one minus the probability on the set; and the probability of a countable, pairwise disjoint union of sets is the sum of the probabilities on the sets).

Note that we cannot describe the probabilistic behavior of an adversary by a probability *distribution*, assigning a probability to each element (in this case a maximal path), because

this does not provide enough information. For instance, consider the adversaries A_1 and A_2 from Example 2.3.7. Their maximal paths are all infinite and both adversaries assign probability 0 to each infinite path. However, they differ on many sets of maximal paths, e.g. the probability that the first action in a path is $snd(1)$ equals 1 for A_1 and $\frac{1}{2}$ for A_2.

Definition 2.3.12 A *probability space* is a triple $(\Omega, \mathcal{F}, \mathbf{P})$, where

1. Ω is a set, called the *sample space*,
2. $\mathcal{F} \subseteq 2^{\Omega}$ is σ*-field*, i.e. a collection of subsets of Ω which is closed under countable[5] union and complement and which contains Ω,
3. $\mathbf{P} : \mathcal{F} \to [0,1]$ is a *probability measure* on $\mathcal{F}$, which means that $\mathbf{P}[\Omega] = 1$ and for any countable collection $\{X_i\}_i$ of pairwise disjoint subsets in $\mathcal{F}$ we have $\mathbf{P}[\cup_i X_i] = \sum_i \mathbf{P}[X_i]$.

Note that it now follows that $\mathbf{P}[\emptyset] = 0$ and $\mathbf{P}[\Omega - X] = 1 - \mathbf{P}[X]$. It is also obvious that $\mathcal{F}$ is closed under intersection.

The idea behind the definition of a probability space is as follows. One can prove that it is not possible to assign a probability to each set and to respect the axioms of probability at the same time. Therefore, we collect those sets to which we can assign a probability into a σ–field $\mathcal{F}$. A σ–field is a collection of sets that contains the set Ω of all events and that is closed under complementation and countable union. The rationale behind this is that we can follow the axioms of probability. Thus, we can assign probability one to Ω and therefore $\Omega \in \mathcal{F}$. Moreover, if we assign probability $\mathbf{P}[X]$ to a set $X \in \mathcal{F}$, then we can also assign a probability to its complement, viz. $1 - \mathbf{P}[X]$. Therefore, the $\mathcal{F}$ is closed under complementation. Similarly, if we have a collection of pairwise disjoint sets $\{X_i\}_i$ which are all assigned a probability then $\mathbf{P}[\cup_i X_i] = \sum_i \mathbf{P}[X_i]$. Hence, the union can also be given a probability and therefore $\mathcal{F}$ is closed under countable unions.

The probability space associated to an adversary is generated from the sets C_π. Here C_π is the *cone above* π, the set containing all maximal paths that start with the finite path π. Since we know the probabilities on the set C_π – namely $\mathbf{Q}^A(\pi)$ – and we need to have a σ–field, we simply consider the smallest σ–field that contains these sets. A fundamental theorem from measure theory now states that, under the conditions met here, we can give a probability measure on all sets in $\mathcal{F}_A$ by specifying it on the sets C_π only, see for instance [Hal50] and [Seg95b].

Definition 2.3.13 The *probability space* associated to a partial adversary A starting in s_0 is the probability space given by

1. $\Omega_A = Path^{max}(A)$,
2. $\mathcal{F}_A$ is the smallest σ–field that contains the set $\{C_\pi \mid \pi \in Path^*(A)\}$, where $C_\pi = \{\pi' \in \Omega_A \mid \pi \sqsubseteq \pi'\}$ and $\sqsubseteq$ denotes the prefix relation on paths,
3. $\mathbf{P}_A$ is the unique measure on $\mathcal{F}_A$ such that $\mathbf{P}_A[C_\pi] = \mathbf{Q}^A(\pi)$ for all $\pi \in Path^*(s, \mathcal{A})$.

The fact that $(\Omega_A, \mathcal{F}_A, \mathbf{P}_A)$ is a probability space follows from standard measure theory arguments, see for instance [Hal50] or [Coh80]. Note that Ω_A and $\mathcal{F}_A$ do not depend on A but only on $\mathcal{A}$, and that $\mathbf{P}_A$ is fully determined by the function $\mathbf{Q}^A$.

[5] In our terminology, countable objects include finite ones.

Note that the cone C_π is contained in $\mathcal{F}_A$ for every finite path in A, but that the cone itself contains finite and infinite *maximal* paths. The reason for requiring π to be a path in the adversary A rather than a path in $\mathcal{A}$ is that in this way $\mathcal{F}_A$ is generated by countably many cones, even if the set of states or actions of $\mathcal{A}$ is uncountable (as is the case for PTAs).

Furthermore, it is not difficult to see that if the set Ω_A is countable, then $\mathcal{F}_A$ is simply the power set of A. However, if Ω_A is uncountable (and this is the case for which probability spaces have been designed), then the set $\mathcal{F}_A$ is quite complicated — probably more complicated than it seems at first sight. Obviously, this collection can be generated inductively by starting from the cones and by taking complementation and countable unions. This requires ordinal induction, rather than ordinary induction. Moreover, the construction of a set not being in $\mathcal{F}_A$ crucially depends on the axiom of choice. The branch of mathematics that is concerned with probability spaces and, more general, measure spaces is called *measure theory*.

The following example presents a few sets that are contained in $\mathcal{F}_A$.

Example 2.3.14 The collection $\mathcal{F}_A$ contains many sets of traces that occur in practice, or that come easily to one's mind. For instance, the set of paths containing at most three elements a is given by

$$\bigcup_{\rho \in X} C_\rho,$$

where $X = \{\alpha \in Path^*(\mathcal{A}) \mid \alpha \text{ contains at most three } a\text{'s}\}$. Since X is countable, the set above is an element of $\mathcal{F}_A$. The set containing the single infinite path π equals

$$\bigcap_{\rho \sqsubseteq \pi, \rho \neq \pi} C_\rho.$$

Example 2.3.15 Consider the adversary A_2 from Example 2.3.7. Then Ω_{A_2} is just the sets of all infinite paths. The set C_π contains the infinite paths extending the path π and $\mathcal{F}_{A_2}$ is the smallest σ–algebra containing those cones. Some values of the function $\mathbf{P}^{A_2}$ are

$$\mathbf{P}^{A_2}[C_{\langle \varepsilon \; snd(0) \; \mu^1_{100} \; 1 \rangle}] = \mathbf{Q}^{A_2}(\langle \varepsilon \, snd(0) \mu^1_{100} 1 \rangle) = \tfrac{1}{4} \cdot \tfrac{99}{100}.$$

and

$$\begin{aligned}
&\mathbf{P}^{A_2}[\text{a max. path generated by } A \text{ contains at most three actions } snd(0)] \leq \\
&\mathbf{P}^{A_2}[\text{a max. path generated by } A \text{ contains finitely many actions } snd(0)] = \\
&\mathbf{P}^{A_2}\left[\bigcup_i \text{a max. path generated by } A \text{ contains no } snd(0) \text{ after position } i\right] = \\
&\lim_{i \to \infty} \mathbf{P}^{A_2}[\text{a max. path generated by } A \text{ contains no } snd(0) \text{ after position } i] = \\
&\lim_{i \to \infty} 0 = 0.
\end{aligned}$$

The third step in this computation follows easily from the definition of probability space.

The trace distribution of an adversary

Now, the trace distribution H generated by an adversary A is obtained by removing all states and internal actions from the probability space associated to A. The probability on a set of traces X is now the probability $\mathbf{P}_H[X]$ that A generates a maximal path with a trace in X.

Definition 2.3.16 The *trace distribution* H of an adversary A, denoted by $trdistr(A)$, is the probability space given by

1. $\Omega_H = {\rm Act}_{\mathcal{A}}{}^* \cup {\rm Act}_{\mathcal{A}}{}^\infty$,
2. $\mathcal{F}_H$ is the smallest σ-field that contains the sets $\{C_\beta \mid \beta \in {\rm Act}_{\mathcal{A}}{}^*\}$, where $C_\beta = \{\beta' \in \Omega_H \mid \beta \sqsubseteq \beta'\}$,
3. $\mathbf{P}_H$ is given by $\mathbf{P}_H[X] = \mathbf{P}_A[\{\pi \in \Omega_A \mid trace(\pi) \in X\}]$ for all $X \in \mathcal{F}_H$.

Standard measure theory arguments [Hal50] together with the fact that the function $trace$ is measurable ensure that $(\Omega_H, \mathcal{F}_H, \mathbf{P}_H)$ is well-defined. We denote the set of trace distributions of $\mathcal{A}$ by $trdistr(\mathcal{A})$.

Example 2.3.17 Consider the trace distribution H of adversary A_2 from Example 2.3.7 again. The sets Ω_H and $\mathcal{F}_H$ need no further explanation. The probability on the set $\{\pi \mid trace(\pi) = snd(1)\}$, i.e. the maximal paths whose trace starts with $snd(1)$, is given as follows.

$$\begin{aligned}\mathbf{P}_H[C_{snd(1)}] = {} & \mathbf{P}_{A_2}[C_{\langle \varepsilon\, snd(1)\, \mu^1_{100}\, 1\rangle}] \\ & + \mathbf{P}_{A_2}[C_{\langle \varepsilon\, snd(1)\, \mu^1_{100}\, \varepsilon\rangle}] \\ & + \mathbf{P}_{A_2}[C_{\langle \varepsilon\, snd(1)\, \mu^1_{200}\, 1\rangle}] \\ & + \mathbf{P}_{A_2}[C_{\langle \varepsilon\, snd(1)\, \mu^1_{200}\, \varepsilon\rangle}] \\ & = \tfrac{1}{4}\cdot\tfrac{1}{100} + \tfrac{1}{4}\cdot\tfrac{99}{100} + \tfrac{1}{4}\cdot\tfrac{1}{200} + \tfrac{1}{4}\cdot\tfrac{199}{200} = \tfrac{1}{2}\end{aligned}$$

2.3.3 Alternative Interpretations for PAs

The theory of PAs takes the standpoint that the behavior of a PA is obtained by resolving the nondeterministic choices first and by then resolving the probabilistic choices. This view is widely adopted, but not the only one. We briefly discuss two different interpretations of PAs.

The work [NCI99] resolves the probabilistic choices first and then the nondeterministic ones. This means that, for each outgoing transition of a state, one of the probabilistic alternatives in the target of the transition is chosen. Since the resolution of the probabilistic choices can be history–dependent, this yields a tree. Then a probability space can be associated to this process along the same lines as we did: The σ–field is generated by a collection of cones, which now contain trees extending a certain finite tree and the probability measure extends the probability on the cones. (The latter are obtained by multiplying the probabilities on the alternatives chosen in this tree.)

This approach leads to a completely different notion of external behavior of a PA, which assigns the same behavior to certain PAs with different trace distributions and which also assigns a different behavior to certain PAs with the same trace distribution. Therefore, this approach is considered controversial.

Secondly, the interpretation of Andova [And99b, And99a] is that the probabilistic resolution is internal to a process and therefore one cannot say when exactly the nondeterministic choices are resolved until one sees an action of the system. Andova works in the alternating model and, although her work is concerned more with bisimulation relations than with trace distributions, one can say that her approach differs on the probabilistic states, but boils down to the same for the nondeterministic states.

2.4 Implementation Relations for PAs

A common approach in verification of concurrent systems is to describe both the implementation of a system and its specification by automata. An *implementation relation* then expresses when one automaton is a correct implementation of another. There are many different implementation relations around for various types of automata, including timed, hybrid and probabilistic ones. For NAs, the *trace inclusion* relation, denoted by $\sqsubseteq_{\mathrm{TR}}$, is often used. This relation defines $\mathcal{A}$ to be a correct implementation of $\mathcal{B}$ if and only if $trace(\mathcal{A}) \subseteq trace(\mathcal{B})$. Trace inclusion is one of the simplest implementation relations and many others are based on it. Trace inclusion preserves *safety properties*, i.e. if $\mathcal{A} \sqsubseteq_{\mathrm{TR}} \mathcal{B}$ then $\mathcal{A}$ meets all safety properties imposed by the specification $\mathcal{B}$. Safety properties typically state that "some bad thing never happens," for instance, that an error never occurs or that the temperature in a microwave oven remains within the safety bounds.

There are two properties, viz. transitivity and substitutivity, which are crucial for the applicability of an implementation relation. These are important, because they allow a complex verification or implementation task to be chopped up in smaller ones. Recall that a relation $\sqsubseteq$ is *transitive* if $\mathcal{A}_1 \sqsubseteq A_2$ and $\mathcal{A}_2 \sqsubseteq A_3$ imply $\mathcal{A}_1 \sqsubseteq A_3$. A relation $\sqsubseteq$ is *substitutive* if $\mathcal{A}_1 \sqsubseteq \mathcal{B}_1$ and $\mathcal{A}_2 \sqsubseteq \mathcal{B}_2$ imply $\mathcal{A}_1 \parallel \mathcal{A}_2 \sqsubseteq \mathcal{B}_1 \parallel \mathcal{B}_2$, where $\parallel$ is the parallel composition operator introduced in Definition 2.2.14. Transitivity is essential because it allows for hierarchical design and verification. This means that from a specification $\mathcal{B}$ one derives an implementation $\mathcal{A}$ via several intermediate specifications $\mathcal{I}_1, \mathcal{I}_2, \ldots \mathcal{I}_n$, which are successively more operational. If one ensures that $\mathcal{A} \sqsubseteq \mathcal{I}_n \sqsubseteq \mathcal{I}_{n-1} \sqsubseteq \ldots \mathcal{I}_1 \sqsubseteq \mathcal{B}$, then transitivity ensures that $\mathcal{A} \sqsubseteq \mathcal{B}$. Substitutivity is crucial because it supports compositional design and verification. This means that the correctness of a composite system follows from the correctness of its components. Hence, we can implement a(n intermediate) specification $\mathcal{B}_1 \parallel \mathcal{B}_2$ by implementing $\mathcal{B}_1$ as $\mathcal{A}_1$ and $\mathcal{B}_2$ as $\mathcal{A}_2$. If the implementation was done correctly (that is, $\mathcal{A}_1 \sqsubseteq \mathcal{B}_1$ and $\mathcal{A}_2 \sqsubseteq \mathcal{B}_2$), then a substitutive relation guarantees $\mathcal{A}_1 \parallel \mathcal{A}_2 \sqsubseteq \mathcal{B}_1 \parallel \mathcal{B}_2$, that is, the whole system is correct.

Since trace distributions are the natural counterparts of traces, one might propose trace distribution inclusion $\sqsubseteq_{\mathrm{TD}}$ as an implementation relation for PAs. The trace distribution equivalence $\equiv_{\mathrm{TD}}$ expresses that two systems have the same external behavior.

Definition 2.4.1 Define the relation $\sqsubseteq_{\mathrm{TD}}$ on PAs by $\mathcal{A} \sqsubseteq_{\mathrm{TD}} \mathcal{B}$ if and only if $trdistr(\mathcal{A}) \subseteq trdistr(\mathcal{B})$. Furthermore, define $\equiv_{\mathrm{TD}}$ by $\mathcal{A} \equiv_{\mathrm{TD}} \mathcal{B}$ if and only if $trdistr(\mathcal{A}) = trdistr(\mathcal{B})$.

It is easy to see that $\sqsubseteq_{\mathrm{TR}}$ and $\sqsubseteq_{\mathrm{TD}}$ are both transitive. However, $\sqsubseteq_{\mathrm{TD}}$ is not substitutive as is shown in the following example. This example is an adaptation from an example by Segala [Seg95b] and is the result of a discussion with Segala. We find this example here more convincing, because the adversaries used here are very natural and, unlike the ones in [Seg95b] do not have unexpected properties, (c.f. Example 2.4.4).

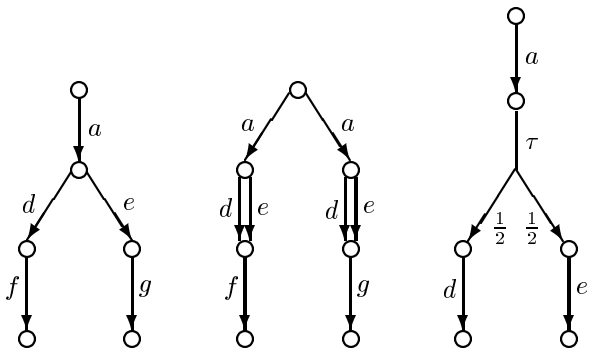


Figure 2.10: The automata $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{B}$ showing that trace distribution inclusion is not compositional

Example 2.4.2 Consider the PAs $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{B}$ in Figure 2.10. It is not difficult to see that $\mathcal{A}_1 \sqsubseteq_{\mathrm{TD}} \mathcal{A}_2$. However, $\mathcal{A}_1 \parallel \mathcal{B} \not\sqsubseteq_{\mathrm{TD}} \mathcal{A}_2 \parallel \mathcal{B}$. The automata $\mathcal{A}_1 \parallel \mathcal{B}$ and $\mathcal{A}_2 \parallel \mathcal{B}$ are represented in Figure 2.11.

Now, consider the adversary A_1 of $\mathcal{A}_1 \parallel \mathcal{B}$ that, in each state having an outgoing transition, schedules this unique outgoing transition with probability one. (The tree representing this adversary has the same structure as the PA $\mathcal{A}_1 \parallel \mathcal{B}$.) It is not difficult to see that there is no adversary of $\mathcal{A}_2 \parallel \mathcal{B}$ with the same trace distribution $H_1 = \mathit{trdistr}(A_1)$: assume that there is one, say A_2. Let $H_2 = \mathit{trdistr}(A_2)$. As $\mathbf{P}_{H_1}[\{adf\}] = \frac{1}{2}$, A_2 should schedule the leftmost transition of $\mathcal{A}_2 \parallel \mathcal{B}$ with probability one. But then $\mathbf{P}_{H_2}[\{aeg\}] = 0$, whereas $\mathbf{P}_{H_1}[\{aeg\}] = \frac{1}{2}$.

Three aspects in the example above are worth noticing. Firstly, when considered as NAs, then we have $\mathcal{A}_1 \sqsubseteq_{\mathrm{TR}} \mathcal{A}_2$. Thus, probabilistic environments (modeled as a PA) can distinguish more than nonprobabilistic environments, even if the automata considered do not contain probabilistic choices.

Secondly, the distinguishing PA $\mathcal{B}$ is very natural, there is nothing tricky going on: The PA $\mathcal{B}$ takes an a–transition, then it decides via an internal probabilistic transition whether to take a d or an e transition.

Nonsubstitutivity seems to be a fundamental property of trace distribution inclusion. Small adaptations to the theory, such as enforcing an I/O distinction on PAs, do not seem to work. In fact, the previous example can easily be transformed into an example for I/O automata.

Since substitutivity is essential in verification of real–life applications, we need a notion of implementation for PAs that is preserved under parallel composition. Segala proposes to simply consider the coarsest (i.e. the largest) precongruence contained in $\sqsubseteq_{\mathrm{TD}}$, denoted by $\sqsubseteq_{\mathrm{PTD}}$. Chapter 5 presents several simulation relations for PAs, which can be used to prove that $\mathcal{A} \sqsubseteq_{\mathrm{PTD}} \mathcal{B}$ for given PAs $\mathcal{A}$ and $\mathcal{B}$. An important result is the alternative characterization of $\sqsubseteq_{\mathrm{PTD}}$ by the principal context $\mathcal{C}$.

Theorem 2.4.3 ([Seg95b]) *Let $\mathcal{C}$ be the PA shown in Figure 2.12, where we suppose that the actions pleft and pright are fresh. Then $\mathcal{A} \sqsubseteq_{\mathrm{PTD}} \mathcal{B}$ iff $\mathcal{A} \parallel \mathcal{C} \sqsubseteq_{\mathrm{TD}} \mathcal{B} \parallel \mathcal{C}$.*

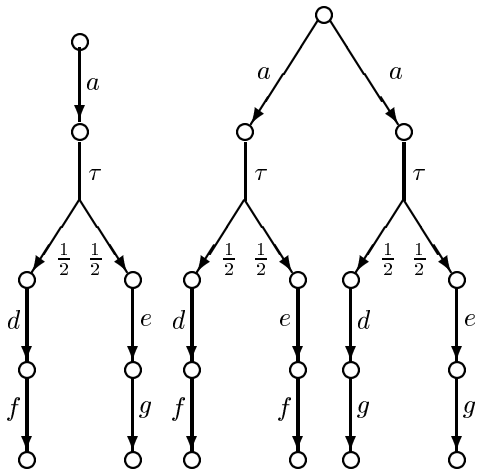


Figure 2.11: The PAs $\mathcal{A}_1 \parallel \mathcal{B}$ and $\mathcal{A}_2 \parallel \mathcal{B}$, showing that trace distribution inclusion is not compositional

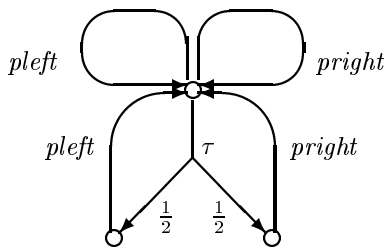


Figure 2.12: Principal context

The relation $\sqsubseteq_{\mathrm{PTD}}$ preserves probabilistic safety properties. These are properties expressing that for a given probability p, "some bad thing happens with a probability smaller than p." For instance, the probability that an error occurs is smaller that 10^{-10} or the probability that the temperature of the microwave is 1% too high is smaller than $\frac{1}{10}$. Moreover, it is transitive and substitutive and therefore suitable as an implementation relation.

One can prove that $\mathcal{A} \sqsubseteq_{\mathrm{TR}} \mathcal{B}$ via simulation relations. Section 2.5 presents two simple kinds of simulations for PAs and Chapter 5 is entirely devoted to simulation and bisimulation relations.

2.4.1 Miscellaneous Remarks on Trace Distributions

This section discusses several technical points with respect to the definition of trace distribution and the relations $\sqsubseteq_{\mathrm{TD}}$ and $\sqsubseteq_{\mathrm{PTD}}$. The reader may skip this section and still understand the rest of this thesis.

The definition of trace distribution

In his definition of trace distribution, Segala takes the completion of the probability space $(\Omega_H, \mathcal{F}_H, \mathbf{P}_H)$ that we used in our definition. The completion is a measure theoretic operation that yields a measure space with several nice mathematical properties. Both conceptually and technically, there is no problem at all in dealing with the definition we gave.

As observed by Segala, the motivation for not taking $trace(\Omega_A)$ as a sample space Ω_H, but dealing with ${V_A}^* \cup {V_A}^\infty$ is that the latter avoids distinguishing between two trace distributions that are the same except that one has more elements whose total probability mass is 0.

However, one might wonder why the σ–field $\mathcal{F}_H$ of a trace distribution is defined as the σ–field generated by the cones and not as the collection of pre–images of sets in $\mathcal{F}_A$. The latter is the set $\{X \in \Omega \mid trace^{-1}(X) \in \mathcal{F}_A\}$. In general, this set contains more elements than $\mathcal{F}_A$ and having more sets whose probability is defined might be useful. However, the set $\mathcal{F}_H$ might have nicer mathematical properties than the set of pre–images.

Compositionality and trace distribution inclusion

Example 2.4.2 demonstrated that $\sqsubseteq_{\mathrm{TD}}$ is not compositional. Therefore, Segala proposed $\sqsubseteq_{\mathrm{PTD}}$ as an implementation relation for PAs. An alternative would be to keep the implementation relation as it is, but to change the parallel composition operator. We believe that the latter is not the right thing to do. Although the parallel composition operator for PAs might have some unexpected properties, we believe it is the implementation relation which causes the troubles with substitutivity. One of these unexpected properties is shown in the following example. It presents an adversary of PAs consisting of two components, where one component looks into the state of a the other. That is, the former component bases its decision on the other one's state, whereas states are supposed internal to a process.

Example 2.4.4 Consider the systems $\mathcal{A}_1$ and $\mathcal{B}$ in Figure 2.10 and the adversary in Figure 2.13. This adversary has quite much power: depending on the state of $\mathcal{B}$, it decides to schedule an f or a g transition in $\mathcal{A}_1$. One might wonder whether this is reasonable, because $\mathcal{A}_1$ and $\mathcal{B}$ are independent systems, whose state space is not supposed to be visible for the environment. On the other hand, one might argue that an adversary can really be the worst

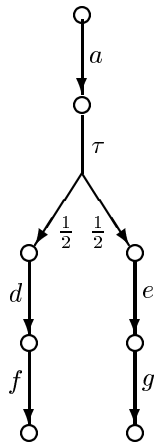


Figure 2.13: An adversary of $\mathcal{A}_1 \parallel \mathcal{B}$ with much power

devil ever, which has knowledge about the entire world and even about dependencies that are unknown to us.

Moreover, also in the nondeterministic case, we can define a scheduler which schedules an a or a b transition in one component, depending on the state of another component. The difference, however, is that in the nondeterministic case, one only sees one trace at the same time (i.e. each behavior corresponds to a single trace) whereas in the probabilistic case one sees many traces at the same time "in superposition," (i.e. each behavior is a trace distribution consisting of several traces with different probabilities).

We believe that, in order to obtain a substitutivity, the implementation relation should be changed, rather than the notion of parallel composition. The reason for this is that the adversary in Example 2.4.2 does not use the strange kind of information as the scheduler in the example above: in order to determine the next step to be taken in a component, it only uses observable information from the other component. So, even when reasonable adversaries are used, trace distribution inclusion is not a precongruence.

The recent work [AHJ01] provides a compositional notion of behavior for probabilistic systems in a variable based setting, which is essentially different ours.

Trace distributions versus traces

Automata can be interpreted as probabilistic automata by simply considering the target state of a transition as a Dirac distribution, see Remark 2.2.8. Then the natural question arises whether two trace equivalent NAs Surprisingly, this is not the case. For infinitely branching NAs, that is, for NAs that can have infinitely many transitions leaving the same state, this result fails.

For finitely branching NAs, the result follows easily by combining the Induction Approximation principle (Theorem 4.3.1) and Theorem 5.3.24.

We conjecture that the result also holds for countably branching PAs. What remains then are the uncountably branching PAs and it is not too much surprising that the combination of uncountably branching and discrete probabilities does not work: discrete probabilities are

tailored for countable structures. For uncountable ones, richer structures, notably probability spaces, have been developed. However, it is important to note that uncountably branching PAs are needed to encode real–time into PAs.

Theorem 2.4.5 *Let $\mathcal{A}$ and $\mathcal{B}$ be automata, which we consider as PAs. If $\mathcal{B}$ is finitely branching, then*

$$\mathcal{A} \sqsubseteq_{\mathrm{TD}} \mathcal{B} \iff \mathcal{A} \sqsubseteq_{\mathrm{TR}} \mathcal{B}.$$

PROOF: (sketch) In this proof, we use the terminology and results from Chapters 5 and 4.

$\Longrightarrow$ Immediate.

$\Longleftarrow$ Assume $\mathcal{A} \sqsubseteq_{\mathrm{TR}} \mathcal{B}$. Since $\mathcal{B}$ is finitely branching, it follows from the Induction Approximation Principle that, in order to prove $\mathcal{A} \sqsubseteq_{\mathrm{TD}} \mathcal{B}$, it suffices to prove that each finite trace distribution of $\mathcal{A}$ is also one of $\mathcal{B}$. Thus, let H be a finite trace distribution of $\mathcal{A}$ and let E be the adversary generating H. By Theorem 5.3.24, we can write E as a convex combination of deterministic adversaries. Let $E = \sum_i p_i \cdot D_i$, where D_i is a deterministic adversary and $p_i \in [0,1]$. Deterministic adversaries generate trace distributions that are isomorphic to finite traces. Therefore, we can find finite adversaries D_i' of $\mathcal{B}$ with $trdistr(D_i) = trdistr(D_i')$. It now follows easily that $D' = \sum_i p_i \cdot D_i'$ is an adversary of $\mathcal{B}$ such that $trdistr(D') = H$.

□

2.5 Step Refinements and Hyperstep Refinements

This section is concerned with proof methods for $\sqsubseteq_{\mathrm{PTD}}$ based on simulation techniques. In the nonprobabilistic case, these techniques have been successfully applied in the verification of a number of systems [DGRV00]. The idea is that simulation relations reduce global reasoning about adversaries and trace distributions (in NAs of executions and traces) to local reasoning in terms of states and transitions. Since the latter is far more easy, the verification effort is reduced significantly.

This section presents two simple simulations for PAs, namely step refinements and hyperstep refinement. They first appeared in [SV99b] and simplify the more complex probabilistic simulation relations introduced in [SL95, Seg95b]. Those ones, and other simulations are studied in Chapter 5.

Probabilistic step refinements

The simplest form of simulations between probabilistic automata that we consider are the probabilistic step refinements. These are mappings from the states of one automaton to the states of another automaton that preserve the initial states and the probabilistic transitions. For the latter, both the external actions and the probabilistic information have to be preserved. We first need an auxiliary definition.

Definition 2.5.1 Let X and Y be sets, $\mu \in \mathrm{Distr}(X)$ and $f : X \to Y$. The *image* of μ under f, notation $f_*(\mu)$, is the probability distribution $\nu \in \mathrm{Distr}(Y)$ satisfying $\nu(y) = \sum_{x \in f^{-1}(y)} \mu(x)$.

Informally, the image distribution $f_*(\mu)$ yields the probability distribution on Y that arises by first picking an element from X according to the distribution μ and by then applying f. If several elements of X are mapped to the same element in Y, then one adds their probabilities.

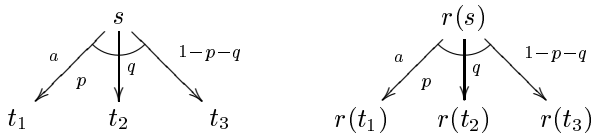
Definition 2.5.2 Let $\mathcal{A}$ and $\mathcal{B}$ be two PAs with $V_{\mathcal{A}} = V_{\mathcal{B}}$. A *probabilistic step refinement* from $\mathcal{A}$ to $\mathcal{B}$ is a function $r : S_{\mathcal{A}} \to S_{\mathcal{B}}$ such that:

1. for all $s \in S^0_{\mathcal{A}}, r(s) \in S^0_{\mathcal{B}}$;
2. for all steps $s \xrightarrow{a}_{\mathcal{A}} \mu$ with $s \in reach_{\mathcal{A}}$, at least one of the following conditions holds:
   (a) $r(s) \xrightarrow{a}_{\mathcal{B}} r_*(\mu)$, or
   (b) $a \in I_{\mathcal{A}} \wedge r(s) \xrightarrow{b}_{\mathcal{B}} r_*(\mu)$, for some $b \in I_{\mathcal{B}}$, or
   (c) $a \in I_{\mathcal{A}} \wedge r_*(\mu) = \{r(s) \mapsto 1\}$.

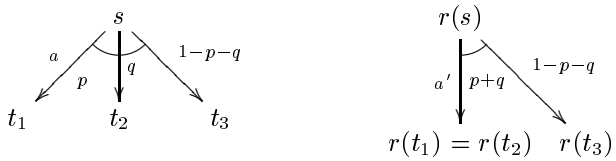
We write $\mathcal{A} \sqsubseteq_{\mathrm{PSR}} \mathcal{B}$ if there is a probabilistic step refinement from $\mathcal{A}$ to $\mathcal{B}$. Note that condition 2(c) is equivalent to $a \in I_{\mathcal{A}} \wedge \forall s'[\mu(s') > 0 \implies r(s') = r(s)]$.

Example 2.5.3 The following diagrams illustrate three typical situations that may occur if r is a probabilistic step refinement from $\mathcal{A}$ to $\mathcal{B}$. The transitions on the left are steps of the probabilistic automaton $\mathcal{A}$, those on the right of $\mathcal{B}$.

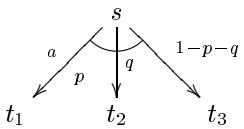
Condition 2a and $r(t_1) \neq r(t_2) \neq r(t_3) \neq r(t_1)$:



Condition 2b, $a \in I_{\mathcal{A}}$, $a' \in I_{\mathcal{B}}$ and $r(t_1) = r(t_2) \neq r(t_3)$:



Condition 2c and $a \in I_{\mathcal{A}}$:



$$r(s) = r(t_1) = r(t_2) = r(t_3)$$

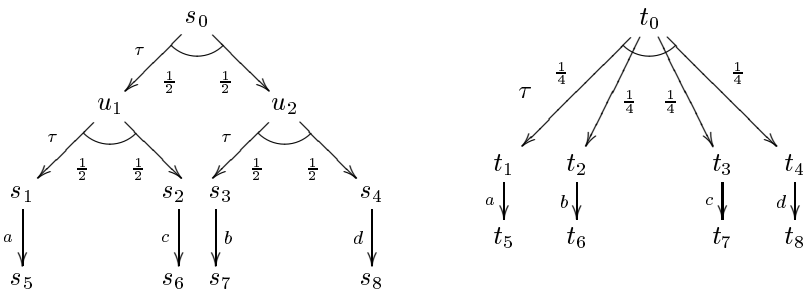


Figure 2.14: The PAs $\mathcal{A}_1$ and $\mathcal{A}_2$.

Probabilistic hyperstep refinements

Probabilistic hyperstep refinements generalize the probabilistic step refinements introduced above. Both step refinements and hyperstep refinements are mappings that preserve initial states and probabilistic transitions in some sense. Whereas probabilistic step refinement map states to states, probabilistic hyperstep refinements map states to distributions over states. This requires a more involved notion of preserving transitions, which is formalized by lifting both the hyperstep refinement and the transition relation. Unlike step refinements, are hyperstep refinements able to contract two subsequent probabilistic choices into a single one, as is shown by the following example (c.f. [Seg95b], Example 8.5.1).

Example 2.5.4 Consider the following PAs $\mathcal{A}_1$ and $\mathcal{A}_2$ in Figure 2.14. Both automata have the same trace distributions: $\mathcal{A}_2$ just contracts the first two τ–transitions of $\mathcal{A}_1$. However, there does not exist a probabilistic step refinement from the first to the second. In order to see this, examine the state u_1. We cannot relate this state to t_0 because then we loose the information that we have chosen for the future a and c actions already. However, u_1 cannot be related to t_1 or t_2 either, because unlike t_1 or t_2 does the state u_1 provide a probabilistic choice between the future a and b actions. The solution provided by the concept of probabilistic hyperstep refinement is to relate u_1 to t_1 and to t_2 with probability $\frac{1}{2}$ each, i.e. to the distribution $\{t_1 \mapsto \frac{1}{2}, t_2 \mapsto \frac{1}{2}\}$.

Definition 2.5.5 Let X, Y be sets and $R \subseteq X \times \mathrm{Distr}(Y)$. The *lifting* of R is the relation $R_{**} \subseteq \mathrm{Distr}(X) \times \mathrm{Distr}(Y)$ given by: $(\mu, \nu) \in R_{**}$ if and only if there is a choice function $r : \mathit{support}(\mu) \to \mathrm{Distr}(Y)$ for R, i.e., a function such that $(x, r(x)) \in R$ for all $x \in \mathit{support}(\mu)$, satisfying

$$\nu(y) = \sum_{x \in \mathrm{supp}(\mu)} \mu(x) \cdot r(x)(y).$$

The idea is that we obtain ν by choosing the probability distribution $r(x)$ with probability $\mu(x)$.

Note that R_{**} is a function if R is so.

Example 2.5.6 Given a probabilistic automaton $\mathcal{A}$ and an action $a \in \mathrm{Act}_{\mathcal{A}}$, we can lift the relation $\xrightarrow{a}$ over $S_{\mathcal{A}} \times \mathrm{Distr}(S_{\mathcal{A}})$ to the relation $\xrightarrow{a}_{**}$ over $\mathrm{Distr}(S_{\mathcal{A}}) \times \mathrm{Distr}(S_{\mathcal{A}})$. For instance, if $s_1 \xrightarrow{a} \mu_1$, $s_2 \xrightarrow{a} \mu_2$ and $s_1 \neq s_2$, then

$$\{s_1 \mapsto \tfrac{1}{3}, s_2 \mapsto \tfrac{2}{3}\} \xrightarrow{a}_{**} \tfrac{1}{3} \cdot \mu_1 + \tfrac{2}{3} \cdot \mu_2.$$

The idea is as follows. Assume that $s_1 \xrightarrow{a} \mu_1$ and $s_2 \xrightarrow{a} \mu_2$ are transitions in $\mathcal{A}$ and that the probability to be in s_1 is $\frac{1}{3}$ and the probability to be in s_2 is $\frac{2}{3}$. Then, after performing the a–transitions mentioned, we choose the next state according to μ_1 with probability $\frac{1}{3}$ and according to μ_2 with probability $\frac{2}{3}$. This yields the distribution $\frac{1}{3} \cdot \mu_1 + \frac{2}{3} \cdot \mu_2$. If there is another a–transition leaving from s_2, say $s_2 \xrightarrow{a} \nu$, then we may take this step instead of $s_2 \xrightarrow{a} \mu_2$. Hence

$$\{s_1 \mapsto \tfrac{1}{3}, s_2 \mapsto \tfrac{2}{3}\} \xrightarrow{a}_{**} \tfrac{1}{3} \cdot \mu_1 + \tfrac{2}{3} \cdot \nu.$$

Note that we do *not* have

$$\{s_1 \mapsto \tfrac{1}{3}, s_2 \mapsto \tfrac{2}{3}\} \xrightarrow{a}_{**} \tfrac{1}{3} \cdot \mu_1 + \tfrac{1}{3} \cdot \mu_2 + \tfrac{1}{3} \cdot \nu.$$

Obviously, we have $s \mapsto 1 \xrightarrow{a} \mu$ whenever $s \mapsto 1 \xrightarrow{a} \mu$.

Example 2.5.7 For the PA $\mathcal{A}_1$ in Figure 2.14, we have that

$$\begin{aligned}\{u_1 \mapsto \tfrac{1}{2}, u_2 \mapsto \tfrac{1}{2}\} &\xrightarrow{\tau}_{**} \tfrac{1}{2} \cdot \{s_1 \mapsto \tfrac{1}{2}, s_2 \mapsto \tfrac{1}{2}\} + \tfrac{1}{2} \cdot \{s_3 \mapsto \tfrac{1}{2}, s_4 \mapsto \tfrac{1}{2}\} \\ &= \{s_1 \mapsto \tfrac{1}{4}, s_2 \mapsto \tfrac{1}{4}, s_3 \mapsto \tfrac{1}{4}, s_4 \mapsto \tfrac{1}{4}\}\end{aligned}$$

Definition 2.5.8 Let $\mathcal{A}$ and $\mathcal{B}$ be probabilistic automaton with $V_{\mathcal{A}} = V_{\mathcal{B}}$. A *probabilistic hyperstep refinement* from $\mathcal{A}$ to $\mathcal{B}$ is a function $h : S_{\mathcal{A}} \to \mathrm{Distr}(S_{\mathcal{B}})$ such that:

1. for all $s \in S^0_{\mathcal{A}}$, $h(s) = \{s' \mapsto 1\}$ for some $s' \in S^0_{\mathcal{B}}$;
2. for all steps $s \xrightarrow{a}_{\mathcal{A}} \mu$ with $s \in \mathit{reach}_{\mathcal{A}}$, at least one of the following conditions holds:
   (a) $h(s) \xrightarrow{a}_{\mathcal{B}^{**}} h_{**}(\mu)$, or
   (b) $a \in I_{\mathcal{A}} \wedge h(s) \xrightarrow{b}_{\mathcal{B}^{**}} h_{**}(\mu)$, for some $b \in I_{\mathcal{B}}$, or
   (c) $a \in I_{\mathcal{A}} \wedge h(s) = h_{**}(\mu)$.

Write $\mathcal{A} \sqsubseteq_{\mathrm{PHSR}} \mathcal{B}$ if there is a probabilistic hyperstep refinement from $\mathcal{A}$ to $\mathcal{B}$.

Example 2.5.9 Consider the function h given by

$$\begin{aligned} s_i &\mapsto t_i, && \text{for all } i \\ u_i &\mapsto \{t_1 \mapsto \tfrac{1}{2}, t_2 \mapsto \tfrac{1}{2}\} && \text{for } i = 1, 2.\end{aligned}$$

Then we have for the lifting h_{**} that

$$\begin{aligned} \{s_i \mapsto 1\} &\mapsto \{t_i \mapsto 1\}, && \text{for all } i \\ \{u_1 \mapsto 1\} &\mapsto \{t_1 \mapsto \tfrac{1}{2}, t_2 \mapsto \tfrac{1}{2}\}, \\ \{u_2 \mapsto 1\} &\mapsto \{t_3 \mapsto \tfrac{1}{2}, t_4 \mapsto \tfrac{1}{2}\}, \\ \{s_i \mapsto \tfrac{1}{2}, s_{i+1} \mapsto \tfrac{1}{2}\} &\mapsto \{t_i \mapsto \tfrac{1}{2}, t_{i+1} \mapsto \tfrac{1}{2}\} && \text{for } i = 1, 3 \\ \{u_1 \mapsto \tfrac{1}{2}, u_2 \mapsto \tfrac{1}{2}\} &\mapsto \tfrac{1}{2} \cdot \{t_1 \mapsto \tfrac{1}{2}, t_2 \mapsto \tfrac{1}{2}\} + \tfrac{1}{2} \cdot \{t_3 \mapsto \tfrac{1}{2}, t_4 \mapsto \tfrac{1}{2}\} \\ &= \{t_1 \mapsto \tfrac{1}{4}, t_2 \mapsto \tfrac{1}{4}, t_3 \mapsto \tfrac{1}{4}, t_4 \mapsto \tfrac{1}{4}\}.\end{aligned}$$

Notice that h_{**} is only partially specified above. Now, it is easy to see that h is a hyperstep refinement from the $\mathcal{A}_1$ to $\mathcal{A}_2$ in Figure 2.14. For this, consider the transitions in $\mathcal{A}_2$. We have $s_0 \xrightarrow{a} \{u_1 \mapsto \frac{1}{2}, u_2 \mapsto \frac{1}{2}\}$ in $\mathcal{A}_1$, and indeed,

$$h_{**}(s_0) \xrightarrow{\tau}_{**} h_{**}(\{u_1 \mapsto \tfrac{1}{2}, u_2 \mapsto \tfrac{1}{2}\}),$$

which is just $\{t_0 \mapsto 1\} \xrightarrow{\tau} \{t_1 \mapsto \frac{1}{4}, t_2 \mapsto \frac{1}{4}, t_3 \mapsto \frac{1}{4}, t_4 \mapsto \frac{1}{4}\}$, which is a lifted transition in $\mathcal{A}_2$. Moreover, we have $u_1 \xrightarrow{\tau} \{t_1 \mapsto \frac{1}{2}, t_2 \mapsto \frac{1}{2}\}$ in $\mathcal{A}_1$ and we have $h_{**}(u_1) = h_{**}(\{s_1 \mapsto \frac{1}{2}, s_2 \mapsto \frac{1}{2}\})$ indeed. The other transitions are now easy.

The following theorem states that probabilistic (hyper-)step refinements are a sound proof method for establishing trace distribution precongruence.

Theorem 2.5.10 *Let $\mathcal{A}$ and $\mathcal{B}$ be probabilistic automata with $V_{\mathcal{A}} = V_{\mathcal{B}}$.*

1. *If $\mathcal{A} \sqsubseteq_{\mathrm{PSR}} \mathcal{B}$ then $\mathcal{A} \sqsubseteq_{\mathrm{PHSR}} \mathcal{B}$.*
2. *If $\mathcal{A} \sqsubseteq_{\mathrm{PHSR}} \mathcal{B}$ then $\mathcal{A} \sqsubseteq_{\mathrm{PTD}} \mathcal{B}$.*

PROOF: For (1), suppose that $\mathcal{A} \sqsubseteq_{\mathrm{PSR}} \mathcal{B}$. Then there exists a probabilistic step refinement r from $\mathcal{A}$ to $\mathcal{B}$. Let $h : S_{\mathcal{A}} \to \mathrm{Distr}(S_{\mathcal{B}})$ be given by $h(s) = \{r(s) \mapsto 1\}$. It is routine to check that h is a probabilistic hyperstep refinement from $\mathcal{A}$ to $\mathcal{B}$. Use that

$$\begin{array}{lcl} h_{**}(\mu) & = & r_*(\mu), \\ s \xrightarrow{a}_{\mathcal{B}} \nu & \iff & \{s \mapsto 1\} \xrightarrow{a}_{\mathcal{B}^{**}} \nu. \end{array}$$

Hence $\mathcal{A} \sqsubseteq_{\mathrm{PHSR}} \mathcal{B}$.

For (2), suppose that $\mathcal{A} \sqsubseteq_{\mathrm{PHSR}} \mathcal{B}$. Then there exists a probabilistic hyperstep refinement h from $\mathcal{A}$ to $\mathcal{B}$. We claim that h is a probabilistic forward simulation in the sense of [Seg95b, SL95]. Now $\mathcal{A} \sqsubseteq_{\mathrm{PTD}} \mathcal{B}$ follows from the soundness result for probabilistic forward simulations, see Section 8.7 in [Seg95b].

For a simple, direct proof of (2) we refer to [SV02a]. □

2.6 Other Models for Probabilistic Systems

Several models have been proposed in the literature to model systems with probabilistic choice. These models can be classified according to the types nondeterministic and probabilistic choices they incorporate: They may either not allow nondeterministic choices at all, they may allow external nondeterminism only, or any kind of nondeterminism. With respect to probabilistic choice, we distinguish between models that deal with discrete probabilistic choice, with exponential probability distributions choice or with arbitrary probabilistic choice. [6] Moreover, various languages have been developed to express and analyze the models conveniently. We mention a few of these.

The reader is referred to Table 2.15 for a schematic overview of the models discussed below.

[6] Note that not every probability distribution is either discrete or continuous, for instance the outcome of first rolling a dice and if 6 comes up throwing a dice is 'hybrid.'

2.6.1 Probabilistic Models without Nondeterminism

First, we discuss several purely probabilistic models. These are models that deal with probabilistic choice but not with nondeterministic choice. They have a long mathematical history. Discrete time Markov chains (DTMCs) have been invented by A.A. Markov (1856 – 1922) and further developed A. N. Kolmogorov (1903 – 1987), who was also a pioneer in Continuous time Markov chains (CTMCs). Semi–Markov chains are much newer. All these models have been applied in a wide variety of applications, including genetics, physics, manufacturing systems and queuing networks, and so on. We refer the reader to [Put94] for a comprehensive study of these models.

This section focuses on models with a countable state space. The ideas for systems with uncountable state spaces are the same, but their formulation requires greater care. We present the models as they traditionally are, without action labels. Under the influence of the automaton model, several labeled variants exists nowadays.

Discrete time Markov chains

A DTMC is basically a PA without nondeterministic choices and without action labels. In each state of the system, there is a unique probabilistic transition specifying the probability to reach a certain target state. Rather than a set of start states, a DTMC specifies a initial probability distribution.

Definition 2.6.1 A *discrete time Markov chain* (DTMC) $\mathcal{A}$ consists of three components:

1. A countable set $S_{\mathcal{A}}$ of *states*,
2. an *initial probability distribution* $\pi \in \mathrm{Distr}(S_{\mathcal{A}})$,
3. a *transition matrix* $\Delta_{\mathcal{A}} : S_{\mathcal{A}} \to \mathrm{Distr}(S_{\mathcal{A}})$.

The semantics of a DTMC is given by a sequence of random variables X_1, X_2, $X_3, \ldots$ on the state space $S_{\mathcal{A}}$, where X_i gives the probability that the DTMC ends up in state s after exactly i transitions. We deviate slightly from the usual terminology in the sense that traditionally the approach is converse: one starts with a sequence of random variables having the so-called Markov Property and shows that this sequence can be represented by a transition matrix and an initial distribution. Basically, the Markov Property for DTMCs states that X_{i+1} depends on X_i, but not X_j for $j < i$. Moreover, this dependence is required to be time–invariant. In other words, $\mathbf{P}[X_{i+1} = s \mid X_i = s']$ does not depend on i. Thus, the sequence $X_1, X_2 \ldots$ is *memoryless*, meaning that the future behavior of the system only depends on the present state and not on the past, i.e. the path leading to this state.

Continuous time Markov chains

A *continuous time Markov chain (CTMC)* specifies a unique probability to move from one state to another, as does a DTMC. In addition, a CTMC assigns a *rate* λ_s to each state s.

This rate determines the amount of time that can be spent in the state s. The latter is called the *delay*, *residence time* or *sojourn time* in s and denoted by D_s. In CTMCs, D_s is (negatively) exponentially distributed with parameter λ_s. This means that the probability to stay in s for at most t time units is given by $1 - e^{-\lambda_s \cdot t}$. As a formula, $\mathbf{P}[D_s < t] = 1 - e^{\lambda_s \cdot t}$.

An exponential distribution with parameter λ has mean value $\frac{1}{\lambda}$. So, the larger the rate of a state, the faster the process leaves this state. Note that the probability to stay in s for *exactly* t time units is 0.

One of the key features of the exponential distributions, which make CTMCs relatively easy to analyze, is the memoryless property. This means that the probability to stay in a state s for at most another t time units does not depend on the amount of time that has already been spent in s, say t'. Mathematically, this is expressed by $\mathbf{P}[D_s < t + t' \mid D_s > t'] = \mathbf{P}[D_s < t] = 1 - e^{-\lambda_s \cdot t}$. Thus, in a CTMC, the entire future behavior, that is the forthcoming states and the future timing behavior, depend only on the present state and not on past states or on the amount of time already spent in this state.

Definition 2.6.2 A *continuous time Markov chain* (CTMC) $\mathcal{A}$ consists of four components:

1. A countable set $S_{\mathcal{A}}$ of *states*,
2. an *initial probability distribution* $\pi \in \mathrm{Distr}(S_{\mathcal{A}})$,
3. a *transition matrix* $\Delta_{\mathcal{A}} : S_{\mathcal{A}} \to \mathrm{Distr}(S_{\mathcal{A}})$.
4. a *rate function* $\lambda_{\mathcal{A}} : S_{\mathcal{A}} \to \mathbb{R}^{>0}$.

The semantics of a CTMC is given by a family of random variables $\{X_t\}_{t \in \mathbb{R}^{>0}}$, where X_t yields the distributions of being in state s at time t. Also here, we deviate from the tradition to start with the random variables.

Sometimes, the transition matrix and the rate function are integrated into a single function $\overline{\lambda} : S_{\mathcal{A}} \times S_{\mathcal{A}} \to \mathbb{R}^{>0}$ by setting $\overline{\lambda}(s, s') = \Delta(s, s') \cdot \lambda(s)$ for $s \neq s'$. Note that the original rates and probabilities can be derived by $\lambda(s) = \sum_{t' \in S_{\mathcal{A}}} \overline{\lambda}(s, t')$ and $\Delta(s, t) = \frac{\overline{\lambda}(s,t)}{\sum_{t' \in S_{\mathcal{A}}} \overline{\lambda}(s,t')}$. Then $\overline{\lambda}(s, s')$ expresses that the time to move from s to s' is exponentially distributed with parameter $\lambda(s, s')$.

Stochastic Petri Nets (SPNs) constitute another model based on exponential distributions and can be translated into CTMCs.

Semi–Markov chains

Semi–Markov chains are similar to CTMCs, except that the sojourn time associated to a state can have an arbitrary distribution. As a consequence, the *timing behavior* in a SMC is not memoryless anymore. The remaining time to be spent in a state may depend on the time already spent in there, but not depend on (the time spent in) previous states. In other words, the *state behavior* of a SMC is still memoryless.

Definition 2.6.3 A *Semi–Markov chain* (SMC) $\mathcal{A}$ consists of four components:

1. A set $S_{\mathcal{A}}$ of *states*,
2. an *initial probability distribution* $\pi \in \mathrm{Distr}(S_{\mathcal{A}})$,
3. a *transition matrix* $\Delta_{\mathcal{A}} : S_{\mathcal{A}} \to \mathrm{Distr}(S_{\mathcal{A}})$.
4. a function $F_{\mathcal{A}} : S_{\mathcal{A}} \to (\mathbb{R} \to [0, 1])$ that assigns an arbitrary *cumulative distribution function* to each state.

As before, the transition matrix and sojourn time distributions can be combined into a single function $\overline{F}_{\mathcal{A}} : S_{\mathcal{A}} \times S_{\mathcal{A}} \to (\mathbb{R} \to [0,1])$ and the semantics of a SMC can be given as a sequence of random variables $\{X_t\}_{t\in\mathbb{R}}$.

A model that can be used to specify SMCs in a more syntactic way are *Generalized Stochastic Petri Nets (GSPNs)* [MCB84]. One can also use *Generalized Semi–Markov Processes* to specify systems with arbitrary distributions. This model is more oriented to system simulation [Gly89, Cas93].

Properties of purely probabilistic models

Unlike for models with probabilistic and nondeterministic choice, where only bounds on the probabilities can be given, allows the purely probabilistic nature of DTMCs, CTMCs and SMCs one to calculate the exact probabilities on many events. A long research tradition in these models has put forward algebraic, analytical and numerical techniques for doing so.

A central issue in these models is *steady state analysis*. Under certain weak assumptions, the probability $\mathbf{P}[X_t = s]$ of being in state s at time t (or after t steps in a DTMC) converges to a limit when t goes to infinity. This means that on the long run, the behavior of the system hardly changes any more.

Another issue is *transient analysis*, which involves for instance the expected number of steps before a certain state is reached.

As remarked before, nondeterminism is essential to define a notion of asynchronous parallel composition and therefore, this operator is not present in these three models without nondeterministic choice. One can however define a synchronous parallel composition operation in which all components operate in lock step [GJS90, GSST95] or a probabilistic merge operator, in which the component that takes the next step is chosen probabilistically [BBS95].

2.6.2 Probabilistic Models with External Nondeterminism

In models that combine probabilistic choice with external nondeterminism, all outgoing edges of a state have different labels. Thus, given the current state and the next action, the next state is determined by a unique probability distribution. Internal nondeterministic choices, expressed by transitions labeled by internal actions or equally labeled transitions leaving from one state, are not allowed in these models. The advantage of such models is that, unlike in purely probabilistic models, a parallel composition operator can be defined. This allows a large system to be specified by several smaller components. On the other hand, when put in an environment that is purely probabilistic (and each transition synchronizes with the environment), the whole system becomes purely probabilistic. Therefore, it can be mapped to one of the models without nondeterminism and the mentioned analysis techniques can be used.

Markov decision processes

A Markov decision process is basically a PA with external nondeterministic choices only. This model is sometimes called the *reactive model*.

Definition 2.6.4 A *Markov Decision Process* (MDP) $\mathcal{A}$ consists of four components:

1. A set $S_{\mathcal{A}}$ of *states*,

2. A set $\mathrm{Act}_{\mathcal{A}}$ of *actions*,
3. an *initial probability distribution* $\pi \in \mathrm{Distr}(S_{\mathcal{A}})$,
4. a *transition matrix* $\Delta_{\mathcal{A}} : S_{\mathcal{A}} \times \mathrm{Act}_{\mathcal{A}} \to \mathrm{Distr}(S_{\mathcal{A}})$.

Note the difference between a transition matrix above, which is a function, and a transition relation in the definition of PAs.

Probabilistic I/O automata

In models with exponential distributions, there is still a problem to define a parallel composition operator. On the one hand, exponential distributions, being memoryless, are very convenient for modeling transitions that are taken independently. This is so because the composite automaton leaves a state (s,t) whenever one of the component processes takes a transition. Therefore the delay in this state is the minimum of two exponential distributions with parameters λ_s and λ_t. This is again an exponential distribution with parameter $\lambda_s + \lambda_t$.

Problems arise, however, when two transitions have to synchronize. It is reasonable to require that synchronization can take place as soon as the sojourn time of both processes involved has expired. As a consequence, the delay of the synchronized transition is given by the maximum of the exponential distributions. Since the maximum of two exponential distributions is not an exponential distribution, the delay cannot be expressed by a rate.

Several solutions for this problem have been proposed. We find the solution adopted by [WSS97] one of the cleanest. It distinguishes between input and output actions. Input actions are required to be always enabled and, importantly, may be taken before the sojourn time in a state has expired. The choice between output and internal actions is purely probabilistic and these actions can only be taken after the sojourn time has passed. Therefore, a closed system (i.e. a system without input actions) can always be mapped to a (labeled) CTMC.

Definition 2.6.5 A *Probabilistic I/O Automaton* (PIOA) $\mathcal{A}$ consists of five components:

1. A set $S_{\mathcal{A}}$ of *states*,
2. An initial state $S^0_{\mathcal{A}} \in S_{\mathcal{A}}$,
3. A set $\mathrm{Act}_{\mathcal{A}}$ of *actions*, which are partitioned into a set of *internal actions* $I_{\mathcal{A}}$, a set of *input actions* $in_{\mathcal{A}}$ and a set of *output actions* $out_{\mathcal{A}}$. The set $out_{\mathcal{A}} \cup I_{\mathcal{A}}$ contains the *locally controlled actions*.
4. a *transition probability function* $\Delta_{\mathcal{A}} : S_{\mathcal{A}} \times \mathrm{Act}_{\mathcal{A}} \times S_{\mathcal{A}} \to [0,1]$. satisfying
   (a) For all $s \in S_{\mathcal{A}}$ and $a \in in_{\mathcal{A}}$, $\sum_{s' \in S_{\mathcal{A}}} \Delta(s,a,s') = 1$.
   (b) For all $s \in S_{\mathcal{A}}$, $\sum_{a \in I_{\mathcal{A}} \cup out_{\mathcal{A}}} \sum_{s' \in S_{\mathcal{A}}} \Delta(s,a,s') \in \{0,1\}$.
5. a *rate transition relation* $\lambda_{\mathcal{A}} : S_{\mathcal{A}} \times S_{\mathcal{A}} \to \mathbb{R}^{>0}$. The rate of a state can be 0 if there are no outgoing transitions labeled by locally controlled actions.

The reason why the PIOA model can have internal actions is that the system has to leave the current state if the sojourn time has expired. Therefore, there is no choice between remaining in the same state and taking an internal transition is present and thus, internal transitions do not introduce nondeterministic choice.

The semantics of a PIOA is given as a transformation of so–called observables, where an observable measures a particular quantity of input.

Semi–Markov decision processes

Puterman [Put94] discusses Semi–Markov decision processes. These are basically Semi–Markov chains with external nondeterministic choice. The definition below is a simplified version of Puterman's. A special class of SMDPs are those with exponential sojourn times. In our classification, these models fall in the same class as PIOAs.

Definition 2.6.6 A *Semi–Markov decision process* (SMDP) $\mathcal{A}$ consists of four components:

1. A set $S_{\mathcal{A}}$ of *states*,
2. an *initial probability distribution* $\pi \in \mathrm{Distr}(S_{\mathcal{A}})$,
3. a *transition matrix* $\Delta_{\mathcal{A}} : S_{\mathcal{A}} \times \mathrm{Act}_{\mathcal{A}} \to \mathrm{Distr}(S_{\mathcal{A}})$.
4. a function $F_{\mathcal{A}} : S_{\mathcal{A}} \to (\mathbb{R} \to [0,1])$ that assigns an arbitrary *cumulative distribution function* to each state.

Given an adversary A, the behavior of an SMDP is purely probabilistic. Therefore, the semantics of a SMDP under the adversary A can be given by a sequence of random variables $\{X_t^A\}_{t\in\mathbb{R}}$ that describe the probability distribution at time t. The semantics of an SMDP can then be given by the set of sequences of random variables arising from an arbitrary adversary of the system.

2.6.3 Probabilistic Models with Full Nondeterminism

Probabilistic automata

We have already seen that probabilistic automata and variants thereof such as the alternating model combine nondeterministic and discrete probabilistic choice. There are several process algebras such as ACP [And99b, And99a], the probabilistic π calculus and the probabilistic process algebra defined in [Han94] that allow us to specify such models at a more abstract level.

Interactive Markov chains

Interactive Markov Chains [Her99] distinguish between *interactive transitions*, which are as in nondeterministic automata and allow one to specify external and internal nondeterministic choices, and *Markovian transitions*, which specify the rate with which a transition is taken, as in CTMCs. Moreover, internal transitions are taken as soon as they become enabled. This separation allows for a clean notion of parallel composition which implicitly realizes synchronization as a maximum of exponential distributions.

Definition 2.6.7 A *Interactive Markov chain* (IMC) $\mathcal{A}$ consists of five components:

1. A set $S_{\mathcal{A}}$ of *states*,
2. An initial state $S^0_{\mathcal{A}} \in S_{\mathcal{A}}$,
3. A set $\mathrm{Act}_{\mathcal{A}}$ of *actions*,
4. a *transition relation* $\Delta_{\mathcal{A}} \subseteq S_{\mathcal{A}} \times \mathrm{Act}_{\mathcal{A}} \times S_{\mathcal{A}}$.

5. a *rate transition relation* $\lambda_{\mathcal{A}} : S_{\mathcal{A}} \times S_{\mathcal{A}} \to \mathbb{R}$.

Moreover, [Her99] introduces a process algebra for IMC as a high level specification language for IMCs. Several notions of bisimulation for IMCs exists, together with their process algebraic axiomatizations, and these can be decided efficiently. A behavior semantics (such as the set of trace distributions for PAs or the sequence of random variables of DTMCs) is not given in [Her99].

Process algebras that combine exponential distributions with nondeterminism are PEPA [Hil96], EMPA [BDG97] and TIPP [GHR93].

SPADES

SPADES, also written ♤, [D'A99], is a process algebra that allows the definitions of *stochastic automata* (SAs). The sojourn time in a state of an SA is specified via clocks, which can have arbitrary probability distributions. That is, each transitions is decorated with a (possibly empty) set of clocks, expressing that this transition can only be taken if these have expired. Immediately afterwards, all clocks in this set are assigned new values according to their probability distributions.

This also allows for a clean notion of parallel composition, because when two transitions are required to synchronize, one just takes the union of their clock reset sets.

Definition 2.6.8 A *Stochastic Automaton* consists of the following components

1. A set $S_{\mathcal{A}}$ of *states*,
2. A set $\mathrm{Act}_{\mathcal{A}}$ of *actions*,
3. A set $\mathcal{C}$ of *random clocks*,
4. a *transition relation* $\Delta_{\mathcal{A}} \subseteq S_{\mathcal{A}} \times \mathrm{Act}_{\mathcal{A}} \times \mathcal{P}(\mathcal{C}) \times S_{\mathcal{A}}$.
5. a *clock resetting function* $\kappa_{\mathcal{A}} : S_{\mathcal{A}} \to \mathcal{P}(\mathcal{C})$,

where $\mathcal{P}(\mathcal{C})$ denotes the powerset of $\mathcal{C}$.

The semantics of stochastic automata is given in terms of stochastic transition systems. These are transition systems in which the target of a transition can be an arbitrary probability space. A behavior semantics for SAs is not given in [D'A99].

2.6.4 Stochastic and Nondeterministic Timing

All the models mentioned above that deal with continuous distributions (i.e. those in the bottom two rows in Figure 2.15) use these to specify probabilistic choices over time (so–called stochastic timing). The probabilistic choice over the state space in these models remains discrete.

Continuous distributions over the states can be useful in various cases, for instance to describe the trajectory of a particle. For stochastic processes (i.e. the sequences of random variables $\{X_t\}_t$), continuous distributions over the state are well studied, but automaton models describing these are less common.

As mentioned before, the SA model generalizes PAs by specifying a general probability space as the target of a transition. In a similar way, the work [DGJP99] defines a MDP with

	none	external	full nondeterminism
discrete	DTMC	MDP	PAs, GPA, AM pr ACP, pr π–calculus
exponential	CTMC SPN	PIOA	IMC algebra for IMC
general	SMP GSNP		SAs ⏢

Figure 2.15: Classification of probabilistic system models according to the types of nondeterministic and probabilistic choices

continuous state space. By restricting the set of possible target probability spaces, this work enables a natural notion of bisimulation for these processes. Ongoing work in that direction [DP01] concerns the development of a CTMC model with a continuous space.

Another useful direction to extend the models discussed above is nondeterministic timing. This type of timing allows a(n external) nondeterministic choice between several (possibly uncountably many) potential delays, which can be used to specify bounds such as "this delay lies between 1 and 2 seconds." The PA model has been extended in this way, leading to (two different types of) PTAs, one in [Seg95b] and another in [KNSS01].

Similar extensions could be made to the MDP model. For the PIOA, IMC and ⏢ model, the situation is more complicated because these models already deal with stochastic sojourn times and equipping them with nondeterministic time would require new analysis methods to compute for instance expected times. Recently, the modeling language MoDest has been proposed [DHKK01], which combines discrete probabilistic and nondeterministic choices over states with both stochastic and nondeterministic timing.

2.6.5 This Thesis

In this thesis, we are interested in the verification and analysis of randomized, distributed algorithms and probabilistic communication protocols. Then the PA model is a natural choice. The applications often contain discrete probabilistic choices and — as it is the case for non-probabilistic distributed algorithms and communication protocols — external and internal nondeterminism are needed to model implementation freedom, scheduling freedom, incomplete information and the environment.

We use the PA model with minor variations and also the notion of behavior differs in minor details throughout the chapters of this thesis. As a consequence, the basic definitions treated here are briefly recalled at several places in this thesis.

The tables below present an overview of variant models and definitions of behavior we used.

	start states	I/O distinction	internal actions	timing
This chapter	set	no	set	no/yes
Chapter 4	single	no	single	no
Chapter 5	none/single	no	single	no
Chapter 7	set single	yes yes	set single	no/yes yes

Figure 2.16: Overview of the variants of the PA model used in this thesis

chapter	notion of behavior based on
This chapter	adversary over all paths starting in start state
Chapter 4	adversary, starting in the start state
Chapter 5	adversary, defined over all paths in a PA

Figure 2.17: Overview of the variants of the notions of behavior used in this thesis

2.7 Summary

The probabilistic automaton model introduced in this chapter combines discrete probabilistic choice and nondeterministic choice in an orthogonal way. This allows us to define an asynchronous parallel composition operator and makes the model suitable for reasoning about randomized distributed algorithms, probabilistic communication protocols and systems with failing components. PAs subsume nonprobabilistic transition systems, Markov decision processes and Markov chains. Nondeterministic timing can be naturally incorporated in this model, but stochastic time, allowing for continuous probabilistic choice over time, cannot.

The behavior of a PA relies on randomized, partial adversaries. These resolve the nondeterministic choices in the model and replace them by probabilistic ones. By ranging over all possible adversaries, one obtains the set of associated probability spaces of a PA. These, in their turn, yield the set of trace distributions, describing the external behavior of a PA.

The implementation relation proposed for PAs is the trace distribution precongruence. This is the largest precongruence relation contained in the trace distribution inclusion relation. The latter is not a precongruence. The trace distribution precongruence can be characterized alternatively by a principal context. Surprisingly, this context can also distinguish between nonprobabilistic automata that are trace equivalent.

We ended this section with an overview of probabilistic models and classified them according to their treatment of probabilistic choice, nondeterministic choice and the nature of time (nondeterministic or probabilistic).