Chapter 9

Probabilistic Timed Automata

9.1 Adding Time

So far we have extended labeled transition systems to handle probabilistic behavior; however, we have not addressed any real-time issue yet. The main objective of this chapter is to add time to probabilistic automata.

Following an approach that Abadi and Lamport [AL91] call the "old-fashioned recipe", we address real-time issues by augmenting probabilistic automata with some structure that models passage of time. In particular, we adopt the solution of Lynch and Vaandrager [LV95], where a *timed automaton* is an ordinary automaton whose actions include the positive real numbers. The occurrence of a real number d means that time d elapses. In addition, a timed automaton of [LV95] is required to satisfy two *trajectory axioms*: the first axiom says that if time d can elapse and immediately afterwards time d' can elapse, then time $d + d'$ can elapse; the second axiom says that if time d can elapse, then there is a *trajectory* that allows us to associate every real time in the interval $[0, d]$ with a state.

The introduction of real-time in probabilistic automata presents two main problems.

1. Time is a continuous entity, and the time that elapses between the occurrence of two separate actions may depend on a probability distribution that is not discrete. For example, the response time of a system may be distributed exponentially. On the other hand, the probability distributions that we allow in the untimed model are only discrete.

2. In the untimed model the parallel composition operator is defined only for simple probabilistic automata. Since time-passage is modeled by actions of $\Re^+$, in a simple probabilistic timed automaton it is not possible to let time pass according to some probability distribution.

The first problem could be solved by removing the requirement that the probability distribution associated with a transition is discrete. However, in such case we would need to redevelop the whole theory, while if we force each probability distribution to be discrete we can reuse most of the results of the untimed model. For this reason, we choose to work only with discrete probability distributions and we defer to further work the extension of the model to non-discrete probability distributions (cf. Section 13.2.1).

For the second problem the reader may object that it originates from the choice of using a distinct time-passage action for each amount of time that elapses in a transition, and thus we may conclude that the problem would be solved by using a unique action that expresses passage of time [LV93b] rather than a different action for every time; however, the problem has deeper roots.

Example 9.1.1 (Problems with probabilistic passage of time) Suppose that from state s_1 a probabilistic timed automaton M_1 lets time pass for 1 second with probability 1/2 and for 2 seconds with probability 1/2 before performing an action a, and suppose that from state s_2 a probabilistic timed automaton M_2 lets time pass for 0.5 seconds with probability 1/2 and for 1.5 seconds with probability 1/2 before performing action a. What is the probability distribution on the time that elapses from state (s_1, s_2) of $M_1 \| M_2$ before performing a? What can we say about the projections of a probabilistic execution of $M_1 \| M_2$? The reader may note the similarity with the problems encountered in the definition of parallel composition for general probabilistic automata (cf. Section 4.3.3). ■

In order to simplify the handling of trajectories, in this thesis we impose an additional restriction on the time-passage transitions of a probabilistic timed automaton; namely, each transition involving time-passage is required to lead to a Dirac distribution. Probabilistic behavior associated with passage of time is allowed only within a probabilistic execution. Even though this timed model may appear to be restrictive, it is sufficiently powerful to analyze non-trivial timed properties of randomized algorithms (cf. Chapter 10).

In the rest of this chapter we concentrate on the definition of the timed model as an extension of the probabilistic automata of Chapter 4. Most of the concepts are extensions of the definitions of [LV95] to the probabilistic framework; the non-trivial part of the chapter is the definition of a *probabilistic timed execution*, where some measure-theoretical complications arise.

9.2 The Timed Model

In this section we define probabilistic timed automata as an extension of the probabilistic automata of Chapter 4, and we extend the timed executions of [LV95] to our framework. Due to the complications that arise in the definition of a probabilistic timed execution, we define probabilistic timed executions in a separate section.

9.2.1 Probabilistic Timed Automata

A *probabilistic semi-timed automaton* M is a probabilistic automaton whose set of external actions includes $\Re^+$, the set of positive reals, and whose transitions with some action in $\Re^+$ are non-probabilistic, i.e., they lead to a Dirac distribution. Actions from $\Re^+$ are referred to as *time-passage actions*, while non-time-passage actions are referred to as *discrete actions*. We let $d, d', \ldots$ range over $\Re^+$ and more generally, $t, t', \ldots$ range over the set $\Re \cup \{\infty\}$ of real numbers plus infinity. The set of *visible* actions is defined by $vis(M) \triangleq ext(M) \setminus \Re^+$.

A *probabilistic timed automaton* is a probabilistic semi-timed automaton M that satisfies the following two axioms.

A1 If $s \xrightarrow{d} s'$ and $s' \xrightarrow{d'} s''$, then $s \xrightarrow{d+d'} s''$.

For the second axiom, we need an auxiliary definition of a *trajectory*, which describes the state changes that can occur during time-passage. Namely, if I is any left-closed interval of $\Re$ beginning with 0, then an *I-trajectory* is a function $\omega : I \rightarrow \mathit{states}(M)$, such that

$$\omega(t) \xrightarrow{t'-t} \omega(t') \text{ for all } t, t' \in I \text{ with } t < t'.$$

Thus, a trajectory assigns a state to each time t in the interval I in a "consistent" manner. We define $\mathit{ltime}(\omega)$, the "last time" of ω, to be the supremum of I. We define $\mathit{fstate}(\omega)$ to be $\omega(0)$, and if I is right-closed, we also define $\mathit{lstate}(\omega)$ to be $\omega(\mathit{ltime}(\omega))$. A trajectory for a transition $s \xrightarrow{d} s'$ is a $[0, d]$-trajectory such that $\mathit{fstate}(\omega) = s$ and $\mathit{lstate}(\omega) = s'$. Now we can state the second axiom.

A2 Each time-passage transition $s \xrightarrow{d} s'$ has a trajectory.

A probabilistic timed automaton M is *simple* if M is a simple probabilistic automaton.

Axioms **A1** and **A2** express natural properties of time: Axiom **A1** says that if time can elapse in two transitions, then it can also elapse in a single transition; Axiom **A2** says that if time d can elapse, then it is possible to associate states with all times in the interval $[0, d]$ in a consistent way.

Example 9.2.1 (The patient construction) A simple way to add time to a probabilistic automaton is to add arbitrary self-loop timed transitions to each state of a probabilistic automaton. Specifically, given a probabilistic automaton M, we define $\mathit{patient}(M)$ to be the probabilistic timed automaton M' such that

1. $\mathit{states}(M') = \mathit{states}(M)$,
2. $\mathit{start}(M') = \mathit{start}(M)$,
3. $\mathit{acts}(M') = \mathit{acts}(M) \cup \Re^+$,
4. $\mathit{trans}(M') = \mathit{trans}(M) \cup \{(s, d, s) \mid s \in \mathit{states}(M), d \in \Re^+\}$.

Thus, $\mathit{patient}(M)$ is like M except that an arbitrary amount of time can elapse between two discrete transitions. It is immediate to verify that $\mathit{patient}(M)$ satisfies axioms **A1** and **A2**. The patient construction was first defined for ordinary automata in [VL92]. ■

Example 9.2.2 (Simple restrictions on time passage) The patient construction does not specify any limitations to the way time can elapse. Sometimes we may want to specify upper and lower bounds to the time it takes for some transition to take place. Such a limitation can be imposed easily by augmenting the states of a probabilistic automaton with variables that express the time limitations that are imposed. As an easy example consider a probabilistic automaton M with a unique state s and a unique discrete transition (s, a, s). Suppose that we want to add time to M and impose that action a occurs once every at least 1 time unit and at most 2 time units. Then the corresponding probabilistic timed automaton M' can be specified as follows.

1. $\mathit{states}(M') = \{(s, l, h) \mid 0 \leq l \leq 1, 0 \leq l \leq h \leq 2\}$,

2. $start(M') = \{(s,0,2)\}$,

3. $acts(M') = \{a\} \cup \Re^+$,

4. $trans(M') = \{((s,0,h),a,(s,1,2)) \mid 0 \leq h \leq 2\} \cup \{((s,l,h),d,(s,l-d,h-d)) \mid d \leq l \leq h\} \cup \{((s,0,h),d,(s,0,h-d))d \leq h\}$.

The variables l and h keep track of the time that must or can elapse before performing a. Time passage decreases both the variables unless they are 0. Action a can occur only when $l = 0$ and leads to a state where $l = 1$. This means that at least 1 time unit must elapse before a can be performed again. No time can elapse if $h = 0$. At thet point the only transition that can be performed is the transition labeled with a. Thus, no more than 2 time units can elapse between the occurrence of two actions a. It is immediate to verify that M' satisfies axioms **A1** and **A2**. ■

9.2.2 Timed Executions

Since a probabilistic timed automaton is also a probabilistic automaton, the executions of the untimed model carry over to the timed case. However, an execution associates states with just a countable number of points in time, whereas the trajectory axiom **A2** allows us to associate states with all real times. Also, our intuition about the executions of a timed system is that visible actions occur at points in time, and that time passes "continuously" between these points. In other words, at each point in time a system is in some state. This leads to the definition of a timed execution.

Timed Executions

A *timed execution fragment* α of a probabilistic timed automaton M is a finite or infinite alternating sequence, $\alpha = \omega_0 a_1 \omega_1 a_2 \omega_2 \cdots$, where

1. Each ω_i is a trajectory and each a_i is a discrete action.

2. If α is a finite sequence then it ends with a trajectory.

3. If ω_i is not the last trajectory in α then its domain is a right-closed interval, and there exists a transition $(lstate(\omega_i), \mathcal{P})$ of M such that $(a, fstate(\omega_{i+1})) \in \Omega$.

A timed execution fragment describes all the discrete changes that occur, plus the evolution of the state during time-passage transitions. If α is a timed execution fragment, then we let $ltime(\alpha)$ denote $\sum_i ltime(\omega_i)$. Note that we allow the case where the domain of the final trajectory is of the form $[0,\infty)$; in this case $ltime(\alpha) = \infty$. We define the initial state of α, $fstate(\alpha)$, to be $fstate(\omega_0)$

A *timed execution* is a timed execution fragment whose first state is a start state.

The timed executions and timed execution fragments of a probabilistic timed automaton can be partitioned into *finite*, *admissible*, and *Zeno* timed executions and timed execution fragments. A timed execution (fragment) α is *finite*, if it is a finite sequence and the domain of its final trajectory is right-closed; a timed execution (fragment) α is *admissible* if $ltime(\alpha) = \infty$; a timed execution (fragment) α is *Zeno* if it is neither finite nor admissible.

There are basically two types of Zeno timed executions: those containing infinitely many discrete actions in finite time, and those containing finitely many discrete actions and for which the time interval associated with the last trajectory is right-open. Thus, Zeno timed executions represent executions of a probabilistic timed automaton where an infinite amount of activity occurs in a bounded period of time. (For the second type of Zeno timed executions, the infinitely many time-passage transitions needed to span the right-open interval should be thought of the "infinite amount of activity".)

We will be interested mostly in the admissible timed executions of a probabilistic timed automaton since they correspond to our intuition that time is a force beyond our control that happens to approach infinity. However, according to our definition of a probabilistic timed automaton, it is possible to specify probabilistic timed automata in which from some states no admissible timed execution fragments are possible. This can be because only Zeno timed execution fragments are possible from that state, or because time cannot advance at all (in which case a *time deadlock* has occurred). Although Zeno timed executions are usually non-desirable, research experience has shown that the analysis of a model would be more complicated if Zeno timed executions are ruled out.

Denote by $\textit{t-frag}^*(M)$, $\textit{t-frag}^\infty(M)$, and $\textit{t-frag}(M)$ the sets of finite, admissible, and all timed execution fragments of M. Similarly, denote by $\textit{t-exec}^*(M)$, $\textit{t-exec}^\infty(M)$, and $\textit{t-exec}(M)$ the sets of finite, admissible, and all timed executions of M.

A *timed extended execution fragment* of M, denoted by α, is either a timed execution fragment of M or a sequence $\alpha'\delta$ where α' is a timed execution fragment of M. Denote by $\textit{t-exec}^*_\delta(M)$ and $\textit{t-exec}_\delta(M)$ the sets of finite and all timed extended executions of M.

Concatenations, Prefixes and Suffixes

If ω is an I-trajectory where I is right-closed, and ω' is an I'-trajectory such that $\textit{lstate}(\omega) = \textit{fstate}(\omega')$, then ω and ω' can be concatenated. The concatenation, denoted by $\omega\omega'$ is the least trajectory (the trajectory with the smallest domain) ω'' such that $\omega''(t) = \omega(t)$ for $t \in I$, and $\omega''(t + \textit{ltime}(\omega)) = \omega(t)$ for $t \in I'$. It is easy to show that ω'' is a trajectory.

Likewise, we may combine a countable sequence of "compatible" trajectories into one: if ω_i is an I_i-trajectory, $0 \leq i < \infty$, where all I_i are right-closed, and if $\textit{lstate}(\omega_i) = \textit{fstate}(\omega_{i+1})$ for all i, then the infinite concatenation $\omega_1\omega_2\cdots$ is the least function ω such that for all i and all $t \in I_i$, $\omega(t + \sum_{j<i} \textit{ltime}(\omega_j)) = \omega_i(t)$. It is easy to show that ω is a trajectory.

A finite timed execution fragment $\alpha = \omega_0 a_1 \omega_1 \cdots a_n \omega_n$ of M and a timed (extended) execution fragment $\alpha' = \omega'_n a_{n+1} \omega_{n+1} \cdots$ of M can be *concatenated* if $\textit{lstate}(\alpha) = \textit{fstate}(\alpha')$. In this case the concatenation, written $\alpha \frown \alpha'$, is defined to be $\alpha'' \triangleq \omega_0 a_1 \omega_1 \cdots a_n (\omega_n \omega'_n) a_{n+1} \omega_{n+1} \cdots$. It is easy to see that α is a timed (extended) execution fragment of M.

The notion of prefix for timed execution fragments and timed extended execution fragments is defined as follows. A timed (extended) execution fragment α of M is a *prefix* of a timed (extended) execution fragment α' of M, written $\alpha \leq \alpha'$, if either $\alpha = \alpha'$ or α is finite and there exists a timed (extended) execution fragment α'' of M such that $\alpha' = \alpha \frown \alpha''$. Likewise, α is a *suffix* of α' if there exists a finite timed execution fragment α'' such that $\alpha' = \alpha'' \frown \alpha$. Denote α by $\alpha' \triangleright \alpha''$.

The *length* of a timed execution fragment α expresses the number of discrete actions in α. Thus, even though α is admissible or Zeno (and thus not finite), its length may be finite.

Formally, define the length of $\alpha = \omega_0 a_1 \omega_1 a_2 \omega_2 \cdots$ as

$$|\alpha| \triangleq \begin{cases} n & \text{if } \alpha \text{ is a finite sequence and ends in } \omega_n \\ \infty & \text{if } \alpha \text{ is an infinite sequence.} \end{cases}$$

9.3 Probabilistic Timed Executions

Since a probabilistic timed automaton is also a probabilistic automaton, it is possible to talk about the probabilistic executions of a probabilistic timed automaton. However, as we have pointed out already for ordinary executions, a probabilistic execution does not describe completely the evolution of a probabilistic timed automaton since it does not allow us to associate every real time with the states that are reached at that time. We need a structure that extends probabilistic executions in the same way as a timed execution extends an execution. A timed execution differs from an execution in two aspects:

1. a timed execution has trajectories to express passage of time;

2. a timed execution does not contain any time-passage actions.

In particular, a timed execution hides the time-passage transitions that are scheduled in an execution to let time pass. Given a trajectory ω, there are infinitely many ways to schedule time-passage transitions to move in time $ltime(\omega)$ from $fstate(\omega)$ to $lstate(\omega)$ ($lstate(\omega)$ is meaningful only if the domain of ω is right-closed); the trajectory ω represents all those possible ways. In a similar way, a probabilistic timed execution should not contain any information on the specific time-passage transitions that are scheduled. Thus, a probabilistic timed execution should be a structure where each state records the past history and each transition contains information on the trajectories that are spanned till the occurrence of the next action. However, it may be the case that there is no next action since the next trajectory is right-open. This would not be a problem except for the fact that from a state there can be uncountably many right-open trajectories that leave even though they are generated by scheduling time-passage transitions according to a discrete probability distribution.

Example 9.3.1 (Uncountable branching from countable branching) Consider a probabilistic automaton M that can increase or decrease a variable x of its state at a constant speed, and suppose that every one time unit the speed of x can be complemented nondeterministically. A valid scheduler $\mathcal{A}$ for M is a scheduler that every one time unit chooses the sign of the speed of x according to a uniform binary distribution. As a result, there are uncountably many trajectories leaving from the start state of M if we use $\mathcal{A}$ to resolve the nondeterminism. Thus, if in a probabilistic timed execution we do not allow for a trajectory to be split into pieces, the probabilistic timed execution of M generated by $\mathcal{A}$ would have a non-discrete probability distribution in its transition relation. ■

To express the fact that we allow only discrete probability distributions on a scheduler, we define probabilistic timed executions in two steps. First we define probabilistic time-enriched executions, which contain closed trajectories and time-passage actions (the time-passage transitions that are scheduled are visible); then, we remove the time-passage actions from probabilistic time-enriched executions to yield probabilistic timed executions.

At the end of this section we show that probabilistic executions, probabilistic time-enriched executions, and probabilistic timed executions are strongly related. Specifically, we show that each probabilistic execution is a *sampling* of a probabilistic time-enriched execution where the information contained in the trajectories is lost, and that each probabilistic time-enriched execution is sampled by some probabilistic execution. Furthermore, we show that it is possible to define an equivalence relation directly on probabilistic time-enriched executions that expresses the fact that two probabilistic time-enriched executions denote the same probabilistic timed execution (they just schedule time-passage transitions in a different way).

All the equivalence results that we prove in this section allow us to use the kind of probabilistic execution that is best suited for each problem. In particular, we use probabilistic timed executions for the theorems of Chapter 10, and we use probabilistic time-enriched executions and probabilistic executions for the results of Chapters 11 and 12. Due to the purely technical content of the comparison section (Section 9.3.3), the reader may focus just on the definitions and on the informal explanations (Sections 9.3.1 and 9.3.2) at a first reading. Most of the concepts are simple modifications of concepts defined for probabilistic executions.

9.3.1 Probabilistic Time-Enriched Executions

Time-Enriched Executions

Let M be a probabilistic timed automaton. A *time-enriched execution fragment* of M is a finite or infinite alternating sequence $\alpha = \omega_0 a_1 \omega_1 a_2 \omega_2 \cdots$ where

1. The domain of ω_0 is $[0,0]$.
2. Each ω_i is a trajectory with a closed domain and each a_i is an action.
3. If a_i is a visible action, then the domain of ω_i is $[0,0]$, and there exists a transition $(\mathit{lstate}(\omega_{i-1}), \mathcal{P})$ of M such that $(a_i, \mathit{fstate}(\omega_i)) \in \Omega$.
4. If a_i is a time-passage action, then the domain of ω_i is $[0, a_i]$ and $\mathit{lstate}(\omega_{i-1}) = \mathit{fstate}(\omega_i)$.

Denote by $\mathit{te\text{-}frag}^*(M)$ and $\mathit{te\text{-}frag}(M)$ the set of finite and all time-enriched execution fragments of M, respectively. The notation for $\mathit{fstate}(\alpha)$, $\mathit{lstate}(\alpha)$ and $\mathit{ltime}(\alpha)$ extends trivially.

A time-enriched execution fragment α contains more information than a timed execution fragment since it is possible to observe what time-passage transitions are used to generate α.

A time-enriched *extended* execution fragment of M is either a time-enriched execution fragment of M or a sequence $\alpha\delta$ where α is a finite time-enriched execution fragment of M. The notation for $\mathit{lstate}(\alpha)$ extends trivially.

A finite time-enriched execution fragment $\alpha = \omega_0 a_1 \omega_1 \cdots a_n \omega_n$ of M and a time-enriched extended execution fragment $\alpha' = \omega'_n a_{n+1} \omega_{n+1} \cdots$ of M can be *concatenated* if $\mathit{lstate}(\alpha) = \mathit{fstate}(\alpha')$. In this case the concatenation is defined to be $\alpha'' \triangleq \omega_0 a_1 \omega_1 \cdots a_n \omega_n a_{n+1} \omega_{n+1} \cdots$, and is denoted by $\alpha \frown \alpha'$. It is easy to see that α'' is a time-enriched extended execution fragment of M. A time-enriched extended execution fragment α of M is a *prefix* of a time-enriched extended execution fragment α' of M, written $\alpha \leq \alpha'$, if either $\alpha = \alpha'$ or α is finite and there exists a time-enriched extended execution fragment α'' of M such that $\alpha' = \alpha \frown \alpha''$. Likewise, α is a *suffix* of α' if there exists a finite time-enriched execution fragment α'' such that $\alpha' = \alpha'' \frown \alpha$. Denote α by $\alpha' \triangleright \alpha''$.

Time-Enriched Transitions

Let $(s, \mathcal{P})$ be a combined transition of M. For each pair (a, s') of Ω, if a is a discrete action, then let $\mathcal{P}_{(a,s')}$ be $\mathcal{D}((a, s'))$; if a is a time-passage action, then let $\mathcal{P}_{(a,s')}$ be a discrete probability distribution of *Probs*(*trajectories*(M, s, a, s')), where *trajectories*(M, s, a, s') denotes the set of trajectories for $s \xrightarrow{a} s'$. The pair $\sum_{(a,s')\in\Omega} P[(a, s')](s, \mathcal{P}_{(a,s')})$ is called a *time-enriched transition* of M.

Thus, a time-enriched transition adds information to a combined transition by specifying what state is reached at each intermediate time. A combined transition gives just the extremes of a trajectory, dropping all the information about what happens in the middle.

Probabilistic Time-Enriched Executions

A *probabilistic time-enriched execution fragment* H of a timed probabilistic automaton M is a fully probabilistic automaton such that

1. $states(H) \subseteq te\text{-}frag^*(M)$
2. for each transition $tr = (q, \mathcal{P})$ of H there is a time-enriched transition $tr' = (lstate(q), \mathcal{P}')$ of M, called the *corresponding time-enriched transition*, such that $\mathcal{P} = q \frown \mathcal{P}'$.
3. each state of H is reachable and enables one transition.

A *probabilistic time-enriched execution* is a probabilistic time-enriched execution fragment whose start state is a start state of M. Denote by *te-prfrag*(M) the set of probabilistic time-enriched execution fragments of M, and by *te-prexec*(M) the set of probabilistic time-enriched executions of M. Also, denote by q_0^H the start state of a generic probabilistic time-enriched execution fragment H.

As for the untimed case, there is a strong relationship between the time-enriched extended execution fragments of a probabilistic timed automaton and the extended executions of one of its probabilistic time-enriched execution fragments. Specifically, let M be a probabilistic timed automaton and let H be a probabilistic time-enriched execution fragment of M. Let q_0 be the start state of H. For each extended execution $\alpha = q_0 a_1 q_1 \cdots$ of H, let

$$\alpha\!\downarrow \;\triangleq\; \begin{cases} q_0 \frown lstate(q_0) a_1 ltraj(q_1) a_2 \cdots & \text{if } \alpha \text{ does not end in } \delta, \\ q_0 \frown lstate(q_0) a_1 ltraj(q_1) a_2 \cdots a_n ltraj(q_n)\delta & \text{if } \alpha = q_0 a_1 q_1 \cdots a_n q_n \delta, \end{cases} \tag{9.1}$$

where $ltraj(q_i)$ denotes the last trajectory of q_i. It is immediate to observe that $\alpha\!\downarrow$ is a time-enriched extended execution fragment of M. For each time-enriched extended execution fragment α of M such that $q_0 \leq \alpha$, i.e., $\alpha = q_0 \frown \omega_0 a_1 \omega_1 \cdots$, let

$$\alpha\!\uparrow\! q_0 \;\triangleq\; \begin{cases} q_0 a_1 (q_0 a_1 \omega_1) a_2 (q_0 a_1 \omega_1 a_2 \omega_2) \cdots & \text{if } \alpha \text{ does not end in } \delta, \\ q_0 a_1 (q_0 a_1 \omega_1) \cdots (q_0 a_1 \omega_1 \cdots a_n \omega_n)\delta & \text{if } \alpha = q_0 a_1 \omega_1 \cdots a_n \omega_n \delta. \end{cases} \tag{9.2}$$

It is immediate to observe that $\alpha\!\uparrow\! q_0$ is an extended execution of some probabilistic timed execution fragment of M. Moreover, the following proposition holds.

Proposition 9.3.1 *Let H be a probabilistic time-enriched execution fragment of a probabilistic timed automaton M. Then, for each extended execution α of H,*

$$(\alpha\downarrow)\uparrow q_0 = \alpha, \tag{9.3}$$

and for each time-enriched extended execution fragment α of M starting with q_0,

$$(\alpha\uparrow q_0)\downarrow = \alpha. \tag{9.4}$$

Events

The probability space $\mathcal{P}_H$ associated with a probabilistic time-enriched execution H is defined as for the untimed case. Thus, Ω'_H is the set of time-enriched extended execution fragments of M that correspond to complete extended executions of H, i.e.,

$$\Omega'_H \triangleq \{\alpha\downarrow \mid \alpha \text{ is a complete extended execution of } H\}, \tag{9.5}$$

where an extended execution α of H is complete iff either α is infinite, or $\alpha = \alpha'\delta$, α' is a finite execution of H, and $\delta \in \Omega^H_{lstate(\alpha)}$. For each finite time-enriched extended execution fragment α of M, let C^H_α denote the *cone*

$$C^H_\alpha \triangleq \{\alpha' \in \Omega_H \mid \alpha \leq \alpha'\}. \tag{9.6}$$

Let $\mathcal{C}_H$ be the set of cones of H. Then define $\mathcal{F}'_H$ to be the σ-field generated by $\mathcal{C}_H$, i.e.,

$$\mathcal{F}'_H \triangleq \sigma(\mathcal{C}_H). \tag{9.7}$$

Define a measure μ on $\mathcal{C}_H$ such that the measure $\mu_H(C^H_\alpha)$ of a cone C^H_α is the product of the probabilities associated with each edge that generates α in H. Formally, let q_0 be the start state of H. If $\alpha \leq q_0$, then

$$\mu_H(C^H_\alpha) \triangleq 1; \tag{9.8}$$

if $\alpha = q_0 \frown \omega_0 a_1 \omega_1 \cdots \omega_{n-1} a_n \omega_n$, then

$$\mu_H(C^H_\alpha) \triangleq P^H_{q_0}[(a_1, q_1)] \cdots P^H_{q_{n-1}}[(a_n, q_n)], \tag{9.9}$$

where for each i, $1 \leq i < n$, $q_i = q_0 \frown \omega_0 a_1 \omega_1 \cdots \omega_{i-1} a_i \omega_i$; if $\alpha = q_0 \frown \omega_0 a_1 \omega_1 \cdots \omega_{n-1} a_n \omega_n \delta$, then

$$\mu_H(C^H_\alpha) \triangleq P^H_{q_0}[(a_1, q_1)] \cdots P^H_{q_{n-1}}[(a_n, q_n)] P_{q_n}[\delta], \tag{9.10}$$

where for each i, $1 \leq i \leq n$, $q_i = q_0 \frown \omega_0 a_1 \omega_1 \cdots \omega_{i-1} a_i \omega_i$. Then the probability measure P'_H is the unique measure on $\mathcal{F}_H$ that extends μ_H, and $\mathcal{P}_H$ is the completion of $\mathcal{P}_H$.

Finite Probabilistic Time-Enriched Executions, Prefixes, Conditionals, and Suffixes

Since a probabilistic time-enriched execution is a fully probabilistic automaton, the definitions of finiteness, prefix, conditional and suffix of Section 4.2.6 extend directly: we just need to define the length of a time-enriched execution fragment α as the number of actions that occur in α.

9.3.2 Probabilistic Timed Executions

We now define the probabilistic timed executions of a probabilistic timed automaton. We use probabilistic time-enriched executions to characterize those transitions that originate from discrete schedulers.

Timed Transitions

A timed transition expresses the result of choosing either an infinite trajectory or a finite trajectory followed by some discrete action at random. However, a timed transition should be the result of scheduling a collection of time-enriched transitions, so that we are guaranteed that it is due to a discrete scheduler. For this reason, we derive a timed transition from the probability distribution associated with a time-enriched probabilistic execution. The derivation proceeds in two steps: first all the time-passage actions are removed and the corresponding trajectories are concatenated; then the resulting structure is truncated at the occurrence of the first action.

Removing Time-Passage Actions. Let $\alpha = \omega_0 a_1 \omega_1 a_2 \omega_2 \cdots$ be a time-enriched execution fragment of a probabilistic timed automaton M. The timed execution represented by α, denoted by $t\text{-}exec(\alpha)$, is the sequence obtained from α by removing all the time-passage actions and by concatenating all the trajectories whose intermediate action is removed.

Let H be a probabilistic time-enriched execution fragment of a probabilistic timed automaton M. Let

$$\Omega \triangleq t\text{-}exec(\Omega_H) \cup limits(t\text{-}exec(\Omega_H)), \tag{9.11}$$

where $limits(t\text{-}exec(\Omega_H))$ is the set of timed executions α of M that end with an open trajectory and such that for each finite prefix α' of α there is an element α'' of $t\text{-}exec(\Omega_H)$ such that $\alpha' \leq \alpha''$. Then, $t\text{-}exec(\mathcal{P}_H)$ denotes the probability space $completion((\Omega, \mathcal{F}, P))$ where $\mathcal{F}$ is the σ-field generated by the cones on Ω, and P is $t\text{-}exec(P_H)$.

The reason for the definition of the sample space of $t\text{-}exec(P_H)$ is mainly technical: we want to establish a relationship between probabilistic time-enriched executions and probabilistic timed executions, and we want the relationship to be preserved by projection of probabilistic timed executions in a parallel composition context. Informally, we are interested in a distribution over trajectories, possibly followed by an action, without keeping any information on how such a distribution is obtained. The elements of the sample space that end with right open trajectories can be affected by the way the transitions are scheduled in a probabilistic time-enriched execution. Moreover, these elements of Ω can create problems for parallel composition. Closing the sample space under limit makes such differences invisible. The reader interested in more details is referred to Sections 9.3.3 and 9.5, and specifically to Examples 9.3.3 and 9.5.1.

Example 9.3.2 (What *t-exec* identifies) Figure 9-1 gives an example of two probabilistic time-enriched executions that are mapped to the same structure by $t\text{-}exec()$. We assume to have two functions ω and ω' defined on the real numbers, and we denote by $\omega_{d,d'}$ the trajectory ω'' with domain $[0, d'-d]$ such that for each $t \leq d'-d$, $\omega''(t) = \omega(t-d)$. A similar notation is used for ω'. ■

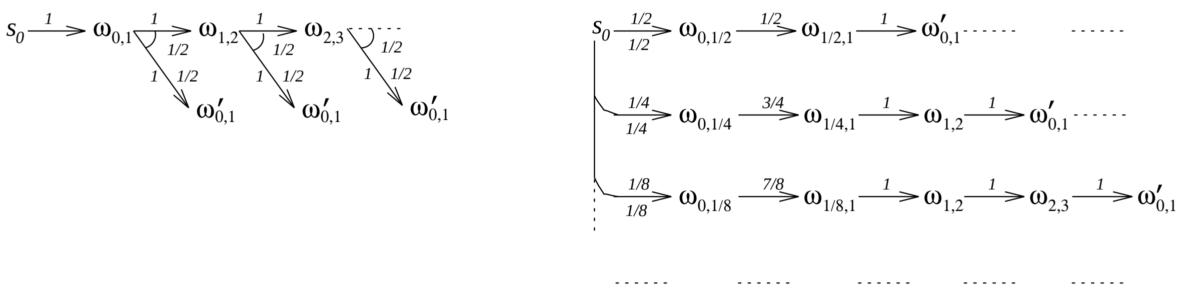

Figure 9-1: Probabilistic time-enriched executions that are mapped to the same structure.

Truncation at the First Action. Let M be a probabilistic timed automaton, and let q be a finite timed execution fragment of M. For each extended timed execution fragment α of M such that $q \leq \alpha$, let

$$truncate_q(\alpha) \triangleq \begin{cases} \alpha & \text{if no action occurs in } \alpha \triangleright q \\ q \frown \omega_0 a_1 fstate(\omega_1) & \text{if } \alpha \triangleright q = \omega_0 a_1 \omega_1 \cdots \end{cases} \tag{9.12}$$

Let H be a probabilistic time-enriched execution fragment of M, and let q be a prefix of the start state of H. Then define $truncate_q(t\text{-}exec(\mathcal{P}_H))$ to be the probability space $\mathcal{P}$ where $\Omega = truncate_q(t\text{-}exec(\Omega_H))$, $\mathcal{F}$ is the σ-field generated by the cones of Ω, and P is the measure $truncate_q(t\text{-}exec(P_H))$.

Timed Transitions. A *timed transition* of M leaving from a state s is a pair $(s, \mathcal{P})$ such that there is a probabilistic time-enriched execution fragment H of M starting in s, and $\mathcal{P} = truncate_s(t\text{-}exec(\mathcal{P}_H))$.

Probabilistic Timed Executions

A *probabilistic timed execution fragment* of a probabilistic timed automaton M, denoted by H, consists of four components.

1. A set $states(H) \subseteq t\text{-}frag_\delta(M)$ of states.
2. A unique start state q_0^H.
3. An action signature $sig(H) = sig(M)$.
4. A transition relation $trans(M)$ consisting of pairs $(q, \mathcal{P})$ such that there exists a timed transition $(lstate(q), \mathcal{P}')$ of M satisfying $\mathcal{P} = q \frown \mathcal{P}'$. Observe that, from the discussion in Section 3.1.5, $q \frown \mathcal{P}'$ is well defined.

Moreover, each state of H is reachable, enables at most one transition, and enables one transition iff it is a finite timed execution fragment of M. A *probabilistic timed execution* of M is a probabilistic timed execution fragment of M whose start state is a start state of M.

An execution of H is a sequence of states of H, $\alpha = q_0 q_1 \cdots$, such that for each i, $q_{i+1} \in \Omega^H_{q_i}$. As for the untimed case, there is a strong correspondence between the timed extended execution fragments of a probabilistic timed execution H of M and the executions of H. Specifically, let

M be a probabilistic timed automaton and let H be a probabilistic timed execution fragment of M. Let q_0 be the start state of H. For each execution $\alpha = q_0 q_1 \cdots$ of H, let

$$\alpha\downarrow \;\triangleq\; \lim_i q_i, \tag{9.13}$$

where the limit is taken under prefix ordering. It is immediate to observe that $\alpha\downarrow$ is a timed extended execution fragment of M. For each timed extended execution fragment α of M such that $q_0 \leq \alpha$, i.e., $\alpha = q_0 \frown \omega_0 a_1 \omega_1 \cdots$, let q_i be $q_0 \frown \omega_0 a_1 \omega_1 \cdots a_i fstate(\omega_i)$, and if $\alpha \triangleright q_0$ is a finite sequence with n discrete actions, let q_{n+1} be α. Then let

$$\alpha\uparrow q_0 \;\triangleq\; q_0 q_1 q_2 \cdots. \tag{9.14}$$

It is immediate to observe that $\alpha\uparrow q_0$ is an execution of some probabilistic timed execution fragment of M. Moreover, the following proposition holds.

Proposition 9.3.2 *Let H be a probabilistic timed execution fragment of a probabilistic timed automaton M. Then, for each execution α of H,*

$$(\alpha\downarrow)\uparrow q_0 = \alpha, \tag{9.15}$$

and for each timed extended execution fragment α of M starting with q_0,

$$(\alpha\uparrow q_0)\downarrow = \alpha. \tag{9.16}$$

Events

The probability space $\mathcal{P}_H$ associated with a probabilistic timed execution fragment H is defined similarly to the untimed case. The set Ω'_H the set of extended timed execution fragments of M that correspond to complete executions of H, where an execution of H is complete iff it is either infinite or it leads to a state that does not enable any transition. The σ-field $\mathcal{F}'_H$ is the minimum σ-field that contains the class of cones of Ω'_H. The measure P'_H is the unique measure that extends the measure defined on cones as follows: if $\alpha = q_0^H \frown \omega_0 a_1 \omega_1 a_2 \cdots a_n \omega_n$, then

$$P'_H[C_\alpha] = P^H_{q_0}[q_1] \cdots P^H_{q_{n-1}}[q_n] P^H_{q_n}[C_\alpha] \tag{9.17}$$

where for each $i \leq n$, $q_i = q_0^H \frown \omega_0 a_1 \omega_1 \cdots a_n fstate(\omega_i)$; if $\alpha = q_0^H \frown \omega_0 a_1 \omega_1 a_2 \cdots a_n \omega_n \delta$, then

$$P'_H[C_\alpha] = P^H_{q_0}[q_1] \cdots P^H_{q_{n-1}}[q_n] P^H_{q_n}[\alpha] \tag{9.18}$$

where for each $i \leq n$, $q_i = q_0^H \frown \omega_0 a_1 \omega_1 \cdots a_n fstate(\omega_i)$. Observe that although there are uncountably many cones in $\mathcal{F}'_H$, every union of cones is expressible as a countable union of disjoint cones. Then, $\mathcal{P}_H$ is the completion of $\mathcal{P}'_H$.

Finite Probabilistic Timed Executions, Prefixes, Conditionals, and Suffixes

Finiteness and prefix are defined similarly to the untimed case, and thus we do not repeat the definitions here.

Conditionals and suffixes differ in a small detail concerning the start state. The reader should observe the similarity of these definitions to those for the untimed case. Also, observe that the properties of conditionals and suffixes (Propositions 9.3.3 and 9.3.4) are the same as for the untimed case. This is what allows us to extend the results for the untimed case directly to the timed case.

Let H be a probabilistic timed execution fragment of a probabilistic timed automaton M, and let q be a prefix of some state of H such that q_0^H is a prefix of q. Then $H|q$ is a new probabilistic execution fragment defined as follows:

1. $states(H|q) = \{q\} \cup \{q' \in states(H) \mid q \leq q'\}$;
2. $start(H|q) = \{q\}$.
3. for each state q' of $H|q$ different from q, $tr_{q'}^{H|q} = tr_{q'}^{H}$.
4. let $\bar{q}$ be the maximum state of H that is a prefix of q. Then, $tr_q^{H|q} = (q, \mathcal{P}_{\bar{q}}^H|C_q)$.

$H|q$ is called a *conditional* probabilistic timed execution fragment. We show later that $H|q$ is a probabilistic timed execution. Observe that $(\Omega_{H|q}, \mathcal{F}_{H|q}, P_{H|q})$ and $(\Omega_H|C_q, \mathcal{F}_H|C_q, P_H|C_q)$ are the same probability space (cf. Section 3.1.8): the sample spaces are the same, the generators are the same, and the probability measures coincide on the generators. Thus, the following proposition is true.

Proposition 9.3.3 *Let H be a probabilistic timed execution fragment of a probabilistic timed automaton M, and let q be a prefix of a state of H such that $q_0^H \leq q$. Then, for each subset E of $\Omega_{H|q}$,*

1. *$E \in \mathcal{F}_{H|q}$ iff $E \in \mathcal{F}_H$.*
2. *If E is an event, then $P_H[E] = P_H[C_q]P_{H|q}[E]$.* ■

Let H be a probabilistic timed execution fragment of a probabilistic timed automaton M, and let q be a prefix of some state of H such that q_0^H is a prefix of q. Then $H \triangleright q$ is a new probabilistic execution fragment defined as follows:

1. $states(H \triangleright q) = \{q' \triangleright q \mid q' \in states(H|q)\}$;
2. $start(H|q) = \{lstate(q)\}$.
3. for each state q' of $H \triangleright q$, $tr_{q'}^{H \triangleright q} = tr_{q \frown q'}^{H|q} \triangleright q$.

$H \triangleright q$ is called a *suffix* of H. It is easy to check that the probability spaces $\mathcal{P}_{H \triangleright q}$ and $\mathcal{P}_{H|q}$ are in a one-to-one correspondence through the measurable function $f : \Omega_{H \triangleright q} \to \Omega_{H|q}$ such that for each $\alpha \in \Omega_{H \triangleright q}$, $f(\alpha) = q \frown \alpha$. The inverse of f is also measurable and associates $\alpha \triangleright q$ with each timed execution α of $\Omega_{H|q}$. Thus, directly from Proposition 9.3.3, we get the following proposition.

Proposition 9.3.4 *Let H be a probabilistic timed execution fragment of a probabilistic timed automaton M, and let q be a prefix of a state of H such that $q_0^H \leq q$. Then, for each subset E of $\Omega_{H \triangleright q}$,*

1. *$E \in \mathcal{F}_{H \triangleright q}$ iff $(q \frown E) \in \mathcal{F}_H$.*

2. *If E is an event, then $P_H[q \frown E] = P_H[C_q] P_{H \triangleright q}[E]$.* ∎

We are left with showing that $H|q$ is well defined. The proof of this apparently obvious fact is not simple and contains several technical details.

Proposition 9.3.5 *Let H be a probabilistic timed execution fragment of a probabilistic timed automaton M, and let q be a prefix of a state of H such that $q_0^H \leq q$. Then, $H|q$ is a probabilistic timed execution fragment of M.*

Proof. We just need to verify that the transition leaving from state q in $H|q$ is a timed transition. Let $\bar{q}$ be the maximum state of H that is a prefix of q. Then, from the definition of a timed transition, there is a probabilistic time-enriched execution fragment $H_{\bar{q}}$ of M such that $\mathcal{P}_{\bar{q}}^H = \bar{q} \frown truncate_{lstate(\bar{q})}(t\text{-}exec(\mathcal{P}_{H_{\bar{q}}}))$. From the definition of $tr_q^{H|q}$, we need to find a probabilistic time-enriched execution fragment H_q of M such that

$$(\bar{q} \frown truncate_{lstate(\bar{q})}(t\text{-}exec(\mathcal{P}_{H_{\bar{q}}})))|C_q = q \frown truncate_{lstate(q)}(t\text{-}exec(\mathcal{P}_{H_q})). \tag{9.19}$$

Let q' be $q \triangleright \bar{q}$. From the definition of $\bar{q}$, q' is just one closed trajectory. Thus, if we build H_q such that

$$(t\text{-}exec(\mathcal{P}_{H_{\bar{q}}}))|C_{q'} = q' \frown t\text{-}exec(\mathcal{P}_{H_q}), \tag{9.20}$$

then Equation 9.19 follows easily using simple properties of *truncate*. Thus, the rest of this proof is dedicated to the construction of an H_q that satisfies (9.20).

Let $q_1, q_2, \ldots$ be an enumeration of the minimal states q'' of H such that $q' \leq t\text{-}exec(q'')$. We distinguish two cases.

1. For each i, $t\text{-}exec(q_i) = q'$.

   The construction for H_q in this case is carried out in the proof of Proposition 9.3.8 (cf. Equation 9.29). We give a forward pointer to avoid too many technical details at this point.

2. There is an i such that $q' < t\text{-}exec(q_i)$.

   We prove this case by reducing the problem to the previous case. That is, we build a new probabilistic time-enriched execution fragment $H'_{\bar{q}}$ such that $t\text{-}exec(\mathcal{P}_{H_{\bar{q}}}) = t\text{-}exec(\mathcal{P}_{H'_{\bar{q}}})$ and such that the minimal states q'' of $H'_{\bar{q}}$ such that $q' \leq t\text{-}exec(q'')$ satisfy $q' = t\text{-}exec(q')$.

   Recall first that q' is a trajectory whose domain is $[0, d]$ for some $d > 0$. Define a collection of finite time-enriched execution fragments $q'_1, q'_2, \cdots$ as follows: for each i, if $t\text{-}exec(q_i) = q'$ then $q'_i = q_i$; otherwise, represent q_i as $\bar{q}_i \frown lstate(\bar{q}_i) d_i \omega_i$, where $\bar{q}_i$ is a state of $H_{\bar{q}}$, and let q'_i be $\bar{q}_i \frown lstate(\bar{q}_i) d_{i,1} \omega_{i,1} d_{i,2} \omega_{i,2} d_{i,3} \omega_{i,3}$ where $\omega_i = \omega_{i,1}\omega_{i,2}\omega_{i,3}$, $t\text{-}exec(\bar{q}_i \frown lstate(\bar{q}_i) d_{i,1} \omega_{i,1} d_{i,2} \omega_{i,2}) = q'$, and the actions $d_{i,1}$ and $d_{i,2}$ are chosen in such a way that for each i $\bar{q}_i \frown lstate(\bar{q}_i) d_{i,1} \omega_{i,1}$ is not a prefix of any of the q'_j's, $j \neq i$. In other words, we split all the q_i's in such a way that a state that corresponds to q' is reached always and such that none of the states of $H_{\bar{q}}$ are identified. Then,

$$\begin{aligned} states(H'_{\bar{q}}) &= \{q'' \mid \exists_i q'' \leq q'_i\} \\ &\cup \left(\bigcup_i \{q'_i \frown (q'' \triangleright q_i) \mid q'' \in states(H_{\bar{q}}), q_i < q''\} \right). \end{aligned} \tag{9.21}$$

The transition relation of $H'_{\bar{q}}$ is obtained from the transition relation of $H_{\bar{q}}$ by scheduling the same time-enriched transitions of M as before except for the states $\bar{q}_i$ where the intermediate transitions leading to the q'_i's are scheduled. It is simple to check that $H'_{\bar{q}}$ satisfies the desired properties. ∎

9.3.3 Probabilistic Executions versus Probabilistic Timed Executions

In this section we show the relationship between probabilistic executions, probabilistic time-enriched executions, and probabilistic timed executions. The main idea is that they all represent the same structures with different levels of detail. We show that a probabilistic execution is a sampling of a probabilistic time-enriched execution, where the information given by the trajectories is lost. Conversely, we show that each probabilistic time-enriched execution is sampled by some probabilistic execution. We show that each probabilistic time-enriched execution represents a probabilistic timed execution and that each probabilistic timed execution is represented by some probabilistic time-enriched execution. Essentially, a probabilistic time-enriched execution is a probabilistic timed execution with the additional information of what time-passage transitions are scheduled. Finally, we define an equivalence relation on probabilistic time-enriched executions that captures the idea of representing the same probabilistic timed execution. This equivalence relation will be useful for parallel composition.

Probabilistic Executions versus Probabilistic Time-Enriched Executions

There is a close relationship between the probabilistic executions of a probabilistic timed automaton and its probabilistic time-enriched executions. Informally, a probabilistic time-enriched execution contains more information than a probabilistic execution because it associates a state with every real time rather than with a countable set of times. In other words, a probabilistic execution can be seen as a sampling of a probabilistic time-enriched execution at countably many points. In later chapters we will see that probabilistic executions are sufficient for the study of the properties of a system whenever such properties do not depend on the actual states that are reached at each time. For the moment we just define what it means for a probabilistic execution to sample a probabilistic time-enriched execution, and we show that each probabilistic time-enriched execution is sampled by some probabilistic execution and that each probabilistic execution samples some probabilistic time-enriched execution. We start by defining a function *sample* that applied to a probabilistic time-enriched execution H of a probabilistic timed automaton M gives a probabilistic execution H' of M, which by definition samples H.

Let $\alpha = \omega_0 a_1 \omega_1 a_2 \omega_2 \cdots$ be a time-enriched execution of a probabilistic timed automaton M, and let $sample(\alpha)$ be the sequence $\alpha' = lstate(\omega_0) a_1 lstate(\omega_1) a_2 lstate(\omega_2) \cdots$. Then, it is easy to check that α' is an execution of M. We say that α' *samples* α. Define

$$states(H') \triangleq sample(states(H)). \tag{9.22}$$

Let $(q, \mathcal{P})$ be a transition of H. Define *sample* on Ω as follows: $sample((a, q')) = (a, sample(q'))$, and $sample(\delta) = \delta$. Then, define the transition $sample((q, \mathcal{P}))$ to be

$$sample((q, \mathcal{P})) \triangleq (sample(q), sample(\mathcal{P})). \tag{9.23}$$

For each state q of H', let $sample^{-1}(q)$ be the set of states q' of H such that $sample(q') = q$. Observe that all the states of $sample^{-1}(q)$ are incomparable under prefix. For each $q' \in sample^{-1}(q)$, let

$$\bar{p}_{q'}^{sample^{-1}(q)} \triangleq \frac{P_H[C_{q'}]}{\sum_{q'' \in sample^{-1}(q)} P_H[C_{q''}]}. \tag{9.24}$$

Then, the transition enabled from q in H' is defined to be

$$tr_q^{H'} \triangleq \sum_{q' \in sample^{-1}(q)} \bar{p}_{q'}^{sample^{-1}(q)} sample(tr_{q'}^{H}). \tag{9.25}$$

Observe the similarity of Equations (9.24) and (9.25) with the equations that the fine the projection of a probabilistic execution (cf. Equations (4.21) and (4.22)).

Proposition 9.3.6 below shows that H' is a probabilistic execution of M. We say that H' *samples* H. Then, Proposition 9.3.7 shows that each probabilistic execution samples some probabilistic time-enriched execution.

Proposition 9.3.6 *For each probabilistic time-enriched execution H of a probabilistic timed automaton M, $sample(H)$ is a probabilistic execution of M.*

Proof. Let H' denote $sample(H)$. The fact that each state of H' is reachable can be shown by a simple inductive argument; the fact that each state of H' is a finite execution fragment of M follows from a simple analysis of the definition of *sample* and of a time-enriched execution.

We need to check that for each state q of H' the transition enabled from q in H' is generated by a combined transition of M. From (9.25), it is enough to show that for each state q' of $sample^{-1}(q)$ the transition $sample(tr_{q'}^{H})$ is generated by a combined transition of M.

Since H is a probabilistic time-enriched execution of M, then there is a time-enriched transition $(lstate(q'), \mathcal{P})$ of M such that $\mathcal{P}_{q'}^{H} = q' \frown \mathcal{P}$. From the definition of *sample* and the definition of a time-enriched transition, $(lstate(q), sample(\mathcal{P}))$ is a combined transition of M, and $sample(\mathcal{P}_{q'}^{H}) = sample(q') \frown sample(\mathcal{P})$, which means that $sample(\mathcal{P}_{q}^{H'}) = q \frown sample(\mathcal{P})$. This is enough to conclude. ∎

Proposition 9.3.7 *Let H be a probabilistic execution of a probabilistic timed automaton M. Then there is a probabilistic time-enriched execution H' of M such that $H = sample(H')$.*

Proof. We build H' inductively in such a way that for each state q of H there is exactly one state q' of H' in $sample^{-1}(q)$. The start state of H' is the same as the start state of H.

Suppose that the transition relation of H' is defined for each state of length at most $i-1$ and assume that for each state q of H of length at most i there is exactly one state q' of H' in $sample^{-1}(q)$. Let q be a state of H of length i and let q' be the state of $sample^{-1}(q)$. Observe from the definition of *sample* that the length of q' is i. Let $(lstate(q), \mathcal{P})$ be the combined transition of M that corresponds to tr_q^H. For each pair (a, s) of Ω, if a is a discrete action, then let $\mathcal{P}_{(a,s')}$ be $\mathcal{D}((a, s'))$; if a is a time-passage action, then let $\mathcal{P}_{(a,s')}$ be $\mathcal{D}(w_{a,s'})$, where $w_{a,s'} \in trajectories(M, s, a, s')$. Let $\mathcal{P}' = \sum_{(a,s)\in\Omega} P[(a, s)]\mathcal{P}_{(a,s)}$. Then, $(lstate(q), \mathcal{P}')$ is a time-enriched transition of M. Let $tr_{q'}^{H'}$ be $(q', q' \frown \mathcal{P}')$. Then, $tr_{q'}^{H'}$ is a legal transition for H'. Moreover, from the definition of $\mathcal{P}'$, each state of $\mathcal{P}_q^H$ is the sampling of exactly one state of $\mathcal{P}_{q'}^{H'}$, and, vice versa, the sample of each state of $\mathcal{P}_{q'}^{H'}$ is a state of $\mathcal{P}_q^H$. ∎

Probabilistic Time-Enriched Executions versus Probabilistic Timed Executions

We define a function *t-sample* that, given a probabilistic time-enriched execution fragment H of M, builds a probabilistic timed execution H' as follows.

$$\begin{aligned} \mathit{states}(H') \;=\; & \{\textit{t-exec}(q_0^H) \cup \\ & \{q \in \Omega_{\textit{t-exec}(H)} \mid q \text{ contains finitely many actions}\} \cup \\ & \{q \in \textit{t-frag}^*(M) \mid \mathit{ltraj}(q) \text{ is a } [0,0]\text{-trajectory and } \exists_{q' \in \Omega_{\textit{t-exec}(H)}} q \leq q'\}. \end{aligned} \tag{9.26}$$

The start state of H' is $\textit{t-exec}(q_0^H)$, and for each state q of H' the transition enabled from q is $(q, \mathit{truncate}_q(\textit{t-exec}(\mathcal{P}_H)|C_q))$.

Proposition 9.3.8 *$\textit{t-sample}(H)$ is a probabilistic timed execution fragment of M.*

Proof. We need to show that for each state q of H' that enables some transition there is a probabilistic time-enriched execution fragment H_q of M starting from $\mathit{lstate}(q)$ such that $\mathcal{P}_q^H = \mathit{truncate}_{\mathit{lstate}(q)}(\textit{t-exec}(\mathcal{P}_{H_q}))$.

Let $q_1, q_2, \ldots$ be an enumeration of the states q' of H such that $\textit{t-exec}(q') = q$, and for each i let p_i denote $P_H[C_{q_i}]$. Observe that, since q ends with the occurrence of a discrete action, for each state q'' of H such that $q' \leq \textit{t-exec}(q'')$ there is an i such that $q_i \leq q''$. Define H_q as follows.

$$\mathit{states}(H_q) \triangleq \bigcup_i \mathit{states}(H \triangleright q_i). \tag{9.27}$$

For each state q' of H_q, let

$$tr_{q'}^{H_q} \triangleq \frac{\sum_{i \mid q' \in \mathit{states}(H \triangleright q_i)} P_H[C_{q_i \frown q'}](tr^H_{q_i \frown q'} \triangleright q_i)}{\sum_{i \mid q' \in \mathit{states}(H \triangleright q_i)} P_H[C_{q_i \frown q'}]}. \tag{9.28}$$

Then, it is enough to prove that

$$q \frown \textit{t-exec}(\mathcal{P}_{H_q}) = \textit{t-exec}(\mathcal{P}_H)|C_q. \tag{9.29}$$

Before proving (9.29), we show the following property: for each state q' of H_q,

$$P_{H_q}[C_{q'}] = \frac{\sum_{i \mid q' \in \mathit{states}(H \triangleright q_i)} P_H[C_{q_i \frown q'}]}{\sum_i P_H[C_{q_i}]}. \tag{9.30}$$

This follows easily by induction using Equation (9.28) for the inductive step. The denominator is necessary for the base case to work.

We now turn to Equation (9.29). Consider an extended timed execution fragment α of M, and distinguish the following two cases.

1. α does not end with an open trajectory.

   Suppose that $\alpha \in \Omega_{\textit{t-exec}(\mathcal{P}_H)|C_q}$. Then, from the definition of $\textit{t-exec}()$ and of the conditional operation, $q \leq \alpha$ and there is a time-enriched execution α' of Ω_H such that $\textit{t-exec}(\alpha') = \alpha$. This means that there is a time-enriched execution α' of Ω_H such that $\textit{t-exec}(\alpha') = \alpha$ and there is a state q_i of H such that $q_i \leq \alpha'$. From the construction of H_q, each prefix of α' is a state of H_q, and thus $\alpha' \in \Omega_{\textit{t-exec}(H_q)}$. The argument can be reversed.

2. α ends with an open trajectory.

   Suppose that $\alpha \in \Omega_{t\text{-}exec(\mathcal{P}_H)|C_q}$. Then, from the definition of *t-exec*() and of the conditional operation, $q \leq \alpha$ and for each finite prefix α' of α there is a timed execution α'' of $t\text{-}exec(\Omega_H)$ such that $\alpha' \leq \alpha''$. It is sufficient to show that for each finite prefix α' of α there is a timed execution α''_q of $t\text{-}exec(\Omega_{H_q})$ such that $\alpha' \leq (q \frown \alpha''_q)$. Consider a prefix α' of α, and let α'' be an element of $t\text{-}exec(\Omega_H)$ such that $\alpha' \leq \alpha''$. Then there is a time-enriched execution α''' of Ω_H such that $\alpha' \leq t\text{-}exec(\alpha''')$, which means that there is a finite prefix α'''' of α''' such that $\alpha' \leq t\text{-}exec(\alpha''')$ and $q \leq t\text{-}exec(\alpha''')$. Let q_i be the prefix of α''''. We know that such prefix exists. Then, from the definition of H_q, $\alpha''''\triangleright q_i$ is a state of H_q, and thus there is a time-enriched execution α'_q of Ω_{H_q} such that $\alpha' \leq (q \frown t\text{-}exec(\alpha'_q))$. Moreover, $t\text{-}exec(\alpha'_q) \in t\text{-}exec(\mathcal{P}_{H_q})$, which is sufficient to conclude. The argument can be reversed.

Finally, we need to show that $P_{t\text{-}exec(\mathcal{P}_H)|C_q}$ and $P_{t\text{-}exec(\mathcal{P}_{H_q})}$ coincide on the cones of their sample spaces. Thus, consider a finite timed execution fragment α of M. From the definition of *t-exec*(),

$$P_{t\text{-}exec(\mathcal{P}_{H_q})}[C_\alpha] = \sum_{q' \in min(\{q' \in states(H_q) | \alpha \leq t\text{-}exec(q')\})} P_{H_q}[C_{q'}]. \tag{9.31}$$

From (9.30),

$$P_{t\text{-}exec(\mathcal{P}_{H_q})}[C_\alpha] = \sum_{q' \in min(\{q' \in states(H_q) | \alpha \leq t\text{-}exec(q')\})} \frac{\sum_{i | q' \in states(H \triangleright q_i)} P_H[C_{q_i \frown q'}]}{\sum_i P_H[C_{q_i}]}. \tag{9.32}$$

From the definition of the states of H_q, (9.32) can be rewritten into

$$P_{t\text{-}exec(\mathcal{P}_{H_q})}[C_\alpha] = \frac{\sum_i \sum_{q' \in min(\{q' \in states(H \triangleright q_i) | q \frown \alpha \leq t\text{-}exec(q_i \frown q')\})} P_H[C_{q_i \frown q'}]}{\sum_i P_H[C_{q_i}]}. \tag{9.33}$$

By simplifying the concatenations we obtain

$$P_{t\text{-}exec(\mathcal{P}_{H_q})}[C_\alpha] = \frac{\sum_{q' \in min(\{q' \in states(H) | q \frown \alpha \leq t\text{-}exec(q')\})} P_H[C_{q'}]}{\sum_i P_H[C_{q_i}]}. \tag{9.34}$$

From the definition of *t-exec*(), the definition of a conditional space, and the definition of the q_i's,

$$P_{t\text{-}exec(\mathcal{P}_H)|C_q}[C_\alpha] = \frac{\sum_{q' \in min(\{q' \in states(H) | q \frown \alpha \leq t\text{-}exec(q')\})} P_H[C_{q'}]}{\sum_i P_H[C_{q_i}]}. \tag{9.35}$$

Since the right sides of Equations (9.34) and (9.35) are the same, we conclude that

$$P_{t\text{-}exec(\mathcal{P}_{H_q})}[C_\alpha] = P_{t\text{-}exec(\mathcal{P}_H)|C_q}[C_{q \frown \alpha}]. \tag{9.36}$$

This completes the proof. ∎

Conversely, we show that every probabilistic timed execution of M is sampled by some probabilistic time-enriched execution of M. Let H be a probabilistic timed execution of M. Then, build H' as follows. Let H_0 be a probabilistic timed execution consisting of a single state that is t-sampled by q_0^H, i.e., $\textit{t-sample}(q_0^{H_0}) = q_0^H$. Strictly speaking H_0 is not a probabilistic timed execution because $q_0^{H_0}$ should enable a transition in general. Suppose now that H_i is defined. Then build H_{i+1} be extending the transition relation of H_i from all the states of H_i that do not end in δ and do not have any outgoing transition as follows. Consider a state q of H_i that do not end in δ and do not have any outgoing transition, and let q' be the state of H such that $\textit{t-exec}(q) = q'$ (our construction ensures that there is always such a state since q ends with a $[0,0]$-trajectory). From the definition of a probabilistic timed execution fragment, there is a probabilistic time-enriched execution fragment $H_{q'}$ of M starting from $\textit{lstate}(q')$ such that $\mathcal{P}_{q'}^H = \textit{truncate}_{\textit{lstate}(q')}(\textit{t-exec}(\mathcal{P}_{H_{q'}}))$. Let $H'_{q'}$ be obtained from $H_{q'}$ by removing all the transitions from states where an action has occurred and by removing all the states that become unreachable. Then, extend H_i from q' with $q' \frown H'_{q'}$, i.e., $H_{i+1} \triangleright q' = H'_{q'}$.

Then the states of H' are the union of the states of the H_i's, the start state of H' is $q_0^{H_0}$, and for each state q of H', if q is a state of H_i, then $tr_q^{H'} = tr_q^{H_{i+1}}$.

Proposition 9.3.9 $\textit{t-sample}(H') = H$.

Proof. We prove that $\mathcal{P}_H = \textit{t-exec}(\mathcal{P}_{H'})$. Then the equality between $\textit{t-sample}(H')$ and H follows by induction after observing that $\textit{t-sample}(H')$ and H have the same start state and that for each state q, $step_q^{\textit{t-sample}(H')} = (q, \textit{truncate}_q(\textit{t-exec}(\mathcal{P}_{H'})|C_q))$, and that $step_q^H = (q, \textit{truncate}_q(\mathcal{P}_H|C_q))$.

For the sample spaces, consider an element α of Ω_H. Then, by definition of Ω_H, there is an execution $\alpha_0\alpha_1\cdots$ of H such that $\lim_i \alpha_i = \alpha$, and such that either α is not a finite execution, or the last element of α ends in δ. We distinguish two cases.

1. α is either an infinite sequence or a finite sequence $\alpha_0\alpha_2\cdots\alpha_n$ where α_n ends with δ.

   From the definition of the transition relation of H', there is a sequence of extended time-enriched execution fragments $q_0, q_1, \ldots$ such that for each i $\alpha_i = \textit{t-exec}(q_0 \frown \cdots \frown q_i)$, $q_0 \frown q_1 \frown \cdots$ is an element of $\Omega_{H'}$, and $\textit{t-exec}(q_0 \frown q_1 \frown \cdots) = \alpha$. Thus, $\alpha \in \Omega_{\textit{t-exec}(H')}$. The converse argument is a reversal of the argument above.

2. $\alpha = \alpha_0\alpha_2\cdots\alpha_n$ where α_n ends with an open trajectory.

   From the definition of the transition relation of H', there is a sequence of extended time-enriched execution fragments $q_0, q_1, \ldots, q_{n-1}$ such that for each $i \leq n-1$ $\alpha_i = \textit{t-exec}(q_0 \frown \cdots \frown q_i)$ and $q_0 \frown \cdots \frown q_i$ is a state of H'. Furthermore, for each finite prefix α' of α there is a time-enriched execution fragment q_n such that $\alpha' \leq \textit{t-exec}(q_0 \frown \cdots \frown q_n)$ and $q_0 \frown \cdots \frown q_{n-1} \frown q_n$ is an element of $\Omega_{H'}$. This means that for each finite prefix α' of α there is an element α'' of $\textit{t-exec}(\Omega_{H'})$ such that $\alpha' \leq \alpha''$, and thus $\alpha \in \Omega_{\textit{t-exec}(\mathcal{P}_{H'})}$. The argument can be reversed.

Consider now a cone C_α. From the definition of $\textit{t-exec}()$,

$$P_{\textit{t-exec}(H')}[C_\alpha] = \sum_{q \in min(\{q \in states(H') | \alpha \leq \textit{t-exec}(q)\})} P_{H'}[C_q]. \tag{9.37}$$

If C_α is not empty, then $\alpha = \alpha_1 \cdots \alpha_n$, where $\alpha_n = \alpha$, $\alpha_0 \cdots \alpha_{n-1}$ is an execution of H, and there is a α'_n such that $\alpha_n \leq \alpha'_n$ and $\alpha_1 \cdots \alpha'_n$ is an execution of H. We show by induction on

n that

$$P_H[C_{\alpha_n}] = \sum_{q \in min(\{q \in states(H') \mid \alpha \leq t\text{-}exec(q)\})} P_{H'}[C_q]. \tag{9.38}$$

The base case is trivial since C_{α_0} denotes the whole sample space. For the inductive case, from the definition of the probability of a cone,

$$P_H[C_{\alpha_n}] = P_H[C_{\alpha_{n-1}}] P^H_{\alpha_{n-1}}[C_{\alpha_n}]. \tag{9.39}$$

From the definition of the transition relation of H,

$$P^H_{\alpha_{n-1}}[C_{\alpha_n}] = \frac{\sum_{q \in states(H') \mid t\text{-}exec(q)=\alpha_{n-1}} P_{H'}[C_q] P_{t\text{-}exec(H' \triangleright q)}[C_{\alpha_n \triangleright \alpha_{n-1}}]}{\sum_{q \in states(H') \mid t\text{-}exec(q)=\alpha_{n-1}} P_{H'}[C_q]}, \tag{9.40}$$

where

$$P_{t\text{-}exec(H' \triangleright q)}[C_{\alpha_n \triangleright \alpha_{n-1}}] = \sum_{q' \in min(\{q' \in states(H' \triangleright q) \mid \alpha_n \leq t\text{-}exec(q \frown q')\})} P_{H' \triangleright q}[C_{q'}]. \tag{9.41}$$

Since α_{n-1} is a state of H, the last trajectory of α_{n-1} has domain $[0,0]$, and the set $\{q \in states(H') \mid t\text{-}exec(q) = \alpha_{n-1}\}$ is a set of minimal states. Thus, by substituting (9.41) in (9.40), simplifying the numerator of (9.40), we obtain

$$P_{t\text{-}exec(H' \triangleright q)}[C_{\alpha_n \triangleright \alpha_{n-1}}] = \frac{\sum_{q' \in min(\{q' \in states(H') \mid \alpha_n \leq t\text{-}exec(q')\})} P_{H'}[C_{q'}]}{\sum_{q \in states(H') \mid t\text{-}exec(q)=\alpha_{n-1}} P_{H'}[C_q]}. \tag{9.42}$$

By substituting (9.42) in (9.39), using induction and simplifying algebraically, we get (9.38). ■

Equivalent Probabilistic Time-Enriched Executions

It is possible to define an equivalence relation on probabilistic time-enriched executions that captures exactly the probabilistic timed executions that they represent.

Let H_1 and H_2 be two probabilistic time-enriched execution fragments of a probabilistic timed automaton M. Then $t\text{-}exec(\mathcal{P}_{H_1})$ and $t\text{-}exec(\mathcal{P}_{H_2})$ are said to be *equivalent*, denoted by $t\text{-}exec(\mathcal{P}_{H_1}) \equiv t\text{-}exec(\mathcal{P}_{H_2})$, iff

1. for each timed extended execution fragment α of M that does not contain infinitely many discrete actions, $\alpha \in \Omega_{t\text{-}exec(\mathcal{P}_{H_1})}$ iff $\alpha \in \Omega_{t\text{-}exec(\mathcal{P}_{H_2})}$;
2. for each finite timed extended execution fragment α of M, $P_{t\text{-}exec(\mathcal{P}_{H_1})}[C_\alpha] = P_{t\text{-}exec(\mathcal{P}_{H_2})}[C_\alpha]$.

H_1 and H_2 are said to be *equivalent*, denoted by $H_1 \equiv H_2$, iff $t\text{-}exec(q_0^{H_1}) = t\text{-}exec(q_0^{H_2})$ and $t\text{-}exec(\mathcal{P}_{H_1}) \equiv t\text{-}exec(\mathcal{P}_{H_2})$.

Example 9.3.3 (Two equivalent probabilistic time-enriched executions) In the definition above we do not require the sample spaces of the given probabilistic time-enriched execution fragments to contain the same timed executions with infinitely many discrete actions. Figure 9-2 shows an example of two probabilistic time-enriched executions whose corresponding sample spaces differ from a timed execution with infinitely many discrete actions and such that

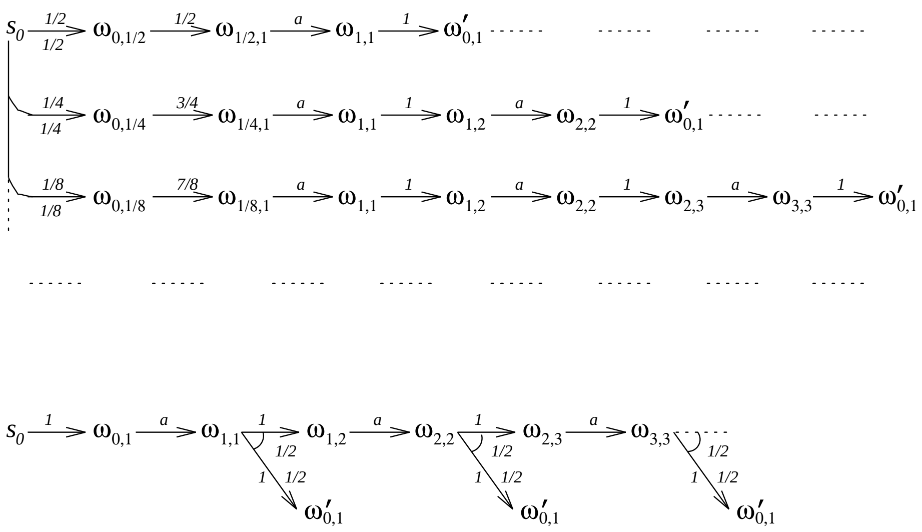

Figure 9-2: Probabilistic time-enriched executions that represent the same probabilistic timed execution.

t-sample() gives the same probabilistic timed execution. The important aspect of this example is that in the upper probabilistic time-enriched execution the explicit time-passage actions are used to let 1 time unit elapse in infinitely many different ways. However, the trajectory that is spanned before the first occurrence of action a is always the same. Observe that the fact that the two probabilistic time-enriched executions of Figure 9-2 represent the same structure is not a consequence of the limit closure of the sample space of *t-exec*(), since *t-exec*(Ω_{H_1}) and *t-exec*(Ω_{H_2}) do not differ in timed executions that end with an open trajectory. Rather, by analyzing this example again in the context of parallel composition we will discover the reason for our definition of *t-exec*() (cf. Example 9.5.1). ■

The rest of this section is dedicated to showing that $\equiv$ characterizes the probabilistic timed executions represented by probabilistic time-enriched executions. We do it by showing two results: the first result says that two equivalent probabilistic time-enriched executions describe the same probabilistic timed execution, and the second result says that for each probabilistic time-enriched execution H, $\mathcal{P}_{t\text{-}sample(H)} \equiv t\text{-}exec(\mathcal{P}_H)$.

Proposition 9.3.10 *If* $t\text{-}exec(H_1) \equiv t\text{-}exec(H_2)$*, then* $t\text{-}sample(H_1) = t\text{-}sample(H_2)$.

Proof. Let $q \in states(t\text{-}sample(H_1))$. If $q = t\text{-}exec(q_0^{H_1})$ or $q \in \Omega_{t\text{-}exec(H_1)}$ and contains finitely many discrete actions, then $q \in states(t\text{-}sample(H_2))$ trivially. Thus, suppose that $ltraj(q)$ is a [0,0]-trajectory and that there is a $q' \in \Omega_{t\text{-}exec(H_1)}$ such that $q \leq q'$. Then, $P_{t\text{-}exec(H_1)}[C_q] > 0$, and, since $t\text{-}exec(H_1) = t\text{-}exec(H_2)$, $P_{t\text{-}exec(H_2)}[C_q] > 0$. Thus, there is a $q'' \in \Omega_{t\text{-}exec(H_2)}$ such that $q \leq q''$, which means that $q \in states(t\text{-}sample(H_2))$. The converse argument is identical.

Consider now a state q of $t\text{-}sample(H_1)$ and $t\text{-}sample(H_2)$. We need to show that $tr_q^{t\text{-}sample(H_1)}$ and $tr_q^{t\text{-}sample(H_2)}$ are the same transition. From the definition of *t-sample*(), it is enough to show that $truncate_q(t\text{-}exec(\mathcal{P}_{H_1})|C_q) = truncate_q(t\text{-}exec(\mathcal{P}_{H_2})|C_q)$. Since $t\text{-}exec(\mathcal{P}_{H_1}) \equiv t\text{-}exec(\mathcal{P}_{H_2})$, a direct analysis of the definition of *t-exec*() shows that $t\text{-}exec(\mathcal{P}_{H_1})|C_q \equiv t\text{-}exec(\mathcal{P}_{H_2})|C_q$. The truncation operation is independent of the elements of Ω that contains infinitely many discrete actions, and thus $\Omega_{truncate_q(t\text{-}exec(\mathcal{P}_{H_1})|C_q)} = \Omega_{truncate_q(t\text{-}exec(\mathcal{P}_{H_2})|C_q)}$. Furthermore, directly from the definition of $\equiv$, $P_{truncate_q(t\text{-}exec(\mathcal{P}_{H_1})|C_q)}$ and $P_{truncate_q(t\text{-}exec(\mathcal{P}_{H_2})|C_q)}$ coincide on the cones, and thus $truncate_q(t\text{-}exec(\mathcal{P}_{H_1})|C_q) = truncate_q(t\text{-}exec(\mathcal{P}_{H_2})|C_q)$. ■

Proposition 9.3.11 *Let H be a probabilistic time-enriched execution of a probabilistic timed automaton M. Then, $\mathcal{P}_{t\text{-}sample(H)} \equiv t\text{-}exec(\mathcal{P}_H)$.*

Proof. Consider a finite timed execution α of M. We prove the proposition in three steps.

1. For each finite timed extended execution α of M, there is a timed extended execution α' of $\Omega_{t\text{-}sample(H)}$ such that $\alpha \leq \alpha'$ iff there is a timed extended execution α'' of $\Omega_{t\text{-}exec(\mathcal{P}_H)}$ such that $\alpha \leq \alpha''$.

   Let $\alpha' \in \Omega_{t\text{-}sample(H)}$ such that $\alpha \leq \alpha'$. Then there is a complete execution $q_0 q_1 \cdots$ of *t-sample*(H) such that $lim_i q_i = \alpha'$. In particular, there is a value n such that $\alpha \leq q_n$. From the definition of the transition relation of *t-sample*(H), $P_{t\text{-}exec(H)}[C_{q_n}] > 0$, and thus there is a timed execution α'' of $\Omega_{t\text{-}exec(\mathcal{P}_H)}$ such that $q_n \leq \alpha''$, which means that $\alpha \leq \alpha''$. Conversely, suppose that there is a timed execution α'' of $\Omega_{t\text{-}exec(\mathcal{P}_H)}$ such that $\alpha \leq \alpha''$. If α'' contains finitely many actions, then $\alpha'' \in \Omega_{t\text{-}sample(H)}$ by definition. Otherwise, there is a finite prefix α''' of α'' such that $\alpha \leq \alpha'''$ and the last trajectory of α''' has domain $[0, 0]$. From the definition of *t-sample*(H), α''' is a state of *t-sample*(H), and thus there is a timed execution α' of $\Omega_{t\text{-}sample(H)}$ such that $\alpha''' \leq \alpha'$, which means that $\alpha \leq \alpha'$.

2. For each timed extended execution fragment α of M that does not contain infinitely many discrete actions, $\alpha \in \Omega_{t\text{-}sample(H)}$ iff $\alpha \in \Omega_{t\text{-}exec(\mathcal{P}_H)}$.

   Let α be a timed extended execution of M that does not contain infinitely many discrete actions, and suppose that $\alpha \in \Omega_{t\text{-}sample(H)}$. If α ends with δ, then Item 1 is sufficient to conclude that $\alpha \in \Omega_{t\text{-}exec(\mathcal{P}_H)}$. If α does not end with δ, then there is a finite execution $q_0 q_1 \cdots q_n$ of *t-sample*(H) such that q_n ends with a right-open trajectory. From the definition of the transition relation of *t-sample*(H), $q_n \in truncate_{q_{n-1}}(t\text{-}exec(\mathcal{P}_H)|C_{q_{n-1}})$. Since q_n ends with an open trajectory, $q_n \in \Omega_{t\text{-}exec(\mathcal{P}_H)}$, i.e., $\alpha \in \Omega_{t\text{-}exec(\mathcal{P}_H)}$.

   Conversely, suppose that $\alpha \in \Omega_{t\text{-}exec(\mathcal{P}_H)}$. If α ends with δ, then Item 1 is sufficient to conclude that $\alpha \in \Omega_{t\text{-}sample(H)}$. If α does not end with δ, then there is a finite prefix α' of α such that $\alpha \triangleright \alpha'$ does not contain any action, and either α' is the start state of *t-sample*(H), or the last trajectory of α' has domain $[0, 0]$. Thus, from the definition of *t-sample*(), α' is a state of *t-sample*(H). From the definition of *truncate*, $\alpha \in truncate_{\alpha'}(t\text{-}exec(\mathcal{P}_H)|C_{\alpha'})$, and thus, from the definition of the transition relation of *t-sample*(H), $\alpha \in \Omega_{\alpha'}^{t\text{-}sample(H)}$. Since α ends with an open trajectory, $\alpha \in \Omega_{t\text{-}sample(H)}$.

3. For each finite timed extended execution fragment α of M, $P_{t\text{-}sample(H)}[C_\alpha] = P_{t\text{-}exec(\mathcal{P}_H)}[C_\alpha]$.

   Let α be a finite timed execution. From Item 1, $C_\alpha^{t\text{-}sample(H)} = \emptyset$ iff $C_\alpha^{t\text{-}exec(\mathcal{P}_H)} = \emptyset$. Suppose that $C_\alpha^{t\text{-}sample(H)}$ is not empty. Then there is an execution of *t-sample*(H),

$\alpha_0\alpha_1\cdots\alpha_{n-1}\alpha_n$ such that $\alpha_{n-1} < \alpha \leq \alpha_n$. From the definition of the probability of a cone,

$$P_{t\text{-}sample(H)}[C_\alpha] = P_{\alpha_0}[C_{\alpha_1}]P_{\alpha_1}[C_{\alpha_2}]\cdots P_{\alpha_{n-2}}[C_{\alpha_{n-1}}]P_{\alpha_{n-1}}[C_\alpha]. \tag{9.43}$$

From the definition of *t-sample*(H), for each $i < n$

$$P_{\alpha_i}[C_{\alpha_{i+1}}] = P_{t\text{-}exec(H)|C_{\alpha_i}}[C_{\alpha_{i+1}}]. \tag{9.44}$$

Thus, by substituting (9.44) in (9.43) and simplifying, we obtain

$$P_{t\text{-}sample(H)}[C_\alpha] = P_{t\text{-}exec(H)}[C_\alpha]. \tag{9.45}$$

This completes the proof. ∎

9.4 Moves

In the non-timed framework we have introduced the notion of a weak transition to abstract from internal computation. Informally, a weak transition is obtained by concatenating several internal and external transitions so that overall the system emulates a unique transition labeled with at most one external action. In the timed framework, due to the presence of explicit time-passage actions, it may be the case that some time t cannot elapse without performing some internal transitions in the middle. This problem becomes more evident when we extend the simulation relations to the timed framework (cf. Chapter 12). For this reason we introduce the concept of a *move*, which extends weak transitions and abstracts from internal transitions interleaved with time-passage transitions..

Let M is a probabilistic timed automaton, s be a state of M, $\mathcal{P}$ be a discrete probability distribution over states of M, and a be an action of M or the value 0. If a is a visible action of M then we use the expression $s \overset{a}{\leadsto} \mathcal{P}$ to denote $s \overset{a}{\Longrightarrow} \mathcal{P}$; if $a = 0$, then we use the expression $s \overset{0}{\leadsto} \mathcal{P}$ to denote $s \leadsto \mathcal{P}$, which is the same as $s \Longrightarrow \mathcal{P}$; if a is a time-passage action, i.e., $a = d$ for some $d \in \Re^+$, then we use the expression $s \overset{d}{\leadsto} \mathcal{P}$ to denote that $\mathcal{P}$ is reached from s by means of several internal and time-passage transitions so that in each situation time d has elapsed. Formally, $s \overset{d}{\leadsto} \mathcal{P}$ iff there is a probabilistic execution fragment H such that

1. the start state of H is s;
2. $P_H[\{\alpha\delta \mid \alpha\delta \in \Omega_H\}] = 1$, i.e., the probability of termination in H is 1;
3. for each $\alpha\delta \in \Omega_H$, $t\text{-}trace(\alpha) = t\text{-}trace(a)$;
4. $\mathcal{P} = lstate(\delta\text{-}strip(\mathcal{P}_H))$, where $\delta\text{-}strip(\mathcal{P}_H)$ is the probability space $\mathcal{P}'$ such that $\Omega' = \{\alpha \mid \alpha\delta \in \Omega_H\}$, and for each $\alpha \in \Omega'$, $P'[\alpha] = P_H[C_{\alpha\delta}]$;

The notion of a generator for a weak transition can be extended to moves in a straightforward way.

9.5 Parallel Composition

The parallel composition operator for probabilistic timed automata is exactly the same as the parallel composition operator for probabilistic automata. Thus, we omit the formal definition. According to the definition of the transition relation of $M_1\|M_2$, M_1 and M_2 synchronize on all their time-passage transitions, and thus time advances always at the same speed in M_1 and M_2.

The definition of a projection of a probabilistic time-enriched execution is the same as the definition of a projection of a probabilistic execution, except that the states of a probabilistic time-enriched execution fragment are time-enriched execution fragments rather than ordinary execution fragments. Thus, we need to extend the definition of a projection to time-enriched execution fragments and time-enriched transitions.

Let M be $M_1\|M_2$, and let α be a time-enriched execution of M. The projection of α onto M_i, $i=1,2$, is the sequence obtained from α by projecting the codomain of each trajectory onto M_i, by removing all the actions not in $acts(M_i)$, and by concatenating all the trajectories whose intermediate actions are removed. It is straightforward to check that α is a time-enriched execution of M_i.

Let H be a probabilistic time-enriched execution of M, and let $tr = (q, \mathcal{P})$ be an action restricted transition of H such that only actions of M_i, $i=1,2$, appear in tr. Define the projection operator on the elements of Ω as follows: $(a,q')\lceil M_i = (a, q'\lceil M_i)$, and $\delta\lceil M_i = \delta$. The projection of tr onto M_i, denoted by $tr\lceil M_i$, is the pair $(q\lceil M_i, \mathcal{P}\lceil M_i)$.

Proposition 9.5.1 *Let $M = M_1\|M_2$, and let H be a probabilistic time-enriched execution fragment of M. Then $H\lceil M_1 \in$ t-prexec(M_1) and $H\lceil M_2 \in$ t-prexec(M_2).*

Proof. The structure of the proof is the same as the proof of Proposition 4.3.4. This time it is necessary to observe that for each state q of H the transition $(tr_{q'}^H \upharpoonright acts(M_1))\lceil M_1$ is generated by a time-enriched transition of M_i. ■

Proposition 9.5.2 *Let $M = M_1\|M_2$, and let H be a probabilistic time-enriched execution fragment of M. Let H_i be $H\lceil M_i$, $i=1,2$. Let q be a state of H_i. Then,*

$$P_{H_i}[C_q] = \sum_{q' \in min(q\rceil H)} P_H[C_{q'}]. \tag{9.46}$$

Proof. This proof has the same structure as the proof of Proposition 4.3.5. ■

In the rest of this section we extend the results of Section 9.3.3 to account for parallel composition. We show that *sample* commutes with projections and that the projections of two equivalent probabilistic time-enriched executions are equivalent. The first result guarantees that *sample* and projection are well defined for probabilistic time-enriched executions; the second result allows us to define indirectly a projection operator on probabilistic timed executions: namely, given a probabilistic timed execution H of $M_1\|M_2$, let H' be any probabilistic time-enriched execution of $M_1\|M_2$ such that *t-sample*$(H') = H$. Then, $H\lceil M_i$ is defined to be *t-sample*$(H'\lceil M_i)$. Before proving these two results, we show why in the definition of *t-exec*() we force probabilistic time-enriched executions like those of Figure 9-1 to be mapped to the same structure (cf. Example 9.3.2).

Example 9.5.1 (Reason for the definition of *t-exec*) We have already seen that the probabilistic time-enriched executions of Figure 9-2 are t-samples of the same probabilistic timed execution. Suppose now the probabilistic time-enriched executions of Figure 9-2 to be probabilistic time-enriched executions of the parallel composition of two probabilistic timed automata M_1 and M_2, and suppose that a is an action of M_2 only. By projecting the probabilistic time-enriched executions of Figure 9-2 onto M_1 we obtain two probabilistic time-enriched executions like those of Figure 9-1, which must denote the same probabilistic timed execution if we want *t-sample* to be preserved by the projection operation. ■

Proposition 9.5.3 *Let M be $M_1 \| M_2$, and let H be a probabilistic time-enriched execution of M. Then, $sample(H\lceil M_i) = sample(H)\lceil M_i$.*

Proof. Since the sampling function commutes with the projection function, $sample(H\lceil M_i)$ and $sample(H)\lceil M_i$ have the same states.

For convenience, denote $sample(H)$ by H'. Let q be one of the states of $sample(H)\lceil M_i$. Below we show that the equation for the transition leaving from q in $sample(H)\lceil M_i$ and the equation for the transition leaving from q in $sample(H\lceil M_i)$ denote the same transition. This is sufficient to show that $sample(H)\lceil M_i$ and $sample(H\lceil M_i)$ have the same transition relation. We use implicitly the fact that the projection onto M_i distributes over the sum of transitions restricted to $acts(M_i)$.

From (9.25), Proposition 4.3.2, and an algebraic simplification, the expression

$$\sum_{q' \in q\rceil H'} \bar{p}_{q'}^{q\rceil H'} P_{q'}^{H'}[acts(M_i)](tr_{q'}^{H'} \upharpoonright acts(M_i))\lceil M_i \tag{9.47}$$

can be rewritten into

$$\sum_{q' \in q\rceil H'} \sum_{q'' \in sample^{-1}(q')} \bar{p}_{q'}^{q\rceil H'} \bar{p}_{q''}^{sample^{-1}(q')} sample(tr_{q''}^{H} \upharpoonright acts(M_i))\lceil M_i, \tag{9.48}$$

which becomes

$$\sum_{q'' \in sample^{-1}(q\rceil H')} \bar{p}_{sample(q'')}^{q\rceil H'} \bar{p}_{q''}^{sample^{-1}(sample(q''))} sample(tr_{q''}^{H} \upharpoonright acts(M_i))\lceil M_i, \tag{9.49}$$

after grouping the two sums.

Denote $H\lceil M_i$ by H''. From (4.22), Proposition 4.3.2, and an algebraic simplification,

$$\sum_{q' \in sample^{-1}(q)} \bar{p}_{q'}^{sample^{-1}(q)} sample(tr_{q'}^{H''}) \tag{9.50}$$

can be rewritten into

$$\sum_{q' \in sample^{-1}(q)} \sum_{q'' \in q'\rceil H} \bar{p}_{q'}^{sample^{-1}(q)} \bar{p}_{q''}^{q'\rceil H} P_{q''}^{H}[acts(M_i)] sample(tr_{q''}^{H} \upharpoonright acts(M_i))\lceil M_i, \tag{9.51}$$

which becomes

$$\sum_{q'' \in (sample^{-1}(q))\rceil H} \bar{p}_{q''\lceil M_i}^{sample^{-1}(q)} \bar{p}_{q''}^{(q''\lceil M_i)\rceil H} P_{q''}^{H}[acts(M_i)] sample(tr_{q''}^{H} \upharpoonright acts(M_i))\lceil M_i \tag{9.52}$$

after grouping the two sums.

From the commutativity of *sample* and projection, $sample^{-1}(q\rceil H') = sample^{-1}(q)\rceil H$. Thus, in order to show that (9.49) and (9.52) denote the same transition, it is sufficient to show that for each state q'' of $sample^{-1}(q\rceil H')$,

$$\bar{p}^{q\rceil H'}_{sample(q'')}\bar{p}^{sample^{-1}(sample(q''))}_{q''} = \bar{p}^{sample^{-1}(q)}_{q''\lceil M_i}\bar{p}^{(q''\lceil M_i)\rceil H}_{q''}. \tag{9.53}$$

By expanding the expressions above with their definitions, (9.53) becomes

$$\begin{aligned} &\frac{P_{H'}[C_{sample(q'')}]P_H[C_{q''}]}{(\sum_{\bar{q}'\in min(q\rceil H')} P_{H'}[C_{\bar{q}'}])(\sum_{\bar{q}''\in sample^{-1}(sample(q''))} P_H[C_{\bar{q}''}])} \\ &= \frac{P_{H''}[C_{q''\lceil M_i}]P_H[C_{q''}]}{(\sum_{\bar{q}'\in sample^{-1}(q)} P_{H''}[C_{\bar{q}'}])(\sum_{\bar{q}''\in min((q''\lceil M_i)\rceil H)} P_H[C_{\bar{q}''}])}. \end{aligned} \tag{9.54}$$

By simplifying common subexpressions, using Proposition 4.3.5, and observing that

$$P_{H'}[C_{sample(q'')}] = \sum_{\bar{q}''\in sample^{-1}(sample(q''))} P_H[C_{\bar{q}''}], \tag{9.55}$$

(we have verified properties like (9.55) several times) Equation (9.54) becomes

$$\sum_{\bar{q}'\in min(q\rceil H')} P_{H'}[C_{\bar{q}'}] = \sum_{\bar{q}'\in sample^{-1}(q)} P_{H''}[C_{\bar{q}'}], \tag{9.56}$$

which can be shown as follows:

$$\begin{aligned} &\sum_{\bar{q}'\in min(q\rceil H')} P_{H'}[C_{\bar{q}'}] \\ &= \sum_{\bar{q}'\in min(q\rceil H')}\ \sum_{q''\in sample^{-1}(\bar{q}')} P_H[C_{q''}] \\ &= \sum_{q''\in min(sample^{-1}(q\rceil H'))} P_H[C_{q''}] \\ &= \sum_{q''\in min((sample^{-1}(q))\rceil H)} P_H[C_{q''}] \\ &= \sum_{\bar{q}'\in sample^{-1}(q)}\ \sum_{q''\in min(\bar{q}'\rceil H)} P_H[C_{q''}] \\ &= \sum_{\bar{q}'\in sample^{-1}(q)} P_{H''}[C_{\bar{q}'}], \end{aligned}$$

where the first step follows from (9.55), the second and fourth steps follow from grouping and ungrouping sums, the third step follows from the commutativity of *sample* and projection, and the fifth step follows from Proposition 4.3.5. ∎

Proposition 9.5.4 *Let H_1 and H_2 be two probabilistic time-enriched executions of $M_1\|M_2$. If $H_1 \equiv H_2$, then $H_1\lceil M_i \equiv H_2\lceil M_i$, $i = 1, 2$.*

Proof. We show first that *t-exec*$(\mathcal{P}_{H_1\lceil M_i})$ and *t-exec*$(\mathcal{P}_{H_2\lceil M_i})$ assign the same probabilities to the same cones; then we show that the sample spaces of *t-exec*$(\mathcal{P}_{H_1\lceil M_i})$ and *t-exec*$(\mathcal{P}_{H_2\lceil M_i})$ satisfy the condition for $\equiv$. This part of the proof relies on the way we have defined the sample spaces of the objects produced by *t-exec*(). For the cones, we show that for each finite timed extended execution α of M_i,

$$P_{t\text{-}exec(\mathcal{P}_{H_1\lceil M_i})}[C_\alpha] = \sum_{\alpha'\in min(\{\alpha'\in t\text{-}frag^*_\delta(M_1\|M_2)\mid\alpha=\alpha'\lceil M_i\})} P_{t\text{-}exec(H_1)}[C_{\alpha'}]. \tag{9.57}$$

and

$$P_{t\text{-}exec(\mathcal{P}_{H_2\lceil M_i})}[C_\alpha] = \sum_{\alpha'\in min(\{\alpha'\in t\text{-}frag^*_\delta(M_1\|M_2)\mid\alpha=\alpha'\lceil M_i\})} P_{t\text{-}exec(H_2)}[C_{\alpha'}]. \tag{9.58}$$

Then, since $H_1 \equiv H_2$, we conclude that the right sides of (9.57) and (9.58) are equal, and thus, $H_1\lceil M_i \equiv H_2\lceil M_i$. We prove only (9.57); the proof for (9.58) is symmetric. From the definition of *t-exec*(),

$$P_{t\text{-}exec(\mathcal{P}_{H_1\lceil M_i})}[C_\alpha] = \sum_{q\in min(\{q\in states(H_1\lceil M_i)\mid\alpha\leq t\text{-}exec(q)\})} P_{H_1\lceil M_i}[C_q]. \tag{9.59}$$

From (4.31),

$$P_{t\text{-}exec(\mathcal{P}_{H_1\lceil M_i})}[C_\alpha] = \sum_{q\in min(\{q\in states(H_1\lceil M_i)\mid\alpha\leq t\text{-}exec(q)\})} \left(\sum_{q'\in min(q\rceil H_1)} P_{H_1}[C_{q'}]\right). \tag{9.60}$$

Consider a state q of $min(\{q \in states(H_1\lceil M_i) \mid \alpha \leq t\text{-}exec(q)\})$ and a state q' of $min(q\rceil H_1)$. Then, from the definition of *t-exec*(), there is at least one $\alpha' \in t\text{-}frag^*_\delta(M_1\|M_2)$ such that $\alpha = \alpha'\lceil M_i$ and $q' \in min(\{q' \in states(H_1) \mid \alpha' \leq t\text{-}exec(q')\})$. Moreover, there is exactly one minimum α'. Conversely, consider one $\alpha' \in min(\{\alpha' \in t\text{-}frag^*_\delta(M_1\|M_2) \mid \alpha = \alpha'\lceil M_i\})$, and consider a state q' of $min(\{q' \in states(H_1) \mid \alpha' \leq t\text{-}exec(q')\})$. Let $q = q'\lceil M_i$. Then, $q' \in min(q\rceil H_1)$ and q is a state of $min(\{q \in states(H_1\lceil M_i) \mid \alpha \leq t\text{-}exec(q)\})$. Thus, from (9.60) we obtain (9.57).

We now move to the sample spaces. Let α be an element of $\Omega_{t\text{-}exec(\mathcal{P}_{H_1\lceil M_i})}$ that does not contain infinitely many discrete actions. If α ends with δ, then α is trivially an element of $\Omega_{t\text{-}exec(\mathcal{P}_{H_2\lceil M_i})}$ since $P_{t\text{-}exec(\mathcal{P}_{H_2\lceil M_i})}[C_\alpha] = P_{t\text{-}exec(\mathcal{P}_{H_2\lceil M_i})}[C_\alpha] > 0$. Otherwise, α ends with an open trajectory. Then, from the definition of $\Omega_{t\text{-}exec(\mathcal{P}_{H_1\lceil M_i})}$, for each finite prefix α' of α there is an element α_1 of *t-exec*$(\Omega_{H_1\lceil M_i})$ such that $\alpha' \leq \alpha_1$. It is enough to show that for each finite prefix α' of α there is also an element α_2 of *t-exec*$(\Omega_{H_2\lceil M_i})$ such that $\alpha' \leq \alpha_2$.

Let α' be a finite prefix of α such that there is an element α_1 of *t-exec*$(\Omega_{H_1\lceil M_i})$ such that $\alpha' \leq \alpha_1$. Thus, there is a time-enriched execution α_1' of $\Omega_{H_1\lceil M_i}$ such that $\alpha' \leq t\text{-}exec(\alpha_1')$. This means that there is a state q_1 of $H_1\lceil M_i$ such that $\alpha' \leq t\text{-}exec(q_1)$. From the definition of projection, there is a state q_1' of H_1 such that $\alpha' \leq t\text{-}exec(q_1'\lceil M_i)$, and thus there is a timed execution α_1'' of *t-exec*(Ω_{H_1}) such that $\alpha' \leq (\alpha_1''\lceil M_i)$. Consider a finite prefix α_1''' of α_1'' such that $\alpha' \leq (\alpha_1'''\lceil M_i)$. Then, $P_{t\text{-}exec(\mathcal{P}_{H_1})}[C_{\alpha_1'''}] > 0$. Since $H_1 \equiv H_2$, $P_{t\text{-}exec(\mathcal{P}_{H_2})}[C_{\alpha_1'''}] > 0$, which means that there is a timed execution α_2'' of $\Omega_{t\text{-}exec(\mathcal{P}_{H_2})}$ such that $\alpha' \leq (\alpha_2''\lceil M_i)$. Thus, there is a state q_2' of H_2 such that $\alpha' \leq t\text{-}exec(q_2'\lceil M_i)$, and from the definition of projection, there is a state q_2 of $H_2\lceil M_i$ such that $\alpha' \leq t\text{-}exec(q_2)$. This implies that there is an element α_2' of *t-exec*$(\Omega_{H_2\lceil M_i})$ such that $\alpha' \leq \alpha_2'$, which is sufficient to conclude. ∎

9.6 Discussion

To our knowledge, no general probabilistic models with dense time have been proposed except for the automata of Courcoubetis, Alur and Dill [ACD91a, ACD91b]. In our model no probability distributions over passage of time are allowed within a probabilistic timed automaton; time can elapse probabilistically only within a probabilistic timed execution, and the associated probability distributions can be only discrete. We have chosen to define the timed model with such a restriction so that all the theory for the untimed model carries over.

Further work should investigate on the extension of our model to non-discrete probability distributions. A starting point could be the study of restricted forms of non-discrete distributions as it is done by Courcoubetis, Alur and Dill in [ACD91a, ACD91b]. Useful ideas can come from the work on stochastic process algebras of Götz, Herzog and Rettelbach [GHR93], Hillston [Hil94], and Bernardo, Donatiello and Gorrieri [BDG94].

Chapter 10

Direct Verification: Time Complexity

Part of this chapter is based on joint work with Anna Pogosyants and Isaac Saias; some of the ideas have been influenced by discussion with Lenore Zuck. The verification of the randomized dining philosophers algorithm of Lehmann and Rabin (Section 10.6) is based on joint work with Nancy Lynch and Isaac Saias [LSS94]; the verification of the randomized algorithm for agreement of Ben-Or (Section 10.8) is joint work with Anna Pogosyants and is a formalization of a proof that appears in the book on distributed algorithms of Nancy Lynch [Lyn95]. Close interaction with Anna Pogosyants lead us to the idea of the abstract complexity measures of Section 10.7.

10.1 General Considerations About Time

The direct analysis of a probabilistic timed automaton is carried out exactly in the same way as for untimed probabilistic automata. Thus, probabilistic statements and progress statements can be generalized directly, and the coin lemmas can be applied without any modification.

In this chapter we concentrate more on topics that are specific to the presence of time. In particular, it is now possible to enrich the notation for progress statements and verify some of the real-time properties of a probabilistic timed automaton. We extend the progress statements of Chapter 5 by adding a time parameter t: the expression $U \xrightarrow[p]{t} U'$ means that, starting from a state of U, a state of U' is reached within time t with probability at least p. Based on the new *timed progress statements* we show how to derive upper bounds on the worst expected time for progress.

We generalize the method for time complexity analysis to more abstract complexity measures. Then, rather than studying the expected time for progress, we study the expected abstract complexity for progress. We use abstract complexity to derive an upper bound on the worst expected time for decision of the randomized algorithm for agreement of Ben-Or that we presented in Chapter 5. Specifically, we show that under some conditions on the scheduling policy, each non-faulty process completes its i^{th} stage within some upper bound, and we show an upper bound on the expected number of stages that are necessary to reach agreement. In this case the abstract complexity is the number of stages. A direct analysis of the expected time for success in Ben-Or's algorithm would not be as easy since there is no useful upper bound on the time it takes to a process to move from a stage to the next stage.

Sections 10.2, 10.3, and 10.4 simply extend the definitions of Chapter 5 to the timed case; Section 10.5 shows how to derive upper bounds on the worst expected time for progress given a timed progress statement, and Section 10.7 shows how to derive upper bounds on the worst expected abstract complexity for progress given a timed progress statement with abstract complexity; Sections 10.6 and 10.8 present examples of application by proving that the randomized dining philosophers algorithm of Lehmann and Rabin guarantees progress in expected constant time and that the randomized agreement algorithm of Ben-Or guarantees agreement in expected exponential time.

10.2 Adversaries

An *adversary* for a probabilistic timed automaton M is a function $\mathcal{A}$ that takes a finite timed execution fragment α of M and returns a timed transition of M that leaves from $lstate(\alpha)$. Formally,

$$\mathcal{A} : t\text{-}frag^*(M) \rightarrow t\text{-}trans(M)$$

such that if $\mathcal{A}(\alpha) = (s, \mathcal{P})$, then $s = lstate(\alpha)$. Moreover, an adversary satisfies the following *consistency* condition: if $\mathcal{A}(\alpha) = (s, \mathcal{P})$, then for each prefix α' of some element α'' of Ω, $\mathcal{A}(\alpha \frown \alpha') = (lstate(\alpha'), \mathcal{P} \triangleright \alpha')$. Informally, consistency says that an adversary does not change its mind during a timed transition.

An adversary is *deterministic* if it returns either deterministic timed transitions of M or pairs of the form $(s, \mathcal{D}(s\delta))$, i.e., the next timed transition is chosen deterministically. Denote the set of adversaries and deterministic adversaries for a probabilistic timed automaton M by $Advs(M)$ and $DAdvs(M)$, respectively.

The definitions of an adversary schema and of the result of the interaction between an adversary and a probabilistic timed automaton is the same as for the untimed case (cf. Section 5.2), and thus we do not repeat them here.

To guarantee that our adversaries are well defined, we need to prove the following lemma.

Lemma 10.2.1 *If $(s, \mathcal{P})$ is a timed transition of a probabilistic timed automaton M, then for each prefix α' of some element α'' of Ω, $(lstate(\alpha'), \mathcal{P} \triangleright \alpha')$ is a timed transition of M.*

Proof. This is proved already in Proposition 9.3.5. ∎

10.3 Event Schemas

As for the untimed case we need a mechanism to associate an event with each probabilistic timed execution fragment of a probabilistic timed automaton. Thus, an *event schema* is a function e that associates an event of the space $\mathcal{P}_H$ with each probabilistic timed execution fragment H of M. The notion of finite satisfiability extends directly from the untimed case. Observe that, although in $\mathcal{P}_H$ there can be uncountably many cones, each finitely satisfiable event can be expressed as the union of countably many disjoint cones. Furthermore, every uncountable family of cones contains at least two cones that are not disjoint.

The definition of a timed probabilistic statement extends directly from the untimed case, and similarly the definition of the concatenation of two event schemas extends directly. Therefore, we omit the definitions, which are identical to those of Chapter 5.

Proposition 10.3.1 *The concatenation of two event schemas is an event schema. That is, if* $e = e_1 \circ_{Cones} e_2$*, then* e *is an event schema.*

Proof. Consider a probabilistic timed execution fragment H. From Proposition 9.3.3 each set $e_2(H|q)$ is an event of $\mathcal{F}_H$. From the closure of a σ-field under countable union, $e(H)$ is an event of $\mathcal{F}_H$. ■

Proposition 10.3.2 $P_H[e_1 \circ_{Cones} e_2(H)] = \sum_{q \in Cones(H)} P_H[C_q] P_{H|q}[e_2(H|q)]$.

Proof. Since $Cones(H)$ represents a collection of disjoint cones, from (5.13) we obtain

$$P_H[e_1 \circ_{Cones} e_2(H)] = \sum_{q \in Cones(H)} P_H[e_2(H|q)]. \tag{10.1}$$

From Proposition 9.3.3, for each $q \in Cones(H)$

$$P_H[e_2(H|q)] = P_H[C_q] P_{H|q}[e_2(H|q)]. \tag{10.2}$$

By substituting (10.2) in (10.1) we obtain the desired result. ■

Now it is possible to prove a concatenation property similar to the one for the untimed case.

Proposition 10.3.3 *Consider a probabilistic timed automaton* M*. Let*

1. $\Pr_{Advs,\Theta}(e_1) \mathcal{R} p_1$ *and,*
2. *for each* $\mathcal{A} \in Advs$*,* $q \in \Theta$*, let* $\Pr_{Advs, Cones(prexec(M,\mathcal{A},q))}(e_2) \mathcal{R} p_2$.

Then, $\Pr_{Advs,\Theta}(e_1 \circ_{Cones} e_2) \mathcal{R} p_1 p_2$.

Proof. Consider an adversary $\mathcal{A} \in Advs$ and any finite timed execution fragment $q \in \Theta$. Let $H = prexec(M, \mathcal{A}, q)$. From Proposition 10.3.2,

$$P_H[e_1 \circ_{Cones} e_2(H)] = \sum_{q' \in Cones(H)} P_H[C_{q'}] P_{H|q'}[e_2(H|q')]. \tag{10.3}$$

Consider an element q' of $Cones(H)$. It is a simple inductive argument to show that

$$H|q' = prexec(M, \mathcal{A}, q'), \tag{10.4}$$

where we use consistency for the base case. Thus, from our second hypothesis,

$$P_{H|q'}[e_2(H|q')] \mathcal{R} p_2. \tag{10.5}$$

By substituting (10.5) in (10.3), we obtain

$$P_H[e_1 \circ_{Cones} e_2(H)] \mathcal{R} p_2 \sum_{q' \in Cones(e_1(H))} P_H[C_{q'}]. \tag{10.6}$$

By using the fact that $Cones(H)$ is a characterization of $e_1(H)$ as a disjoint union of cones, Equation (10.6) can be rewritten into

$$P_H[e_1 \circ_{Cones} e_2(H)] \mathrel{\mathcal{R}} p_2 P_H[e_1(H)]. \tag{10.7}$$

From the first hypothesis, $P_H[e_1(H)] \mathrel{\mathcal{R}} p_1$; therefore, from Proposition 5.4.1,

$$P_H[e_1 \circ_{Cones} e_2(H)] \mathrel{\mathcal{R}} p_1 p_2. \tag{10.8}$$

This completes the proof. ■

10.4 Timed Progress Statements

As a special case of a probabilistic statement for the timed case we can add some features to the notation $X \xrightarrow[p]{}_{Advs} X'$. In particular we define a *timed progress statement* to assert that starting from a set of states U some other state of a set U' is reached within time t with probability at least p. Such a statement, which we denote by $U \xrightarrow[p]{t}_{Advs} U'$, or by $U \xrightarrow[p]{t} U'$ if *Advs* is clear from the context, is expressed by the probabilistic statement $\Pr_{Advs,U}(e_{U',t}) \geq p$, where the event schema $e_{U',t}$ applied to a timed probabilistic execution fragment H returns the set of timed executions α of Ω_H where a state from U' is reached within time t in $\alpha \triangleright q_0^H$. Such a set can be expressed as a union of cones, and therefore it is an event.

Similarly, the progress statements involving actions can be generalized to the timed framework. Thus, $V \xrightarrow[p]{t}_{Advs} V'$ is the probabilistic statement $\Pr_{Advs,\Theta_{V,V'}}(e_{V',t}) \geq p$, where $\Theta_{V,V'}$ is the set of finite timed execution fragments of M where an action from V occurs and no action from V' occurs after the last occurrence of an action from V, and the event schema $e_{V',t}$ applied to a timed probabilistic execution fragment H returns the set of timed executions α of Ω_H such that an action from V occurs in $\alpha \triangleright q_0^H$ within time t.

In order to generalize the concatenation theorem for progress statements, we need to extend the definition of a finite-history-insensitive adversary schema. Thus, an adversary schema *Advs* is *finite-history-insensitive* iff for each adversary $\mathcal{A}$ of *Advs* and each finite timed execution fragment α of M there is an adversary $\mathcal{A}'$ of *Advs* such that for each timed execution fragment α' such that $\alpha \leq \alpha'$, $\mathcal{A}(\alpha') = \mathcal{A}'(\alpha' \triangleright \alpha)$. Then, the following theorem is shown in the same way as for the untimed case.

Theorem 10.4.1 *Let Advs be finite-history-insensitive. If $X \xrightarrow[p_1]{t_1}_{Advs} X'$ and $X' \xrightarrow[p_2]{t_2}_{Advs} X''$, then $X \xrightarrow[p_1 p_2]{t_1+t_2}_{Advs} X''$.* ■

10.5 Time Complexity

In this section we show how to study the time complexity of a randomized distributed algorithm. We start by defining how to compute a worst expected time, and then we show how it is possible to derive upper bounds on the worst expected running time of an algorithm based on timed progress statements.

10.5.1 Expected Time of Success

Let e be a finitely satisfiable event schema and suppose that $P_H[e(H)] = 1$, i.e., that the property described by e is satisfied in H with probability 1. Let $Cones(H)$ be a characterization of $e(H)$ as a disjoint union of cones, where each element of $Cones(H)$ identifies the first point along a timed execution where the property denoted by e is satisfied. Then, we can compute the expected time to satisfy the property identified by e as

$$\sum_{q \in Cones(H)} P_H[C_q](ltime(q \triangleright q_0^H)). \tag{10.9}$$

In general, if e is a finitely satisfiable event-schema and $Cones(H)$ identifies the first point along a timed execution where the property identified by e is satisfied, then for each probabilistic timed execution fragment H of M we define $E_H[e]$, the expected time to satisfy e in H, as follows.

$$E_H[e] = \begin{cases} \sum_{q \in Cones(H)} P_H[C_q](ltime(q \triangleright q_0^H)) & \text{if } P_H[e(H)] = 1 \\ \infty & \text{otherwise.} \end{cases} \tag{10.10}$$

Then, the question is the following: are there easy ways to compute upper bounds on the expected time for success in a randomized algorithm without computing explicitly (10.10)? We give a positive answer to this question.

10.5.2 From Timed Progress Statements to Expected Times

Timed progress statements can be used to analyze the time complexity of a randomized algorithm. The main idea for the analysis is expressed by Proposition 10.5.1. Suppose that we know the following:

$$\begin{cases} U \xrightarrow[p]{t}_{Advs} U' \\ U \Rightarrow (U \ Unless \ U'). \end{cases} \tag{10.11}$$

Then, if $Advs$ is finite-history-insensitive and $s\delta \notin \Omega_{\mathcal{A}(s)}$ for each $\mathcal{A} \in Advs$ and each $s \in U$, we know from Proposition 5.5.6 that $U \xrightarrow[1]{}_{Advs} U'$. Let e be a finitely satisfiable event schema, and let $Cones$ express the points of satisfaction of e. Suppose that for each probabilistic timed execution fragment H and each state q of H, if there is no prefix q' of q such that $q' \in Cones(H)$, then $e(H \triangleright q) = e(H) \triangleright q$ and $Cones(H \triangleright q) = Cones(H) \triangleright q$ (e.g., e can express the property of reaching some state in a set U'', or the property of performing some action). Let

$$E_{U,Advs}[e] \ \triangleq \ sup_{s \in U, \mathcal{A} \in Advs} E_{prexec(M,\mathcal{A},s)}[e]. \tag{10.12}$$

Then the following property is valid.

Proposition 10.5.1

$$E_{U,Advs}[e] \leq t + pE_{U',Advs}[e] + (1-p)E_{U,Advs}[e]. \tag{10.13}$$

Proof. We prove (10.13) by distinguishing four cases.

1. $E_{U',Advs}[e] \geq E_{U,Advs}[e]$.

   In this case (10.13) is satisfied trivially.

2. $E_{U,Advs}[e] = \infty$ and $p < 1$.

   Also in this case (10.13) is satisfied trivially.

3. $E_{U,Advs}[e] = \infty$ and $p = 1$.

   We show that $E_{U',Advs}[e] = \infty$, which is enough to satisfy (10.13). Suppose by contradiction that $E_{U',Advs}[e] < \infty$. Then we distinguish the following cases.

   (a) There is an adversary $\mathcal{A}$ of *Advs* and a state s of U such that $P_{prexec(M,\mathcal{A},s)}[e(prexec(M,\mathcal{A},s))] < 1$.

   (b) It is not the case that there is an adversary $\mathcal{A}$ of *Advs* and a state s of U such that $P_{prexec(M,\mathcal{A},s)}[e(prexec(M,\mathcal{A},s))] < 1$.

   For Case (a), let $Cones_{U'}$ be the function that expresses the points of satisfaction of $e_{U'}$, and let H be $prexec(M,\mathcal{A},s)$, where $P_{prexec(M,\mathcal{A},s)}[e(prexec(M,\mathcal{A},s))] < 1$. Then,

$$P_H[e(H)] \geq \sum_{q \in Cones_{U'}(H)} P_H[C_q] P_{H \triangleright q}(e(H \triangleright q)), \tag{10.14}$$

   i.e., the probability of satisfying e is not smaller than the probability of reaching U' and then from there satisfying e. From the finite-history-insensitivity of *Advs*, for each state q of $Cones_{U'}(H)$ there is an adversary $\mathcal{A}'$ of *Advs* such that $H \triangleright q = prexec(M,\mathcal{A}',lstate(q))$, and thus, since $E_{U',Advs}[e] < \infty$, $P_{H \triangleright q}(e(H \triangleright q)) = 1$. By substituting this result in (10.14), we get

$$P_H[e(H)] \geq \sum_{q \in Cones_{U'}(H)} P_H[C_q]. \tag{10.15}$$

   Since $p = 1$, the right side of (10.15) is equal to 1, i.e., $P_H[e(H)] \geq 1$, a contradiction.

   For Case (b), let $Cones_{U'}$ be a function that expresses the points of satisfaction of e_U, and, for each $d > 0$, let $Cones_d$ be a function that expresses the event of reaching time d as a union of disjoint cones. From the definition of a probabilistic timed execution, we know that $Cones_d$ exists and that for each probabilistic timed execution fragment H and each $q \in Cones_d(H)$, $ltime(q \triangleright q_0^H) = d$. Let H be $prexec(M,\mathcal{A},s)$. From (10.10) the expected time for success for e is

$$E_H[e] = \sum_{q \in Cones(H)} P_H[C_q] ltime(q \triangleright q_0^H). \tag{10.16}$$

   Let ϵ be an arbitrary positive number. Let Θ_1 be the set of elements q of $Cones_{U'}(H)$ such that $ltime(q \triangleright q_0^H) < t + \epsilon$, and let H_2 be the set of elements q of $Cones_{t+\epsilon}(H)$ that do not have any prefix in Θ_1. Since $P_H[e_U(H)] = 1$, then $P_H[\cup_{q \in \Theta_1 \cup \Theta_2} C_q] = 1$. Moreover, by hypothesis, $P_H[\cup_{q \in Cones(H)} C_q] = 1$. Thus, observe that each element of $Cones(H)$ has either a proper prefix or a suffix in $\Theta_1 \cup \Theta_2$. In fact, if there is an element q of $Cones(H)$ that has no prefix nor suffix in $\Theta_1 \cup \Theta_2$, then the cone C_q would not be part of $\cup_{q \in \Theta_1 \cup \Theta_2} C_q$, contradicting the hypothesis that $P_H[\cup_{q \in Cones(H)} C_q] = 1$. Similarly, we can show that for each element q of $\Theta_1 \cup \Theta_2$ has either a prefix or a proper suffix in $Cones(H)$. Thus, $Cones(H)$ can be partitioned into two sets Θ^p and Θ^s of elements that have a proper prefix and a suffix, respectively, in $\Theta_1 \cup \Theta_2$, and $\Theta_1 \cup \Theta_2$ can be partitioned into two sets $\Theta^p_{1,2}$ and $\Theta^s_{1,2}$ of elements that have a prefix and a proper suffix, respectively, in $Cones(H)$. Based on these observations, the right side of Equation(10.16) can be rewritten into

$$\left(\sum_{q\in\Theta^p}\sum_{q'\in\Theta^s_{1,2}|q'\le q} P_H[C_{q'}]P_{H\triangleright q'}[C_{q\triangleright q'}](ltime(q'\triangleright q_0^H)+ltime(q\triangleright q'))\right) \tag{10.17}$$
$$+\left(\sum_{q\in\Theta^s}\sum_{q'\in\Theta^p_{1,2}|q\le q'} P_H[C_q]P_{H\triangleright q}[C_{q'\triangleright q}]ltime(q\triangleright q_0^H)\right).$$

Observe that for each $q\in\Theta^s$, $\sum_{q'\in\Theta^p_{1,2}|q\le q'}P_{H\triangleright q}[C_{q'\triangleright q}]=1$, and observe that for each $q'\in\Theta^s_{1,2}$, $\sum_{q\in\Theta^p|q'\le q}P_{H\triangleright q'}[C_{q\triangleright q'}]=1$. By exchanging the sums in (10.17) and using some simple algebraic manipulations, we obtain

$$\left(\sum_{q'\in\Theta^s_{1,2}} P_H[C_{q'}]\left(ltime(q'\triangleright q_0^H)+\sum_{q\in\Theta^p|q'\le q}P_{H\triangleright q'}[C_{q\triangleright q'}]ltime(q\triangleright q')\right)\right) \tag{10.18}$$
$$+\left(\sum_{q'\in\Theta^p_{1,2}}\sum_{q\in\Theta^s|q\le q'} P_H[C_q]P_{H\triangleright q}[C_{q'\triangleright q}]ltime(q\triangleright q_0^H)\right).$$

In the first summand, since from the properties of e for each $q'\in\Theta^s_{1,2}$, $e(H\triangleright q')=e(H)\triangleright q'$, the subexpression $\sum_{q\in\Theta^p|q'\le q}ltime(q\triangleright q')P_{H\triangleright q'}[C_{q\triangleright q'}]$ denotes $E_{H\triangleright q'}[e]$. In the second summand, observe that for each $q'\in\Theta^p_{1,2}$ there is exactly one element q of Θ^s such that $q\le q'$. Moreover, $P_H[C_q]P_{H\triangleright q}[C_{q'\triangleright q}]=P_H[C_{q'}]$. Thus, from (10.18) we obtain

$$\begin{aligned} E_H[e] \;\le\; & \left(\sum_{q'\in\Theta^s_{1,2}} P_H[C_{q'}](ltime(q'\triangleright q_0^H)+E_{H\triangleright q'}[e])\right) \\ & + \left(\sum_{q'\in\Theta^p_{1,2}} P_H[C_{q'}]ltime(q'\triangleright q_0^H)\right). \end{aligned} \tag{10.19}$$

By repartitioning $\Theta^s_{1,2}\cup\Theta^p_{1,2}$ into Θ_1 and Θ_2, and by observing that for each element q of Θ_1 $ltime(q\triangleright q_0^H)<t+\epsilon$, and for each element q of Θ_2 $ltime(q\triangleright q_0^H)=t+\epsilon$, (10.19) can be rewritten into

$$E_H[e]\le(t+\epsilon)\left(\sum_{q\in\Theta^s_{1,2}\cap\Theta_1}P_H[C_q]E_{H\triangleright q}[e])\right)+\left(\sum_{q\in\Theta^p_{1,2}\cap\Theta_1}P_H[C_q]E_{H\triangleright q}[e]\right) \tag{10.20}$$
$$+\left(\sum_{q\in\Theta^s_{1,2}\cap\Theta_2}P_H[C_q]E_{H\triangleright q}[e]\right)+\left(\sum_{q\in\Theta^p_{1,2}\cap\Theta_2}P_H[C_q]E_{U,Advs}[e]\right),$$

where we have added $E_{H \triangleright q}[e]$ in the upper right summand and $E_{U,Advs}[e]$ in the lower right summand. Since *Advs* is finite history insensitive, for each $q \in \Theta_1 \cup \Theta_2$ there is an adversary $\mathcal{A}'$ of *Advs* such that $(H \triangleright q) = \mathit{prexec}(M, \mathcal{A}, \mathit{lstate}(q))$. Thus, (10.20) can be rewritten into

$$E_H[e] \leq (t + \epsilon) \left(\sum_{q \in \Theta_1} P_H[C_q] E_{U',Advs}[e]) \right) + \left(\sum_{q \in \Theta_2} P_H[C_q] E_{U,Advs}[e] \right), \qquad (10.21)$$

where we have used $U \Rightarrow (U \text{ } \mathit{Unless} \text{ } U')$ to say that the last states of the elements of Θ_2 are in U. Observe that $\sum_{q \in \Theta_1} P_H[C_q]$ is $P_H[e_{U',t}(H)]$, which is 1 by hypothesis. Since by hypothesis $E_{U',Advs}[e] < \infty$, from (10.21) we derive that $E_{U,Advs}[e] < \infty$, a contradiction.

4. $E_{U,Advs}[e] < \infty$, $E_{U',Advs}[e] < \infty$, and $E_{U',Advs}[e] \leq E_{U',Advs}[e]$.

   Let $\mathcal{A}$ be an adversary of *Advs* and s be a state of U. Let H be $\mathit{prexec}(M, \mathcal{A}, s)$. Let ϵ be any positive real number. Equation (10.21) can be derived also in this case using the same identical argument as before. Since we have assumed that $E_{U',Advs}[e] \leq E_{U,Advs}[e]$, the lowest possible value of the right side of (10.21) occurs by giving U' the lowest possible probability, which is p. Thus, (10.21) becomes

$$E_H[e] \leq (t + \epsilon) p E_{U',Advs}[e] + (1 - p) E_{U,Advs}[e]. \qquad (10.22)$$

   Since Equation (10.22) is valid for any adversary *Advs* and any state of U, we obtain timed execution fragment

$$E_{U,Advs}[e] \leq (t + \epsilon) p E_{U',Advs}[e] + (1 - p) E_{U,Advs}[e]. \qquad (10.23)$$

   Since Equation (10.23) is valid for every ϵ, Equation (10.23) is valid also for the infimum of the values that ϵ can have, i.e., 0, and thus,

$$E_{U,Advs}[e] \leq t + p E_{U',Advs}[e] + (1 - p) E_{U,Advs}[e]. \qquad (10.24)$$

   This completes the proof. ∎

Example 10.5.1 (From timed progress to expected time) As a simple example of application of Proposition 10.5.1, suppose that e expresses the property of reaching U'. Then, we know by definition that $E_{U',Advs}[e] = 0$. By applying Equation (10.13), we obtain $E_{U,Advs}[e] \leq t + (1 - p) E_{U,Advs}[e]$, which gives $E_{U,Advs}[e] \leq t/p$, i.e., the expected time to reach U' from U is at most t/p. Informally speaking, we can view the process of reaching U' as a sequence of Bernoulli trials, each one performed every t time units. At time t, with probability p we have reached U', and with probability $(1 - p)$ we are still in U, and thus we apply the same experiment again. The expected number of rounds of such a process is $1/p$, and thus the expected time for success is t/p. Suppose now that we know the following,

$$\begin{cases} U_0 \xrightarrow[p_1]{t_1}_{Advs} U_1 & U_0 \Rightarrow (U_0 \text{ } \mathit{Unless} \text{ } U_1) \\ U_1 \xrightarrow[p_2]{t_2}_{Advs} U_2 & U_1 \Rightarrow (U_1 \text{ } \mathit{Unless} \text{ } U_2), \end{cases} \qquad (10.25)$$

and suppose that e expresses the property of reaching U_2. Then, we know that $E_{U_2,Advs}[e] = 0$. By applying Proposition 10.5.1, we obtain

$$\left\{\begin{array}{l} E_{U_0,Advs}[e] \leq t_1 + p_1 E_{U_1,Advs}[e] + (1-p_1)E_{U_0,Advs}[e] \\ E_{U_1,Advs}[e] \leq t_2 + (1-p_2)E_{U_1,Advs}[e]. \end{array}\right. \tag{10.26}$$

From simple algebraic manipulations (10.26) becomes

$$\left\{\begin{array}{l} E_{U_0,Advs}[e] \leq t_1/p_1 + E_{U_1,Advs}[e] \\ E_{U_1,Advs}[e] \leq t_2/p_2, \end{array}\right. \tag{10.27}$$

and thus, after substituting the second inequality in the first inequality,

$$\left\{\begin{array}{l} E_{U_0,Advs}[e] \leq t_1/p_1 + t_2/p_2 \\ E_{U_1,Advs}[e] \leq t_2/p_2. \end{array}\right. \tag{10.28}$$

Suppose now that in addition to (10.25) we know that

$$\left\{\begin{array}{l} U_0 \xrightarrow[p_3]{t_3}_{Advs} U_2 \\ U_0 \Rightarrow (U_0 \ \mathit{Unless}\ U_2), \end{array}\right. \tag{10.29}$$

which is possible if $U_1 \subseteq U_0 \cup U_2$. Then, from Proposition 10.5.1 we get

$$E_{U_0,Advs}[e] \leq t_3/p_3, \tag{10.30}$$

which added to (10.28) gives

$$\left\{\begin{array}{l} E_{U_0,Advs}[e] \leq min(t_1/p_1 + t_2/p_2, t_3/p_3) \\ E_{U_1,Advs}[e] \leq t_2/p_2. \end{array}\right. \tag{10.31}$$

Therefore, more information may give us the possibility to prove better bounds. ∎

Proposition 10.5.1 can be proved also for timed progress statements that involve sets of actions rather than sets of states. Let V, V' denote two sets of actions, and let $Advs$ be an adversary schema. Suppose that

$$V \xrightarrow[p]{t}_{Advs} V'. \tag{10.32}$$

Let e be a finitely satisfiable event schema, and let $Cones$ express the points of satisfaction of e. Suppose that for each probabilistic timed execution fragment H and each state q of H, if there is no prefix q' of q such that $q' \in Cones(H)$, then $e(H \triangleright q) = e(H) \triangleright q$ and $Cones(H \triangleright q) = Cones(H) \triangleright q$. Let $E_{V,V',Advs}[e]$ denote $sup_{q \in \Theta_{V,V'}, \mathcal{A} \in Advs} E_{prexec(M,\mathcal{A},q)}[e]$. Let $\Theta_{V'}$ denote the set of finite execution fragments of M whose last action is in V', and let $E_{V',Advs}[e]$ denote $sup_{q \in \Theta_{V'}, \mathcal{A} \in Advs} E_{prexec(M,\mathcal{A},q)}[e]$. Suppose that $q'\delta \notin \Omega_{\mathcal{A}(q)}$ for each q', each $\mathcal{A} \in Advs$ and each $q \in \Theta_{V,V'}$. Then the following proposition is valid.

Proposition 10.5.2

1. *$E_{V,V',Advs}[e] \leq t + pE_{V',Advs}[e] + (1-p)E_{V,V',Advs}[e]$, and*
2. *for each set of actions V'', $E_{V',Advs}[e] \leq E_{V',V'',Advs}[e]$.*

Proof. The proof of the first item follows the lines of the proof of Proposition 10.5.1; the proof of the second item follows from the fact that $\Theta_{V'} \subseteq \Theta_{V',V''}$. ∎

10.6 Example: Randomized Dining Philosophers

To illustrate the use of timed progress statements for the analysis of an algorithm, we reconsider the randomized dining philosophers algorithm of Lehmann and Rabin, and we show that, under the condition that each process has a minimum speed, progress is guaranteed within expected constant time. First, we show how to add time to the probabilistic automaton that describes the algorithm; then, we add time limitations to the progress statements that we used in Section 6.3.3 and we derive the upper bound on the expected time for progress; finally we repeat the low level proof observing that the coin lemmas are applied in the same way as for the untimed case.

10.6.1 Representation of the Algorithm

The probabilistic timed automaton that represent the Algorithm of Lehmann and Rabin can be obtained directly from the probabilistic automaton of Section 6.3.2 by adding arbitrary self-loop time-passage transition from each state (same as the patient construction of Example 9.2.1). Then, in order to enforce a lower bound on the speed of each process, we impose some limitations on the adversaries that act on M. For convenience, but without loss of generality, we assume that from any point each process in its trying or exit region performs one transition within time 1. Thus, the adversary schema that we use on M is the set of adversaries $\mathcal{A}$ for M such that for each finite timed execution fragment α of M,

1. $P_{prexec(M,\mathcal{A},\alpha)}[frag^{\infty}(M)] = 1$, and
2. for each element α' of $\Omega_{prexec(M,\mathcal{A},\alpha)}$ there is no pair of prefixes $\alpha_1 \leq \alpha_2$ of $\alpha' \triangleright \alpha$ and no process i such that process i is in its trying or exit region in $lstate(\alpha_1)$, $ltime(\alpha_2 \triangleright \alpha_1) > 1$, and process i does not perform any discrete transition in $\alpha_2 \triangleright \alpha_1$.

We call this adversary schema *Unit-Time*.

Remark 10.6.1 Observe that in Condition 1 we require the probability of the admissible executions to be 1 rather than requiring the sample space to contain only admissible executions. The reason for using probabilities is technical and is due to the fact that the sample space of a probabilistic timed executions always contains Zeno timed executions, even though they occur with probability 0. From the practical point of view all the Zeno timed executions can be ignored.

In other words, it is not necessary to know the intricacies of the definition of a probabilistic timed executions since they are used only to guarantee that the events of interest are measurable. From the point of view of verifying the correctness of a randomized distributed algorithm, as long as Zeno timed executions occur only with probability 0, it is possible to think that Zeno timed executions do not occur at all. ■

Remark 10.6.2 (Alternative approach) Another alternative approach to modeling the algorithm of Lehmann and Rabin, which we do not use here, is to augment the probabilistic automaton of Section 6.3.2 with an upper bound for each process i to the time by which process i must perform a transition, and to allow a time-passage transition only when no process goes beyond its upper bound. Of course the upper bounds need to be updated opportunely within a transition. In this case the condition imposed on an adversary would be just that time advances unboundedly with probability 1. ■

10.6.2 The High Level Proof

The high level proof consists of the same progress statements that we used in Section 6.3.3 together with a time bound. Specifically, we use the following timed progress statements.

$$
\begin{array}{lll}
\mathcal{T} \xrightarrow[1]{2} \mathcal{RT} \cup \mathcal{C} & & \text{(Proposition 10.6.3)}, \\
\mathcal{RT} \xrightarrow[1]{3} \mathcal{F} \cup \mathcal{G} \cup \mathcal{P} & & \text{(Proposition 10.6.15)}, \\
\mathcal{F} \xrightarrow[1/2]{2} \mathcal{G} \cup \mathcal{P} & & \text{(Proposition 10.6.14)}, \\
\mathcal{G} \xrightarrow[1/4]{5} \mathcal{P} & & \text{(Proposition 10.6.11)}, \\
\mathcal{P} \xrightarrow[1]{1} \mathcal{C} & & \text{(Proposition 10.6.1)}.
\end{array}
$$

By combining the statements above by means of Proposition 5.5.3 and Theorem 10.4.1 we obtain

$$\mathcal{T} \xrightarrow[1/8]{13} \mathcal{C}. \tag{10.33}$$

Observing that if some process is in the trying region then some process is in the trying region unless some process gets to the critical region, we apply Proposition 10.5.1 and we obtain that the expected time to reach $\mathcal{C}$ from $\mathcal{RT}$ is at most 104, i.e., the algorithm of Lehmann and Rabin guarantees progress within expected constant time.

10.6.3 The Low Level Proof

We now prove the timed progress statements of Section 10.6.2. The proofs are exactly the same as the proofs given in Section 6.3.4 with the difference that in this case we consider also time bounds and we consider only admissible timed execution fragments since we know that they occur with probability 1.

Proposition 10.6.1 *If some process is in P, then some process enters C within time 1, i.e.,*

$$\mathcal{P} \xrightarrow[1]{1} \mathcal{C}.$$

Proof. Let i be the process in P. Then, from the definition of *Unit-Time*, process i is scheduled within time 1, and enters C. ∎

Lemma 10.6.2 *If some process is in its Exit region, then it will enter R within time 3.*

Proof. The process needs to perform two transitions to relinquish its two resources, and then one transition to send a `rem` message to the user. Every adversary of *Unit-Time* guarantees that those three transitions are performed within time 3. ∎

Proposition 10.6.3 $\mathcal{T} \xrightarrow{2} \mathcal{RT} \cup \mathcal{C}$.

Proof. From Lemma 6.3.2, every process that begins in E_F or E_S relinquishes its resources within time 2 . If no process begins in C or enters C in the meantime, then the state reached at this point is a state of $\mathcal{RT}$; otherwise, the starting state or the state reached when the first process enters C is a state of $\mathcal{C}$. ■

We now turn to the proof of $\mathcal{G} \xrightarrow[1/4]{5} \mathcal{P}$. The following lemmas form a detailed cases analysis of the different situations that can arise in states of $\mathcal{G}$. Informally, each lemma shows that a specific coin event is a sub-event of the properties of reaching some other state. Here we do not repeat the proof of Lemma 6.3.4 since it does not depend on timing issues.

Lemma 10.6.4

1. Let $X_{i-1} \in \{E_R, R, F\}$ and $X_i = \underleftarrow{W}$. If $FIRST(\texttt{flip}_{i-1}, \texttt{left})$, then, within time 1, either $X_{i-1} = P$ or $X_i = S$.

2. Let $X_{i-1} = D$ and $X_i = \underleftarrow{W}$. If $FIRST(\texttt{flip}_{i-1}, \texttt{left})$, then, within time 2, either $X_{i-1} = P$ or $X_i = S$.

3. Let $X_{i-1} = S$ and $X_i = \underleftarrow{W}$. If $FIRST(\texttt{flip}_{i-1}, \texttt{left})$, then, within time 3, either $X_{i-1} = P$ or $X_i = S$.

4. Let $X_{i-1} = W$ and $X_i = \underleftarrow{W}$. If $FIRST(\texttt{flip}_{i-1}, \texttt{left})$, then, within time 4, either $X_{i-1} = P$ or $X_i = S$.

Proof. The four proofs start in the same way. Let s be a state of M satisfying the respective properties of items *1* or *2* or *3* or *4*. Let $\mathcal{A}$ be an adversary of *Unit-Time*, and let α be an admissible timed execution of $\Omega_{prexec(M,\{s\},\mathcal{A})}$ where the result of the first coin flip of process $i-1$, if it occurs, is `left`.

1. By hypothesis and Lemma 6.3.4, $i-1$ does not hold any resource at the beginning of α and has to obtain Res_{i-2} (its left resource) before pursuing Res_{i-1}. From the definition of *Unit-Time*, i performs a transition within time 1 in α. If $i-1$ does not hold Res_{i-1} when i performs this transition, then i progresses into configuration S. If not, it must be the case that $i-1$ succeeded in getting it in the meanwhile. But, in this case, since $i-1$ flips `left`, Res_{i-1} was the second resource needed by $i-1$ and $i-1$ therefore entered P.

2. If $X_i = S$ within time 1, then we are done. Otherwise, process $i-1$ performs a transition within time 1. Let $\alpha = \alpha_1 \frown \alpha_2$ such that the last transition of α_1 is the first transition taken by process $i-1$. Then $X_{i-1}(\mathit{fstate}(\alpha_2)) = F$ and $X_i(\mathit{fstate}(\alpha_2)) = \underleftarrow{W}$. Since process $i-1$ did not flip any coin during α_1, from the finite-history-insensitivity of *Unit-Time* and Item *1* we conclude.

3. If $X_i = S$ within time 1, then we are done. Otherwise, process $i-1$ performs a transition within time 1. Let $\alpha = \alpha_1 \frown \alpha_2$ such that the last transition of α_1 is the first transition taken by process $i-1$. If $X_{i-1}(\mathit{fstate}(\alpha_2)) = P$ then we are also done. Otherwise it must be the case that $X_{i-1}(\mathit{fstate}(\alpha_2)) = D$ and $X_i(\mathit{fstate}(\alpha_2)) = \underleftarrow{W}$. Since process $i-1$ did not flip any coin during α_1, from the finite-history-insensitivity of *Unit-Time* and Item *2* we conclude.

4. If $X_i = S$ within time 1, then we are done. Otherwise, process i checks its left resource within time 1 and fails, process $i-1$ gets its right resource before, and hence reaches at least state S. Let $\alpha = \alpha_1 \frown \alpha_2$ where the last transition of α_1 is the first transition of α that leads process $i-1$ to state S. Then $X_{i-1}(\mathit{fstate}(\alpha_2)) = S$ and $X_i(\mathit{fstate}(\alpha_2)) = \underleftarrow{W}$. Since process $i-1$ did not flip any coin during α_1, from the finite-history-insensitivity of *Unit-Time* and Item *3* we conclude. ∎

Lemma 10.6.5 *Assume that $X_{i-1} \in \{E_R, R, T\}$ and $X_i = \underleftarrow{W}$. If FIRST*(flip$_{i-1}$, left)*, then, within time 4, either $X_{i-1} = P$ or $X_i = S$.*

Proof. Follows directly from Lemma 10.6.4 after observing that $X_{i-1} \in \{E_R, R, T\}$ is equivalent to $X_{i-1} \in \{E_R, R, F, W, S, D, P\}$. ∎

The next lemma is a useful tool for the proofs of Lemmas 10.6.7, 10.6.8, and 10.6.9. It is just repeated from Section 6.3.4.

Lemma 10.6.6 *Let $X_i \in \{\underleftarrow{W}, \underleftarrow{S}\}$ or $X_i \in \{E_R, R, F, \underleftarrow{D}\}$ with FIRST*(flip$_i$, left)*. Furthermore, let $X_{i+1} \in \{\underrightarrow{W}, \underrightarrow{S}\}$ or $X_{i+1} \in \{E_R, R, F, \underrightarrow{D}\}$ with FIRST*(flip$_{i+1}$, right)*. Then the first of the two processes i or $i+1$ testing its second resource enters P after having performed this test (if this time ever comes).*

Proof. By Lemma 6.3.4 Res_i is free. Moreover, Res_i is the second resource needed by both i and $i+1$. Whichever tests for it first gets it and enters P. ∎

Lemma 10.6.7 *If $X_i = \underleftarrow{S}$ and $X_{i+1} \in \{\underrightarrow{W}, \underrightarrow{S}\}$ then, within time 1, one of the two processes i or $i+1$ enters P. The same result holds if $X_i \in \{\underleftarrow{W}, \underleftarrow{S}\}$ and $X_{i+1} = \underrightarrow{S}$.*

Proof. Being in state S, process i tests its second resource within time 1. An application of Lemma 10.6.6 finishes the proof. ∎

Lemma 10.6.8 *Let $X_i = \underleftarrow{S}$ and $X_{i+1} \in \{E_R, R, F, \underrightarrow{D}\}$. If FIRST*(flip$_{i+1}$, right)*, then, within time 1, one of the two processes i or $i+1$ enters P. The same result holds if $X_i \in \{E_R, R, F, D\}$, $X_{i+1} = \underrightarrow{S}$ and FIRST*(flip$_i$, left)*.*

Proof. Being in state S, process i tests its second resource within time 1. An application of Lemma 10.6.6 finishes the proof. ∎

Lemma 10.6.9 *Assume that $X_{i-1} \in \{E_R, R, T\}$, $X_i = \underleftarrow{W}$, and $X_{i+1} \in \{E_R, R, F, \underrightarrow{W}, \underrightarrow{D}\}$. If FIRST*(flip$_{i-1}$, left) *and FIRST*(flip$_{i+1}$, right)*, then, within time 5, one of the three processes $i-1$, i or $i+1$ enters P.*

Proof. Let s be a state of M such that $X_{i-1}(s) \in \{E_R, R, T\}$, $X_i(s) = \underleftarrow{W}$, and $X_{i+1}(s) \in \{E_R, R, F, \underrightarrow{W}, \underrightarrow{D}\}$. Let $\mathcal{A}$ be an adversary of *Unit-Time*, and let α be an admissible timed execution of $\Omega_{prexec(M,\{s\},\mathcal{A})}$ where the result of the first coin flip of process $i-1$ is `left` and the result of the first coin flip of process $i+1$ is `right`. By Lemma 10.6.5, within time 4 either process $i-1$ reaches configuration P in α or process i reaches configuration $\underleftarrow{S}$ in α. If $i-1$

reaches configuration P, then we are done. If not, then let $\alpha = \alpha_1 \frown \alpha_2$ such that $\mathit{lstate}(\alpha_1)$ is the first state s' of α with $X_i(s') = \underleftarrow{S}$. If $i+1$ enters P before the end of α_1, then we are done. Otherwise, $X_{i+1}(\mathit{fstate}(\alpha_2))$ is either in $\{\underrightarrow{W}, \underrightarrow{S}\}$ or it is in $\{E_R, R, F, \underrightarrow{D}\}$ and process $i+1$ has not flipped any coin yet in α. From the finite-history-insensitivity of *Unit-Time* we can then apply Lemma 10.6.6: within time 1 process i tests its second resource and by Lemma 10.6.6 process i enters P if process $i+1$ did not check its second resource in the meantime. If process $i+1$ checks its second resource before process i does the same, then by Lemma 10.6.6 process $i+1$ enters P. ■

Lemma 10.6.10 *Assume that* $X_{i+2} \in \{E_R, R, T\}$*,* $X_{i+1} = \underrightarrow{W}$*, and* $X_i \in \{E_R, R, F, \underleftarrow{W}, \underleftarrow{D}\}$*. If* $\mathit{FIRST}(\mathtt{flip}_i, \mathtt{left})$ *and* $\mathit{FIRST}(\mathtt{flip}_{i+2}, \mathtt{right})$*, then, within time* 5*, one of the three processes* i*,* $i+1$ *or* $i+2$*, enters* P*.*

Proof. The proof is analogous to the one of Lemma 10.6.9. This lemma is the symmetric case of Lemma 10.6.9. ■

Proposition 10.6.11 *Starting from a global configuration in* $\mathcal{G}$*, then, with probability at least* 1/4*, some process enters* P *within time* 5*. Equivalently:*

$$\mathcal{G} \xrightarrow[1/4]{5} \mathcal{P}.$$

Proof. Lemmas 10.6.7 and 10.6.8 jointly treat the case where $X_i = \underleftarrow{S}$ and $X_{i+1} \in \{E_R, R, F, \underrightarrow{\#}\}$ and the symmetric case where $X_i \in \{E_R, R, F, \underleftarrow{\#}\}$ and $X_{i+1} = \underrightarrow{S}$; Lemmas 10.6.9 and 10.6.10 jointly treat the case where $X_i = \underleftarrow{W}$ and $X_{i+1} \in \{E_R, R, F, \underrightarrow{W}, \underrightarrow{D}\}$ and the symmetric case where $X_i \in \{E_R, R, F, \underleftarrow{W}, \underleftarrow{D}\}$ and $X_{i+1} = \underrightarrow{W}$.

Specifically, each lemma shows that a compound event of the kind $\mathit{FIRST}(\mathtt{flip}_i, x)$ and $\mathit{FIRST}(\mathtt{flip}_j, y)$ leads to $\mathcal{P}$. Each of the basic events $\mathit{FIRST}(\mathtt{flip}_i, x)$ has probability at least 1/2. From Lemma 6.2.4 each of the compound events has probability at least 1/4. Thus the probability of reaching $\mathcal{P}$ within time 5 is at least 1/4. ■

We now turn to $\mathcal{F} \xrightarrow[1/2]{2} \mathcal{G} \cup \mathcal{P}$. The proof is divided in two parts and constitute the global argument of the proof of progress, i.e., the argument that focuses on the whole system rather than on a couple of processes.

Lemma 10.6.12 *Start with a state* s *of* $\mathcal{F}$*. If there exists a process* i *for which* $X_i(s) = F$ *and* $(X_{i-1}, X_{i+1}) \neq (\underrightarrow{\#}, \underleftarrow{\#})$*, then, with probability at least* 1/2 *a state of* $\mathcal{G} \cup \mathcal{P}$ *is reached within time* 1*.*

Proof. If $s \in \mathcal{G} \cup \mathcal{P}$, then the result is trivial. Let s be a state of $\mathcal{F} - (\mathcal{G} \cup \mathcal{P})$ and let i be such that $X_i(s) = F$ and $(X_{i-1}, X_{i+1}) \neq (\underrightarrow{\#}, \underleftarrow{\#})$. Assume without loss of generality that $X_{i+1} \neq \underleftarrow{\#}$, i.e., $X_{i+1} \in \{E_R, R, F, \underrightarrow{\#}\}$. The case for $X_{i-1} \neq \underrightarrow{\#}$ is similar. Furthermore, we can assume that $X_{i+1} \in \{E_R, R, F, \underrightarrow{D}\}$ since if $X_{i+1} \in \{\underrightarrow{W}, \underrightarrow{S}\}$ then s is already in $\mathcal{G}$. We show that the event schema $\mathit{FIRST}((\mathtt{flip}_i, \mathtt{left}), (\mathtt{flip}_{i+1}, \mathtt{right}))$, which by Lemma 6.2.2 has probability at least 1/2, leads eventually to a state of $\mathcal{G} \cup \mathcal{P}$. Let $\mathcal{A}$ be an adversary of *Unit-Time*, and let α be an admissible timed execution of $\Omega_{prexec(M,\{s\},\mathcal{A})}$ where if process i flips before process $i+1$ then process i flips left, and if process $i+1$ flips before process i then process $i+1$ flips right.

Then, within time 1, i performs one transition and reaches W. Let $j \in \{i, i+1\}$ be the first of i and $i+1$ that reaches W and let s_1 be the state reached after the first time process j reaches W. If some process reached P in the meantime, then we are done. Otherwise there are two cases to consider. If $j = i$, then, $\texttt{flip}_i$ gives $\texttt{left}$ and $X_i(s_1) = \underleftarrow{W}$ whereas X_{i+1} is (still) in $\{E_R, R, F, \underrightarrow{D}\}$. Therefore, $s_1 \in \mathcal{G}$. If $j = i+1$, then $\texttt{flip}_{i+1}$ gives $\texttt{right}$ and $X_{i+1}(s_1) = \underrightarrow{W}$ whereas $X_i(s_1)$ is (still) F. Therefore, $s_1 \in \mathcal{G}$. ■

Lemma 10.6.13 *Start with a state s of $\mathcal{F}$. If there exists a process i for which $X_i(s) = F$ and $(X_{i-1}(s), X_{i+1}(s)) = (\underrightarrow{\#}, \underleftarrow{\#})$. Then, with probability at least $1/2$, a state of $\mathcal{G} \cup \mathcal{P}$ is reached within time 2.*

Proof. The hypothesis can be summarized into the form $(X_{i-1}(s), X_i(s), X_{i+1}(s)) = (\underrightarrow{\#}, F, \underleftarrow{\#})$. Since $i-1$ and $i+1$ point in different directions, by moving to the right of $i+1$ there is a process k pointing to the left such that process $k+1$ either points to the right or is in $\{E_R, R, F, P\}$, i.e., $X_k(s) \in \{\underleftarrow{W}, \underleftarrow{S}, \underleftarrow{D}\}$ and $X_{k+1}(s) \in \{E_R, R, F, \underrightarrow{W}, \underrightarrow{S}, \underrightarrow{D}, P\}$.

If $X_k(s) \in \{\underleftarrow{W}, \underleftarrow{S}\}$ and $X_{k+1}(s) \neq P$ then $s \in \mathcal{G}$ and we are done; if $X_{k+1}(s) = P$ then $s \in \mathcal{P}$ and we are done. Thus, we can restrict our attention to the case where $X_k(s) = \underleftarrow{D}$.

We show that $FIRST((\texttt{flip}_k, \texttt{left}), (\texttt{flip}_{k+1}, \texttt{right}))$, which by Lemma 6.2.2 has probability at least $1/2$, leads to $\mathcal{G} \cup \mathcal{P}$ within time 2. Let $\mathcal{A}$ be an adversary of *Unit-Time*, and let α be an admissible timed execution of $\Omega_{prexec(M,\{s\},\mathcal{A})}$ where if process k flips before process $k+1$ then process k flips left, and if process $k+1$ flips before process k then process $k+1$ flips right.

Within time 2 process k performs at least two transitions and hence goes to configuration W. Let $j \in \{k, k+1\}$ be the first of k and $k+1$ that reaches W and let s_1 be the state reached after the first time process j reaches W. If some process reached P in the meantime, then we are done. Otherwise, we distinguish two cases. If $j = k$, then, $\texttt{flip}_k$ gives $\texttt{left}$ and $X_k(s_1) = \underleftarrow{W}$ whereas X_{k+1} is (still) in $\{E_R, R, F, \underrightarrow{\#}\}$. Thus, $s_1 \in \mathcal{G}$. If $j = k+1$, then $\texttt{flip}_{k+1}$ gives $\texttt{right}$ and $X_{k+1}(s_1) = \underrightarrow{W}$ whereas $X_k(s_1)$ is (still) in $\{\underleftarrow{D}, F\}$. Thus, $s_1 \in \mathcal{G}$. ■

Proposition 10.6.14 *Start with a state s of $\mathcal{F}$. Then, with probability at least $1/2$, a state of $\mathcal{G} \cup \mathcal{P}$ is reached within time 2. Equivalently:*

$$\mathcal{F} \xrightarrow[1/2]{2} \mathcal{G} \cup \mathcal{P}.$$

Proof. The hypothesis of Lemmas 10.6.12 and 10.6.13 form a partition of $\mathcal{F}$. ■

Finally, we prove $\mathcal{RT} \xrightarrow[1]{} \mathcal{F} \cup \mathcal{G} \cup \mathcal{P}$.

Proposition 10.6.15 *Starting from a state s of $\mathcal{RT}$, then a state of $\mathcal{F} \cup \mathcal{G} \cup \mathcal{P}$ is reached within time 3 Equivalently:*

$$\mathcal{RT} \xrightarrow[1]{3} \mathcal{F} \cup \mathcal{G} \cup \mathcal{P}.$$

Proof. Let s be a state of $\mathcal{RT}$. If $s \in \mathcal{F} \cup \mathcal{G} \cup \mathcal{P}$, then we are trivially done. Suppose that $s \notin \mathcal{F} \cup \mathcal{G} \cup \mathcal{P}$. Then in s each process is in $\{E_R, R, W, S, D\}$ and there exists at least process in $\{W, S, D\}$. Let $\mathcal{A}$ be an adversary of *Unit-Time*, and let α be an admissible timed execution of $\Omega_{prexec(M,\{s\},\mathcal{A})}$.

We first argue that within time 1 some process reaches a state of $\{S, D, F\}$ in α. This is trivially true if in state s there is some process in $\{S, D\}$. If this is not the case, then all processes are either in E_R or R or W. Eventually, some process in R or W performs a transition. If the first process not in E_R performing a transition started in E_R or R, then it reaches F and we are done; if the first process performing a transition is in W, then it reaches S since in s no resource is held. Once a process i is in $\{S, D, F\}$, then within time 2 process i reaches either state F or P, and we are done. ∎

10.7 Abstract Complexity Measures

We have seen how to measure the expected time to satisfy a property. However, the technique can be extended to other kinds of measures of complexity. Specifically, let ϕ be a complexity measure on timed execution fragments that is additive under concatenation, i.e., $\phi(q_1 \frown q_2) = \phi(q_1) + \phi(q_2)$. Then we can compute the expected ϕ rather than the expected time, where the ϕ of a state q of H is defined to be $\phi(q \triangleright q_0^H)$. We generalize the notation for timed progress statements by writing

$$U \xrightarrow[p]{\phi(c)}_{Advs} U' \tag{10.34}$$

with the meaning that $\Pr_{Advs,U}(e_{U',\phi(c)}) \geq p$, where the event schema $e_{U',\phi(c)}$ applied to a timed probabilistic execution fragment H returns the set of timed executions α of Ω_H where a state from U' is reached within complexity c. More specifically, let $Cones_{U',\phi(c)}(H)$ be the set of minimal timed execution fragments q of M such that C_q^H is not empty, $lstate(q) \in U'$, and $\phi(q \triangleright q_0^H) \leq c$. Then, $e_{U',\phi(c)}(H) = \cup_{q \in Cones_{U',\phi(c)}(H)} C_q^H$. Observe that time is just one of the possible complexity measures.

The same definition can be extended to sets of actions as we have done previously, and the concatenation theorem is still valid.

The expected complexity of a finitely satisfiable event schema can be defined easily. Specifically, if e is a finitely satisfiable event-schema and $Cones(H)$ identifies the points of satisfaction of e, then for each probabilistic timed execution fragment H of M we define $E_{H,\phi}[e]$, the expected complexity to satisfy e in H, as follows.

$$E_{H,\phi}[e] = \begin{cases} \sum_{q \in Cones(H)} P_H[C_q](\phi(q \triangleright q_0^H)) & \text{if } P_H[e(H)] = 1 \\ \infty & \text{otherwise.} \end{cases} \tag{10.35}$$

Then, a proposition similar to Proposition 10.5.1 can be proved.

Proposition 10.7.1 *Suppose that*

$$\begin{cases} U \xrightarrow[p]{\phi(c)}_{Advs} U' \\ U \Rightarrow (U \ \mathit{Unless}\ U'), \end{cases} \tag{10.36}$$

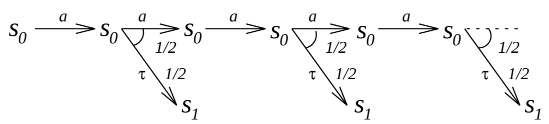

Figure 10-1: An example of the use of ξ.

and suppose that Advs is finite history insensitive and that $s\delta \notin \Omega_{\mathcal{A}(s)}$ for each $\mathcal{A} \in Advs$ and each $s \in U$. Then,

$$E_{U,Advs,\phi}[e] \leq c + pE_{U',Advs,\phi}[e] + (1-p)(\xi + E_{U,Advs,\phi}[e]), \tag{10.37}$$

where

$$\xi = sup_{q \in t\text{-}frag^*(M) \mid lstate(q) \in U}\left(sup_{q'>q}\left(inf_{q'' \mid q<q'' \leq q'}(\phi(q'' \triangleright q))\right)\right). \tag{10.38}$$

Proof. This proof has the same structure as the proof of Proposition 10.5.1. Here we describe in detail only the main differences. In particular, we show part of the derivation from Equation (10.16) to Equation (10.21), where the constant ξ is used. Observe that if we use ϕ to express time complexity, then $\xi = 0$.

From (10.35) the expected complexity for success for e is

$$E_{H,\phi}[e] = \sum_{q \in Cones(H)} P_H[C_q]\phi(q \triangleright q_0^H). \tag{10.39}$$

For each $d > 0$, let $Cones_d$ be a function that expresses the event of reaching complexity d as a union of disjoint cones. From the definition of a probabilistic timed execution, we know that $Cones_d$ exists and, from (10.38), we know that for each probabilistic timed execution fragment H and each $q \in Cones_d(H)$, $d \leq \phi(q \triangleright q_0^H) \leq d + \xi$. Let ϵ be any positive number. Following the same derivation as in the proof of Proposition 10.5.1, we obtain

$$E_{H,\phi}[e] \leq (c+\epsilon)\left(\sum_{q \in \Theta_1} P_H[C_q]E_{U',Advs,\phi}[e]\right) + \left(\sum_{q \in \Theta_2} P_H[C_q](\xi + E_{U,Advs,\phi}[e])\right). \tag{10.40}$$

∎

One of the novel aspects of Proposition 10.7.1 is the constant ξ. Roughly speaking, ξ gives us a lower bound to the minimum complexity increase that we can obtain by moving along a timed execution fragment.

Example 10.7.1 (Why ξ is necessary) For example, if the abstract complexity that we use is the number of discrete actions that appear in a timed execution fragment, then $\xi = 1$. In fact, whenever we perform a discrete action, the complexity increases by 1. Figure 10-1 shows an example where $\xi = 1$ and where Equation (10.37) is invalidated if we do not include ξ. Denote the probabilistic timed execution fragment of Figure 10-1 by H. Let U be $\{s_0\}$, U' be $\{s_1\}$, and let e express the property of reaching U'. Let $Advs$ contain only one adversary that generates H when applied to s_0. Let ϕ count the number of external actions in a timed execution fragment (no time-passage actions in H). Then, it is immediate to verify that the statement $U \xrightarrow[1/2]{\phi(1)} U'$ is valid in H and that also $U \Rightarrow (U\ \mathit{Unless}\ U')$ is valid. By applying Equation (10.37) with $\xi = 1$, we obtain

$$E_{U,Advs,\phi}[e] \leq t + 1/2(1 + E_{U,Advs,\phi}[e]), \tag{10.41}$$

which leads to $E_{U,Advs,\phi}[e] \leq 3$. If we did not use ξ in Equation (10.37) we would have obtained $E_{U,Advs,\phi}[e] \leq 2$. We now show that $E_{H,\phi}[e] = 3$. In fact,

$$E_{H,\phi}[e] = \frac{1}{2} + 3\frac{1}{4} + 5\frac{1}{8} + 7\frac{1}{16} + \cdots \tag{10.42}$$

By rearranging the terms, we obtain

$$E_{H,\phi}[e] = \sum_{i \geq 0} \frac{1}{2^i} \left(\frac{1}{2} + \frac{2}{4} + \frac{2}{8} + \frac{2}{16} + \cdots \right). \tag{10.43}$$

Recall that $\sum_{i \geq 0} 1/2^i = 2$. Thus, by rearranging the terms again,

$$E_{H,\phi}[e] = 2 + 1/2 \left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \cdots \right) = 3. \tag{10.44}$$

Roughly speaking, the transition relation of H is structured in such a way that whenever the experiment of reaching U' from U fails, the system looses one additional complexity unit during the random draw. In the proof of Proposition 10.7.1 this phenomenon is detected when we define the partition Θ_1 and Θ_2. To make sure that Θ_1 and Θ_2 partition an event with probability 1 and that Θ_1 captures all the places where U' is reached within time t, Θ_2 must be based on states reached after time t. In the probabilistic execution H of this example the states of Θ_2 have complexity $t + 1$. ■

10.8 Example: Randomized Agreement with Time

Using abstract complexity measures it is possible to show that the randomized agreement algorithm of Ben-Or guarantees agreement within an expected exponential time. This is not an exceptional complexity result, but it corresponds to the time complexity of the algorithm.

In more detail, we add time to the probabilistic automaton that describes Ben-Or's protocol in the same way as we have done for the Dining Philosophers algorithm of Lehmann and Rabin. In this case each adversary is required to schedule every process that enables some transition within time 1 from every point. Then we show an upper bound linear in st on the time it takes to all processes to complete a specific stage st. Finally, we derive an upper bound on the expected number of stages it takes for all processes to decide. This is achieved by defining an abstract complexity on the timed executions of M that checks the highest stage reached at every point. A direct extension of the untimed proof without abstract complexities would not be possible. In fact, given a reachable state s, the validity of the progress statement of Chapter 6 relies on completing the highest stage reached in s, and we cannot establish any useful upper bound on the time to complete such stage: there is no useful bound on the difference between the highest and the lowest stages reached in s, and the adversary may stop the processes with the highest values of st. We start by proving the upper bound on the time it takes to each process to complete some stage st.

Lemma 10.8.1 *There is a constant d such that, for each stage st, each process completes stage st within time $d \cdot st$.*

Proof. Let d_1 be the maximum time it takes to each process from the moment it reaches a new stage st to the moment it broadcasts its value and its value is delivered; let d_2 be the maximum time it takes to each process to broadcast and deliver its second message after receiving enough messages from the first round; let d_3 be the maximum time it takes to each process to move to a new stage once it has received enough messages from the second round. Then $d = d_1 + d_2 + d_3$. Since we have not defined formally M, we cannot say explicitly what is the value of d.

We show the result by induction on st where for the base case we assume that $st = 0$ and that stage 0 is completed by time 0. By induction, by time $d \cdot st$ each non-faulty process has completed round st. Then, by time $d_1 + d \cdot st$ each non-faulty process has broadcasted and delivered its first round message, and thus every non-faulty process has received enough messages for the first round of stage $st + 1$. Within additional time d_2 each non-faulty process delivers its second message, and within additional time d_3 each non-faulty process reaches stage $st + 2$, i.e., within time $d(st + 1)$ each non-faulty process completes stage $st + 1$. ■

For each finite timed execution fragment α of M define $\phi(\alpha)$, the stage complexity of α, to be $\mathit{max\text{-}stage}(\mathit{lstate}(\alpha)) - \mathit{max\text{-}stage}(\mathit{fstate}(\alpha))$, where for each state s, $\mathit{max\text{-}stage}(s)$ is the maximum stage that is reached in s by some process. Observe that this complexity measure is an upper bound to the stage at which some process decides since if at state s the first process has just decided, then $\mathit{max\text{-}stage}(s)$ is not smaller than the stage of the process that has decided. Thus, an upper bound on the expected ϕ for the decision of the first process is an upper bound on the expected stage at which the first process decides. We show the following two statements.

$$\mathcal{B} \xrightarrow[1]{\phi(1)}_{f\text{-}fair} \mathcal{F} \cup \mathcal{O}. \tag{10.45}$$

$$\mathcal{F} \xrightarrow[1/2^n]{\phi(2)} \mathcal{O}. \tag{10.46}$$

Then, by combining (10.45) and (10.46) with Theorem 5.5.2, we obtain

$$\mathcal{B} \xrightarrow[1/2^n]{\phi(3)} \mathcal{O}. \tag{10.47}$$

From Proposition 10.7.1, we obtain

$$E_{\mathcal{B},\mathit{Unit\text{-}Time},\phi}[e_{\mathcal{O}}] \leq 3 + (1 - 1/2^n)(1 + E_{\mathcal{B},\mathit{Unit\text{-}Time},\phi}[e_{\mathcal{O}}]), \tag{10.48}$$

where 1 is the value of ξ given by (10.38). By solving Equation (10.48) we obtain

$$E_{\mathcal{B},\mathit{Unit\text{-}Time},\phi}[e_{\mathcal{O}}] \leq 2^{n+2} - 1. \tag{10.49}$$

Since if a process decides at stage st then each other non-faulty process decides within stage $st + 1$, then we can derive that the expected stage by which every process decides is at most 2^{n+2}, and thus, from Lemma 10.8.1, each process decides within expected time $d \cdot 2^{n+1}$.

The proofs for (10.45) and (10.46) have the same structure as the corresponding proofs for the untimed case. Recall that the proof of (10.45) consider the maximum stage st of a reachable state s and states that eventually stage $st + 1$ is reached, at which time a state of $\mathcal{F}$ is reached. The proof of (10.46) states that a specific coin lemma leads a process to decide by stage $\mathit{max\text{-}stage}(s) + 1$. Then, since if a process decides a stage st each process decides by stage $st + 1$, the complexity of the state where the first process decides is at most $\mathit{max\text{-}stage}(s) + 2$.

10.9 Discussion

To our knowledge this is the first time that statements similar to our timed progress statements have been used for the analysis of the performance of a randomized distributed algorithm. In particular, we have been able to prove similar results only because we have studied techniques to prove properties that hold with some probability different than 1. This should be a sufficiently strong reason to pursue additional research on methodologies (automatic or not) for the analysis of properties that hold with probabilities different than 1. The work of Hansson [Han94] and the algorithm that Courcoubetis and Yannakakis present in [CY90] are in this direction.

Chapter 11

Hierarchical Verification: Timed Trace Distributions

11.1 Introduction

In this chapter we extend the trace distribution preorder of Chapter 7 to the timed framework. The main difference is that we use *timed traces* rather than traces. A timed trace contains the sequence of discrete actions that occur within a timed execution plus the time of occurrence of each action and the time at which the observation ends. That is, in a timed execution we observe at what time each external action occurs and, if finitely many actions occur, how much time elapses after the occurrence of the last action.

We define a preorder relation based on *timed trace distribution* inclusion, and we characterize the coarsest precongruence that is contained in the timed trace distribution preorder by using a *timed principal context*, which is just the principal context of Chapter 7 augmented with arbitrary time-passage self-loop transitions from its unique state. Most of the proofs follow directly from the results already proved in Chapter 7, since in several cases it is sufficient to study ordinary trace distributions in order to derive properties of timed trace distributions.

11.2 Timed Traces

We start by defining the main object of observation, i.e., timed traces. The definition of a timed trace that we give in this section is taken directly from [LV95].

Timed Sequence Pairs

Let K be any set that does not intersect $\Re^{+}$. Then a *timed sequence* over K is defined to be a (finite or infinite) sequence γ over $K \times \Re^{\geq 0}$ in which the time components are nondecreasing, i.e., if (k, t) and (k', t') are consecutive elements in γ then $t \leq t'$. We say that γ is *Zeno* if it is infinite and the limit of the time components is finite.

A *timed sequence pair* over K is a pair $\beta = (\gamma, t)$, where γ is a timed sequence over K and $t \in \Re^{\geq 0} \cup \{\infty\}$, such that t is greater than or equal to all time components in γ. We write *seq*(β), and *ltime*(β) for the two respective components of β. We denote by *tsp*(K) the set of timed sequence pairs over K. We say that a timed sequence pair β is *finite* if both $seq(\beta)$ and $ltime(\beta)$ are finite, and *admissible* if $seq(\beta)$ is not Zeno and $ltime(\beta) = \infty$.

Let β and β' be timed sequence pairs over K with β finite. Then define $\beta;\beta'$ to be the timed sequence pair $(seq(\beta)\gamma, ltime(\beta) + ltime(\beta'))$, where γ is the modification of $seq(\beta')$ obtained by adding $ltime(\beta)$ to all the time components. If β and β' are timed sequence pairs over a set K, then β is a *prefix* of β', denoted by $\beta \leq \beta'$, if either $\beta = \beta'$, or β is finite and there exists a timed sequence pair β'' such that $\beta' = \beta;\beta''$.

Lemma 11.2.1 *$\leq$ is a partial ordering on the set of timed sequence pairs over K.* ∎

Now we describe how to translate from a sequence over $K \cup \Re^+$, and ordinary trace, to a timed sequence pair over K. First, if β is any sequence over $K \cup \Re^+$, then we define the *time of occurrence* of any K-element in β to be the sum of all the reals that precede that element in β. We also define $ltime(\beta)$ to be the sum of all the reals in β. Finally, we define $t\text{-}trace(\beta)$ to be the timed sequence pair $(\gamma, ltime(\beta))$, where γ is the subsequence of β consisting of all the elements of K, each paired with its time of occurrence.

If β is a sequence over $K \cup \Re^+$ then we say that β is *admissible* if the sum of the positive reals in β is infinite.

Lemma 11.2.2 *If β is a finite or admissible timed sequence pair then $t\text{-}trace(trace(\beta)) = \beta$.* ∎

Lemma 11.2.3 *If β is a sequence over $K \cup \Re^+$ then β is admissible if and only if $t\text{-}trace(\beta)$ is admissible.* ∎

Timed Traces of Timed Probabilistic Automata

Suppose that $\alpha = \omega_0 a_1 \omega_1 a_2 \omega_2 \cdots$ is a timed execution fragment of a timed probabilistic automaton M. For each a_i, define the *time of occurrence* t_i to be $\sum_{j<i} ltime(\omega_j)$, i.e., the sum of the lengths of all the trajectory intervals preceding a_i in α. Let γ be the sequence consisting of the actions in α paired with their times of occurrence:

$$\gamma = (a_1, t_1)(a_2, t_2) \cdots.$$

Then $t\text{-}trace(\alpha)$, the *timed trace* of α, is defined to be the pair

$$(\gamma \restriction (vis(M) \times \Re^+), ltime(\alpha)).$$

Thus, $t\text{-}trace(\alpha)$ records the occurrences of visible actions together with their times of occurrence, and together with the time spanned by α. Note that neither internal actions nor time-passage actions appear explicitly in the timed trace of α.

Proposition 11.2.4 *If α is a timed execution fragment of M then $t\text{-}trace(\alpha)$ is a timed sequence pair over $vis(M)$.* ∎

Proposition 11.2.5 *Let α be a timed execution fragment of M, and let $trace(\alpha)$ denote the ordered sequence of external actions that appear in α. Then, $t\text{-}trace(\alpha) = t\text{-}trace(trace(\alpha))$.* ∎

Proposition 11.2.6 *If $\alpha = \alpha_1 \frown \alpha_2$ is a timed execution fragment of M, then $t\text{-}trace(\alpha) = t\text{-}trace(\alpha_1); t\text{-}trace(\alpha_2)$.* ∎

We write *t-traces*(M) for the set of all timed traces of M, *t-traces*$^*(M)$ for the set of *finite* timed traces of M, and *t-traces*$^\infty(M)$ for the set of *admissible* timed traces of M,

The timed traces of a probabilistic timed automaton M can be characterized also in terms of its time-enriched executions or in terms of its ordinary executions. Specifically, if α is a time-enriched execution of M, then let *t-trace*(α) denote *t-trace*$(\textit{t-exec}(\alpha))$, and if α is an execution of M, then let *t-trace*(α) denote *t-trace*$(\textit{trace}(\alpha))$. The following proposition holds.

Proposition 11.2.7 *Let M be a probabilistic timed automaton.*

1. *If α is a time-enriched execution of M, then there is a timed execution α' of M such that t-trace(α) = t-trace(α').*

2. *If α is a timed execution of M, then there is a time-enriched execution α' of M such that t-trace(α) = t-trace(α').*

3. *If α is a timed execution of M, then there is an execution α' of M such that t-trace(α) = t-trace(α').*

4. *If α is an execution of M, then there is a timed execution α' of M such that t-trace(α) = t-trace(α').*

Proof.

1. Let α' be *t-exec*(α). Then, *t-trace*(α) = *t-trace*(α') by definition.

2. Let α be $\omega_0 a_1 \omega_1 a_2 \cdots$. If α is a finite timed execution or an infinite sequence, then let $\alpha' = \mathit{fstate}(\omega_0) \frown \alpha_1 \frown \alpha_2 \frown \cdots$, where for each i,

$$\alpha_i = \begin{cases} \omega_{i-1} a_i \mathit{fstate}(\omega_i) & \text{if } \omega_{i-1} \text{ has domain } [0,0], \\ \mathit{fstate}(\omega_{i-1}) \mathit{ltime}(\omega_{i-1}) \omega_{i-1} a_i \mathit{fstate}(\omega_i) & \text{otherwise;} \end{cases}$$

   if $\alpha = \omega_0 a_1 \omega_1 a_2 \cdots a_n \omega_n$ and the domain of ω_n is right-open, then let $\alpha' = \mathit{fstate}(\omega_0) \frown \alpha_1 \frown \cdots \frown \alpha_n \frown \alpha'_{n+1}$, where the α_i's are defined above and $\alpha'_{n+1} = \omega'_0 d_1 \omega'_1 d_2 \omega'_2 \cdots$ is an infinite sequence such that $\omega'_0 \omega'_1 \omega'_2 \cdots = \omega_n$. It is immediate to verify that α and α' have the same timed trace since $\alpha = \textit{t-exec}(\alpha')$.

3. Let α be $\omega_0 a_1 \omega_1 a_2 \cdots$. If α is a finite timed execution or an infinite sequence, then let $\alpha' = \mathit{fstate}(\omega_0) \frown \alpha_1 \frown \alpha_2 \frown \cdots$, where for each i,

$$\alpha_i = \begin{cases} \mathit{lstate}(\omega_{i-1}) a_i \mathit{fstate}(\omega_i) & \text{if } \omega_{i-1} \text{ has domain } [0,0], \\ \mathit{fstate}(\omega_{i-1}) \mathit{ltime}(\omega_{i-1}) \mathit{lstate}(\omega_{i-1}) a_i \mathit{fstate}(\omega_i) & \text{otherwise;} \end{cases}$$

   if $\alpha = \omega_0 a_1 \omega_1 a_2 \cdots a_n \omega_n$ and the domain of ω_n is right-open, then let $\alpha'' = \mathit{fstate}(\omega_0) \frown \alpha_1 \frown \cdots \frown \alpha_n \frown \alpha'_{n+1}$, where the α_i's are defined above and $\alpha'_{n+1} = \mathit{fstate}(\omega_n) d_1 \omega_n(d_1) d_2 \omega_n(d_1 + d_2) \cdots$ is an infinite sequence such that $\sum_i d_i = \mathit{ltime}(\omega_n)$. It is immediate to verify that α and α' have the same timed trace.

4. Given $\alpha = s_0 a_1 s_1 a_2 \cdots$, build a time-enriched execution α'' by replacing each state s_i with a trajectory for (s_{i-1}, a_i, s_i) whenever a_i is a time-passage action. Then, $t\text{-}trace(\alpha) = t\text{-}trace(\alpha'')$. Item 2 is enough to conclude. ∎

The bottom line of the proposition above is that for the study of the timed traces of a probabilistic timed automaton it is not necessary to observe the trajectories spanned by a computation. The points of occurrence of discrete actions are sufficient.

11.3 Timed Trace Distributions

In this section we define the timed trace distributions of a probabilistic timed automaton and we extend the action restriction operation. The main result is that it is possible to study the timed trace distributions of a probabilistic timed automaton M by considering either its probabilistic executions, or its probabilistic time-enriched executions, or its probabilistic timed executions.

11.3.1 Three ways to Define Timed Trace Distributions

We now define the timed trace distribution of a probabilistic execution, of a probabilistic time-enriched execution, and of a probabilistic timed execution of a probabilistic timed automaton. The definitions are given in the same style as for the untimed case. Furthermore, we show that the three definitions lead to the same collection of timed trace distributions. This enforces the remark that for the study of the timed trace distributions of a probabilistic timed automaton it is not necessary to observe the trajectories spanned by a computation.

Timed Trace Distribution of a Probabilistic Execution

Let H be a probabilistic execution of a probabilistic timed automaton M, and let f be a function from Ω_H to $\Omega = tsp(vis(M))$ that assigns to each extended execution its timed trace. The *timed trace distribution* of H, denoted by $t\text{-}tdistr(H)$, is the probability space $completion((\Omega, \mathcal{F}, P))$ where $\mathcal{F}$ is the σ-field generated by the cones C_β, where β is a finite timed sequence pair of $tsp(vis(M))$, and $P = f(P_H)$. Note that from Proposition 3.1.4 f is a measurable function from $(\Omega_H, \mathcal{F}_H)$ to $(\Omega, \mathcal{F})$.

Timed Trace Distribution of a Probabilistic Time-Enriched Execution

Let H be a probabilistic time-enriched execution of a probabilistic timed automaton M, and let f be a function from Ω_H to $\Omega = tsp(vis(M))$ that assigns to each time-enriched extended execution its timed trace. The *timed trace distribution* of H, denoted by $t\text{-}tdistr(H)$, is the probability space $(\Omega, \mathcal{F}, P)$ where $\mathcal{F}$ is the σ-field generated by the cones C_β, where β is a finite timed timed sequence pair of $tsp(vis(M))$, and $P = f(P_H)$. Note that from Proposition 3.1.4 f is a measurable function from $(\Omega_H, \mathcal{F}_H)$ to $(\Omega, \mathcal{F})$.

Timed Trace Distribution of a Probabilistic Timed Execution

Let H be a probabilistic timed execution of a probabilistic timed automaton M, and let f be a function from Ω_H to $\Omega = tsp(vis(M))$ that assigns to each timed extended execution its timed trace. The *timed trace distribution* of H, denoted by *t-tdistr*(H), is the probability space $(\Omega, \mathcal{F}, P)$ where $\mathcal{F}$ is the σ-field generated by the cones C_β, where β is a finite timed timed sequence pair of *tsp*(*vis*(M)), and $P = f(P_H)$. Note that from Proposition 3.1.4 f is a measurable function from $(\Omega_H, \mathcal{F}_H)$ to $(\Omega, \mathcal{F})$.

Equivalence of the Definitions

We now show that the three definitions of a timed trace distribution lead to the same collection of timed trace distributions when applied to a probabilistic timed automaton (cf. Propositions 11.3.2 and 11.3.4). Thus, we can freely denote a generic timed trace distribution by $\mathcal{D}$ and denote the timed trace distributions of a probabilistic tomed automaton M by *t-tdistrs*(M).

Lemma 11.3.1 *Let H be a probabilistic time-enriched execution of a probabilistic timed automaton M. Then, t-tdistr(H) = t-tdistr(sample(H)).*

Proof. Let $\mathcal{D}$ be *t-tdistr*(H) and let $\mathcal{D}'$ be *t-tdistr*(*sample*(H)) Consider a finite timed trace β. From the definition of *t-tdistr*(),

$$P_{\mathcal{D}'}[C_\beta] = P_{sample(H)}[\{\alpha \in \Omega_{sample(H)} \mid \beta \leq t\text{-}trace(\alpha)\}]. \tag{11.1}$$

Since C_β is a finitely satisfiable event, there is a set of Θ of states of *sample*(H) such that for each element q of Θ, $\beta \leq t\text{-}trace(q)$, and such that

$$\{\alpha \in \Omega_{sample(H)} \mid \beta \leq t\text{-}trace(\alpha)\} = \cup_{q\in\Theta} C_q^{sample(H)}. \tag{11.2}$$

Thus,

$$P_{\mathcal{D}'}[C_\beta] = \sum_{q\in\Theta} P_{sample(H)}[C_q^{sample(H)}]. \tag{11.3}$$

From Equation (9.55), Equation (11.3) becomes

$$P_{\mathcal{D}'}[C_\beta] = \sum_{q\in sample^{-1}(\Theta)} P_H[C_q^H]. \tag{11.4}$$

Observe that $sample^{-1}(\Theta)$ is a characterization of C_β for $\mathcal{D}$, and thus,

$$P_{\mathcal{D}'}[C_\beta] = P_{\mathcal{D}}[C_\beta]. \tag{11.5}$$

This completes the proof. ∎

Proposition 11.3.2 *Let M be a probabilistic timed automaton. Then, for each probabilistic time-enriched execution H of M there exists a probabilistic execution H' of M such that t-tdistr(H) = t-tdistr(H'), and for each probabilistic execution H of M there exists a probabilistic time-enriched execution H' of M such that t-tdistr(H) = t-tdistr(H').*

Proof. Follows directly from Propositions 9.3.6 and 9.3.7, and from Lemma 11.3.1. ∎

Lemma 11.3.3 *Let H be a probabilistic time-enriched execution of a probabilistic timed automaton M. Then, t-tdistr(H) = t-tdistr(t-sample(H)).*

Proof. Let $\mathcal{D}$ be *t-tdistr*(H), and let $\mathcal{D}'$ be *t-tdistr*(*t-sample*(H)). Consider a finite timed sequence pair $\mathcal{D}$ of *tsp*(*vis*(M)). From the definition of *t-tdistr*,

$$P_{\mathcal{D}}[C_\beta] = P_H[\{\alpha \in \Omega_H \mid \beta \leq \textit{t-trace}(\alpha)\}]. \tag{11.6}$$

From the definition of *t-exec*$(\mathcal{P}_H)$,

$$P_{\mathcal{D}}[C_\beta] = P_{\textit{t-exec}(\mathcal{P}_H)}[\{\alpha \in \Omega_{\textit{t-exec}(H)} \mid \beta \leq \textit{t-trace}(\alpha)\}]. \tag{11.7}$$

With a similar analysis,

$$P_{\mathcal{D}'}[C_\beta] = P_{\textit{t-sample}(H)}[\{\alpha \in \Omega_{\textit{t-sample}(H)} \mid \beta \leq \textit{t-trace}(\alpha)\}]. \tag{11.8}$$

Since from Proposition 9.3.11 *t-exec*$(\mathcal{P}_H) = \mathcal{P}_{\textit{t-sample}(H)}$, and since the events of (11.7) and (11.8) are unions of countably many disjoint cones, we conclude that $P_{\mathcal{D}}[C_\beta] = P_{\mathcal{D}'}[C_\beta]$. ∎

Proposition 11.3.4 *Let M be a probabilistic timed automaton. Then, for each probabilistic time-enriched execution H of M there exists a probabilistic timed execution H' of M such that t-tdistr(H) = t-tdistr(H'), and for each probabilistic timed execution H of M there exists a probabilistic time-enriched execution H' of M such that t-tdistr(H) = t-tdistr(H').*

Proof. Follows directly from Propositions 9.3.8 and 9.3.9, and from Lemma 11.3.3. ∎

Proposition 11.3.5 *Let H_1 and H_2 be two equivalent probabilistic time-enriched executions of a probabilistic timed automaton M. Then, t-tdistr(H_1) = t-tdistr(H_2).*

Proof. From Proposition 9.3.10, *t-sample*(H_1) = *t-sample*(H_2), and from Lemma 11.3.3, *tdistr*(H_1) = *tdistr*(*t-sample*(H_1)) and *tdistr*(H_2) = *tdistr*(*t-sample*(H_2)). Thus, combining the observations above, *t-tdistr*(H_1) = *t-tdistr*(H_2). ∎

11.3.2 Timed Trace Distribution of a Trace Distribution

Given a trace distribution of a probabilistic timed automaton, it is possible to define its timed trace distribution as we have done for ordinary traces. Thus, let $\mathcal{D}$ be a trace distribution of a probabilistic automaton, and let f be a function from $\Omega_{\mathcal{D}}$ to $\Omega = \{\textit{t-trace}(\beta) \mid \beta \in \Omega_{\mathcal{D}}\}$ that assigns to each trace its timed trace. The *timed trace distribution* of $\mathcal{D}$, denoted by *t-tdistr*$(\mathcal{D})$, is the probability space *completion*$((\Omega, \mathcal{F}, P))$ where $\mathcal{F}$ is the σ-field generated by the cones C_β, where β is a finite timed trace, and $P = f(P_{\mathcal{D}})$. Note that from Proposition 3.1.4 f is a measurable function from $(\Omega_{\mathcal{D}}, \mathcal{F}_{\mathcal{D}})$ to $(\Omega, \mathcal{F})$.

Proposition 11.3.6 *Let H be a probabilistic execution of a timed probabilistic automaton M. Then, t-tdistr(H) = t-tdistr(tdistr(H)).*

Proof. Let $\mathcal{D}$ be *t-tdistr*(H), and let $\mathcal{D}'$ be *t-tdistr*(*tdistr*(H)). We show first that $\mathcal{D}$ and $\mathcal{D}'$ have the same sample space. Then, we show that they assign the same probability to each cone.

To show that $\mathcal{D}$ and $\mathcal{D}'$ have the same sample space, it is enough to show that for each timed sequence pair β of *tsp*(*vis*(M)) thehre is a trace β' of $\textit{ext}(M)^* \cup \textit{ext}(M)^\omega$ such that *t-trace*$(\beta') = \beta$. Let $(\beta = (a_1, t_1)(a_2, t_2), (a_3, t_3) \cdots, t)$. If *seq*$(\beta)$ is an infinite sequence, then let $\beta' = \beta_1\beta_2\beta_3\cdots$, where for each i, if $t_{i+1} = t_i$, then $\beta_i = a_i$, and if $t_{i+1} > t_i$, then $\beta_i =$

$a_i(t_{i+1} - t_i)$. If $seq(\beta)$ is a finite sequence, i.e., $seq(\beta) = (a_1, t_1)(a_2, t_2), (a_3, t_3) \cdots, (a_n, t_n)$ then $\beta' = \beta_1\beta_2\beta_3 \cdots \beta_{n-1}\beta'_n$ where the β_i's are defined above, and β'_n is a_n if $t_n = t$, $a_n(t - t_n)$ if $0 < t - t_n < \infty$, and a_n followed by the infinite sequence of 1's if $t = \infty$. It is easy to verify that in every case $t\text{-}trace(\beta') = \beta$.

To show that $\mathcal{D}$ and $\mathcal{D}'$ assign the same probability to each cone, let β be a finite timed trace. From the definition of *t-tdistr* and *tdistr*,

$$P_{\mathcal{D}'}[C_\beta] = P_H[\{\alpha \in \Omega_H \mid \beta \leq t\text{-}trace(trace(\alpha))\}]. \tag{11.9}$$

From Proposition 11.2.5, (11.9) becomes

$$P_{\mathcal{D}'}[C_\beta] = P_H[\{\alpha \in \Omega_H \mid \beta \leq t\text{-}trace(\alpha)\}], \tag{11.10}$$

which is the definition of $P_{\mathcal{D}}[C_\beta]$. ■

11.3.3 Action Restriction

Finally, we extend the action restriction operator to timed trace distributions. Let M be a probabilistic timed automaton, and let V be a set of visible actions of M. For each timed trace $\beta = (\gamma, t)$ of M, let $\beta \upharpoonright V$ be the pair (γ', t) where γ' is obtained from γ by removing all the pairs whose action is in V. Let $\mathcal{D}$ be a timed trace distribution of M. Define $\mathcal{D} \upharpoonright V$ to be the timed trace distribution $(\Omega, \mathcal{F}, P)$ where $\Omega = \Omega_{\mathcal{D}} \upharpoonright V$, $\mathcal{F}$ is the σ-field generated by the cones C_β, where β is a finite timed trace, and $P = P_{\mathcal{D}} \upharpoonright V$. Note that from Proposition 3.1.4 $\upharpoonright V$ is a measurable function from $(\Omega_{\mathcal{D}}, \mathcal{F}_{\mathcal{D}})$ to $(\Omega, \mathcal{F})$. Action restriction commutes with the operation of taking a timed trace distribution of a trace distribution.

Proposition 11.3.7 *Let $\mathcal{D}$ be a trace distribution of a probabilistic timed automaton M, and let V be a set of visible actions of M. Then, $t\text{-}tdistr(\mathcal{D} \upharpoonright V) = t\text{-}tdistr(\mathcal{D}) \upharpoonright V$.*

Proof. Let $\mathcal{D}'$ be $t\text{-}tdistr(\mathcal{D} \upharpoonright V)$, and let $\mathcal{D}''$ be $t\text{-}tdistr(\mathcal{D}) \upharpoonright V$. Let β be a finite timed trace. By applying the definitions of *t-tdistr* and of $\upharpoonright$, we obtain the following two equations.

$$P_{\mathcal{D}'}[C_\beta] = P_{\mathcal{D}}[\{\beta' \in \Omega_{\mathcal{D}} \mid \beta \leq t\text{-}trace(\beta' \upharpoonright V)\}]. \tag{11.11}$$

$$P_{\mathcal{D}''}[C_\beta] = P_{\mathcal{D}}[\{\beta' \in \Omega_{\mathcal{D}} \mid \beta \leq t\text{-}trace(\beta') \upharpoonright V\}]. \tag{11.12}$$

Observe that for each β' of $\Omega_{\mathcal{D}}$, $t\text{-}trace(\beta' \upharpoonright V) = t\text{-}trace(\beta') \upharpoonright V$. Thus, the right expressions of (11.11) and (11.12) denote the same value. That is, $P_{\mathcal{D}'}[C_\beta] = P_{\mathcal{D}''}[C_\beta]$. ■

11.4 Timed Trace Distribution Precongruence

Let M_1, M_2 be two probabilistic timed automata with the same external actions. The *timed trace distribution preorder* is defined as follows.

$$M_1 \sqsubseteq_{Dt} M_2 \text{ iff } t\text{-}tdistrs(M_1) \subseteq t\text{-}tdistrs(M_2).$$

As for the untimed case, the timed trace distribution preorder is not a precongruence. A counterexample can be created directly from the counterexample of Chapter 7 by augmenting the probabilistic automata of Figure 7-4 with arbitrary self-loop time-passage transitions from their deadlock states (the states that do not enable any transition). Thus, we define the *timed trace distribution precongruence*, denoted by $\sqsubseteq_{DCt}$, as the coarsest precongruence that is contained in the timed trace distribution preorder.

11.5 Alternative Characterizations

The timed trace distribution precongruence can be characterized by a timed version of the principal context of Chapter 7. Namely, let the *timed principal context*, denoted by C_P be the principal context of Figure 7-6 augmented with self-loop time-passage transitions for each time-passage action d. Then, the following holds.

Theorem 11.5.1 *$M_1 \sqsubseteq_{DCt} M_2$ iff $M_1 \| C_P \sqsubseteq_{Dt} M_2 \| C_P$.*

Thus, if we define the *principal timed trace distributions* of a probabilistic timed automaton M, denoted by *pt-tdistrs*(M), to be the timed trace distributions of $M \| C_P$, then we get the following.

Corollary 11.5.2 *$M_1 \sqsubseteq_{DCt} M_2$ iff $ext(M_1) = ext(M_2)$ and $\textit{pt-tdistrs}(M_1) \subseteq \textit{pt-tdistrs}(M_2)$.*
■

The rest of this section is dedicated to the proof of Theorem 11.5.1. The structure of the proof follows the same lines as the proof of Theorem 7.5.1, where only one additional transformation step is added: a distinguishing context is transformed into a new *time-deterministic* context where each state enables either discrete actions only or time-passage actions only. A time-deterministic context is a probabilistic automaton such that for each state s and each time-passage action d, if $s \xrightarrow{d} s_1$ and $s \xrightarrow{d} s_2$, then $s_1 = s_2$. All the lemmas except for one are proved by reducing the problem to the untimed framework.

Lemma 11.5.3 *Let C be a distinguishing context for two probabilistic timed automata M_1 and M_2. Then there exists a distinguishing context C' for M_1 and M_2 with no discrete actions in common with M_1 and M_2. C' is called a* separated context.

Proof. The context C' is built from C in the same way as in the proof of Lemma 7.5.3. The constructions *clp* and *exch* work as well (they never exchange transitions involving time-passage), and the proof is carried out at the level of probabilistic executions rather than probabilistic timed executions.

Specifically, let $\mathcal{D}$ be a timed trace distribution of $M_1 \| C$ that is not a timed trace distribution of $M_2 \| C$. Consider a probabilistic execution H_1 of $M_1 \| C$ such that $\textit{t-tdistr}(H_1) = \mathcal{D}$, and consider the scheduler that leads to H_1. Apply to $M_1 \| C'$ the same scheduler with the following modification: whenever a transition $((s_1, c), a, \mathcal{P}_1 \otimes \mathcal{P})$ is scheduled in $M_1 \| C$, schedule $((s_1, c), a_1, \mathcal{D}((s_1, c')))$, where c' is $c_{(c,a,\mathcal{P})}$, followed by $((s_1, c'), a, \mathcal{P}_1 \otimes \mathcal{D}(c'))$, and, for each $s_1' \in \Omega_1$, followed by $((s_1', c'), a_2, \mathcal{D}(s_1') \otimes \mathcal{P})$. Denote the resulting probabilistic execution by H_1' and the resulting timed trace distribution by $\mathcal{D}'$. From Lemma 7.5.3, $tdistr(H_1) = tdistr(H_1') \restriction vis(M_1 \| C)$, and thus, from Propositions 11.3.6 and 11.3.7, $\mathcal{D} = \mathcal{D}' \restriction vis(M_1 \| C)$.

Suppose by contradiction that it is possible to obtain $\mathcal{D}'$ from $M_2 \| C'$. Consider the scheduler that leads to $\mathcal{D}'$ in $M_2 \| C'$, and let H_2' be the corresponding probabilistic execution. Then, from Lemma 7.5.3, $clp(exch(H_2'))$ is a probabilistic execution of $M_2 \| C'$, and $tdistr(clp(exch(H_2'))) = tdistr(H_2') \restriction acts(M_1 \| C)$. From Propositions 11.3.6 and 11.3.7, $\mathcal{D} = \textit{t-tdistr}(clp(exch(H_2')))$, which is a contradiction. ■

Lemma 11.5.4 *Let C be a distinguishing separated context for two probabilistic timed automata M_1 and M_2. Then there exists a distinguishing cycle-free separated context C' for M_1 and M_2.*

Proof. The context C' can be built by unfolding C. Every scheduler for C can be transformed into a scheduler for C' and vice versa, leading to the same timed trace distributions. ■

Lemma 11.5.5 *Let C be a distinguishing cycle-free, separated context for two probabilistic timed automata M_1 and M_2. Then there exists a distinguishing time-deterministic, cycle-free separated context C' for M_1 and M_2 that from any state enables either time-passage actions only or discrete actions only.*

Proof. The context C' is built from C as follows:

1. for each time-passage transition $s \xrightarrow{d} s'$ of C and each trajectory ω for $s \xrightarrow{d} s'$, add an action $start_\omega$ and an action end_ω;

2. for each time-passage transition $s \xrightarrow{d} s'$ of C and each trajectory ω for $s \xrightarrow{d} s'$, add a collection of new states $\{s_{\omega,t} \mid 0 \leq t \leq d\}$, a transition $s \xrightarrow{start_\omega} s_{\omega,0}$, a transition $s_{\omega,d} \xrightarrow{end_\omega} s'$, and for each $0 \leq t < t' \leq d$, a transition $s_{\omega,t} \xrightarrow{t'-t} s_{\omega,t'}$;

3. remove all the time-passage transitions leaving from states of C.

Let $\mathcal{D}$ be a timed trace distribution of $M_1\|C$ that is not a timed trace distribution of $M_2\|C$. Consider a probabilistic execution H_1 of $M_1\|C$ such that $t\text{-}tdistr(H_1) = \mathcal{D}$, and consider the scheduler that leads to H_1. Apply to $M_1\|C'$ the same scheduler with the following modification: whenever a time-passage transition $s \xrightarrow{d} s'$ is scheduled, choose a trajectory ω for $s \xrightarrow{d} s'$ and schedule $start_\omega$, followed by d, and followed by end_ω. Denote the resulting probabilistic execution by H_1' and the resulting timed trace distribution by $\mathcal{D}'$. Then,

$$\mathcal{D}' \restriction acts(M_1\|C) = \mathcal{D}. \tag{11.13}$$

To prove (11.13) we prove first that $tdistr(H_1') \restriction acts(M_1\|C) = tdistr(H_1)$, and then we apply Propositions 11.3.6 and 11.3.7. To prove that $tdistr(H_1') \restriction acts(M_1\|C) = tdistr(H_1)$ we define a construction *tclp* to be applied to probabilistic executions of $M_i\|C'$ where each occurrence of a start action is followed eventually by the corresponding end action with probability 1.

Let H' be a probabilistic execution of $M_i\|C'$ where each occurrence of a start action is followed eventually by the corresponding end action with probability 1, and denote $tclp(H')$ by H. For each state q of H', let $tclp(q)$ be obtained from q by replacing each state of the form $s_{\omega,t}$ with the state $\omega(t)$, by removing each occurrence of a start action together with its following state, and by removing each end action together with its following state. Then,

$$states(H) \triangleq tclp(states(H')). \tag{11.14}$$

Let $(q, \mathcal{P})$ be a restricted transition of H', and suppose that no start or end action occurs. Let $\Omega' = \{(a, tclp(q')) \mid (a, q') \in \Omega\}$, and for each $(a, q'') \in \Omega'$, let $P'[(a, q'')] = P[a \times tclp^{-1}(q'')]$, where $tclp^{-1}(q)$ is the set of states q' of H' such that $tclp(q') = q$. Then the transition $tclp((q, \mathcal{P}))$ is defined to be

$$tclp((q, \mathcal{P})) \triangleq (tclp(q), \mathcal{P}). \tag{11.15}$$

For the transition relation of H, consider a state q of H, and let $min(tclp^{-1}(q))$ be the set of minimal states of $tclp^{-1}(q)$ under prefix ordering. For each state $\bar{q} \in tclp^{-1}(q)$, let

$$\bar{p}_{\bar{q}}^{tclp^{-1}(q)} \triangleq \frac{P_{H'}[C_{\bar{q}}]}{\sum_{q' \in min(tclp^{-1}(q))} P_{H'}[C_{q'}]}. \tag{11.16}$$

The transition enabled from q in H is

$$\sum_{q' \in tclp^{-1}(q)} \bar{p}_{\bar{q}}^{tclp^{-1}(q)} P_{q'}^{H'}[acts(M_i \| C)] tclp(tr_{q'}^{H'} \restriction acts(M_i \| C)). \tag{11.17}$$

The probabilistic execution H satisfies the following properties.

a. H is a probabilistic execution of $M_i \| C$.

   The fact that each state of H is reachable can be shown by a simple inductive argument; the fact that each state of H is a finite execution fragment of $M_i \| C$ follows from a simple analysis of the definition of $tclp$.

   From (11.17) it is enough to check that for each state q' of H', the transition $tclp(tr_{q'}^{H'} \restriction acts(M_i \| C))$ is generated a combined transition of $M_i \| C$. Since $tr_{q'}^{H'}$ is a transition of H', $(tr_{q'}^{H'} \restriction acts(M_i \| C))$ can be expressed as $q' \frown tr$, where tr is a combined transition of $M_i \| C'$ and no start or end action occurs in tr. Let tr' be obtained by substituting each state of the form $s_{\omega,t}$ with $\omega(t)$ in tr. Then, tr' is a combined transition of $M \| C$, and, from the definition of $tclp$, $tclp(tr_{q'}^{H'} \restriction acts(M_i \| C)) = tclp(q') \frown tr'$.

b. For each state q of H,

$$P_H[C_q] = \sum_{q' \in min(tclp^{-1}(q))} P_{H'}[C_{q'}]. \tag{11.18}$$

   This is shown by induction on the length of q. If q consists of a start state only, then the result is trivial. Otherwise, from the definition of the probability of a cone, Equation (11.17), and a simple algebraic simplification,

$$P_H[C_{qas}] = P_H[C_q] \left(\sum_{q' \in tclp^{-1}(q)} \bar{p}_{q'}^{tclp^{-1}(q)} P_{q'}^{H'}[a \times tclp^{-1}(qas)] \right). \tag{11.19}$$

   Observe that for each $q' \in tclp^{-1}(q)$ the set $\Omega_{q'}^{H'} \cap (\{a\} \times tclp^{-1}(qas))$ contains only one element, say $(a, q'as'')$, and thus $P_{H'}[C_{q'}] P_{q'}^{H'}[a \times tclp^{-1}(qas)]$ gives $P_{H'}[C_{q'as''}]$. Moreover, observe that the states of $min(tclp^{-1}(qas))$ are the states of the form described in Equation (11.19) (simple cases analysis). Thus, by applying induction to (11.19), using (11.16), simplifying algebraically, and using the observations above,

$$P_H[C_{qas}] = \sum_{q' \in min(tclp^{-1}(qas))} P_{H'}[C_{q'}]. \tag{11.20}$$

c. $tdistr(H) = tdistr(H') \upharpoonright acts(M_i \| C)$.

Let β be a finite trace of H or H'. Then $\{\alpha \in \Omega_{H'} \mid \beta \leq trace(\alpha) \upharpoonright acts(M_i \| C)\}$ can be expressed as a union of disjoint cones $\cup_{q \in \Theta} C_q$ where

$$\Theta = \{q \in states(H') \mid trace(q) \upharpoonright acts(M_i \| C) = \beta, lact(q) = lact(\beta)\}. \tag{11.21}$$

The set $tclp(\Theta)$ is the set

$$tclp(\Theta) = \{q \in states(H) \mid trace(q) = \beta, lact(q) = lact(\beta)\}, \tag{11.22}$$

which is a characterization of $\{\alpha \in \Omega_H \mid \beta \leq trace(\alpha)\}$ as a union of disjoint cones. Observe that $min(tclp^{-1}(tclp(\Theta))) = \Theta$. Moreover, for each $q_1 \neq q_2$ of $tclp(\Theta)$, $tclp^{-1}(q_1) \cap tclp^{-1}(q_2) = \emptyset$. Thus, from (11.18), $P_{H'}[\cup_{q \in \Theta} C_q] = P_H[\cup_{q \in tclp(\Theta)} C_q]$. This is enough to conclude.

To complete the proof of (11.13) it is enough to observe that $H_1 = tclp(H_1')$. Property (11.13) is then expressed by property (c).

Suppose by contradiction that it is possible to obtain $\mathcal{D}'$ from $M_2 \| C'$. Consider the scheduler that leads to $\mathcal{D}'$ in $M_2 \| C'$, and let H_2' be the corresponding probabilistic execution. Observe that, since the timed trace distribution of H_2' is $\mathcal{D}'$, and since by construction in $\mathcal{D}'$ each occurrence of a start action is followed eventually by the corresponding end action with probability 1, in H_2' each occurrence of a start action is followed eventually by the corresponding end action with probability 1. Thus, $tclp$ can be applied, and $t\text{-}tdistr(tclp(H_2')) = \mathcal{D}$, which is a contradiction. ∎

Lemma 11.5.6 *Let C be a distinguishing time-deterministic, cycle-free, separated context for two probabilistic timed automata M_1 and M_2 that from any state enables either time-passage actions only or discrete actions only. Then there exists a distinguishing time-deterministic, cycle-free separated context C' for M_1 and M_2 that from any state enables either time-passage actions only or discrete actions only, and such that the transition relation from any state enabling discrete actions is at most countably branching. C' is called a* time-deterministic, countably-branching, cycle-free separated context*.*

Proof. Let $\mathcal{D}$ a timed trace distribution of $M_1 \| C$ that is not a timed trace distribution of $M_2 \| C$. Consider one of the corresponding probabilistic executions H. Observe that H has at most countably many states that enable discrete actions, and that at each state of H there are at most countably many transitions of C that are scheduled. Thus, in total, only countably many discrete transitions of C are used to generate $\mathcal{D}$. Then C' is C without the useless discrete transitions. ∎

Lemma 11.5.7 *Let C be a distinguishing time-deterministic, countably-branching, cycle-free separated context for two probabilistic timed automata M_1 and M_2. Then there exists a distinguishing cycle-free separated context C' for M_1 and M_2 that at each state enabling discrete actions either enables two deterministic transitions or a unique probabilistic transition with two possible outcomes. C' is called a* time-deterministic, binary separated context*.*

Proof. The context C' is built from C in the same way as in the proof of Lemma 7.5.6. The constructions *shr* and *shf* work as well. The specific procedure is the same as the procedure followed in the proof of Lemma 11.5.3. ■

Lemma 11.5.8 *Let C be a distinguishing time-deterministic, binary separated context for two probabilistic timed automata M_1 and M_2. Then there exists a distinguishing time-deterministic, binary separated context C' for M_1 and M_2 where all the probabilistic transitions have a uniform distribution over two states. C' is called a* time-deterministic, balanced separated context.

Proof. The context C' is built from C in the same way as in the proof of Lemma 7.5.7. The specific procedure is the same as the procedure followed in the proof of Lemma 11.5.3. ■

Lemma 11.5.9 *Let C be a distinguishing time-deterministic, balanced separated context for two probabilistic timed automata M_1 and M_2. Then there exists a distinguishing time-deterministic, binary separated context C' for M_1 and M_2 with no internal actions and such that for each time t each discrete action appears exactly in one edge of the transition tree that leaves from a state whose time is t. C' is called a* time-deterministic, total balanced separated context.

Proof. The context C' is obtained from C by renaming all of its discrete actions so that for each time t each edge of the new transition relation leaving from a state whose current time is t has its own action. The proof of Lemma 7.5.8 applies. ■

Lemma 11.5.10 *Let C be a distinguishing time-deterministic, total balanced separated context for two probabilistic timed automata M_1 and M_2. Then there exists a distinguishing time-deterministic, total, cycle-free separated context C' for M_1 and M_2 that from every state enables one time-passage transition for each timed-action d, two deterministic transitions, and a probabilistic transition with a uniform distribution over two choices. C' is called a* complete context.

Proof. In this case it is enough to complete C by adding all the missing transitions and states. If $\mathcal{D}$ is a timed trace distribution of $M_1\|C$ that is not a timed trace distribution of $M_2\|C$, then it is enough to use on $M_1\|C'$ the same scheduler that is used in $M_1\|C$. In fact, since each new discrete transition of C' has a distinct action, none of the new discrete transitions of C' can be used in $M_2\|C'$ to generate $\mathcal{D}$, and since each state of C' is uniquely determined by the timed trace of all the executions leading to that state, none of the new time-passage transitions can be scheduled (this would affect the resulting timed trace distribution). ■

Lemma 11.5.11 *Let C be a distinguishing complete context for two probabilistic timed automata M_1 and M_2. Then the timed principal context is a distinguishing context for M_1 and M_2.*

Proof. The result is achieved in two steps. First the actions of C are renamed so that each state enables two deterministic transitions with actions *left* and *right*, a probabilistic transition with actions *pleft* and *pright*, and one transition for each time-passage action d. Call this context C_1. Then, by observing that the state of C_1 is uniquely determined by the timed trace of any timed execution leading to it, all the states of C_1 are collapsed into a unique one.

Thus, we need to show only that C_1 is a distinguishing context. The proof of Lemma 7.5.10 applies. ■

Lemma 11.5.12 *Let C_P be a distinguishing context for two probabilistic timed automata M_1 and M_2. Then the simple context C of Figure 7-6 augmented with a self-loop time-passage transition from state s_0 for each time-passage action d, where start is an action that does not appear in M_1 and M_2, is a distinguishing context for M_1 and M_2.*

Proof. The proof of Lemma 7.5.11 applies. ■

Proof of Theorem 11.5.1. Let $M_1 \sqsubseteq_{DCt} M_2$. Then, from Lemma 11.5.12, $M_1 \| C_P \sqsubseteq_{Dt} M_2 \| C_P$. Conversely, let $M_1 \| C_P \sqsubseteq_{Dt} M_2 \| C_P$. Then, from Lemmas 11.5.3, 11.5.4, 11.5.5, 11.5.6, 11.5.7, 11.5.8, 11.5.9, 11.5.10, and 11.5.11, $M_1 \sqsubseteq_{DCt} M_2$. ■

Chapter 12

Hierarchical Verification: Timed Simulations

12.1 Introduction

The simulation method extends to the timed framework almost directly. The main difference is that in a timed simulation that abstracts from internal computation we use moves (cf. Section 9.4) rather than weak combined transitions. The kind of results that we prove are a direct extension of similar results for the untimed model. In particular, probabilistic timed forward simulations are sound for the timed trace distribution precongruence.

12.2 Probabilistic Timed Simulations

We start directly with simulation relations that abstract from internal computation; the strong relations are essentially the same as for the untimed case.

For convenience assume that M_1 and M_2 do not have common states. A *probabilistic timed bisimulation* between two simple probabilistic timed automata M_1 and M_2 is an equivalence relation $\mathcal{R}$ over $states(M_1) \cup states(M_2)$ such that

1. each start state of M_1 is related to at least one start state of M_2, and vice versa;
2. for each pair of states $s_1 \ \mathcal{R} \ s_2$ and each transition $s_1 \xrightarrow{a} \mathcal{P}_1$ of either M_1 or M_2, there exists a move $s_2 \overset{a\restriction ext(M_2)}{\rightsquigarrow} \mathcal{P}_2$ of either M_1 or M_2 such that $\mathcal{P}_1 \equiv_{\mathcal{R}} \mathcal{P}_2$.

We write $M_1 \simeq_{\mathrm{Pt}} M_2$ whenever $ext(M_1) = ext(M_2)$ and there is a probabilistic timed bisimulation between M_1 and M_2.

A *probabilistic timed simulation* between two simple probabilistic timed automata M_1 and M_2 is a relation $\mathcal{R} \subseteq states(M_1) \times states(M_2)$ such that

1. each start state of M_1 is related to at least one start state of M_2;
2. for each pair of states $s_1 \ \mathcal{R} \ s_2$ and each transition $s_1 \xrightarrow{a} \mathcal{P}_1$ of M_1, there exists a move $s_2 \overset{a\restriction ext(M_2)}{\rightsquigarrow} \mathcal{P}_2$ of M_2 such that $\mathcal{P}_1 \sqsubseteq_{\mathcal{R}} \mathcal{P}_2$.

We write $M_1 \sqsubseteq_{\mathrm{Pt}} M_2$ whenever $ext(M_1) = ext(M_2)$ and there is a probabilistic timed simulation from M_1 to M_2. We denote the kernel of probabilistic timed simulation by $\equiv_{\mathrm{Pt}}$.

It is easy to check that $\simeq_{\mathrm{Pt}}$ is an equivalence relation, that $\sqsubseteq_{\mathrm{Pt}}$ is a preorder relation, and that both $\simeq_{\mathrm{Pt}}$ and $\sqsubseteq_{\mathrm{Pt}}$ are preserved by the parallel composition operator. It is also easy to verify that a weak probabilistic bisimulation is a probabilistic timed bisimulation and that a weak probabilistic simulation is a probabilistic timed bisimulation.

12.3 Probabilistic Timed Forward Simulations

A *probabilistic timed forward simulation* between two simple probabilistic timed automata M_1, M_2 is a relation $\mathcal{R} \subseteq states(M_1) \times Probs(states(M_2))$ such that

1. each start state of M_1 is related to at least one Dirac distribution over a start state of M_2;
2. for each $s \mathrel{\mathcal{R}} \mathcal{P}'$, if $s \xrightarrow{a} \mathcal{P}_1$, then
   (a) for each $s' \in \Omega'$ there exists a probability space $\mathcal{P}_{s'}$ such that $s' \overset{a\lceil ext(M_2)}{\leadsto} \mathcal{P}_{s'}$, and
   (b) there exists a probability space $\mathcal{P}'_1$ of $Probs(Probs(states(M_2)))$ satisfying $\mathcal{P}_1 \sqsubseteq_{\mathcal{R}} \mathcal{P}'_1$,

   such that $\sum_{s' \in \Omega'} P'[s']\mathcal{P}_{s'} = \sum_{\mathcal{P} \in \Omega'_1} P'_1[\mathcal{P}]\mathcal{P}$.

Denote the existence of a probabilistic timed forward simulation from M_1 to M_2 by $M_1 \sqsubseteq_{FSt} M_2$.

Proposition 12.3.1 $\sqsubseteq_{FSt}$ *is preserved by the parallel composition operator.*

Proof. Let $M_1 \sqsubseteq_{FSt} M_2$, and let $\mathcal{R}$ be a probabilistic timed forward simulation from M_1 to M_2. Let $\mathcal{R}'$ be a relation between $states(M_1) \times states(M_3)$ and $Probs(states(M_2) \times states(M_3))$, defined as follows:

$$(s_1, s_3) \mathrel{\mathcal{R}'} \mathcal{P} \text{ iff } \mathcal{P} = \mathcal{P}_2 \otimes \mathcal{D}(s_3) \text{ for some } \mathcal{P}_2 \text{ such that } s_1 \mathrel{\mathcal{R}} \mathcal{P}_2.$$

The proof that $\mathcal{R}'$ satisfies Condition 1 and that Condition 2 is satisfied for each discrete transition of $M_1 \| M_2$ is essentially the proof of Proposition 8.5.1. Thus we need to show only that Condition 2 is satisfied by time-passage transitions.

Let $(s_1, s_3) \mathrel{\mathcal{R}'} \mathcal{P}_2 \otimes \mathcal{D}(s_3)$, and let $(s_1, s_3) \xrightarrow{d} (s'_1, s'_3)$, where $s_1 \xrightarrow{d} s'_1$, and $s_3 \xrightarrow{d} s'_3$. From the definition of a probabilistic timed forward simulation, for each $s \in \Omega_2$ there exists a move $s_2 \overset{d}{\leadsto} \mathcal{P}_s$ of M_2, and there exists a probability space $\mathcal{P}'_2$ of $Probs(Probs(states(M_2)))$, such that

$$\sum_{s \in \Omega_2} P_2[s]\mathcal{P}_s = \sum_{\mathcal{P} \in \Omega'_2} P'_2[\mathcal{P}]\mathcal{P}, \tag{12.1}$$

and

$$\mathcal{D}(s'_1) \sqsubseteq_{\mathcal{R}} \mathcal{P}'_2. \tag{12.2}$$

Moreover, from the definition of a probabilistic timed automaton, there is a trajectory ω_3 for $s_3 \xrightarrow{d} s'_3$.

For each $s \in \Omega_2$, let $\mathcal{O}_s$ be a generator for $s \overset{d}{\leadsto} \mathcal{P}_s$. Define a new generator $\mathcal{O}'_s$ as follows: for each finite execution fragment α of $M_2 \| M_3$ starting in (s, s_3),

1. if $\mathcal{O}_s(\alpha\lceil M_2) = (s', \mathcal{P})$, where $(s', \mathcal{P}) = \sum_i p_i(s', a_i, \mathcal{P}_i)$, each $(s', a_i, \mathcal{P}_i)$ is a transition of M_2, and $\alpha\lceil M_3$ is consistent with ω_3, i.e., for each prefix α' of α, $lstate(\alpha')\lceil M_3 = \omega_3(ltime(\alpha'))$, then letting s_3'' denote $lstate(\alpha\lceil M_3)$,

$$\mathcal{O}'_s(\alpha) = \sum_i p_i((s', s_3''), a_i, \mathcal{P}_i \otimes \mathcal{P}'_i),$$

where $\mathcal{P}'_i = \mathcal{D}(s_3'')$ if a_i is a discrete action, and $\mathcal{P}'_i = \mathcal{D}(\omega_3(ltime(\alpha) + a_i))$ if a_i is a time-passage action.

2. otherwise, $\mathcal{O}'_s(\alpha) = \mathcal{D}(\delta)$.

The move generated by each $\mathcal{O}'_s$ is $(s, s_3) \overset{d}{\leadsto} \mathcal{P}_s \otimes \mathcal{D}(s_3')$. In fact, an execution fragment α of $M_2 \| M_3$ is terminal for $\mathcal{O}'_s$ iff $\alpha\lceil M_2$ is terminal for $\mathcal{O}_s$ and $lstate(\alpha\lceil M_3) = s_3'$, and thus $\Omega_{\mathcal{O}'_s} = \Omega_s \times \mathcal{D}(s_3')$. Moreover, for each $\alpha \in \Omega_{\mathcal{O}'_s}$, $P_\alpha^{\mathcal{O}'_s} = P_{\alpha\lceil M_2}^{\mathcal{O}_s}$.

Denote $\mathcal{P}_s \otimes \mathcal{D}(s_3')$ by $\mathcal{P}_{(s,s_3)}$. Then, for each $(s, s_3) \in \Omega_2 \otimes \mathcal{D}(s_3)$, we have identified a move $(s, s_3) \overset{a}{\leadsto} \mathcal{P}_{(s,s_3)}$. These are the spaces of Condition 2.a in the definition of a probabilistic timed forward simulation.

From this point the proof proceeds exactly in the same way as the proof of Proposition 8.5.1. No modification of the text is necessary. ∎

12.4 The Execution Correspondence Theorem: Timed Version

The execution correspondence theorem of Chapter 8 extends easily to the timed framework. In this section we define the notion of a timed execution correspondence structure, show the timed version of the execution correspondence theorem, and, as a consequence, show that probabilistic timed forward simulations are transitive.

The timed execution correspondence theorem is stated in terms of the probabilistic executions of a probabilistic timed automaton; however, it is easy to see that the same result can be extended to probabilistic timed executions: the execution correspondence theorem talks about countably many states of a probabilistic timed execution; all the other points can be described by arbitrary trajectories.

12.4.1 Timed Execution Correspondence Structure

The definition of a fringe for a probabilistic timed execution is the same as the definition of a fringe for a probabilistic execution. For the definition of $fringe(H, i)$ the only difference is in the way the length of a state of H is measured, and thus the definition given for probabilistic automata is still valid.

Let $\mathcal{R}$ be a probabilistic timed forward simulation from M_1 to M_2. A *timed execution correspondence structure* via $\mathcal{R}$ is a tuple (H_1, H_2, m, S), where H_1 is a probabilistic execution of M_1, H_2 is a probabilistic execution of M_2, m is a mapping from natural numbers to fringes of M_2, and S is a mapping from natural numbers to probability distributions of $Probs(Probs(states(H_2)))$, such that

1. For each i, $m(i) \leq m(i+1)$;

2. For each state q_2 of H_2, $\lim_{i\to\infty} \sum_{q\in\Omega_i | q_2 \leq q} P_i[q] = P_H[C_q]$;

3. Let $q_1\ \mathcal{R}\ (\Omega, \mathcal{F}, P)$ iff for each $q \in \Omega$, $t\text{-}trace(q) = t\text{-}trace(q_1)$, and either

   (a) q_1 does not end in δ, each state of Ω does not end in δ, and $lstate(q_1)\ \mathcal{R}\ lstate(\mathcal{P})$, or

   (b) q_1 and each state of Ω end in δ and $lstate(\delta\text{-}strip(q_1))\ \mathcal{R}\ lstate(\delta\text{-}strip(\mathcal{P}))$.

   Then, for each $i \geq 0$, $m(i) = \sum_{\mathcal{P}\in\Omega_{S(i)}} P_{S(i)}[\mathcal{P}]\mathcal{P}$, and $fringe(H_1, i) \sqsubseteq_{\mathcal{R}} S(i)$.

4. Let, for each $i \geq 0$, each $q_1 \in fringe(H_1, i)$, and each $q_2 \in states(H_2)$, $W_i(q_1, q_2) \triangleq \sum_{\mathcal{P}} w_i(q_1, \mathcal{P}) P[q_2]$. If $W_i(q_1, q_2') = 0$ for each prefix or extension q_2' of q_2, then, for each extension q_1' of q_1 such that $q_1' \in fringe(H_1, i+1)$, and each prefix or extension q_2' of q_2, $W_{i+1}(q_1', q_2') = 0$.

12.4.2 The Main Theorem

Theorem 12.4.1 *Let $M_1 \sqsubseteq_{FS} M_2$ via the probabilistic timed forward simulation $\mathcal{R}$, and let H_1 be a probabilistic execution of M_1. Then there exists a probabilistic execution H_2 of M_2, a mapping m from natural numbers to fringes of M_2, and a mapping S from natural numbers to probability distributions of $Probs(Probs(states(H_2)))$, such that (H_1, H_2, m, S) is an execution correspondence structure via $\mathcal{R}$.*

Proof. The proof has exactly the same structure as the proof of Theorem 8.6.1. Note that the only difference between this theorem and Theorem 8.6.1 is in Condition 3, where we use timed traces rather than traces. ∎

12.4.3 Transitivity of Probabilistic Timed Forward Simulations

The timed execution correspondence theorem can be used to show that probabilistic timed forward simulations are transitive, i.e., if $M_1 \sqsubseteq_{FSt} M_2$ and $M_2 \sqsubseteq_{FSt} M_3$, then $M_1 \sqsubseteq_{FSt} M_3$. The proof of this result follows the same lines as the corresponding proof in the untimed case (cf. Section 8.6.4), where combined transitions are replaced by moves and traces are replaced by timed traces. We leave the details of the proof to the reader.

12.5 Soundness for Timed Trace Distributions

As for the untimed model, the timed execution correspondence theorem can be used to show that probabilistic timed forward simulations are sound for the timed trace distribution precongruence. Since $\sqsubseteq_{FSt}$ is a precongruence, it is enough to show that $\sqsubseteq_{FSt}$ is sound for the timed trace distribution preorder.

Proposition 12.5.1 *If $M_1 \sqsubseteq_{FSt} M_2$, then $M_1 \sqsubseteq_{Dt} M_2$.*

Proof. Let $M_1 \sqsubseteq_{FSt} M_2$, and let H_1 be a probabilistic execution of M_1 that leads to a timed trace distribution $\mathcal{D}_1$. From Lemma 12.4.1, there exists a probabilistic execution H_2 of M_2 that corresponds to H_1 via some mappings m, S. We show that H_2 leads to a timed trace distribution $\mathcal{D}_2$ that is equivalent to $\mathcal{D}_1$.

Consider a cone C_β of $\mathcal{D}_1$. The cone C_β can be expressed as a union of cones of $\mathcal{P}_{H_1}$, and thus its measure can be expressed as

$$\lim_{i\to\infty} \sum_{q_1 \in fringe(H_1,i) | \beta \leq t\text{-}trace(q_1)} P_{H_1}[C_{q_1}]. \tag{12.3}$$

Consider a cone C_β of $\mathcal{D}_2$. The cone C_β can be expressed as a union of cones of $\mathcal{P}_{H_2}$, and thus its measure can be expressed as

$$\lim_{i\to\infty} \sum_{q_2 \in m(i) | \beta \leq t\text{-}trace(q_2)} P_{m(i)}[q_2]. \tag{12.4}$$

The reason for Expression (12.4) is that at the limit each cone expressing the occurrence of β is captured completely.

Thus, it is sufficient to show that for each finite β and each i,

$$\sum_{q_1 \in fringe(H_1,i) | \beta \leq t\text{-}trace(q_1)} P_{H_1}[C_{q_1}] = \sum_{q_2 \in m(i) | \beta \leq t\text{-}trace(q_2)} P_{m(i)}[q]. \tag{12.5}$$

From this point the proof proceeds exactly as the proof of Proposition 8.7.1. ■

Chapter 13

Conclusion

13.1 Have we Met the Challenge?

We have developed a model for the description of randomized distributed real-time systems, and we have investigated how the new model can be used for the analysis of algorithms. The main idea behind the model is to extend labeled transition systems to account for randomization in such a way that probabilistic behavior and nondeterministic behavior are clearly distinct.

We have shown how commonly used informal statements can be formulated in the new formalism, and we have shown how such statements can be proved to be correct in a formal and rigorous way. In particular, we have developed verification techniques that resemble the common ways in which randomized algorithms are analyzed. The main improvement is that now we have a collection of results that allow us to determine when a specific argument can be used safely. Furthermore, we have shown how to derive upper bounds to the complexity of a randomized distributed algorithm using an ordinary time complexity measure as well as more abstract complexity measures like "number of rounds in an asynchronous computation".

Finally, we have extended several verification techniques that are commonly used within the labeled transition system model. We have extended the trace semantics of labeled transition systems and several of the existing simulation relations for labeled transition systems. In particular, all our preorder relations are compositional and the simulation relations are sound for the trace-based semantics. Although we have not presented any example of verification using simulations, except for two toy examples based on coin flips, we are confident that in the future the method based on simulations will become of practical relevance as it happened for ordinary automata.

Therefore, we can claim that we have met the challenge given by randomization at least partially. Surely we understand much more of the problem than before. The fact that we have been able to prove new results about randomized algorithms is a positive sign. In particular, Aggarwal [Agg94] used successfully the technique presented in this thesis for the verification of the randomized self-stabilizing algorithm of Aggarwal and Kutten [AK93], which is not trivial at all; during the verification process Aggarwal discovered also a subtle bug in the original protocol. In the measure in which the power of a proof method is evaluated based on the bugs that such method helps to discover, our methodology has achieved something. Indeed we have discovered another bug on one existing algorithm, and the main issue is that we did not have to work much to discover such a bug; essentially it was sufficient to try to reformulate the proof of correctness in our framework.

13.2 The Challenge Continues

Although we have improved considerably our understanding of randomization in distributed computation, what we have discovered looks like the tip of the iceberg. We have addressed several problems, and in solving them we have addressed more the basic methodology rather than an extensive analysis of all the possible solutions. Therefore, there are several directions for further research that can be pursued. Here we suggest some of the most important directions.

13.2.1 Discrete versus Continuous Distributions

Throughout this thesis we have assumed that the probability distributions associated with the transitions of a probabilistic automaton are discrete. Although such assumption is sufficiently general for the study of several randomized algorithms, several other real-time systems are better described by using continuous distributions. Examples involve algorithms for transmission of data along a common wire, scheduling algorithms for massively parallel machines, and queuing systems. Moreover, continuous distributions would be more suitable for the study of randomized hybrid systems.

The extension of the theory to continuous distributions involves nontrivial measure theoretical problems. In particular it is not the case any more that any union of cones is measurable; thus, not even the event that expresses the occurrence of an action or the reachability of a state is measurable in general. The events with probability 0 need a more careful treatment within the model with continuous distributions. It is likely that some restrictions must be imposed to the model to ensure that some minimal set of events is measurable. Examples of restricted models with continuous distributions are the automata of Alur, Courcuobetis and Dill [ACD91a, ACD91b], where the time that elapses between two transitions is governed by an exponential distribution or by a distribution which is non zero in a finite collection of closed intervals, and the models of [GHR93, Hil93, BDG94], where the time between the occurrence of two actions is assumed to be distributed exponentially. Exponential distributions occur in several real systems and are easy to model due to their memoryless structure. However, other distributions should be studied.

13.2.2 Simplified Models

Within the context of ordinary automata Lynch and Tuttle [LT87] have developed a model of I/O automata. The model enforces a distinction between Input actions and Output actions within an automaton, and requires that input actions are enabled from every state. Furthermore, in a parallel composition context each action is required to be the output or internal action of at most one process, i.e., each action is under the control of at most one process. Based on the Input/Output distinction Lynch and Tuttle can introduce fairness in the model in a natural way, and in particular they can use the trace semantics as a meaningful notion of implementation. In general the trace semantics is not meaningful as a notion of implementation since, for example, it is not sensitive to deadlock. The advantage of the use of traces is that traces are easy to deal with.

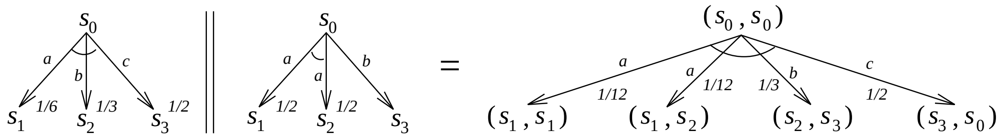

Figure 13-1: Synchronization for probabilistic I/O automata.

For this reason, it makes sense to study a theory of probabilistic I/O automata as an extension of the model of [LT87] and as a restriction of our model. An interesting point of a model with I/O distinction is that it is possible to relax the requirement that all the transitions of a probabilistic I/O automaton are simple. In particular, only the transitions with input actions need to be simple, while all the others can be general. The parallel composition can be defined easily since a non-simple transition synchronizes only with simple transitions. Figure 13-1 gives an example of synchronization between a transition with three output actions a, b, c and two transitions of an I/O automaton with just two input actions a, b. A similar observation was made also by Wu, Stark and Smolka in [WSS94].

A restricted timed model with I/O distinction is introduced by Merrit, Modugno and Tuttle [MMT91]. In particular timing constraints can be described only by giving upper and lower bounds to the time it takes for a process to perform the next transition whenever it is ready to do so. MMT automata turned out to be sufficient for the modeling of several distributed systems, and in particular, due to their simple structure, made the analysis simpler than by using the full automaton model. Once again, a study of the probabilistic version of the MMT model would be useful. The proofs that we have illustrated in Chapter 12 could be carried out in the probabilistic MMT model as well.

Finally, the analysis of a system can be simplified by studying time-deterministic probabilistic timed automata, i.e., probabilistic timed automata such that from each state s and each time d there is at most one state reachable from s in time d. In fact, if a system is time-deterministic, then the end points of a time-passage transition determine completely the trajectory that is spanned. Therefore, trajectories could be removed also from the direct analysis of randomized timed algorithms. It turns out that most of the times an algorithm can be described as a time-deterministic probabilistic automaton. Probabilistic MMT automata are an example of time-deterministic probabilistic automata.

13.2.3 Beyond Simple Probabilistic Automata

The study of parallel composition and of the simulation relations of this thesis is done within the context of simple probabilistic automata. The main problem is that we did not find any reasonable definition of parallel composition for general probabilistic automata that is consistent with our synchronization style. We have just observed that in the presence of an Input/Output distinction it is possible to relax the simplicity condition and yet obtain a meaningful notion of parallel composition. It would be interesting to investigate other mechanisms that give a meaning to general probabilistic automata and yet work as we expect in the simple case.

13.2.4 Completeness of the Simulation Method

We have provided several simulation and bisimulation relations for probabilistic automata and probabilistic timed automata, and we have shown that they are sound for the trace distribution precongruence and the timed trace distribution precongruence, respectively. However, we have not shown any completeness result for probabilistic forward simulations and probabilistic forward timed simulations. In [LV93a, LV95] it is shown that forward simulations together with another kind of simulations called *backward simulations* are sound and complete for the trace preorder. Are probabilistic forward simulations complete for the trace distribution preorder? If not, is there an equivalent of backward simulations that can lead to completeness?

13.2.5 Testing Probabilistic Automata

We have presented the trace distribution semantics as an example of a semantics based on abstract observations. Another widely known semantics for ordinary automata is the failure semantics of Brookes, Hoare and Roscoe [BHR84], which in turn is connected to the testing preorders of De Nicola and Hennessy [DH84]. Similarly to the trace distribution semantics, it should be possible to extend the failure semantics to the probabilistic framework and find a sufficiently powerful context to distinguish probabilistic automata that are not in the corresponding precongruence relation. Possibly, a related theory of testing in the style of [DH84] should be defined. It is very likely that the new testing preorders will be similar to those of Yi and Larsen [YL92]. Other theories of testing for probabilistic automata are studied in [Chr90b, Chr90a, CSZ92, YCDS94] and are explained in Section 2.2.

13.2.6 Liveness in Probabilistic Automata

In the extension of the notion of an execution of an automaton we have obtained a parallelism between the theory of ordinary automata and the theory of probabilistic automata. In this parallelism also the notion of liveness has found its place, although we have not addressed the issue in this thesis. In ongoing research we have given a simple definition of a live probabilistic automaton as a pair (M, L) where L is an arbitrary subset of the probabilistic executions of M, and we have shown that the *live trace distribution precongruence* can be defined easily and can be characterized by a *live principal context*, which is essentially the principal context paired with the set of its probabilistic executions. However, lot of work remains to be done within the theory of liveness.

First of all it would be useful to study how the definition of safety and liveness properties of Alpern and Schneider [AS85] extends to the probabilistic framework and what consequences such extension has. Furthermore, the use of the live trace preorder within ordinary automata makes sense as a notion of implementation in the presence of I/O distinction and of a property called *receptiveness* or *environment-freedom* [Dil88, AL93, GSSL94]. It would be useful to study the theory of receptiveness of [Dil88, AL93] and of environment-freedom of [GSSL94] in the context of randomization. In this case, differently from [GSSL94], the environment is expressed by a function rather than by a sequence of actions. However, non-trivial problems arise in imposing restrictions to the behavior of the environment.

13.2.7 Temporal Logics for Probabilistic Systems

In the chapters on direct analysis we have identified a collection of probabilistic statements that are useful for the analysis of algorithms. However, there are several other statements that can be of interest. It would be desirable to find a probabilistic temporal logic that expresses as many properties as possible. The probabilistic modal logic of [LS89] is a direct extension of the modal logic of Hennessy and Milner [HM85] for reactive processes, but it is not sufficiently powerful to deal with nondeterminism; similarly, the extended probabilistic logic of [LS92] is not sufficiently powerful. The Probabilistic Computation Tree Logic of [HJ89, Han94] captures more the consequences of the interplay between probability and nondeterminism; in [SL94] PCTL is generalized also to probabilistic systems with internal actions (WPCTL). However, there are still properties that are useful and do not seem to be expressible in WPCTL. Specifically, we do not know how to express a property of the kind "after something has happened, no matter where I am, something else will happen with probability at least p". Is there something missing in WPCTL? What would be a more appropriate temporal logic?

Another issue is the relationship between the simulation method and temporal logic. That is, if a probabilistic automaton implements another probabilistic automaton according to some implementation relation (e.g., trace distribution precongruence, probabilistic simulation, probabilistic forward simulation, etc.), what can we say about the implementation? What properties of the specification are satisfied by the implementation? More generally, given a probabilistic temporal logic and a preorder relation, what fragment of the logic is preserved by the preorder relation? Somehow it is implicit that whenever we use some preorder relation as a notion of implementation we are interested only in the properties that are preserved by such relation; however, we need to know what are those properties. In [SL95] we have stated that weak probabilistic simulation preserve a large fragment of WPCTL and that weak probabilistic bisimulations preserve WPCTL. The results of [SL95] can be proved easily given the results of this thesis. However, more work in this direction is necessary. In particular, some completeness results would be useful.

13.2.8 More Algorithms to Verify

In this thesis we have illustrated our direct verification technique by proving the correctness of the randomized dining philosophers algorithm of Lehmann and Rabin [LR81] and of the randomized agreement protocol of Ben-Or [BO83]. In [Agg94] Aggarwal uses our model to verify the correctness of the self-stabilizing minimum weight spanning tree randomized algorithm of Aggarwal and Kutten [AK93]. However, the technique should be tested against many other algorithms. We are currently investigating the agreement protocol of Aspnes and Herlihy [AH90] and the randomized mutual exclusion algorithm of Pnueli and Zuck [PZ86]. Based on the little experience that we have gained, we can say that the model provides us with a systematic way of analyzing those algorithms, and in particular it provides us with a simple methodology to identify the critical points of an algorithm.

It is very likely that new coin lemmas need to be developed together with other techniques for the actual computation of the probability of an event. A technique that needs further development is the partition technique of Section 6.7. The analysis of other algorithms should make clear what other techniques are necessary. Also, playing with the toy resource allocation protocol of Chapter 5 can be very instructive. Although the protocol is simple, its analysis highlights several of the issues that arise in randomized distributed computation.

It is also plausible, as it happened for non-probabilistic distributed algorithms, that some complex protocols can be verified more easily by using the simulation method. Finding those algorithms would be an optimal way to test the hierarchical verification method and possibly to improve it.

13.2.9 Automatic Verification of Randomized Systems

Formal verification usually involves two levels of analysis. First, an algorithm is analyzed at a high level by using the intuition that designers have of their own algorithm; then, a more detailed verification of the high level claims is carried out in order to guarantee correctness. The low level analysis is very tedious and involves checking a whole lot of uninteresting details. On the other hand, several times the low level analysis is the only way to discover flaws in the intuitions about an algorithm.

Fortunately, the low level analysis is amenable to automatic verification, although the research in this area is still in progress. Model checking [EC82, CES83] is certainly a useful technique; in [SGG$^+$93] it is shown how a theorem prover can be used to help in the verification of a protocol using simulations; in [PS95] we have investigated how a randomized algorithm can be verified mechanically once the high level proof is formulated. However, there is still a lot of work that needs to be done. It would be interesting to study how model checking and theorem proving could be integrated to automatize part of the verification of an algorithm.

13.3 The Conclusion's Conclusion

To say what we have done in one sentence, we have provided a new way of reasoning about randomized systems that integrates both the theoretical aspects of modeling and the basic requirements for usage in practice. From the modeling point of view we have distinguished between nondeterminism and probability explicitly and we have extended the main semantics that are available within the labeled transition systems model; from the point of view of verification we have formalized some of the common informal arguments about randomized algorithms and we have provided guidelines to determine whether an argument can be used safely. Furthermore, we have provided a systematic way to analyze the complexity of randomized algorithms. All our results are compatible with previous work.

As we have seen in the previous section, there are still many open problems in this area. Here we hope to have stimulated the curiosity of the reader to go much further. Needless to say that for us (me) working on this project was a continuous discovery.

Bibliography

[ACD91a] R. Alur, C. Courcoubetis, and D.L. Dill. Model-checking for probabilistic real-time systems. In J. Leach Albert, B. Monien, and M. Rodríguez, editors, *Proceedings* 18^{th} *ICALP,* Madrid, volume 510 of *Lecture Notes in Computer Science*, pages 115–136. Springer-Verlag, 1991.

[ACD91b] R. Alur, C. Courcoubetis, and D.L. Dill. Verifying automata specifications of probabilistic real-time systems. In de Bakker et al. [dBHRR91], pages 28–44.

[ACS94] B. Awerbuch, L. Cowen, and M.A. Smith. Efficient asynchronous distributed symmetry breaking. In *Proceedings of the* 26^{th} *Annual ACM Symposium on Theory of Computing*, 1994.

[Agg94] S. Aggarwal. Time optimal self-stabilizing spanning tree algorithms. Technical Report MIT/LCS/TR-632, MIT Laboratory for Computer Science, 1994. Master's thesis.

[AH90] J. Aspnes and M.P. Herlihy. Fast randomized consensus using shared memory. *Journal of Algorithms*, 15(1):441–460, September 1990.

[AK93] S. Aggarwal and S. Kutten. Time optimal self stabilizing spanning tree algorithms. In R.K. Shyamasundar, editor, *13th International Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 761 of *Lecture Notes in Computer Science*, pages 400–410, Bombay, India., December 1993. Springer-Verlag.

[AL91] M. Abadi and L. Lamport. An old-fashioned recipe for real time. In de Bakker et al. [dBHRR91], pages 1–27.

[AL93] M. Abadi and L. Lamport. Composing specifications. *ACM Transactions on Programming Languages and Systems*, 15(1):73–132, 1993.

[AS85] B. Alpern and F.B. Schneider. Defining liveness. *Information Processing Letters*, 21(4):181–185, 1985.

[BBS92] J.C.M. Baeten, J.A. Bergstra, and S.A. Smolka. Axiomatizing probabilistic processes: ACP with generative probabilities. In Cleaveland [Cle92], pages 472–485.

[BDG94] M. Bernardo, L. Donatiello, and R. Gorrieri. Modeling and analyzing concurrent systems with MPA. In U. Herzog and M. Rettelbach, editors, *Proceedings of* *the Second Workshop on Process Algebras and Performance Modelling* (PAPM), Erlangen, Germany, pages 175–189, 1994.

[BFJ+82] J. Burns, M. Fisher, P. Jackson, N.A. Lynch, and G. Peterson. Data requirements for implementation of n-process mutual exclusion using a single shared variable. *Journal of the ACM*, 29(1):183–205, 1982.

[BG91] J.C.M. Baeten and J.F. Groote, editors. *Proceedings of CONCUR 91,* Amsterdam, volume 527 of *Lecture Notes in Computer Science*. Springer-Verlag, 1991.

[BHR84] S.D. Brookes, C.A.R. Hoare, and A.W. Roscoe. A theory of communicating sequential processes. *Journal of the ACM*, 31(3):560–599, 1984.

[BK90] J.C.M. Baeten and J.W. Klop, editors. *Proceedings of CONCUR 90,* Amsterdam, volume 458 of *Lecture Notes in Computer Science*. Springer-Verlag, 1990.

[BM89] B. Bloom and A. Meyer. A remark on bisimulation between probabilistic processes. In *Proceedings of the Symposium on Logical Foundations of Computer Science*, volume 363 of *Lecture Notes in Computer Science*, pages 26–40, 1989.

[BO83] M. Ben-Or. Another advantage of free choice: completely asynchronous agreement protocols. In *Proceedings of the 2^{nd} Annual ACM Symposium on Principles of Distributed Computing,* Montreal, Quebec, Canada, August 1983.

[BPV94] D. Bosscher, I. Polak, and F. Vaandrager. Verification of an audio control protocol. Technical Report CS-R9445, CWI, Amsterdam, July 1994.

[BW90] J.C.M. Baeten and W.P Weijland. *Process Algebra.* Cambridge Tracts in Theoretical Computer Science 18, Cambridge University Press, 1990.

[CES83] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2), 1983.

[Chr90a] I. Christoff. Testing equivalences and fully abstract models for probabilistic processes. In Baeten and Klop [BK90], pages 126–140.

[Chr90b] I. Christoff. *Testing Equivalences for Probabilistic Processes.* PhD thesis, Department of Computer Science, Uppsala University, 1990.

[Chr93] L. Christoff. *Specification and Verification Methods for Probabilistic Processes.* PhD thesis, Department of Computer Science, Uppsala University, 1993.

[Cle92] W.R. Cleaveland, editor. *Proceedings of CONCUR 92,* Stony Brook, NY, USA, volume 630 of *Lecture Notes in Computer Science*. Springer-Verlag, 1992.

[CM88] K.M. Chandi and J. Misra. *Parallel Program Design: A Foundation.* Addison-Wesley, 1988.

[CSZ92] R. Cleaveland, S.A. Smolka, and A. Zwarico. Testing preorders for probabilistic processes (extended abstract). In *Proceedings* 19^{th} *ICALP,* Madrid, volume 623 of *Lecture Notes in Computer Science*, pages 708–719. Springer-Verlag, 1992.

[CY88] C. Courcoubetis and M. Yannakakis. Verifying temporal properties of finite-state probabilistic programs. In 29^{th} *Annual Symposium on Foundations of Computer Science*, pages 338–345, 1988.

[CY90] C. Courcoubetis and M. Yannakakis. Markov decision procedures and regular events. In M. Paterson, editor, *Proceedings* 17^{th} *ICALP,* Warwick, volume 443 of *Lecture Notes in Computer Science*, pages 336–349. Springer-Verlag, July 1990.

[dBHRR91] J.W. de Bakker, C. Huizing, W.P. de Roever, and G. Rozenberg, editors. *Proceedings of the REX Workshop "Real-Time: Theory in Practice"*, volume 600 of *Lecture Notes in Computer Science*. Springer-Verlag, 1991.

[DeN87] R. De Nicola. Extensional equivalences for transition systems. *Acta Informatica*, 24:211–237, 1987.

[DH84] R. De Nicola and M. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133, 1984.

[Dil88] D. Dill. *Trace Theory for Automatic Hierarchical Verification of Speed-Independent Circuits*. ACM Distinguished Dissertations. MIT Press, 1988.

[EC82] E.A. Emerson and E.C. Clarke. Using branching time temporal logic to synthesize synchronous skeletons. *Science of Computer Programming*, 2:241–266, 1982.

[FLP85] M. Fischer, N. Lynch, and M. Paterson. Impossibility of distributed consensus with a family of faulty process. *Journal of the ACM*, 32(2):374–382, April 1985.

[GHR93] N. Götz, U. Herzog, and M. Rettelbach. Multiprocessor and distributed system design: the integration of functional specification and performance analysis using stochastic process algebras. In L. Donatiello and R. Nelson, editors, *Performance Evaluation of Computer and Communication Systems. Joint Tutorial Papers of Performance '93 and Sigmetrics '93*, volume 729 of *Lecture Notes in Computer Science*, pages 121–146. Springer-Verlag, 1993.

[GJS90] A. Giacalone, C.C Jou, and S.A. Smolka. Algebraic reasoning for probabilistic concurrent systems. In *Proceedings of the Working Conference on Programming Concepts and Methods (IFIP TC2),* Sea of Galilee, Israel, 1990.

[Gla90] R.J. van Glabbeek. The linear time – branching time spectrum. In Baeten and Klop [BK90], pages 278–297.

[Gla93] R.J. van Glabbeek. The linear time – branching time spectrum ii. The semantics of sequential systems with silent moves. In E. Best, editor, *Proceedings of CONCUR 93,* Hildesheim, Germany, volume 715 of *Lecture Notes in Computer Science*, pages 66–81. Springer-Verlag, 1993.

[GSB94] R. Gupta, S.A. Smolka, and S. Bhaskar. On randomization in sequential and distributed algorithms. *ACM Computing Surveys*, 26(1):1–86, 1994.

[GSSL94] R. Gawlick, R. Segala, J.F. Søgaard-Andersen, and N.A. Lynch. Liveness in timed and untimed systems. In S. Abiteboul and E. Shamir, editors, *Proceedings* 21^{th} *ICALP,* Jerusalem, volume 820 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994. A full version appears as MIT Technical Report number MIT/LCS/TR-587.

[GSST90] R.J. van Glabbeek, S.A. Smolka, B. Steffen, and C.M.N. Tofts. Reactive, generative, and stratified models of probabilistic processes. In *Proceedings* 5^{th} *Annual Symposium on Logic in Computer Science,* Philadelphia, USA, pages 130–141. IEEE Computer Society Press, 1990.

[Hal50] P.R. Halmos. *Measure Theory*. Springer-Verlag, 1950.

[Han91] H. Hansson. *Time and Probability in Formal Design of Distributed Systems*. PhD thesis, Department of Computer Science, Uppsala University, 1991.

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

[Hil93] J. Hillston. PEPA: Performance enhanced process algebra. Technical Report CSR-24-93, Department of Computer Science, University of Edimburgh (UK), 1993.

[Hil94] J. Hillston. *A Compositional Approach to Performance Modeling*. PhD thesis, Department of Computer Science, University of Edimburgh (UK), 1994.

[HJ89] H. Hansson and B. Jonsson. A framework for reasoning about time and reliability. In *Proceedings of the* 10^{th} *IEEE Symposium on Real-Time Systems,* Santa Monica, Ca., 1989.

[HJ90] H. Hansson and B. Jonsson. A calculus for communicating systems with time and probabilities. In *Proceedings of the* 11^{th} *IEEE Symposium on Real-Time Systems,* Orlando, Fl., 1990.

[HM85] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–161, 1985.

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

[HS85] S. Hart and M. Sharir. How to schedule if you must. *SIAM Journal on Computing*, 14:991–1012, 1985.

[HSP83] S. Hart, M. Sharir, and A. Pnueli. Termination of probabilistic concurrent programs. *ACM Transactions on Programming Languages and Systems*, 5(3):356–380, 1983.

[JHY94] B. Jonsson, C. Ho-Stuart, and W. Yi. Testing and refinement for nondeterministic and probabilistic processes. In Langmaack, de Roever, and Vytopil, editors, *Proceedings of the Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, volume 863 of *Lecture Notes in Computer Science*, pages 418–430, 1994.

[JL91] B. Jonsson and K.G. Larsen. Specification and refinement of probabilistic processes. In *Proceedings of the 6th IEEE Symposium on Logic in Computer Science*, pages 266–277, Amsterdam, July 1991.

[Jon91] B. Jonsson. Simulations between specifications of distributed systems. In Baeten and Groote [BG91], pages 346–360.

[JP89] C. Jones and G. Plotkin. A probabilistic powerdomain of evaluations. In *Proceedings* 4^{th} *Annual Symposium on Logic in Computer Science,* Asilomar, California, pages 186–195. IEEE Computer Society Press, 1989.

[JP94] B. Jonsson and J. Parrow, editors. *Proceedings of CONCUR 94,* Uppsala, Sweden, volume 836 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[JS90] C.C. Jou and S.A. Smolka. Equivalences, congruences, and complete axiomatizations for probabilistic processes. In Baeten and Klop [BK90], pages 367–383.

[JY95] B. Jonsson and W. Yi. Compositional testing preorders for probabilistic processes. In *Proceedings* 10^{th} *Annual Symposium on Logic in Computer Science,* San Diego, California. IEEE Computer Society Press, 1995.

[Kar90] R.M. Karp. An introduction to randomized algorithms. Technical Report TR-90-024, Computer Science Division, University of California, Berkeley, CA, 1990.

[Kel76] R. Keller. Formal verification of parallel programs. *Communications of the ACM*, 7(19):561–572, 1976.

[KR92] E. Kushilevitz and M. Rabin. Randomized mutual exclusion algorithms revisited. In *Proceedings of the* 11^{th} *Annual ACM Symposium on Principles of Distributed Computing,* Quebec, Canada, pages 275–284, 1992.

[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 8^{th} Annual ACM Symposium on Principles of Programming Languages*, pages 133–138, January 1981.

[LS82] D. Lehmann and S. Shelah. Reasoning with time and chance. *Information and Control*, 53:165–198, 1982.

[LS89] K.G. Larsen and A. Skou. Bisimulation through probabilistic testing. In *Conference Record of the* 16^{th} *ACM Symposium on Principles of Programming Languages,* Austin, Texas, pages 344–352, 1989.

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

[LS92] K.G. Larsen and A. Skou. Compositional verification of probabilistic processes. In Cleaveland [Cle92], pages 456–471.

[LSS94] N.A. Lynch, I. Saias, and R. Segala. Proving time bounds for randomized distributed algorithms. In *Proceedings of the* 13^{th} *Annual ACM Symposium on Principles of Distributed Computing,* Los Angeles, CA, pages 314–323, 1994.

[LT87] N.A. Lynch and M.R. Tuttle. Hierarchical correctness proofs for distributed algorithms. In *Proceedings of the* 6^{th} *Annual ACM Symposium on Principles of Distributed Computing*, pages 137–151, Vancouver, Canada, August 1987. A full version is available as MIT Technical Report MIT/LCS/TR-387.

[LV91] N.A. Lynch and F.W. Vaandrager. Forward and backward simulations for timing-based systems. In de Bakker et al. [dBHRR91], pages 397–446.

[LV93a] N.A. Lynch and F.W. Vaandrager. Forward and backward simulations – part I: Untimed systems. Technical Report MIT/LCS/TM-486, MIT Laboratory for Computer Science, May 1993.

[LV93b] N.A. Lynch and F.W. Vaandrager. Forward and backward simulations – part II: Timing-based systems. Technical Report MIT/LCS/TM-487, MIT Laboratory for Computer Science, September 1993.

[LV95] N.A. Lynch and F.W. Vaandrager. Forward and backward simulations – part II: Timing-based systems. Technical Report CS-R95??, CWI, Amsterdam, ?? 1995.

[Lyn95] N.A. Lynch. *Distributed Algorithms.* Morgan Kaufmann Publishers, Inc., 1995. To appear.

[Mil89] R. Milner. *Communication and Concurrency.* Prentice-Hall International, Englewood Cliffs, 1989.

[Mil93] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25:267–310, 1993.

[MMT91] M. Merritt, F. Modugno, and M. Tuttle. Time constrained automata. In Baeten and Groote [BG91], pages 408–423.

[OL82] S. Owicki and L. Lamport. Proving liveness properties of concurrent programs. *ACM Transactions on Programming Languages and Systems*, 4:455–495, 1982.

[Par81] D.M.R. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, 5^{th} *GI Conference*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer-Verlag, 1981.

[Plo81] G.D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Computer science Department, Aarhus University, 1981.

[Pnu82] A. Pnueli. The temporal semantics of concurrent programs. *Theoretical Computer Science*, 13:45–60, 1982.

[Pnu83] A. Pnueli. On the extremely fair treatment of probabilistic algorithms. In *Proceedings of the 15^{th} Annual ACM Symposium on Theory of Computing, Boston, Massachusetts*, May 1983.

[PS95] A. Pogosyants and R. Segala. Formal verification of timed properties of randomized distributed algorithms. In *Proceedings of the 14^{th} Annual ACM Symposium on Principles of Distributed Computing,* Ottawa, Ontario, Canada, August 1995. To appear.

[PZ86] A. Pnueli and L. Zuck. Verification of multiprocess probabilistic protocols. *Distributed Computing*, 1(1):53–72, 1986.

[Rab63] M.O. Rabin. Probabilistic automata. *Information and Control*, 6:230–245, 1963.

[Rab76] M.O. Rabin. Probabilistic algorithms. In J. F. Traub, editor, *Algorithms and Complexity: New Directions and Results*, pages 21–39. Academic Press, 1976.

[Rab82] M.O. Rabin. N-process mutual exclusion with bounded waiting by $4 \log N$ shared variables. *Journal of Computer and System Sciences*, 25:66–75, 1982.

[Rao90] J.R. Rao. Reasoning about probabilistic algorithms. In *Proceedings of the 9^{th} Annual ACM Symposium on Principles of Distributed Computing,* Quebec, Canada, August 1990.

[Rud66] W. Rudin. *Real Complex Analysis.* McGraw-Hill, 1966.

[Sai92] I. Saias. Proving probabilistic correctness: the case of Rabin's algorithm for mutual exclusion. In *Proceedings of the 11^{th} Annual ACM Symposium on Principles of Distributed Computing,* Quebec, Canada, August 1992.

[Sei92] K. Seidel. Probabilistic communicating processes. Technical Report PRG-102, Ph.D. Thesis, Programming Research Group, Oxford University Computing Laboratory, 1992.

[SGG+93] J.F. Søgaard-Andersen, S.J. Garland, J.V. Guttag, N.A. Lynch, and A. Pogosyants. Computer-assisted simulation proofs. In C. Courcoubetis, editor, *Proceedings of the fifth international conference on Computer Aided Verification*, volume 697 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.

[She87] G.S. Shedler. *Regeneration and Networks of Queues.* Springer-Verlag, 1987.

[SL94] R. Segala and N.A. Lynch. Probabilistic simulations for probabilistic processes. In Jonsson and Parrow [JP94], pages 481–496.

[SL95] R. Segala and N.A. Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 1995.

[SLL93] J.F. Søgaard-Andersen, N.A. Lynch, and B.W. Lampson. Correctness of communication protocols. a case study. Technical Report MIT/LCS/TR-589, MIT Laboratory for Computer Science, November 1993.

[SS90] S. Smolka and B. Steffen. Priority as extremal probability. In Baeten and Klop [BK90], pages 456–466.

[Tof90] C. Tofts. A synchronous calculus of relative frequencies. In Baeten and Klop [BK90].

[Var85] M.Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Proceedings of 26th IEEE Symposium on Foundations of Computer Science*, pages 327–338, Portland, OR, 1985.

[VL92] F.W. Vaandrager and N.A. Lynch. Action transducers and timed automata. In Cleaveland [Cle92], pages 436–455.

[VW86] M.Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proceedings Symposium on Logic in Computer Science*, pages 332–344. IEEE Computer Society Press, 1986.

[Whi80] W. Whitt. Continuity of generalized semi-markov processes. *Mathematics of Operations Research*, 5, 1980.

[WLL88] J.L. Welch, L. Lamport, and N.A. Lynch. A lattice-structured proof technique applied to a minimum spanning tree algorithm. Technical Report MIT/LCS/TM-361, MIT Laboratory for Computer Science, June 1988.

[WSS94] S.H. Wu, S. Smolka, and E.W. Stark. Composition and behaviors of probabilistic I/O automata. In Jonsson and Parrow [JP94].

[YCDS94] S. Yuen, R. Cleaveland, Z. Dayar, and S. Smolka. Fully abstract characterizations of testing preorders for probabilistic processes. In Jonsson and Parrow [JP94].

[YL92] W. Yi and K.G. Larsen. Testing probabilistic and nondeterministic processes. In *Protocol Specification, Testing and Verification XII*, pages 47–61, 1992.

[Zuc86] L. Zuck. *Past Temporal Logic*. PhD thesis, The Weizman Institute of Science, 1986.

Table of Symbols

We list the symbols that are used in this thesis in the order they appear in the presentation. Each symbol is listed with a short description and a reference to the pages where it is first defined.

Ω	Sample space.	33
$\mathcal{F}$	σ-field.	33
$\sigma(\mathcal{C})$	σ-field generated by a family of sets $\mathcal{C}$.	33
μ	Measure	33
P	Probability measure.	34
E	Event.	34
$completion()$	Completion of a measure.	34
$\otimes$	Product of σ-fields, of measures, and of discrete probability spaces.	35
$\mid$	Conditional of an event and of an event schema.	36
$\mid$	Conditional of a probabilistic execution fragment.	57
$\mathcal{P}$	Probability space.	37
$\mathcal{D}()$	Dirac distribution.	37
$\mathcal{U}()$	Uniform distribution.	37
$Probs(C)$	Set of discrete probability spaces $(\Omega, \mathcal{F}, P)$ with no 0-probability elements such that $\Omega \subseteq C$.	37
A	Automaton.	37
$states()$	States of.	37
$start()$	Start states of.	37
$sig()$	Action signature of.	37
$ext()$	External actions of.	37
$int()$	Internal actions of.	37
$acts()$	Actions of.	37
$trans()$	Transitions of.	37
$\xrightarrow{a}$	Transition with action a.	38
$\Longrightarrow$	Weak transition.	38
α	Execution (fragment).	39
$fstate()$	First state of.	39
$lstate()$	Last state of.	39
$frag()$	Execution fragments of.	39
$exec()$	Executions of.	39
$\frown$	Concatenation of executions.	39
$\frown$	Transition prefixing operator.	52

$\leq$	Prefix of.	39
$\triangleright$	Suffix operator.	39
$\triangleright$	Transition suffixing operator.	52
β	Trace.	40
$traces()$	Traces of.	40
$\sqsubseteq_T$	Trace preorder.	40
$\Vert$	Parallel composition operator.	41
M	Probabilistic automaton.	46
δ	Termination or deadlock symbol.	46
$ctrans()$	Combined transitions of.	48
H	Probabilistic execution (fragment).	49
$prfrag()$	Probabilistic execution fragments of.	49
$prexec()$	Probabilistic executions of.	49
$\alpha\downarrow$	From an execution of a probabilistic execution fragment to an execution fragment of a probabilistic automaton.	51
$\alpha\uparrow q_0$	From an execution fragment of a probabilistic automaton to an execution of a probabilistic execution fragment.	51
tr	Transition.	51
$\mathcal{P}_{tr}$	Probability space in the transition tr, i.e., $tr = (s, \mathcal{P}_{tr})$ or, if tr is simple, $tr = (s, a, \mathcal{P}_{tr})$.	51
V	Set of actions.	51
U	Set of states.	51
tr_s^M	Transition leaving from state s in the fully probabilistic automaton M.	51
$\mathcal{P}_H$	Probability space associated with the probabilistic execution fragment H.	52
C_α	Cone with prefix α.	53
$\xrightarrow{a}_C$	Combined transition.	58
$\xRightarrow{a}_C$	Weak combined transition.	59
$\mathcal{O}$	Generator of a weak transition.	60
$\upharpoonright$	Action restriction operator.	64
$\lceil$	Projection operator.	65
$\rceil$	Reverse of projection.	66
$Rename_\rho()$	Renaming operator.	72
$Hide_I()$	Hiding operator.	73
$Advs()$	Adversaries for.	80
$prexec(M, \mathcal{A}, \alpha)$	Probabilistic execution fragment of M generated by adversary $\mathcal{A}$ with starting condition α.	80
e	Event schema.	82
$Cones()$	Function that identifies the points of satisfaction of a finitely satisfiable event schema.	83
$\circ_{Cones}$	Concatenation of two event schemas.	83
$\Pr_{Advs,\Theta}(e)\ \mathcal{R}\ p$	Probabilistic statement.	84
$(\equiv, F)$	Oblivious relation.	92
$FIRST(\ldots)$	Coin event: first occurrence of an action among many.	107

Index