An introduction to probabilistic automata

Mariëlle Stoelinga *

Department of Computer Engineering
University of California at Santa Cruz, CA 95064, USA
marielle@cse.ucsc.edu

Abstract

This paper provides an elementary introduction to the probabilistic automaton (PA) model, which has been developed by Segala. We describe how distributed systems with discrete probabilities can be modeled and analyzed by means of PAs. We explain how the basic concepts for the analysis of nonprobabilistic automata 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 simulation relations for PAs. Finally, we give an overview of various other state based models that are used for the analysis of probabilistic systems.

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] and serve 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 IEEE 1394 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 go into the differences and similarities between both types of choices later in this paper. As subsume nonprobabilistic automata, Markov chains and Markov decision processes. The PA framework does not provide any syntax, but several syntaxes have been defined on top of it to facilitate the modeling of a system as a PA. 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 subsequent failures large enough? What is the minimal probability that the system responds within 3 seconds?

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, thus keeping system models modular. PAs have been extended with time in order to model and reason about time.

The aim of this paper is to explain the basic ideas behind PAs and their behavior in an intuitive way. Our explanation focuses on the differences and similarities with nonprobabilistic automata.

Organization of the paper This paper is organized as follows. Section 2 introduces the probabilistic automaton model and Section 3 treats its behavior (semantics). Then Sections 4 and 5 are concerned respectively with implementation and simulation relations for PAs. In Section 6, we give several other models dealing with probabilistic choice and finally, Section 7 presents a summary of the paper.

*This paper is a revised version of a Chapter 2 in [Sto02], which was written while the author working at the Computing Science Institute, University of Nijmegen, the Netherlands.

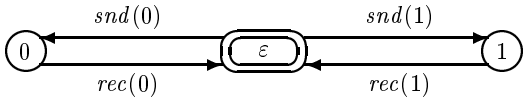

Figure 1: A channel automaton

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 recall the notion of a nonprobabilistic automaton, which also known as a state machine, a labeled transition system, or a state transition system.

2.1 Nonprobabilistic Automata

An *nonprobabilistic automaton*, abbreviated 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 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 2.4.

Example 2.1 Consider the NA in Figure 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.

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

Definition 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. Some 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 the various definitions.

2.1.1 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 nondeterministic behavior is specified 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 including 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 2.13), 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 2.3 In the state ε, the channel NA in Figure 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 2.9 and 2.14.

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.

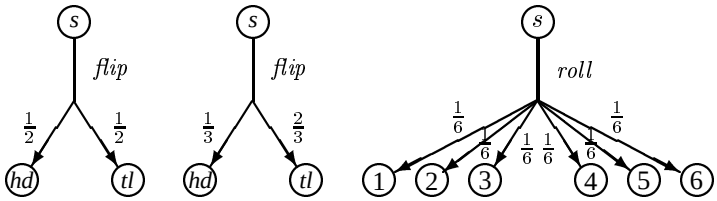

Figure 2: Transitions representing a fair coin flip, an unfair coin flip and a fair dice roll.

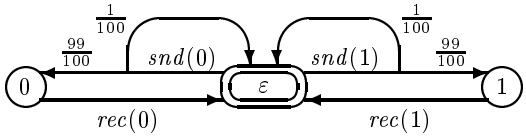

Figure 3: A lossy channel PA

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. Let $\mathrm{Distr}(X)$ denote the set of 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 2.4 Let the set of states be given by s, hd, tl, 1, 2, 3, 4, 5 and 6. The transitions in Figure 2 are respectively given by

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

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 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 2.5 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.

Example 2.6 The PA in Figure 3 represents the same 1–place communication channel as the NA in Figure 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.

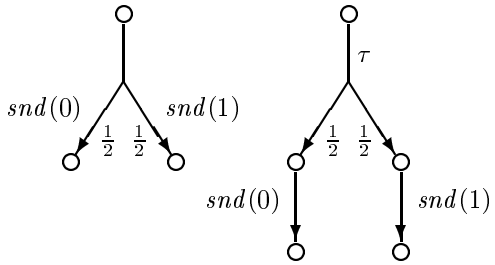

Figure 4: Modeling multi–labeled transitions in a PA

Remark 2.7 Note that each transition in a PA is labeled with a single action. The system depicted on the left in Figure 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 4.

There is, however, a multilabeled version of the PA model, see Section 2.5. That model is technically more complex and, in practice, the PA model is expressive enough.

Remark 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 retaining whether a state can be reached with a positive probability. Precisely, $s \xrightarrow{a} s'$ is a transition in $\mathcal{A}^-$ if and only if there is a transition $s \xrightarrow{a} \mu$ in $\mathcal{A}$ with $\mu(s') > 0$.

2.2.1 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 2.1.1 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 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.

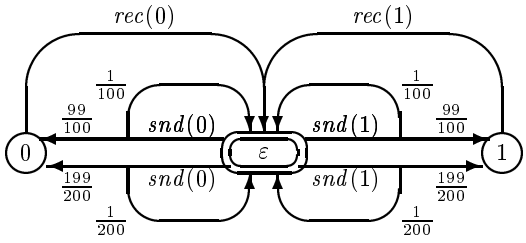

Figure 5: A lossy channel PA with partially unknown probabilities

Example 2.9 Like in Figure 3, the PA in Figure 5 also represents a faulty communication channel. The difference with the PA in Figure 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 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, where the nondeterministic choice represents implementation freedom: It requires that in an implementation 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 by

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

where $x = 100, 200$ and $i = 0, 1$. Thus, the superscripts denote the bit and the subscripts the probability.

Finally, we remark that the philosophical debate on the nature of nondeterminism choice and probability has not ended. In this paper, we take a practical standpoint and postulate that both types of choices are appropriate for describing and predicting certain phenomena in system behavior. For more information on the philosophical issues arising in the area of probability theory, the reader is referred to [Coh89].

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 condition 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 a simplification of the one [Seg95b], which is based on a generalization of Wang's axiom.

Definition 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[1] 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 $s'' \xrightarrow{d-d'}_{\mathcal{A}} s'$.

[1] For simplicity the conditions here are slightly more restrictive than those in [LV96].

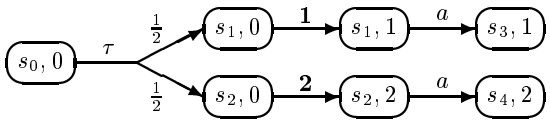

Figure 6: A part of a PTA

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 2.11 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 2.11 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 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 $(s_1, 0) \xrightarrow{0.2} (s_1, 0.2)$ and $(s_1, 0.2) \xrightarrow{0.5} (s_1, 0.7)$, 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 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.

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

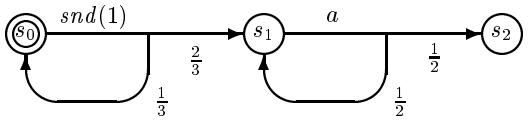

Figure 7: A sender PA

component takes the transition $s_1 \xrightarrow{\tau} \mu_1$ and the other one 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 need to define the probabilistic distribution arising from two independent probabilistic choices. Furthermore, we only take the parallel composition of PAs whose action signatures do not clash.

Notation 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 2.13 We say that two PAs $\mathcal{A}$ and $\mathcal{B}$ are *compatible* if $I_\mathcal{A} \cap \mathrm{Act}_\mathcal{B} = \mathrm{Act}_\mathcal{A} \cap I_\mathcal{B} = \emptyset$. 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 2.14 The system in Figure 7 represents a sender process. Its action set is $\{snd(0), snd(1), a\}$. Figure 8 shows the parallel composition process of this process and the channel process in Figure 3.

2.5 Other Automaton Models with Nondeterminism and Discrete Probabilities

Below, we discuss the GPA model and the alternating model, which are two other automaton models combining discrete probabilities and nondeterminism. Both are equivalent to the PA model in some sense.

General probabilistic automata Segala [Seg95b] introduces a more general notion of probabilistic automata, which we call *general probabilistic automata* (GPAs)[2]. 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 4 on page 5 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). This distinction comes with the requirement that input actions are enabled in every state and

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

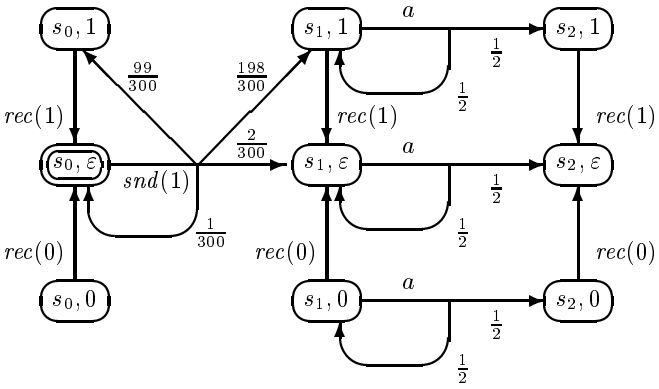

Figure 8: Parallel composition

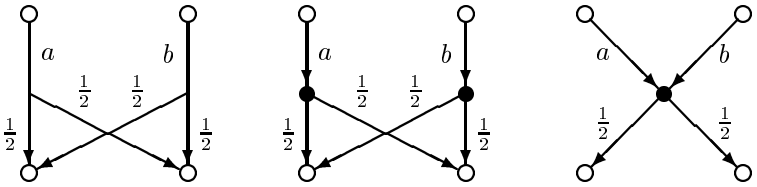

Figure 9: A PA model and two alternating models equivalent to it

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 2.7; 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 bisimulation [LS91]. 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 9.

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 a tree–like structure.

We recall the definition of traces for NAs first.

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 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 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 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 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 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 5. Its trace is $\langle snd(1)\ snd(1)\rangle$.

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 first resolving the nondeterministic choices in the PA (replacing them by probabilistic choices) and then removing the states and the internal actions. Formally, nondeterminism is resolved by *randomized, partial adversaries.* An adversary can be considered as the

equivalent of an execution in an NA. Thus, each adversary yields a different system behavior. To study the probabilistic behavior it generates, each adversary is associated a *probability space*, assigning probabilities to certain sets of paths. The trace distribution of this adversary 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.

3.2.1 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 3 on page 4 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.

3.2.2 Adversaries

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 E schedules the transition $s \xrightarrow{a} \mu$ with probability $E(\pi)(a, \mu)$. The value $E(\pi)(\perp)$ is the probability on an interruption. We let our adversaries start from a fixed start state s_0[3].

[3]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.

Definition 3.5 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

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

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

Example 3.6 Reconsider the lossy communication channel from Example 2.9. Let E_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, E_1 schedules the $rec(i)$–action whenever the system is in i. Then E_1 is defined by

$$\begin{aligned} &E_1(\pi)(snd(1), \mu^1_{100}) = 1, && \text{if } last(\pi) = \varepsilon \\ &E_1(\pi)(rec(i), \{\varepsilon \mapsto 1\}) = 1, && \text{if } last(\pi) = i \end{aligned}$$

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 E_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 E_2 is given by

$$\begin{aligned} &E_2(\pi)(snd(i), \mu^i_j) = \tfrac{1}{4}, && \text{if } last(\pi) = \varepsilon \\ &E_2(\pi)(rec(i), \{\varepsilon \mapsto 1\}) = 1, && \text{if } last(\pi) = i \end{aligned}$$

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

The adversary E_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

$$\begin{aligned} &E_3(\pi)(snd(1), \mu^1_{100}) = \tfrac{1}{3}, && \text{if } last(\pi) = \varepsilon \\ &E_3(\pi)(snd(1), \mu^1_{200}) = \tfrac{1}{3}, && \text{if } last(\pi) = \varepsilon \\ &E_3(\pi)(rec(i), \{\varepsilon \mapsto 1\}) = \tfrac{2}{3}, && \text{if } last(\pi) = i \\ &E_3(\pi)(\bot) = \tfrac{1}{3}, \end{aligned}$$

and 0 otherwise.

Remark 3.7 An adversary E 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 E and whose leaves are the sequences $\pi\bot$, where π is a path satisfying $E(\pi)(\bot) > 0$. The children of a node π are the finite paths $\pi a \mu t$, where $E(\pi)(a, \mu) > 0$ and $\mu(t) > 0$, and the sequence $\pi\bot$ if$E(\pi)(\bot) > 0$. The edge from π to $\pi a \mu t$ is labeled with the probability $E(\pi)(a, \mu) \cdot \mu(t)$ and the edge from π to $\pi\bot$ with the probability $E(\pi)(\bot)$. 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. As said before, nondeterminism is used if we do not wish to specify the factors that influence the choice. Since there is no reason to exclude the possibility that probabilistic factors govern this choice, we need probabilistic adversaries to describe all the possible ways 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 often 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 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.

3.2.3 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 E is obtained by scheduling the transitions described by E and executing them until – possibly – E tells us to stop. The paths obtained in this way are the *maximal paths* in E, 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 E. The associated probability space assigns a probability to certain sets of maximal paths.

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

Definition 3.8 A *path in* E is a finite or infinite path

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

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

Note the difference between paths in a PA and those in an adversary. The former are a superset of the latter: compare Definitions 3.3 and 3.8.

For every finite path π, we can compute the probability $\mathbf{Q}^E(\pi)$ that a path generated by E starts with π. This probability is obtained by multiplying the probabilities that E 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 E is *exactly* π equals $\mathbf{Q}^E(\pi) \cdot E(\pi)(\bot)$.

Definition 3.9 Let $\mathcal{A}$ be a PA and let $s_0 \in S_{\mathcal{A}}$ be a state. Then we define the function $\mathbf{Q}^E : Path^*(s_0, \mathcal{A}) \to [0, 1]$ inductively by

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

Example 3.10 Reconsider the adversaries E_1, E_2 and E_3 from the Example 3.6. The path $\pi = \langle \varepsilon, snd(1), \mu^1_{100}, \varepsilon, snd(1), \mu^1_{100}, 1\rangle$ is a path in E_1, E_2 and in E_3. It is assigned the following probabilities by the adversaries:

$$\mathbf{Q}^{E_1}(\pi) = 1 \cdot \tfrac{1}{100} \cdot 1 \cdot \tfrac{99}{100}, \qquad \mathbf{Q}^{E_2}(\pi) = \tfrac{1}{4} \cdot \tfrac{1}{100} \cdot \tfrac{1}{4} \cdot \tfrac{99}{100} \qquad \mathbf{Q}^{E_3}(\pi) = \tfrac{1}{3} \cdot \tfrac{1}{100} \cdot \tfrac{1}{3} \cdot \tfrac{99}{100}.$$

This path is maximal in E_3, but not in E_1 and E_2. Furthermore, the sequence $\langle \varepsilon, snd(0), \mu^0_{100}, \varepsilon\rangle$ is a path of the system in Figure 5. It is also a path of the adversary E_2, but not of the adversaries E_1 and E_3.

To study the probabilistic behavior generated by an adversary E, 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).

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 E_1 and E_2 from Example 3.6. 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 E_1 and $\frac{1}{2}$ for E_2.

Definition 3.11 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[4] 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 $\mathbf{Q}^E(\pi)$ – 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 $\mathcal{F}_E$ by specifying it on the sets C_π only, see for instance [Hal50] and [Seg95b].

Definition 3.12 The *probability space* associated to a partial adversary E starting in s_0 is the probability space given by

1. $\Omega_E = Path^{max}(E)$,
2. $\mathcal{F}_E$ is the smallest σ–field that contains the set $\{C_\pi \mid \pi \in Path^*(E)\}$, where $C_\pi = \{\pi' \in \Omega_E \mid \pi \sqsubseteq \pi'\}$ and $\sqsubseteq$ denotes the prefix relation on paths,

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

3. $\mathbf{P}_E$ is the unique measure on $\mathcal{F}_E$ such that $\mathbf{P}_E[C_\pi] = \mathbf{Q}^E(\pi)$ for all $\pi \in Path^*(s, \mathcal{A})$.

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

Note that the cone C_π is contained in $\mathcal{F}_E$ for every finite path in E, but that the cone itself contains finite and infinite *maximal* paths. The reason for requiring π to be a path in the adversary E rather than a path in $\mathcal{A}$ is that in this way $\mathcal{F}_E$ 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_E is countable, then $\mathcal{F}_E$ is simply the power set of E. However, if Ω_E is uncountable (and this is the case for which probability spaces have been designed), then the set $\mathcal{F}_E$ 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, but this requires ordinal induction, rather than ordinary induction. Moreover, the construction of a set not being in $\mathcal{F}_E$ 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 $\mathcal{F}_E$.

Example 3.13 The collection $\mathcal{F}_E$ 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 $\mathcal{F}_E$. The set containing the single infinite path π equals

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

Example 3.14 Consider the adversary E_2 from Example 3.6. Then Ω_{E_2} is just the sets of all infinite paths. The set C_π contains the infinite paths extending the path π and $\mathcal{F}_{E_2}$ is the smallest σ–algebra containing those cones. Some values of the function $\mathbf{P}^{E_2}$ are

$$\mathbf{P}^{E_2}[C_{\langle \varepsilon \ snd(0) \ \mu^1_{100} \ 1 \rangle}] = \mathbf{Q}^{E_2}(\langle \varepsilon \, snd(0) \mu^1_{100} 1 \rangle) = \tfrac{1}{4} \cdot \tfrac{99}{100}.$$

and

$$\begin{aligned}
&\mathbf{P}^{E_2}[\text{a max. path generated by } E \text{ contains at most three actions } snd(0)] \leq \\
&\mathbf{P}^{E_2}[\text{a max. path generated by } E \text{ contains finitely many actions } snd(0)] = \\
&\mathbf{P}^{E_2}\left[\bigcup_i \text{a max. path generated by } E \text{ contains no } snd(0) \text{ after position } i\right] = \\
&\lim_{i \to \infty} \mathbf{P}^{E_2}[\text{a max. path generated by } E \text{ contains no } snd(0) \text{ after position } i] = \\
&\lim_{i \to \infty} 0 = 0.
\end{aligned}$$

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

3.2.4 The trace distribution of an adversary

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

Definition 3.15 The *trace distribution* H of an adversary E, denoted by $trdistr(E)$, is the probability space given by

1. $\Omega_H = \mathrm{Act}_{\mathcal{A}}{}^* \cup \mathrm{Act}_{\mathcal{A}}{}^\infty$,
2. $\mathcal{F}_H$ is the smallest σ-field that contains the sets $\{C_\beta \mid \beta \in \mathrm{Act}_{\mathcal{A}}{}^*\}$, where $C_\beta = \{\beta' \in \Omega_H \mid \beta \sqsubseteq \beta'\}$,
3. $\mathbf{P}_H$ is given by $\mathbf{P}_H[X] = \mathbf{P}_E[\{\pi \in \Omega_E \mid trace(\pi) \in X\}]$ 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 3.16 Consider the trace distribution H of adversary E_2 from Example 3.6 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.

$$\begin{aligned}\mathbf{P}_H[C_{snd(1)}] = {} & \mathbf{P}_{E_2}[C_{\langle \varepsilon\, snd(1)\, \mu^1_{100}\, 1\rangle}] \\ & + \mathbf{P}_{E_2}[C_{\langle \varepsilon\, snd(1)\, \mu^1_{100}\, \varepsilon\rangle}] \\ & + \mathbf{P}_{E_2}[C_{\langle \varepsilon\, snd(1)\, \mu^1_{200}\, 1\rangle}] \\ & + \mathbf{P}_{E_2}[C_{\langle \varepsilon\, snd(1)\, \mu^1_{200}\, \varepsilon\rangle}] \\ & = \tfrac{1}{4}\cdot\tfrac{1}{100} + \tfrac{1}{4}\cdot\tfrac{99}{100} + \tfrac{1}{4}\cdot\tfrac{1}{200} + \tfrac{1}{4}\cdot\tfrac{199}{200} = \tfrac{1}{2}\end{aligned}$$

4 Implementation Relations for PAs

4.1 Trace distribution inclusion

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. For NAs, the *trace inclusion* relation, denoted by $\sqsubseteq_{\mathrm{TR}}$, is often used. This means that $\mathcal{A}$ to is considered to be a correct implementation of $\mathcal{B}$ if and only if $traces(\mathcal{A}) \subseteq traces(\mathcal{B})$. Trace inclusion is one of the simplest implementation relations and many others are based on it. Trace inclusion preserves *safety properties*.

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 4.1 For PAs $\mathcal{A}$ and $\mathcal{B}$, define $\mathcal{A} \sqsubseteq_{\mathrm{TD}} \mathcal{B} = trdistr(\mathcal{A}) \subseteq \mathcal{B}$ and $\mathcal{A} \equiv_{\mathrm{TD}} \mathcal{B} = trdistr(\mathcal{A}) = trdistr(\mathcal{B})$.

In order to be useful in verification, an implementation $\sqsubseteq$ should be substitutive with respect to parallel composition. This means that $\mathcal{A}_1 \sqsubseteq \mathcal{A}_2$ implies $\mathcal{A}_1 \parallel \mathcal{B} \sqsubseteq_{\mathcal{A}2} \parallel \mathcal{B}$. Substitutivity guarantees that if $\mathcal{A}_1$ correctly implements $\mathcal{A}_2$, it does so in all environments. Moreover, substitutivity is needed for compositional verification and design, that is, to derive the correctness from a system from the correctness of its components.

As shown below, the relation $\sqsubseteq_{\mathrm{TD}}$ is not substitutive with respect to $\parallel$. The example is the result of a discussion with Segala and is an adaptation of an earlier example by him [Seg95b]. We find the example here more convincing, because these adversaries use only on external information (visible actions) of the other system, whereas the ones in [Seg95b] make different decisions depending on internal information (states) of the environment.

Example 4.2 Consider the PAs $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{B}$ in Figure 10. It is not difficult to see that $\mathcal{A}_1 \sqsubseteq_{\mathrm{TD}} \mathcal{A}_2$. We claim that $\mathcal{A}_1 \parallel \mathcal{B} \not\sqsubseteq_{\mathrm{TD}} \mathcal{A}_2 \parallel \mathcal{B}$. In order to see this, consider the automata $\mathcal{A}_1 \parallel \mathcal{B}$ and $\mathcal{A}_2 \parallel \mathcal{B}$ in Figure 11.

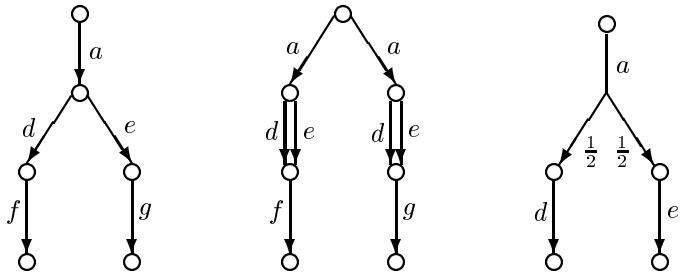

Figure 10: The automata $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{B}$ showing that trace distribution inclusion is not compositional

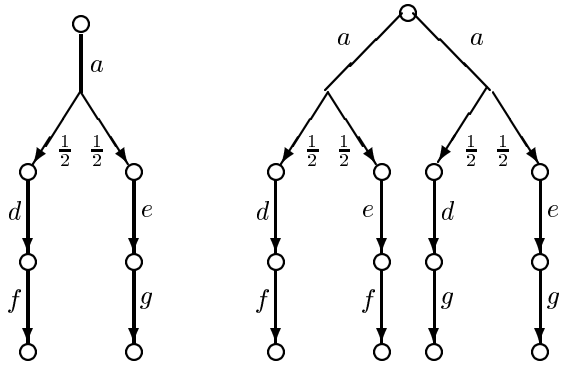

Figure 11: The PAs $\mathcal{A}_1 \parallel \mathcal{B}$ and $\mathcal{A}_2 \parallel \mathcal{B}$, showing that trace distribution inclusion is not compositional

Let E_1 be the adversary 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 $H_1 = trdistr(E_1)$: assume that there is one, say E_2. Let $H_2 = trdistr(E_2)$. As $\mathbf{P}_{H_1}[\{adf\}] = \frac{1}{2}$, E_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 $\mathbf{P}_{H_1}[\{aeg\}] = \frac{1}{2}$.

Note that, 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.

4.2 Trace distribution precongruence

In order to obtain a substitutive implementation relation, Segala proposes to simply consider the coarsest (i.e. the largest) precongruence contained in $\sqsubseteq_{\mathrm{TD}}$, denoted by $\sqsubseteq_{\mathrm{PTD}}$. 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." A direct proof of $\mathcal{A} \sqsubseteq_{\mathrm{PTD}} \mathcal{B}$ from the definitions is usually complicated. Section 5 treats probabilistic simulation relations, which are sound for trace distribution inclusion and much easier to establish. Another important result is the alternative characterization of $\sqsubseteq_{\mathrm{PTD}}$ with the principal context $\mathcal{C}$.

Theorem 4.3 ([Seg95b]) *Let $\mathcal{C}$ be the PA shown in Figure 12, where we suppose that the actions $pleft$ and $pright$ are fresh. Then $\mathcal{A} \sqsubseteq_{\mathrm{PTD}} \mathcal{B}$ iff $\mathcal{A} \parallel \mathcal{C} \sqsubseteq_{\mathrm{TD}} \mathcal{B} \parallel \mathcal{C}$.*

The work [AHJ01] develops notion of behavior for a variable-based, synchronous probabilistic model. The key idea is to retain the variable dependencies when composing two systems. That is, an adversary of

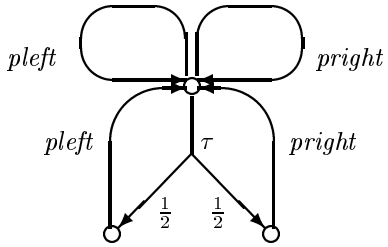

Figure 12: Principal context

the composed system is a product of two adversaries for the component systems and one for the environment.

Remark 4.4 (Trace distributions versus traces) Recall from Remark 2.8 that each NA can be interpreted as a PA by simply considering the target state of a transition as a Dirac distribution. Then the question arises whether two trace equivalent NAs are also trace distribution equivalent. Surprisingly, this is not the case. In [Sto02], it is shown that trace equivalence and trace distribution equivalence coincide for finitely branching NAs, that is, for NAs in which the number of outgoing transitions in a state is finite. It is conjectured that this also holds for countably branching NAs. However, for uncountably branching NAs, the result fails.

5 Probabilistic Simulation and Bisimulation

Simulation and bisimulation relations are a useful tool for system analysis. Both relations are sound for trace-based relations, while they are much easier to establish. Furthermore, bisimulation relations allow one to reduce a system to equivalent but smaller system, which is obtained by replacing each state in a system by its bisimulation equivalence class.

(Bi-)simulation relations have been developed for many different systems, including timed and hybrid systems. This section discusses weak and strong probabilistic (bi-)simulations.

Recall that in the non-probabilistic case a bisimulation is an equivalence R on the state space S such that $(s, s') \in R$ and $s \xrightarrow{a} t$ imply that there is a transition $s' \xrightarrow{a} t'$ and $(s', t') \in R$. Since the target of a transition a PA is a probability distribution rather than a single state, a probabilistic bisimulation has to compare probability distributions μ and μ' when matching transitions $s \xrightarrow{a} \mu$ and $s' \xrightarrow{a} \mu'$ in related states s and s'. The idea is as follows. Since bisimilar states are interchangeable, it does not matter which element within the same bisimulation equivalence class is reached. Hence, a bisimulation relation compares the probability to reach the equivalence classes, rather than the probability to reach a single element. Thus, we lift an equivalence relation R on S to a relation on $\mathrm{Distr}(S)$ in as follows.

Definition 5.1 Let X be a set and let R an equivalence relation on X. Then the lifting of R to $\mathrm{Distr}(S)$, denoted by $\equiv_R$, is defined by

$$\mu \equiv_R \mu' \;=\; \forall C[\mu[C] = \mu'[C]].$$

where C ranges over the set X/R of equivalence classes modulo R.

Definition 5.2 (Strong bisimulation) An equivalence relation $R \subseteq S \times S$ is a *strong simulation* iff for all $(s, s') \in R$

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

The states s and s' are *strongly bisimilar*, notation $s \approx_{sbis} s'$, if there exists a bisimulation R containing (s, s').

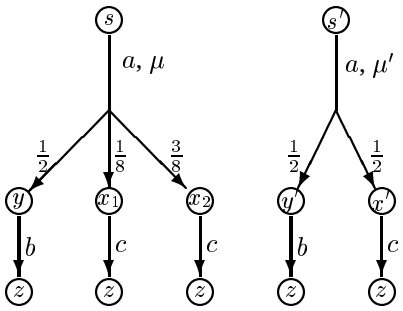

Figure 13: (a) $s \approx_{sbis} s'$

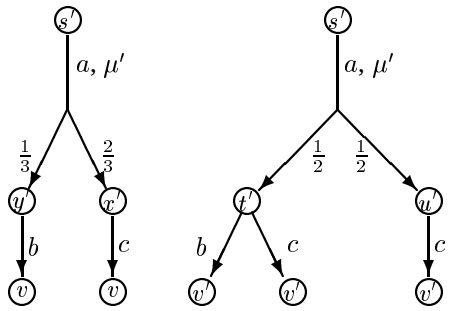

Figure 13: (b) $s \preceq_{ssim} s'$

Example 5.3 The states s and s' in the system in Figure 13(a) are strongly bisimilar and so are y and y' and so are x_1, x_2 and x'. Indeed μ and μ' assign exactly the same probabilities to each of the equivalence classes $\{s, s'\}$, $\{z\}$, $\{y, y'\}$ and $\{x_1, x_2, x\}$.

The situation for simulations is similar, except that we cannot use the relation $\equiv_R$ anymore to compare the target probability distributions of two transitions, because simulation relations need not be equivalences. Instead, $\mu \sqsubseteq_R \mu'$ is established by a weight function. This function quantifies to which extend the probability $\mu(x)$ contributes to the probability $\mu'(x')$ of a related state.

Definition 5.4 (Strong simulation) A relation $R \subseteq S \times S$ is a *strong simulation* iff for all $(s, s') \in R$

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

The states s and s' are *strongly similar*, notation $s \preceq_{ssim} s'$, if there exists a bisimulation R containing (s, s').

Example 5.5 Now, consider the system in Figure 13(b). The relation $R = \{(s, s'), (t, t'), (u, u'), (u, t'), (v, v')\}$ is a strong simulation. For instance, the state u is related to the states t' and u'. We can distribute $\mu(u) = \frac{2}{3}$ over t' and u' by having $wgt(t, t') = 1/3$, $wgt(u, u') = 1/2$. Moreover, take $wgt(u, t') = 1/6$ and $wgt(\cdot, \cdot) = 0$ in the other cases. This shows $\mu \sqsubseteq_R \mu'$.

5.1 Strong Combined (Bi–)simulation

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

Definition 5.6 We write $s \overset{a}{\rightarrowtail} \mu$ iff there is a countable family of transitions $s \xrightarrow{a} \mu_i$, such that μ is a convex combination of the distributions μ_i.

Definition 5.7 (Strong combined (bi–)simulation) A *strong combined simulation* is a relation R on S such that for all $(s, s') \in R$

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

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

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

Example 5.8 Consider the systems in Figure 14. The relation $\{(s_1, t_1), (s_2, t_2), (s_3, t_3)\}$ is a strong combined simulation. The transition $s \xrightarrow{s_1}_a \mu$ corresponds to the combined transition $t_1 \overset{a}{\rightarrowtail} \nu$ with $\nu(t_2) = \nu(t_3) = \frac{1}{2}$ is obtained as a convex combination of the steps $t_1 \xrightarrow{a} \{t_2 \mapsto 1\}$ and $t_1 \xrightarrow{a} \{t_3 \mapsto 1\}$ with $\nu = \frac{1}{2} \cdot \{t_2 \mapsto 1\} + \frac{1}{2} \cdot \{t_3 \mapsto 1\}$. This relation is not a strong simulation because $s \xrightarrow{s_1}_a \mu$ cannot be matched with any of the outgoing transitions in t_1.

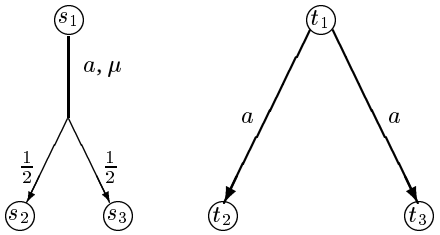

Figure 14: Simulating a probabilistic choice by a nondeterministic one.

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 allow. we consider models that contain either no or only external or external and internal nondeterminism. As far as probability concerns, we consider models with discrete, with exponential and with any kind of probabilities. The reader is referred to Table 15 for a schematic overview of the models discussed below.

6.1 Probabilistic Models without Nondeterministic choice

Probabilistic models without nondeterminism are sometimes called *purely probabilistic models*. Below we discuss discrete time, continuous time, and semi-Markov Chains, which widely used in performance analysis, economics and the social sciences.

Discrete time Markov Chains A *discrete time Markov Chain (DTMC)* is basically an unlabeled PA in which each state has exactly one outgoing probabilistic transition.

Continuous time Markov Chains A *continuous time Markov chain (CTMC)* can be seen as a DTMC in which each state s is assigned a *rate* in $\lambda_s \in \mathbb{R}^{>0}$. The rate λ_s determines the *sojourn time* in s, that is, the amount of time the process can spend in s: the probability to stay in s for at most t time units is given by $1 - e^{-\lambda_s \cdot t}$. 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 for at most another t time units does not depend on the amount of time that has already been spent there.

Semi–Markov chains *Semi–Markov chains (SMCs)* generalize CTMCs, by allowing the sojourn time to be determined by an arbitrary probability distributions.

An advantage of purely probabilistic models over models with nondeterminism is that the probability on a certain event is a single real number, not an interval. A long research tradition in these models has put forward many algebraic, analytical and numerical techniques to compute these probabilities. A disadvantage is that the absence of nondeterminism does not allow an asynchronous parallel composition operator.

6.2 Probabilistic Models with External Nondeterministic Choice

In models that combine probabilistic choice with external nondeterminism, all outgoing edges of a state have different labels. We discuss Markov Decision Processes, Probabilistic I/O automata and Semi–

Markov decision processes. The advantage of these models is that an asynchronous parallel composition operator can be defined, allowing a large system to be split up by several smaller components. Furthermore, when we put the system in an purely probabilistic environment (such that each system transition synchronizes with an environment transition), the whole system becomes purely probabilistic and the analysis techniques for these systems can be used.

Markov Decision Processes A *Markov Decision Process (MDP)* is a PA without internal actions in which each state contains at most one outgoing transition labeled with a.

Probabilistic I/O automata Probabilistic I/O automata (PIOAs) [WSS97] combine external nondeterminism with exponential probability distributions. The memoryless property of these distributions allows a smooth definition of a parallel composition operator, as far as independent transitions are concerned. For synchronizing transitions, the situation is more difficult. Various solutions have been proposed. We find the solution adopted in PIOAs one of the cleanest. This model partitions the visible actions into input and output actions. Output and internal actions are governed by rates. This means that they can only been taken when the sojourn time has expired. Furthermore, the choice between the various output or internal action is purely probabilistic. Input actions, on the other hand, are always enabled and can be taken before the sojourn time has expired.

Semi–Markov decision processes Puterman [Put94] discusses Semi–Markov decision processes (SMDPs), which are basically Semi–Markov chains with external nondeterministic choice.

6.3 Probabilistic Models with Full Nondeterminism

Probabilistic automata We have already seen that probabilistic automata and variants thereof combine nondeterministic and discrete probabilistic choice. Several process algebras, such as ACP [And99b, And99a], the probabilistic π–calculus and the probabilistic process algebra defined in [Han94], allow one to describe such models algebraically.

Interactive Markov chains Interactive Markov Chains (IMCs) [Her99] combine exponential distributions with full nondeterminism. The definition of a parallel composition operator poses the same problems as when one combines exponential distributions with external nondeterminism. IMCs propose an elegant solution by distinguishing between interactive transitions and Markovian transitions. The *interactive transitions* allow one to specify external and internal nondeterministic choices and they synchronize (except for the τ transition) with their environment. The *Markovian transitions* specify the rate with which the transition is taken, similarly to CTMCs.

SPADES Full nondeterminism and arbitrary probability distributions are combined in the process algebra SPADES (also written ♤) and its underlying semantic model *stochastic automata* (SAs). The sojourn time in a state of an SA is specified via clocks, which can have arbitrary probability distributions. More precisely, each transition is decorated with a (possibly empty) set of clocks κ and can only be taken when all clocks in κ have expired. In that case all clocks in κ are assigned new values according to their probability distributions. Stochastic automata in their turn have a semantics in terms of stochastic transition systems. These are transition systems in which the target of a transition can be an arbitrary probability space.

7 Summary

The probabilistic automaton model discussed in this paper 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 incor-

	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	SMDP	SAs Ω

Figure 15: Classification of probabilistic models according to their nondeterministic and probabilistic choices

porated 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 by replacing them by probabilistic ones. When 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).

References

[AD94] R. Alur and D.L. Dill. A Theory of Timed Automata. *Theoretical Computer Science*, 126:183–235, 1994.

[AHJ01] L. de Alfaro, T.A. Henzinger, and R. Jhala. Compositional methods for probabilistic systems. In K. G. Larsen and M. Nielsen, editors, *Proceedings CONCUR 01*, Aalborg, Denmark, volume 2154 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.

[AL92] M. Abadi and L. Lamport. An old-fashioned recipe for real time. In J.W. de Bakker, C. Huizing, W.P. de Roever, and G. Rozenberg, editors, *Proceedings REX Workshop on Real-Time: Theory in Practice*, Mook, The Netherlands, June 1991, volume 600 of *Lecture Notes in Computer Science*, pages 1–27. Springer-Verlag, 1992.

[Alf97] L. de Alfaro. *Formal Verification of Probabilistic Systems*. PhD thesis, Stanford University, 1997.

[Alf99] L. de Alfaro. Computing minimum and maximum reachability times in probabilistic systems. In J.C.M. Baeten and S. Mauw, editors, *Proceedings CONCUR 99*, Eindhoven, The Netherlands, volume 1664 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999.

[And99a] S. Andova. Process algebra with interleaving probabilistic parellel composition. Technical Report CSR 99-04, Eindhoven University of Technology, 1999.

[And99b] S. Andova. Process algebra with probabilistic choice. In *Proceedings of 5th AMAST Workshop on Real-Time and Probabilistic Systems (ARTS'99)* Bamberg, Germany, May 1999, volume 1601 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999.

[Coh80] D.L. Cohn. *Measure Theory*. Birkhaeuser, Boston, 1980.

[Coh89] L.J. Cohen. *An introduction to the philosophy of induction and probability*. Clarendon Press, Oxford, 1989.

[Hal50] P.R. Halmos. *Measure Theory*. Van Nostrand Reinhold Company Inc, New York, 1950.

[Han94] H.A. Hansson. *Time and Probability in Formal Design of Distributed Systems*, volume 1 of *Real-Time Safety Critical Systems*. Elsevier, 1994.

[Her99] H.H. Hermanns. *Interactive Markov Chains*. PhD thesis, University of Erlangen–Nürnberg, July 1999. Available via `http://wwwhome.cs.utwente.nl/~hermanns`.

[HJ94] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):515–535, 1994.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall International, Englewood Cliffs, 1985.

[HP00] O.M. Herescu and C. Palamidessi. Probabilistic asynchronous π-calculus. In J. Tiuryn, editor, *Proceedings of 3rd International Conference on Foundations of Science and Computation Structures (FOSSACS)*, Berlin, Germany, March 2000, volume 1784 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, 2000.

[IEE96] IEEE Computer Society. IEEE Standard for a High Performance Serial Bus. Std 1394-1995, August 1996.

[KNSS01] M. Z. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Automatic verification of real-time systems with discrete probability distributions. *Theoretical Computer Science*, 268, 2001.

[LR81] D. Lehmann and M. Rabin. On the advantage of free choice: a symmetric and fully distributed solution to the dining philosophers problem. In *Proceedings of the 8th ACM Symposium on Principles of Programming Languages*, pages 133–138, 1981.

[LS91] K.G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94:1–28, 1991.

[LT89] N.A. Lynch and M.R. Tuttle. An introduction to input/output automata. *CWI Quarterly*, 2(3):219–246, September 1989.

[LV96] N.A. Lynch and F.W. Vaandrager. Forward and backward simulations, II: Timing-based systems. *Information and Computation*, 128(1):1–25, July 1996.

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

[Seg95a] R. Segala. Compositional trace–based semantics for probabilistic automata. In *Proc. CONCUR'95*, volume 962 of *Lecture Notes in Computer Science*, pages 234–248, 1995.

[Seg95b] R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, June 1995. Available as Technical Report MIT/LCS/TR-676.

[SL95] R. Segala and N.A. Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, 1995.

[Sto02] Mariëlle Stoelinga. *Alea jacta est: verification of probabilistic, real-time and parametric systems*. PhD thesis, University of Nijmegen, the Netherlands, April 2002. Available via `http://www.soe.ucsc.edu/~marielle`.

[Tan81] A.S. Tanenbaum. *Computer networks*. Prentice-Hall International, Englewood Cliffs, 1981.

[WSS97] S.-H. Wu, S.A. Smolka, and E. Stark. Composition and and behaviors of probabilistic I/O automata. *Theoretical Computer Science*, 1-2:1–38, 1997.

[Yi90] Wang Yi. Real-time behaviour of asynchronous agents. In J.C.M. Baeten and J.W. Klop, editors, *Proceedings CONCUR 90*, Amsterdam, volume 458 of *Lecture Notes in Computer Science*, pages 502–520. Springer-Verlag, 1990.