Forward and Backward Simulations
Part II: Timing-Based Systems

Nancy Lynch
MIT Laboratory for Computer Science
Cambridge, MA 02139, USA
`lynch@theory.lcs.mit.edu`

Frits Vaandrager
CWI
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands
`fritsv@cwi.nl`
University of Amsterdam, Programming Research Group
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands

September 9, 1993
(Revision of March 1, 1993 version)

Abstract

A general automaton model for timing-based systems is presented and is used as the context for developing a variety of simulation proof techniques for such systems. These techniques include (1) refinements, (2) forward and backward simulations, (3) forward-backward and backward-forward simulations, and (4) history and prophecy relations. Soundness and completeness results are given for these simulations. These results are largely analogous to the results in Part I of this paper for untimed systems. In fact, many of the results for the timed case are obtained as consequences of the analogous results for the untimed case.

1991 Mathematics Subject Classification: 68Q60, 68Q68.
1991 CR Categories: C.3, F.1.1, F.3.1.
Keywords and Phrases: Timing-based systems, real-time, timed automata, simulations, refinement mappings, forward simulations, backward simulations, forward-backward simulations, backward-forward simulations, history relations, prophecy relations.
Notes: This work was supported by ONR contracts N00014-85-K-0168 and N00014-91-J-1988, by NSF grant CCR-8915206, and by DARPA contracts N00014-89-J-1988 and N00014-92-J-4033. Part of this work took place while the second author was employed by the Ecole des Mines, CMA, Sophia Antipolis, France. The second author also received partial support from the ESPRIT Basic Research Action 7166, CONCUR2. An earlier verion of this paper (Part I+II) appeared as [14].

1 Introduction

In this paper, we describe a basic *timed automaton* model for real-time and other timing-based systems. We intend that the model should provide a suitable basis for formal reasoning about timing-based systems, in particular, for verification of their correctness and for analysis of their complexity. It should support many different kinds of correctness proof techniques, including process algebraic and assertional methods. So far, process algebraic and assertional methods have been used primarily to prove properties of untimed (asynchronous) systems; we would also like to use them for timing-based systems. Also, the kinds of properties generally proved using these methods have been "ordinary" safety properties; we would like also to use similar methods to prove timing properties (e.g., upper and lower bounds on time).

We use our model to express some powerful simulation techniques for proving correctness of timing-based systems. Since simulation techniques are a special case of assertional methods, the automata-theoretic style of our model is the natural style for presenting them. However, we expect that the model can also serve as a semantic model for interesting algebraic languages, and thus that process algebraic methods can also be employed in the same framework.

The kinds of simulation techniques we describe are timed versions of the techniques studied in Part I of this paper, [15]. That is, we consider timed versions of *refinements*, *forward simulations*, *backward simulations*, *forward-backward*, *backward-forward simulations*, *history relations* and *prophecy relations*. Again, we prove soundness and completeness theorems, and theorems relating the different kinds of simulations to each other.

The goal of extending simulation techniques to timing-based systems is also the motivation for the work of Lynch and Attiya in [12]. That work, however, only explores forward simulations. Also, the model used in [12] has considerably more structure than the very general model proposed here; it is based closely on the *timed automaton* model of Merritt, Modugno and Tuttle [16], which assumes that the system being modeled is describable in terms of a collection of separate tasks, each with associated upper and lower bounds on its speed. This extra structure supports the development of a useful *progress measure* proof method, which we do not develop here. On the other hand, the basic theorems about forward simulations that appear in [12] are stated in a setting that has more structure than is really necessary for those theorems. In this paper, we make only those assumptions that are needed for the basic results about simulation proof techniques.

We propose a notion of *timed automaton*, which is just an automaton (or labeled transition system) equipped with some additional structure. Specifically, each state of the automaton has an associated *current time* value. (Thus we use *absolute* time in the sense of [2].) The actions of the automaton are of three kinds: *visible actions*, a special *internal action* τ, and a special *time-passage action* ν. As in many other formalisms for real-time (See, for instance, [2, 5, 8, 16, 18, 19, 23].), all actions except for the time-passage action are modeled as occurring instantaneously, i.e., they do not change the time component of the state.

To specify times, we use a *dense* time domain, specifically, the nonnegative reals (starting with time 0 in the initial state), and we impose no lower bound on the time between events. This choice distinguishes our work from many others', e.g., [7, 8, 18, 19, 21, 24], in which discrete time values or universal positive lower bounds on step time are used. Use of realvalued time is less restrictive, and we believe that the extra flexibility will be useful in the design and analysis of timing-based distributed algorithms. The penalty we pay for this flexibility is that our automata may admit some "Zeno executions", i.e., infinite executions in which the time component is bounded.

Timed automata are required to satisfy a small set of basic axioms that express natural properties of time. For instance, there is an axiom saying that time-passage actions must increase the time, and another saying that all the other actions are instantaneous. Also, if time can advance by a particular amount in two steps, it can also advance by the same amount in a single step. Finally, if time can advance by a particular amount in one time-passage step, then it is possible to assign states to all the times that occur during the step. We attempt to use as few axioms as possible to obtain the results about simulations. Later, as we try to express different proof methods in terms of this model, we expect to have to add additional requirements to obtain the desired properties.

In order to define correctness for timed automata, we require notions of external behavior. We emphasize two such notions. First, as the finite behaviors of a timed automaton, we take the *finite timed traces*, each of which consists of a finite sequence of timed external actions together with a final time. Second, as the infinite behaviors, we take the *admissible timed traces*, each of which consists of a sequence of timed external actions that can occur in an execution in which the time grows unboundedly (i.e., a "non-Zeno" infinite execution). In [22] it is argued that inclusion of finite and admissible timed traces is a good notion of implementation, provided that the implementation automaton satisfies the technical condition of *I/O feasibility*. In this paper we will not be concerned with I/O feasibility. Instead our objective will be to develop simulation-style proof techniques to show inclusion of timed traces.

Even though our notion of timed automata has less structure than those of [16] and [12], it is closely related to those models. Ours can be regarded as a generalization of the model in [12], in which the notion of separate tasks is removed. (There are some minor distinctions; for instance, we do not include names for internal actions, but label them all by the special symbol τ. This distinction is unimportant in a setting without separate tasks.) Also, the model of [16] includes treatment of fairness and liveness, whereas our model does not. (The model in [16] was originally designed as an extension of the untimed input/output automaton model of [13], which emphasizes the notion of *fair execution*.) The reason we have not equipped our model with facilities for handling fairness and liveness is that we believe that in the setting of timing-based systems, most properties of practical importance can be expressed as safety properties, given the admissibility assumption that time increases without bound. The absence of fairness and liveness considerations in our model seems to remove various technical and philosophical complications, and to lead to simpler and more systematic proof techniques. In cases where liveness properties are required, it will be necessary to augment the model of this paper with special liveness conditions; we anticipate that all the results of this paper should extend to this augmented model.

Our model can also be considered to be a generalization of the finite-state model of Alur and Dill [1], since we are not restricted to the special structure of [1] for specifying time bounds. Also, we do not impose any finiteness restrictions on our automata. On the other hand, the model of [1] does have some facility for modeling liveness.

We refer the reader to Part I for a general discussion of the various simulation techniques, for the case of untimed systems. The simulations we consider here are defined similarly to those in Part I but using our new notions of external behavior, and with the addition of the requirement that corresponding states must have the same current time component. Again, we give soundness and completeness results, as well as results relation the various kinds of simulations to each other.

In order to prove our results for timed automata, we find it convenient to use the corresponding results already proved in Part I, for the simpler untimed setting. The results for the timed setting are analogous to those for the untimed setting. In fact, in most cases, our results for the timed setting are derived from those for the untimed setting, while in the remaining cases, new proofs analogous to those in Part I are presented. An advantage of this two-phase approach is that it highlights the adaptability of the various verification techniques from the untimed to the timed setting.

Because of the logical dependence of this paper on Part I, we have not tried to write this paper in a self-contained manner. Thus, we employ freely the notation and definitions of Part I, and refer in many places to the results from Part I.

We consider the main contributions of this paper to be the following. First, we introduce the notions of a timed automaton and its behavior, and extend existing simulation notions to this new setting. Second, we give an organized presentation, in terms of this general timed automaton model, of a wide range of important simulation techniques, together with their basic soundness and completeness properties. Third, our presentation style, which bases the timed case on the untimed case, explains the connections between these two settings.

The rest of the paper is organized as follows. Section 2 contains the definitions for timed automata and their executions and traces. Section 3 contains some definitions and results for restricted types of timed automata. Section 4 discusses the structures that can be obtained as the behaviors of timed automata. Section 5 contains the major results of the paper – the definitions of all the timed simulations and the results that follow from the analogous results in Part I. Section 6 contains the results about the new simulations that require new proofs. Finally, Section 7 contains our conclusions.

2 Timed Automata and Their Behaviors

This section presents the basic definitions and results for timed automata. The development is generally parallel to that in Section 3 of Part I of this paper [15]. A

2.1 Timed Automata

A *timed automaton* A is an automaton whose set of actions contains a special *time-passage* action ν. The set of *visible* actions is defined by $vis(A) \triangleq ext(A) - \{\nu\}$. As an additional component, a timed automaton contains a mapping $.now_A : states(A) \rightarrow \mathsf{R}^{\geq 0}$; this indicates the current time in the given state. As in Part I, we will suppress the subscript A where no confusion is likely. We assume that a timed automaton satisfies the following axioms.

S1 If $s \in start(A)$ then $s.now = 0$.

S2 If $s' \xrightarrow{a} s$ and $a \neq \nu$, then $s'.now = s.now$.

S3 If $s' \xrightarrow{\nu} s$ then $s'.now < s.now$.

S4 If $s' \xrightarrow{\nu} s''$ and $s'' \xrightarrow{\nu} s$, then $s' \xrightarrow{\nu} s$.

In order to state the last axiom, we need an auxiliary definition. Suppose s', s are states of A with $t' = s'.now < s.now = t$. A *trajectory* from s' to s is a function $w : [t', t] \rightarrow states(A)$ satisfying the following conditions:

1. $w(t') = s'$,
2. $w(t) = s$, and
3. $w(t_1) \xrightarrow{\nu} w(t_2)$ for all $t_1, t_2 \in [t', t]$ with $t_1 < t_2$.
4. $w(t'').now = t''$ for all $t'' \in [t', t]$.

Now our final axiom can be formulated as follows:

S5 If $s' \xrightarrow{\nu} s$ then there exists a trajectory from s' to s.

Axiom **S1** says that the time is always 0 in a start state. Axiom **S2** says that non-time-passage actions do not change the time; that is, they occur "instantaneously", at a single point in time. Axiom **S3** says that time-passage actions must cause the time to increase. Axiom **S4** allows repeated time-passage steps to be combined into one step. Axiom **S5** says that if time can pass from time t' to time t then it is possible to associate states with all times in the interval in a "consistent" way.[1]

Suppose S is a nonempty set of states of A. Then $S.ftime$ and $S.ltime$ denote the infimum and supremum of the $.now$ values of the states in S, respectively. If S contains a unique state with minimal $.now$-value, then $S.fstate$ is defined to be this state. Otherwise $S.fstate$ is undefined. Similarly, $S.lstate$ denotes the unique state in S with maximal $.now$-value, if it exists, and is undefined otherwise.

In this paper $A, B, \ldots$ will range over *timed* automata.

2.2 Timed Executions

Since a timed automaton is a special case of an automaton (as defined in Part I), we already have a notion of execution for timed automata. However, this type of execution only describes the system state at a countable number of points in time. Since our trajectory axiom gives us the ability to associate states with all real times, we define a notion of *timed execution*, which includes such information. The usual kind of execution can be regarded as "sampling" our kind of timed execution at countably many points in time, as we show in Subsection 2.4.1 below.

[1] This axiom is a strengthening of a similar axiom used in [14, 20, 23], which, rephrased in our terminology, reads:

S5′ If $s' \xrightarrow{\nu} s$ and $s'.now < t < s.now$, then there is an s'' with $s''.now = t$ such that $s' \xrightarrow{\nu} s''$ and $s'' \xrightarrow{\nu} s$.

We strengthen **S5′** in this paper because the new definition seems natural to us and because we need it for some of our later results (Lemma 3.2, for instance). In Appendix A, we discuss the relationship between axioms **S5** and **S5′** in more detail and show in particular that **S5′** does not in general imply **S5**.

2.2.1 Basic Definitions

A *timed execution fragment* of a timed automaton A is a finite or infinite alternating sequence $\Sigma = S_0 a_1 S_1 a_2 S_2 \cdots$ of nonempty sets of states of A and actions in $acts(A) - \{\nu\}$, beginning with a set of states, and if it is finite also ending with a set of states, such that the following holds for each index i.

1. The function $.now$ is injective on S_i.

2. If S_i is not the last state set in Σ then the set $S_i.now$ is a closed interval; if S_i is the last state set in Σ then $S_i.now$ is a left-closed interval (and either closed or open to the right).

3. For all $s', s \in S_i$, $s'.now < s.now$ implies $s' \xrightarrow{\nu} s$.

4. If S_i is not the last state set, then $S_i.lstate \xrightarrow{a_{i+1}} S_{i+1}.fstate$.

Property 1 says that S_i contains at most one state for every value of the current time variable $.now$. Property 2 says that S_i contains states for exactly the times in some interval of the real numbers. Property 3 describes a consistency condition among the different states in S_i. Property 4 describes a consistency condition between values in consecutive intervals.

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

If Σ is a timed execution fragment, then $\Sigma.ftime$ denotes the minimum $.now$ value in the first state set in Σ, and $\Sigma.ltime$ denotes the supremum of the $.now$ values of all states in Σ. Note that, if Σ is a timed execution, then $\Sigma.ftime = 0$. Also, we define $\Sigma.fstate$ to be the state with the minimum $.now$ value in the first state set in Σ.

2.2.2 Finite, Admissible and Zeno Timed Executions

In this paper, we will be interested in certain subclasses of the set of timed executions: the *finite*, *admissible* and *Zeno* timed executions.

We define a timed execution fragment Σ to be *finite* if Σ is a finite sequence and the interval associated with the last state set is closed. If Σ is finite, then it contains only finitely many events. Moreover, $\Sigma.ltime$ is the largest $.now$ component in the last state set in Σ, and $\Sigma.ltime$ is finite. If Σ is finite then we define $\Sigma.lstate$ to be the unique state in the last state set in Σ with its $.now$ value equal to $\Sigma.ltime$.

We define a timed execution fragment Σ to be *admissible* if $\Sigma.ltime = \infty$.

Timed automata do not include any features for describing liveness or fairness (such as the class structure of I/O automata). We believe that such features are less important in the timed setting than they are in the untimed setting. Instead, we generally focus on the *admissible* timed executions. The notion of an admissible timed execution is more tractable mathematically than the notion of a fair execution in the I/O automaton model; this is because the admissible timed executions of a timed automaton can be expressed as the limits of infinite sequences of finite timed executions.

More precisely, if $\Sigma = S_0 a_1 S_1 \ldots$ and $\Sigma' = S'_0 a'_1 S'_1 \ldots$ are timed execution fragments, then define Σ' to be a *t-prefix* of Σ, denoted by $\Sigma' \preceq \Sigma$, if either $\Sigma' = \Sigma$, or else $\Sigma' = S'_0 a'_1 S'_1 \ldots a'_i S'_i$

is finite, the prefix $S'_0 a'_1 S'_1 \ldots a'_i$ of Σ' is equal to the prefix $S_0 a_1 S_1 \ldots a_i$ of Σ, and the final state set S'_i of Σ' is included in the next state set S_i of Σ. Then the admissible timed executions are exactly the limits of the infinite sequences of finite timed executions, where each timed execution in the sequence is a t-prefix of the next and the maximum $.now$ values of the final state sets go to ∞. This characterization permits the reduction of questions about infinite behaviors to questions about finite behaviors. A similar reduction is not possible in untimed models that incorporate fairness.

A timed execution fragment is said to be *Zeno* if it is neither finite nor admissible. Note that there are two types of Zeno timed executions: those containing infinitely many actions, but for which there is a finite upper bound on the $.now$ values that occur in the state sets, and those containing finitely many actions, but for which the time interval of the final state set is right-open. Zeno timed executions are those in which an infinite amount of activity occurs within a bounded period of time. (For the second type of Zeno timed execution described above, the "infinite amount of activity" corresponds to the infinite number of time-passage steps needed to span the right-open interval.) Some models of real-time computation, for instance the earlier version of Timed CSP [21], exclude Zeno executions altogether, but we allow them in order to make our results as general as possible.

We note that, according to our definitions, there are timed automata in which from some (or even all) states no admissible timed execution fragment is possible. This can be, for instance, because from these states time can continue advancing, but not beyond a certain point (that is, all timed execution fragments starting from these states are Zeno), or because time cannot advance at all (that is, a *time deadlock* occurs). Time deadlocks have also been studied in the context of several process algebraic models ([2, 8, 18]). Our model does allow time deadlocks. However, in several of our theorems we will require that the automata be *feasible*. A timed automaton A is *feasible* provided that each finite timed execution is a t-prefix of some admissible timed execution. Thus, a feasible timed automaton does not have time deadlocks, although it may have Zeno timed executions.

We denote by *t-frag*$^*(A)$, *t-frag*$^\infty(A)$ and *t-frag*(A) the sets of finite, admissible and all timed execution fragments of A. Similarly, we denote by *t-execs*$^*(A)$, *t-execs*$^\infty(A)$ and *t-execs*(A) the sets of finite, admissible and all timed executions of A.

2.3 Timed Traces

Since a timed automaton is an automaton (as defined in Part I), we already have a notion of trace for timed automata. However, the traces of timed automata do not provide a sufficiently abstract view of their behavior, because they do not contain information about the real times at which visible events occur, and because they do not reflect the invisible nature of time-passage actions. In this subsection, we define a notion of external behavior for timed automata which we call *timed traces*. These do not include explicit ν events, but do include information about the real time of visible events, and about the final time up to which the observation is made. As an auxiliary definition along the way to the definition of a timed trace, we define the notion of a *timed sequence pair*.

2.3.1 Timed Sequence Pairs

A *timed sequence* over a given set K is defined to be a (finite or infinite) sequence δ over $K \times \mathsf{R}^{\geq 0}$ in which the time components are nondecreasing, i.e., $t \leq t'$ if (k, t) and (k', t') are consecutive elements in δ. We say that δ is *Zeno* if it is infinite and the limit of the time components is finite.

A *timed sequence pair* over K is a pair $p = (\delta, t)$, where δ is a timed sequence over K and $t \in \mathsf{R}^{\geq 0} \cup \{\infty\}$, such that t is greater or equal than all time components in δ. We write $p.seq$, and $p.ltime$ for the two respective components of p. We define $p.ftime$ to be equal to the time component of the first pair in $p.seq$ in case $p.seq$ is nonempty, and equal to $p.ltime$ otherwise. We denote by $tsp(K)$ the set of timed sequence pairs over K. We say that a timed sequence pair p is *finite* if both $p.seq$ and $p.ltime$ are finite, and *admissible* if $p.seq$ is not Zeno and $p.ltime = \infty$.

Let p and p' be timed sequence pairs over K such that p is finite and $p.ltime \leq p'.ftime$. Then define $p; p'$ to be the timed sequence pair $(p.seq\ p'.seq, p'.ltime)$. If p and q are timed sequence pairs over a set K, then p is a *prefix* of q, notation $p \leq q$, if either $p = q$, or p is finite and there exists a timed sequence pair p' such that $q = p; p'$.

Lemma 2.1 *$\leq$ is a partial ordering on timed sequence pairs over K.*

2.3.2 Timed Traces of Timed Automata

Suppose $\Sigma = S_0 a_1 S_1 a_2 S_2 \cdots$ is a timed execution fragment of a timed automaton A. For each a_i, define the *time of occurrence* t_i to be $S_{i-1}.ltime$, or equivalently, $S_i.ftime$. Let γ be the sequence consisting of the actions in Σ paired with their times of occurrence:

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

Then *t-trace*(Σ), the *timed trace* of Σ, is defined to be the pair

$$\textit{t-trace}(\Sigma) \triangleq (\gamma \lceil (vis(A) \times \mathsf{R}^{\geq 0}), \Sigma.ltime).$$

Thus, *t-trace*(Σ) records the occurrences of visible actions together with their times of occurrence, and the limit time of the timed execution fragment. Note that in a timed trace both τ and ν actions are supressed. It is easily checked that *t-trace*(Σ) is a timed sequence pair over $vis(A)$.

A *timed trace* of A is the timed trace of some finite or admissible timed execution of A. Thus, we explicitly exclude the traces of Zeno executions. We write *t-traces*(A) for the set of all timed traces of A, *t-traces*$^*(A)$ for the set of *finite* timed traces, i.e., those that originate from a finite timed execution of A, and *t-traces*$^\infty(A)$ for the *admissible* timed traces, i.e., those that originate from an admissible timed execution of A. The following proposition is a direct consequence of the definitions.

Proposition 2.2 *The sets t-traces$^*(A)$ and t-traces$^\infty(A)$ consist of finite timed sequence pairs and admissible timed sequence pairs over vis(A), respectively.*

These notions induce three preorders on timed automata in an obvious way: $A \leq^{\mathrm{t}}_{\mathrm{T}} B \triangleq \textit{t-traces}(A) \subseteq \textit{t-traces}(B)$, $A \leq^{\mathrm{t}}_{*\mathrm{T}} B \triangleq \textit{t-traces}^*(A) \subseteq \textit{t-traces}^*(B)$, and $A \leq^{\mathrm{t}}_{\infty\mathrm{T}} B \triangleq \textit{t-traces}^\infty(A) \subseteq \textit{t-traces}^\infty(B)$. The kernels of these preorders are denoted by $\equiv^{\mathrm{t}}_{\mathrm{T}}$, $\equiv^{\mathrm{t}}_{*\mathrm{T}}$ and $\equiv^{\mathrm{t}}_{\infty\mathrm{T}}$, respectively.

2.3.3 Moves

Suppose A is a timed automaton, s' and s are states of A, and p is a timed sequence pair over $vis(A)$. Then we say that (s', p, s) is a *t-move* of A, and write $s' \stackrel{p}{\leadsto}_A s$, or just $s' \stackrel{p}{\leadsto} s$ when A is clear, if A has a finite timed execution fragment Σ with $\Sigma.fstate = s'$, $t\text{-}trace(\Sigma) = p$ and $\Sigma.lstate = s$.

Lemma 2.3 *Suppose $s' \stackrel{p}{\leadsto}_A s$ and $p = q; r$. Then there exists s'' such that $s' \stackrel{q}{\leadsto}_A s''$ and $s'' \stackrel{r}{\leadsto}_A s$.*

2.4 Relationships Between Timed and Untimed Execution Fragments

In this section, we observe some close connections between the timed execution fragments and the ordinary execution fragments of a timed automaton. Note that the rest of the paper does not depend on this subsection, the reader who is comfortable with our definition of a timed execution can skip to Section 3.

2.4.1 Sampling

Roughly speaking, an ordinary execution fragment can be regarded as "sampling" the state information in a timed execution fragment at a countable number of points in time. Formally, we say that an execution fragment $\alpha = s_0 a_1 s_1 \ldots$ of A *samples* a timed execution fragment $\Sigma = S_0 b_1 S_1 \ldots$ of A if there is a monotone increasing function $f : \mathsf{N} \to \mathsf{N}$ such that the following conditions are satisfied.

1. $f(0) = 0$,
2. $b_i = a_{f(i)}$ for all $i \geq 1$,
3. $a_j = \nu$ for all j not in the range of f,
4. For all $i \geq 0$ such that S_i is not the last set in Σ,
   (a) $s_j \in S_i$ for all j, $f(i) \leq j < f(i+1)$,
   (b) $s_{f(i)}.now = S_i.ftime$, and
   (c) $s_{f(i+1)-1}.now = S_i.ltime$.
5. If S_i is the last set in Σ, then
   (a) $s_j \in S_i$ for all j, $f(i) \leq j$,
   (b) $s_{f(i)}.now = S_i.ftime$, and
   (c) the $.now$ values of states in $\{s_j : f(i) \leq j\}$ are cofinal with the $.now$ values of states in S_i.[2]

[2]This simply means that for every state in $\{s_j : f(i) \leq j\}$, there is a state in S_i with a $.now$ value that is at least as great, and vice versa.

In other words, the function f in this definition maps the (indices of) events in Σ to corresponding events in α, in such a way that exactly the non-time-passage events of α are included in the image. Condition 4 is a consistency condition relating the first and last times for each non-final set to the times produced by the appropriate steps of α. Condition 5 gives a similar consistency condition for the first time of the final set (if any); in place of the consistency condition for the last time, there is a "cofinality" condition asserting that the times grow to the same limit in both executions.

The following two straightforward lemmas show the relationship between timed execution fragments and ordinary execution fragments.

Lemma 2.4 *If α is an execution fragment of A then there is a timed execution fragment Σ of A such that α samples Σ.*

Lemma 2.5 *If Σ is a timed execution fragment of A then there is an execution fragment α of A such that α samples Σ.*

Define a state s to be *t-reachable* in timed automaton A provided that there is a finite timed execution Σ such that $\Sigma.lstate = s$. The following lemma shows that t-reachability can equivalently be defined by means of ordinary executions.

Lemma 2.6 *State s is t-reachable in A if and only if it is reachable in A, i.e., there is a finite (ordinary) execution of A that ends in s.*

Proof: Straightforward using Lemmas 2.4 and 2.5. ■

An important consequence of Lemma 2.6 is that any technique that can prove that a property holds for all final states of finite (ordinary) executions is a sound technique for proving that a property holds in all t-reachable states of a timed automaton. Most importantly, induction on the steps of ordinary executions is sound in this sense. Conversely, any technique that can prove that a property holds for all t-reachable states also proves that it holds for all reachable states.

2.4.2 Finite, Admissible and Zeno Execution Fragments

Here we define the finite, admissible and Zeno execution fragments, and relate them to the finite, admissible and Zeno timed execution fragments defined in Section 2.2.2.

As in Part I, an execution fragment α is defined to be *finite* if it is a finite sequence. An execution fragment α is defined to be *admissible* if there is no finite upper bound on the $.now$ values of the states in α. Finally, an execution fragment is said to be *Zeno* if it is neither finite nor admissible.

Lemma 2.7 *If α samples Σ then*

1. *α is finite iff Σ is finite,*
2. *α is admissible iff Σ is admissible, and*
3. *α is Zeno iff Σ is Zeno.*

2.4.3 Timed Traces

It is possible to give a sensible definition of timed trace for an ordinary execution fragment of a timed automaton. Namely, suppose $\alpha = s_0 a_1 s_1 a_2 s_2 \cdots$ is a execution fragment of a timed automaton A. First, define $\alpha.ltime$ to be the supremum of the $:.now$ values of all the states in α. Then let γ be the sequence consisting of the actions in α paired with their times of occurrence:

$$\gamma = (a_1, s_1.now)(a_2, s_2.now) \cdots .$$

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

$$(\gamma\lceil(vis(A) \times \mathsf{R}^{\geq 0}), \alpha.ltime).$$

The following lemma shows that the definitions of timed traces for execution fragments and timed execution fragments are properly related:

Lemma 2.8 *If α samples Σ then $t\text{-}trace(\alpha) = t\text{-}trace(\Sigma)$.*

3 Restricted Kinds of Timed Automata

In this section, we continue our parallel development with Part I by defining certain restricted kinds of automata that are useful in our proofs. Recall that in the earlier paper, we defined what it meant for an untimed automaton to be deterministic, fin and a forest. We define analogous notions here.

Each timed automaton that contains a ν step fails to be deterministic in the sense of Part I, as a consequence of the trajectory axiom **S5**. Thus the notion of determinism is not useful for timed automata, and we will use the following notion of t-determinism instead.

We say that A is *t-deterministic* if $|start(A)| = 1$, and for any state s' and any finite timed sequence pair p over $vis(A)$, there is at most one state s such that $s' \overset{p}{\leadsto}_A s$. The following lemma gives a "local" characterization of t-determinism.

Lemma 3.1 *A timed automaton A is t-deterministic iff $|start(A)| = 1$, every τ transition is of the form (s, τ, s) for some s, and*

$$s \xrightarrow{a} r \;\wedge\; s \xrightarrow{a} r' \;\wedge\; r.now = r'.now \quad\Rightarrow\quad r = r'. \tag{1}$$

Proof: "$\Rightarrow$" $|start(A)| = 1$ by definition of t-determinism. There can be no step $s' \xrightarrow{\tau} s$ with $s' \neq s$, because, taking $p = (\lambda, s.now)$, this would imply $s' \overset{p}{\leadsto}_A s'$ and $s' \overset{p}{\leadsto}_A s$, which contradicts t-determinism. Thus every τ transition is of the form (s, τ, s) for some s, Finally, suppose $s \xrightarrow{a} r$, $s \xrightarrow{a} r'$ and $r.now = r'.now = t$. Define $p = ((a, t), t)$ if $a \in vis(A)$, and $p = (\lambda, t)$ otherwise. Then $s \overset{p}{\leadsto}_A r$ and $s \overset{p}{\leadsto}_A r'$. But this means that $r = r'$, because of t-determinism.

"$\Leftarrow$" Suppose that $p = (\delta, t)$ is a timed sequence pair over $vis(A)$, and s, s', s'' are states such that $s'' \overset{p}{\leadsto}_A s$ and $s'' \overset{p}{\leadsto}_A s'$. We must prove $s = s'$. A has finite timed executions Σ and Σ' with $\Sigma.fstate = \Sigma'.fstate = s''$, $t\text{-}trace(\Sigma) = t\text{-}trace(\Sigma') = p$, $\Sigma.lstate = s$ and $\Sigma'.lstate = s'$. Without loss of generality we can assume that Σ and Σ' contain no τ

steps: since all τ steps of A are of the form (s, τ, s), such steps can be removed from timed executions. Since Σ and Σ' have the same timed trace, $s.now = s'.now$. Also because Σ and Σ' have the same timed trace and because they do not contain τ's, both sequences have the same length. By induction on this length we establish that $\Sigma = \Sigma'$. This then implies that $s = \Sigma.lstate = \Sigma'.lstate = s'$.

Suppose that Σ consists of just one state set, say S. Then Σ' also consists of a single state set, call it S'. In this case it easily follows via (1) that $S = S'$, which immediately implies $\Sigma = \Sigma'$.

Now suppose Σ is of the form $\Sigma_1 a S$. Then Σ' is of the form $\Sigma_2 a S'$, with $t\text{-}trace(\Sigma_1) = t\text{-}trace(\Sigma_2) = p'$, for some p'. Let $\Sigma_1.fstate = s_1$ and $\Sigma_2.fstate = s_2$. By induction hypothesis it follows that $\Sigma_1 = \Sigma_2$, and hence $s_1 = s_2$. Application of (1) now gives that $S.fstate = S'.fstate$, and via a subsequent application of the same property it follows that $S = S'$. From this $\Sigma = \Sigma'$ follows, as required. ■

Likewise, any timed automaton that contains a ν step is not fin, again as a consequence of **S5**, so we define a new notion of t-finite invisible nondeterminism. We say that A has *t-finite invisible nondeterminism (t-fin)* if $start(A)$ is finite, and for any state s' and any finite timed sequence pair p over $vis(A)$, there are only finitely many states s such that $s' \overset{p}{\rightsquigarrow}_A s$.

Also, any timed automaton that contains a ν step cannot be a forest, because if a state has one incoming time-passage step then axiom **S5** implies that it must have an infinite number of them. Recall that a forest is characterized by the property that for each state there is a unique execution that leads to it. In analogy we define a *t-forest* to be a timed automaton with the property that for each state s there is a unique timed execution Σ that leads to it, i.e., such that $\Sigma.lstate = s$. The following lemma gives a "local" characterization of a t-forest.

Lemma 3.2 *A timed automaton A is a t-forest iff all states of A are reachable, start states have no incoming steps, and for every state s, if there are two distinct steps leading to s, $r \xrightarrow{a} s$ and $r' \xrightarrow{a'} s$, then $a = a' = \nu$ and either $r \xrightarrow{\nu} r'$ or $r' \xrightarrow{\nu} r$.*

Proof: "$\Rightarrow$" All states in a t-forest are reachable by definition. It is also easy to see that start states have no incoming steps. Suppose that $r \xrightarrow{a} s$ and $r' \xrightarrow{a'} s$, with $(r, a) \neq (r', a')$. Let Σ and Σ' be the unique timed executions leading to r and r', respectively. We consider four cases.

1. $a = a' \neq \nu$. Then $r \neq r'$ and hence $\Sigma \neq \Sigma'$. It is easy to see that both $\Sigma a\{s\}$ and $\Sigma' a\{s\}$ are timed executions leading to s. Since A is a t-forest we have that $\Sigma a\{s\} = \Sigma' a\{s\}$, which implies $\Sigma = \Sigma'$. This is a contradiction.

2. $a \neq a' = \nu$. Again we derive a contradiction by constructing two different timed executions leading to s. The first one is $\Sigma a\{s\}$. Since $r' \xrightarrow{\nu} s$, there exists a trajectory w from r' to s, by axiom **S5**. Let Σ'' be the sequence obtained from Σ' by extending the last state set of Σ' with the states in the range of w. Then it is easy to check that Σ'' is a timed execution leading to s. Since $\Sigma a\{s\}$ and Σ'' are clearly different, we have a contradiction as required.

3. $\nu = a \neq a'$. Analogous to case (2).

4. $a = a' = \nu$. Using axiom **S5**, we can extend both Σ and Σ' to timed executions Σ_1 and Σ_1', respectively, that lead to s; the construction ensures that r is in the last state set of Σ_1 and r' is in the last state set of Σ_1'. Since A is a t-forest, we have $\Sigma_1 = \Sigma_1'$, so r' is in the last state set of Σ_1. Then by definition of a timed execution, either $r \xrightarrow{\nu} r'$ or $r' \xrightarrow{\nu} r$.

"$\Leftarrow$" Because all states of A are reachable we know that for each state s there is at least one timed execution that leads to it. In order to show uniqueness, suppose that A has two timed executions, Σ and Σ' that lead to s. By induction on the sum of the lengths of Σ and Σ' we prove that $\Sigma = \Sigma'$. Let S be the last state set of Σ, and let S' be the last state set of Σ'. Moreover, let r (resp., r') be the state in S (resp., S') with minimum time.

First suppose that $r'.now < r.now$. Then S' contains a state r'' with $r''.now = r.now$. Since $r \xrightarrow{\nu} s$ and $r'' \xrightarrow{\nu} s$, it follows that $r = r''$. Since $r.now > 0$, S is not the first state set of Σ, so S is immediately preceded in Σ by a (non-ν) action. This implies that there is a non-ν transition of A leading to r. However, since $r = r''$, r also has an incoming ν-transition from r', and we have obtained a contradiction. A similar contradiction is reached if $r.now < r'.now$. Therefore, it must be that $r.now = r'.now$.

We now argue that $S = S'$: if not, then there must be states $u \in S$ and $u' \in S'$ such that $u \neq u'$ but $u.now = u'.now$. But this cannot be because both $u \xrightarrow{\nu} s$ and $u' \xrightarrow{\nu} s$.

Now suppose Σ consists of S only. Then the state r of S with minimum $.now$ value is a start state, and hence has no incoming steps. This implies that also Σ' must consist of S only. Symmetrically, if Σ' consists of S only, then so does Σ.

The remaining case to be considered is the one in which Σ is of the form $\Sigma_1 a S$ and Σ' is of the form $\Sigma_1' a' S$. It is easy to show that $a = a'$ and the last states of Σ_1 and Σ_1' are equal. Thus by induction hypothesis $\Sigma_1 = \Sigma_1'$, and $\Sigma = \Sigma'$ follows. ∎

Suppose A is a timed automaton. In analogy with the untimed case, the relation $\textit{t-after}(A)$ consists of those pairs $(p, s) \in \textit{tsp}(\textit{vis}(A)) \times \textit{states}(A)$ for which there is a finite timed execution of A with timed trace p and last state s.

$$\textit{t-after}(A) \triangleq \{(p, s) \mid \exists \Sigma \in \textit{t-execs}^*(A) : \textit{t-trace}(\Sigma) = p \text{ and } \Sigma.\textit{lstate} = s\}.$$

The relation $\textit{t-past}(A) \triangleq \textit{t-after}(A)^{-1}$ relates a state s of A to the timed traces of timed executions that lead to s.

Lemma 3.3

1. *If A is t-deterministic then $\textit{t-after}(A)$ is a function from $\textit{t-traces}^*(A)$ to $\textit{states}(A)$.*

2. *If A has t-fin then $\textit{t-after}(A)$ is image-finite.*

3. *If A is a t-forest then $\textit{t-past}(A)$ is a function from $\textit{states}(A)$ to $\textit{t-traces}^*(A)$.*

Proof: Parts 1 and 2 are straightforward from the definitions.

For 3, suppose that A is a t-forest. Because all states of A are reachable we know that for each state s of A, $t\text{-}past(A)[s]$ contains at least one element. But this element is uniquely determined by the unique timed execution that leads to s. ■

4 Timed Trace Properties

For each timed automaton A, its *timed behavior*, $t\text{-}beh(A)$, is defined by

$$t\text{-}beh(A) \quad \triangleq \quad (vis(A), t\text{-}traces(A)).$$

Completely analogously to the way in which we characterized, in Section 3.3 of Part I, the behaviors of automata in terms of trace properties, we now characterize the timed behaviors of timed automata in terms of *timed trace properties*.

A set of timed sequence pairs is *prefix closed* if, whenever a timed sequence pair is in the set all its prefixes are also. A *timed trace property* P is a pair (K, L) where K is a set of labels and L is a nonempty, prefix closed set of finite and admissible timed sequence pairs over K. We will refer to the constituents of P as $sort(P)$ and $t\text{-}traces(P)$, respectively. Also, we write $t\text{-}traces^*(P)$ for the set of finite timed sequence pairs in $t\text{-}traces(P)$, and $t\text{-}traces^\infty(P)$ for the set of admissible timed sequence pairs in $t\text{-}traces(P)$. For P and Q timed trace properties, we define $P \leq^{\mathrm{t}}_{*\mathrm{T}} Q \triangleq t\text{-}traces^*(P) \subseteq t\text{-}traces^*(Q)$, $P \leq^{\mathrm{t}}_{\infty\mathrm{T}} Q \triangleq t\text{-}traces^\infty(P) \subseteq t\text{-}traces^\infty(Q)$, and $P \leq^{\mathrm{t}}_{\mathrm{T}} Q \triangleq t\text{-}traces(P) \subseteq t\text{-}traces(Q)$. The kernels of these preorders are denoted by $\equiv^{\mathrm{t}}_{*\mathrm{T}}$, $\equiv^{\mathrm{t}}_{\infty\mathrm{T}}$ and $\equiv^{\mathrm{t}}_{\mathrm{T}}$, respectively.
P is *limit-closed* if each infinite chain $p_1 \leq p_2 \leq p_3 \leq \cdots$ of elements of $t\text{-}traces^*(P)$ in which time grows unboundedly has a limit in $t\text{-}traces^\infty(P)$, i.e., an admissible timed sequence pair p with for all i, $p_i \leq p$.

Lemma 4.1 *Suppose P and Q are timed trace properties with Q limit-closed. Then $P \leq^{\mathrm{t}}_{*\mathrm{T}} Q \Leftrightarrow P \leq^{\mathrm{t}}_{\mathrm{T}} Q$.*

A timed trace property P is *feasible* if every element of $t\text{-}traces^*(P)$ is a prefix of some element of $t\text{-}traces^\infty(P)$.

Lemma 4.2 *Suppose P and Q are timed trace properties such that P is feasible. Then $P \leq^{\mathrm{t}}_{\infty\mathrm{T}} Q \Leftrightarrow P \leq^{\mathrm{t}}_{\mathrm{T}} Q$.*

Lemma 4.3

1. *$t\text{-}beh(A)$ is a timed trace property.*

2. *If A has t-fin then $t\text{-}beh(A)$ is limit-closed.*

3. *If A is feasible then $t\text{-}beh(A)$ is feasible.*

4. *$A \leq^{\mathrm{t}}_{\mathrm{T}} B \Leftrightarrow t\text{-}beh(A) \leq^{\mathrm{t}}_{\mathrm{T}} t\text{-}beh(B)$, $A \leq^{\mathrm{t}}_{*\mathrm{T}} B \Leftrightarrow t\text{-}beh(A) \leq^{\mathrm{t}}_{*\mathrm{T}} t\text{-}beh(B)$, and $A \leq^{\mathrm{t}}_{\infty\mathrm{T}} B \Leftrightarrow t\text{-}beh(A) \leq^{\mathrm{t}}_{\infty\mathrm{T}} t\text{-}beh(B)$.*

Proof: We sketch the proof of 2; it is analogous to that of Lemma 3.5 of Part I. Suppose A has t-fin and $p_1 \leq p_2 \leq \ldots$ is an infinite chain of timed sequence pairs in $t\text{-}traces^*(A)$ such that the limits of the time components of the p_i's is ∞. Assume without loss of generality that $p_i < p_{i+1}$, for all $i \geq 1$. Let p be the limit of the p_i's. We must show that $p \in t\text{-}traces^\infty(A)$.

We use Lemma 2.1 of Part I. This time, G is constructed as follows. The nodes are pairs (p_i, s), where p_i is one of the timed sequence pairs in the sequence above, and s is a state of A, such that $(p, s) \in t\text{-}after(A)$. There is an edge from node (p_i, s') to node (p_{i+1}, s) exactly if $s' \overset{q}{\leadsto}_A s$, where $p_{i+1} = p_i; q$. Using Lemma 2.3, it is not difficult to show that G satisfies the hypotheses of Lemma 2.1 of Part I. Then that lemma implies the existence of an infinite path in G starting at a root; given this path, it is easy to construct an admissible timed execution of A having p as its timed trace. ∎

Proposition 4.4

1. *If B has t-fin then $A \leq^{\mathrm{t}}_{*\mathrm{T}} B \Leftrightarrow A \leq^{\mathrm{t}}_{\mathrm{T}} B$.*
2. *If A is feasible then $A \leq^{\mathrm{t}}_{\infty\mathrm{T}} B \Leftrightarrow A \leq^{\mathrm{t}}_{\mathrm{T}} B$.*

Proof: Part 1 follows from Lemmas 4.1 and 4.3. Part 2 is a corollary of Lemmas 4.2 and 4.3. ∎

Example 4.5 We present two timed automata, TA and TB, which are in a sense the timed analogues of the automata A and B of Example 3.1 of Part I. The example illustrates the necessity of the t-fin condition in Prop. 4.4(1). Timed automaton TA does an a-action at integer times:

- $states(TA) = \mathsf{R}^{\geq 0} \times \mathsf{N}$,
- $(t, n).now_A = t$,
- $start(TA) = \{(0, 0)\}$,
- $acts(TA) = \{\tau, \nu, a\}$, and
- $steps(TA)$ is specified by:

$$(t', n) \xrightarrow{\nu} (t, n) \qquad \text{if } t' < t \leq n,$$

$$(t, n) \xrightarrow{a} (t, n+1) \quad \text{if } t = n.$$

Timed automaton TB also does an a-action at integer times, but only finitely often:

- $states(TB) = \mathsf{R}^{\geq 0} \times \mathsf{N} \times \mathsf{N}$,
- $(t, n, m).now_{TB} = t$,
- $start(TB) = \{(0, 0, m) \mid m \in \mathsf{N}\}$,

- $acts(TB) = \{\tau, \nu, a\}$, and
- $steps(TB)$ is specified by:

$$(t', n, m) \xrightarrow{\nu} (t, n, m) \qquad \text{if } t' < t \leq n,$$

$$(t, n, m) \xrightarrow{a} (t, n+1, m) \quad \text{if } t = n < m.$$

One can check that $TA \leq^{\mathrm{t}}_{*\mathrm{T}} TB$ but $TA \not\leq^{\mathrm{t}}_{\mathrm{T}} TB$.

Example 4.6 In order to see that the feasibility condition in Prop. 4.4(2) is actually needed, we consider a timed automaton *Zeno* with states drawn from the interval $[0, 1)$, $t.now = t$, start state 0, actions from $\{\tau, \nu\}$, and a step $t' \xrightarrow{\nu} t$ whenever $t' < t$. Since *Zeno* has no admissible timed traces, $Zeno \leq^{\mathrm{t}}_{\infty\mathrm{T}} TA$. However, because TA does not allow for initial time-passage steps, $Zeno \not\leq^{\mathrm{t}}_{\mathrm{T}} TA$.

We close this section with the construction of the *canonical timed automaton* for a given timed trace property.

For P a timed trace property, the associated *canonical timed automaton* $t\text{-}can(P)$ is the structure A given by

- $states(A) = t\text{-}traces^*(P)$,
- $(\delta, t).now_A = t$,
- $start(A) = \{(\lambda, 0)\}$,
- $acts(A) = sort(P) \cup \{\tau, \nu\}$, and
- $steps(A)$ is specified by:

$$(\delta', t') \xrightarrow{a}_A (\delta, t) \quad \text{if } a \in vis(A) \wedge t' = t \wedge \delta'(a, t) = \delta,$$

$$(\delta', t') \xrightarrow{\nu}_A (\delta, t) \quad \text{if } t' < t \wedge \delta' = \delta.$$

(It is not hard to check that $t\text{-}can(P)$ is indeed a timed automaton).

Lemma 4.7 *Suppose P is a timed trace property. Then*

1. *$t\text{-}can(P)$ is t-deterministic and is a t-forest,*
2. *$t\text{-}beh(t\text{-}can(P)) \equiv^{\mathrm{t}}_{*\mathrm{T}} P$,*
3. *$P \leq^{\mathrm{t}}_{\mathrm{T}} t\text{-}beh(t\text{-}can(P))$, and*
4. *if P is limit-closed then $t\text{-}beh(t\text{-}can(P)) \equiv^{\mathrm{t}}_{\mathrm{T}} P$.*

Proof: Part 1 follows easily using Lemmas 3.1 and 3.2. Part 2 follow from the definitions. Since *t-can*(P) has t-fin, it follows by Lemma 4.3 that *t-beh*(*t-can*(P)) is limit-closed. Now 3 and 4 follow by combination of 2 and Lemma 4.1. ■

Lemma 4.8 *Suppose A and B are timed automata. Then*

1. *t-can*(*t-beh*(A)) *is t-deterministic and is a t-forest,*
2. *t-can*(*t-beh*(A)) $\equiv^{\mathrm{t}}_{*\mathrm{T}}$ A*,*
3. $A \leq^{\mathrm{t}}_{\mathrm{T}}$ *t-can*(*t-beh*(A))*, and*
4. *if A has t-fin then* *t-can*(*t-beh*(A)) $\equiv^{\mathrm{t}}_{\mathrm{T}}$ A*.*

Proof: By combining Lemmas 4.3 and 4.7. ■

5 Simulations for Timed Automata

Our aim is to develop proof techniques for showing inclusion between the sets of timed traces of timed automata. One way of doing this is to show how this problem can be reduced to the problem of proving inclusion between the sets of traces of certain derived automata. This reduction solves our problem, in a sense, since it allows us to use the various simulation techniques of Part I to prove inclusion results for timed automata. The approach is analogous to that followed for Milner's CCS [17] where the problem of establishing a weak bisimulation is reduced to the problem of finding a strong bisimulation. Another example of this approach appears in [12], where the problem of showing including of timed behaviors of certain kinds of timed automata is reduced to that of proving inclusion between sets of admissible behaviors of certain derived I/O automata. A key role in our reduction is played by the construction of the closure of a timed automaton.

An alternative approach that is sometimes preferable is to define new simulations directly in terms of timed automata. In this section, we also define such simulations, and give soundness and completeness results for them and relationships between them. Proofs of these results again use the closure construction, and are based on corresponding results for the untimed case.

5.1 The Closure Construction

The *closure* of a timed automaton A, denoted by $cl(A)$, is the automaton B given by

- $states(B) = states(A)$,
- $start(B) = start(A)$,
- $acts(B) = (ext(A) \times \mathsf{R}^{\geq 0}) \cup \{\tau\}$,

- $steps(B)$ is the smallest relation such that

$$s' \xrightarrow{a}_A s \ \wedge\ a \in vis(A) \ \wedge\ s.now_A = t \quad \Rightarrow \quad s' \xrightarrow{(a,t)}_B s,$$

$$s' \xrightarrow{\tau}_A s \quad \Rightarrow \quad s' \xrightarrow{\tau}_B s,$$

$$s' \overset{(\lambda,t)}{\rightsquigarrow}_A s \quad \Rightarrow \quad s' \xrightarrow{(\nu,t)}_B s.$$

Thus, the closure construction augments A by attaching a time, more specifically the $.now$ component of the resulting state, to each visible and time-passage action, and by adding new time-passage actions to short-circuit the effects of any number of τ and ν actions. Note that, unlike the ν steps in A, the (ν,t) steps in the closure do not always strictly increase the time. In particular there is an *idling step* $s \xrightarrow{(\nu,t)}_B s$ (where $t = s.now$) from each state s to itself.

We now wish to show the relationship between executions of A and of $cl(A)$. In order to do this, we first need to define a simple well-formedness condition for timed sequences. Let $t', t \in \mathsf{R}^{\geq 0}$ with $t' \leq t$. We say a finite timed sequence δ over $ext(A)$ *passes properly from* t' *to* t if (1) the time components of the pairs in δ are in the interval $[t', t]$, (2) each pair (a,u) in δ with $a \in vis(A)$ and $u > t'$ is preceded (not necessarily immediately) by a pair (ν, u), and (3) if $t' < t$ then δ includes a pair with time component t.

Lemma 5.1 *If β is a finite sequence over $ext(cl(A))$ and $s' \overset{\beta}{\Longrightarrow}_{cl(A)} s$ then β is a finite timed sequence over $ext(A)$ and passes properly from $s'.now$ to $s.now$.*

The next two lemmas show the close relationship between executions of A and of $cl(A)$.

Lemma 5.2 *Suppose s' and s are states of A, and δ is a finite timed sequence over $ext(A)$ that passes properly from $s'.now$ to $s.now$. Then*

$$s' \overset{\delta}{\Longrightarrow}_{cl(A)} s \qquad \Leftrightarrow \qquad s' \overset{p}{\rightsquigarrow}_A s, \ \textit{where } p = (\delta \lceil (vis(A) \times \mathsf{R}^{\geq 0}), s.now)$$

Proof: Straightforward. ■

Lemma 5.3 *Suppose $s' \overset{p}{\rightsquigarrow}_A s$, where $p = (\delta, s.now)$. Let δ' be the sequence obtained from δ by adding a pair (ν, t) just before each pair (a, t) in δ, and also appending a pair $(\nu, s.now)$ at the end of the sequence. Then $s' \overset{\delta'}{\Longrightarrow}_{cl(A)} s$.*

Proof: It is easy to check that δ' is a timed sequence over $ext(A)$ that passes properly from $s'.now$ to $s.now$, and that $p = (\delta' \lceil (vis(A) \times \mathsf{R}^{\geq 0}), s.now)$. From this the lemma follows via an application of Lemma 5.2. ■

We are now prepared to prove some important properties of the closure construction.

Lemma 5.4

1. *A is t-deterministic if and only if $cl(A)$ is deterministic.*

2. *A has t-fin if and only if $cl(A)$ has fin.*

Proof: We prove case (1). Case (2) is similar and left to the reader.

Suppose A is t-deterministic. Then $|start(A)| = 1$ and hence $|start(cl(A))| = 1$. Suppose s' is a state of $cl(A)$ and β is a finite sequence over $ext(cl(A))$. We have to show that there is at most one state s such that $s' \stackrel{\beta}{\Longrightarrow}_{cl(A)} s$. So suppose that $s' \stackrel{\beta}{\Longrightarrow}_{cl(A)} s_1$ and $s' \stackrel{\beta}{\Longrightarrow}_{cl(A)} s_2$. Using Lemma 5.1, it follows that β passes properly from $s'.now$ to $s_1.now$ and to $s_2.now$, which implies that $s_1.now = s_2.now$. By Lemma 5.2, we infer that, for $p = (\beta\lceil(vis(A) \times \mathsf{R}^{\geq 0}), s_1.now)$, $s' \stackrel{p}{\leadsto}_A s_1$ and $s' \stackrel{p}{\leadsto}_A s_2$. Since A is t-deterministic, we have that $s_1 = s_2$, as required.

For the other direction, suppose that $cl(A)$ is deterministic. Then $|start(cl(A))| = 1$ and hence $|start(A)| = 1$. Suppose s' is a state of A and p is a finite timed sequence pair over $vis(A)$. We have to show that there is at most one state s such that $s' \stackrel{p}{\leadsto}_A s$. So suppose that $s' \stackrel{p}{\leadsto}_A s_1$ and $s' \stackrel{p}{\leadsto}_A s_2$. By Lemma 5.3, there is a sequence β such that $s' \stackrel{\beta}{\Longrightarrow}_{cl(A)} s_1$ and $s' \stackrel{\beta}{\Longrightarrow}_{cl(A)} s_2$. Since $cl(A)$ is deterministic, we have that $s_1 = s_2$, as required. ■

Lemma 5.5 *$A \leq^{\mathrm{t}}_{*\mathrm{T}} B \Leftrightarrow cl(A) \leq_{*\mathrm{T}} cl(B)$.*

Proof: "$\Rightarrow$" Suppose $\beta \in traces^*(cl(A))$. Then there are $s' \in start(A)$ and $s \in states(A)$ such that $s' \stackrel{\beta}{\Longrightarrow}_{cl(A)} s$. By Lemma 5.1, β is a finite timed sequence over $ext(A)$ and passes properly from $s'.now$ to $s.now$. Thus we can apply Lemma 5.2 to obtain $s' \stackrel{p}{\leadsto}_A s$, where $p = (\beta\lceil(vis(A) \times \mathsf{R}^{\geq 0}), s.now)$. This means that $p \in t\text{-}traces^*(A)$, and therefore also $p \in t\text{-}traces^*(B)$. So there are $r' \in start(B)$ and $r \in states(B)$ such that $r' \stackrel{p}{\leadsto}_B r$. Applying Lemma 5.2 in the other direction gives $r' \stackrel{\beta}{\Longrightarrow}_{cl(B)} r$. Thus $\beta \in traces^*(cl(B))$, as required.

"$\Leftarrow$" Suppose $p = (\delta, t) \in t\text{-}traces^*(A)$. Then there are $s' \in start(A)$ and $s \in states(A)$ such that $s' \stackrel{p}{\leadsto}_A s$. Let δ' be the sequence obtained from δ by adding a pair (ν, t') just before each pair (a, t') in δ, and also appending a pair (ν, t) at the end of the sequence. By Lemma 5.3, $s' \stackrel{\delta'}{\Longrightarrow}_{cl(A)} s$. This means that $\delta' \in traces^*(cl(A))$, and therefore also $\delta' \in traces^*(cl(B))$. So there are $r' \in start(B)$ and $r \in states(B)$ such that $r' \stackrel{\delta'}{\Longrightarrow}_B r$. By Lemma 5.1, δ' is a finite timed sequence over $ext(A)$ and passes properly from $r'.now$ to $r.now$. Thus we can apply Lemma 5.2 to obtain $r' \stackrel{p'}{\leadsto}_A r$, where $p' = (\delta'\lceil(vis(A) \times \mathsf{R}^{\geq 0}), r.now)$. It easily checked that $p' = p$. Thus $p \in t\text{-}traces^*(B)$, as required. ■

Lemma 5.6 *$cl(A) \leq_{\mathrm{T}} cl(B) \Rightarrow A \leq^{\mathrm{t}}_{\mathrm{T}} B$.*

Proof: Suppose $cl(A) \leq_{\mathrm{T}} cl(B)$. Then certainly $cl(A) \leq_{*\mathrm{T}} cl(B)$, so by Lemma 5.5, $A \leq^{\mathrm{t}}_{*\mathrm{T}} B$. Thus it remains to be shown that $A \leq^{\mathrm{t}}_{\infty\mathrm{T}} B$. For this, suppose that $p = (\delta, \infty) \in t\text{-}traces^{\infty}(A)$. We will prove $p \in t\text{-}traces^{\infty}(B)$. Let δ' be the sequence obtained from δ by adding a pair (ν, t) just before each pair (a, t) in δ, and, if δ is finite, appending to the result of that the sequence $(\nu, u)(\nu, u+1)(\nu, u+2)\cdots$, where $u \in \mathsf{R}^{\geq 0}$ is some arbitrary number larger than all time components in δ. It is routine to check that $\delta' \in traces^{\omega}(cl(A))$. Thus $\delta' \in traces^{\omega}(cl(B))$. It is now easy to find a corresponding admissible timed execution Σ of B with $t\text{-}trace(\Sigma) = p$. Thus $p \in t\text{-}traces^{\infty}(B)$. ■

Example 5.7 The reverse implication of Lemma 5.6 does not hold in general. For a counterexample, take B to be a timed automaton that nondeterministically chooses a positive natural number n, then does action a at times $1-2^{-1}$, $1-2^{-2}$,..., $1-2^{-n}$, and then idles forever. B is a feasible timed automaton with infinite invisible nondeterminism. Let A be the same as B, except that it may also choose ω at the beginning, in which case it subsequently does action a at times $1-2^{-1}$, $1-2^{-2}$,..., $1-2^{-n}$,... Timed automaton A is not feasible because by choosing ω it reaches a state from where only a Zeno execution is possible and no admissible execution. Timed automata A and B have the same timed traces, but $cl(A)$ has an infinite trace $(a, 1-2^{-1})$, $(a, 1-2^{-2})$,..., $(a, 1-2^{-n})$,... which $cl(B)$ does not have.

It turns out that we *do* have the reverse of Lemma 5.6 in case B has t-fin.

Lemma 5.8 *Suppose B has t-fin. Then $cl(A) \leq_{\mathrm{T}} cl(B) \Leftrightarrow A \leq^{\mathrm{t}}_{\mathrm{T}} B$.*

Proof:

$$\begin{array}{rcl} cl(A) \leq_{\mathrm{T}} cl(B) & \Leftrightarrow & \text{(by Lemma 5.4, and Prop. 3.6 of Part I)} \\ cl(A) \leq_{*\mathrm{T}} cl(B) & \Leftrightarrow & \text{(by Lemma 5.5)} \\ A \leq^{\mathrm{t}}_{*\mathrm{T}} B & \Leftrightarrow & \text{(by Prop. 4.4)} \\ A \leq^{\mathrm{t}}_{\mathrm{T}} B & & \end{array}$$

■

Corollary 5.9 *The following statements are equivalent.*

1. $A \leq^{\mathrm{t}}_{*\mathrm{T}} B$.
2. $cl(A) \leq_{\mathrm{FB}} cl(B)$.
3. $cl(A) \leq_{\mathrm{BF}} cl(B)$.

If B has t-fin then also the following statements are equivalent, with each other and the three statements above.

1. $A \leq^{\mathrm{t}}_{\mathrm{T}} B$.
2. $cl(A) \leq_{\mathrm{iFB}} cl(B)$.

Proof:

$$\begin{array}{rcl} A \leq^{\mathrm{t}}_{*\mathrm{T}} B & \Leftrightarrow & \text{(by Lemma 5.5)} \\ cl(A) \leq_{*\mathrm{T}} cl(B) & \Leftrightarrow & \text{(by Theorems 5.5 and 5.6 of Part I)} \\ cl(A) \leq_{\mathrm{FB}} cl(B) & \Leftrightarrow & \text{(by Prop. 5.9 of Part I)} \\ cl(A) \leq_{\mathrm{BF}} cl(B) & & \end{array}$$

If B has t-fin then

$$\begin{array}{rcl} A \leq^{\mathrm{t}}_{*\mathrm{T}} B & \Rightarrow & \text{(by Lemma 5.5)} \\ cl(A) \leq_{*\mathrm{T}} cl(B) & \Rightarrow & \text{(by Lemma 5.4, and Theorem 5.6 of Part I)} \\ cl(A) \leq_{\mathrm{iFB}} cl(B) & \Rightarrow & \text{(by Theorem 5.5 of Part I)} \\ cl(A) \leq_{\mathrm{T}} cl(B) & \Rightarrow & \text{(by Lemma 5.6)} \\ A \leq^{\mathrm{t}}_{\mathrm{T}} B & \Rightarrow & A \leq^{\mathrm{t}}_{*\mathrm{T}} B \end{array}$$

$\blacksquare$

In a sense, we have solved our problem now: we have found a way to prove inclusion of the sets of timed traces of timed automata A and B, under the reasonable assumption that B has t-fin. All we have to do is to establish a forward-backward simulation between two closely related timed automata, $cl(A)$ and $cl(B)$. The automata $cl(A)$ and $cl(B)$ are very similar to A and B: they are the same except for their step relations, which are just a kind of transitive closure of the step relations of A and B.

5.2 Direct Simulations Between Timed Automata

We have already provided sufficient machinery to permit simulation methods to be used for reasoning about timed automata. However, we would like to go further: we would also like to define the various simulations directly on the timed automata themselves. We do this in the present section. Later, we give a simple lemma relating the new simulations to the simulations between the closures of the automata.

Suppose A and B are timed automata. A *timed refinement* from A to B is a function $r : states(A) \rightarrow states(B)$ that satisfies:

1. $r(s).now = s.now$.
2. If $s \in start(A)$ then $r(s) \in start(B)$.
3. If $s' \xrightarrow{a}_A s$ then $r(s') \stackrel{p}{\leadsto}_B r(s)$, where $p = t\text{-}trace((s', a, s))$.

A *timed forward simulation* from A to B is a relation f over $states(A)$ and $states(B)$ that satisfies:

1. If $u \in f[s]$ then $u.now = s.now$.
2. If $s \in start(A)$ then $f[s] \cap start(B) \neq \emptyset$.
3. If $s' \xrightarrow{a}_A s$ and $u' \in f[s']$, then there exists $u \in f[s]$ such that $u' \stackrel{p}{\leadsto}_B u$, where $p = t\text{-}trace((s', a, s))$.

A *timed backward simulation* from A to B is a total relation b over $states(A)$ and $states(B)$ that satisfies:

1. If $u \in b[s]$ then $u.now = s.now$.
2. If $s \in start(A)$ then $b[s] \subseteq start(B)$.
3. If $s' \xrightarrow{a}_A s$ and $u \in b[s]$, then there exists $u' \in b[s']$ such that $u' \stackrel{p}{\leadsto}_B u$, where $p = t\text{-}trace((s', a, s))$.

A *timed forward-backward simulation* from A to B is a relation g over $states(A)$ and $\mathbf{N}(states(B))$ that satisfies:

1. If u is an element of any set in $g[s]$ then $u.now = s.now$.

2. If $s \in start(A)$ then there exists $S \in g[s]$ such that $S \subseteq start(B)$.

3. If $s' \xrightarrow{a}_A s$ and $S' \in g[s']$, then there exists $S \in g[s]$ such that for every $u \in S$ there exists $u' \in S'$ such that $u' \overset{p}{\leadsto}_B u$, where $p = t\text{-}trace((s', a, s))$.

A *timed backward-forward simulation* from A to B is a total relation g over $states(A)$ and $\mathbf{P}(states(B))$ that satisfies:

1. If u is an element of any set in $g[s]$ then $u.now = s.now$.

2. If $s \in start(A)$ then for all $S \in g[s]$, $S \cap start(B) \neq \emptyset$.

3. If $s' \xrightarrow{a}_A s$ and $S \in g[s]$, then there exists $S' \in g[s']$ such that for every $u' \in S'$ there exists $u \in S$ such that $u' \overset{p}{\leadsto}_B u$, where $p = t\text{-}trace((s', a, s))$.

A relation h over $states(A)$ and $states(B)$ is a *timed history relation* from A to B if it is a timed forward simulation from A to B and h^{-1} is a timed refinement from B to A.

A relation p over $states(A)$ and $states(B)$ is a *timed prophecy relation* from A to B if it is a timed backward simulation from A to B and p^{-1} is a timed refinement from B to A.

Analogous to Part I, we write $A \leq^{t}_{R} B$, $A \leq^{t}_{F} B$, etc. in case there exists a timed refinement, timed forward simulation, etc., from A to B.

Without working out the details, we note here that, analogous to the untimed case, there exists a full correspondence between timed history/prophecy relations and the obvious notion of timed history/prophecy variables.

5.3 Synchronicity

A new feature in the definitions of the various timed simulations is the requirement that related states have the same $.now$ component. In this subsection we explore the consequences of this natural restriction.

Suppose A and B are timed automata. A relation f over $states(A)$ and $states(B)$ is *synchronous* if for all $(s, u) \in f$, $u.now = s.now$. For each relation f over $states(A)$ and $states(B)$, we define the subrelation $syn(f)$ to be

$$\{(s, u) \in f \mid u.now = s.now\}.$$

Thus, f is synchronous if and only if $syn(f) = f$.

Similarly, a relation g over $states(A)$ and $\mathbf{P}(states(B))$ is *synchronous* if for all $(s, S) \in g$ and all $u \in S$, $u.now = s.now$. For each relation g over $states(A)$ and $\mathbf{P}(states(B))$, we define the subrelation $syn1(g)$ to be

$$\{(s, S) \in g \mid \forall u \in S : u.now = s.now\}.$$

Thus, g is synchronous if and only if $syn1(g) = g$.

Also, for each relation g over $states(A)$ and $\mathbf{P}(states(B))$, we define the relation $syn2(g)$ to be

$$\{(s, S) | \exists T : (s, T) \in g \text{ and } S = \{u \in T \mid u.now = s.now\}\}.$$

So also g is synchronous if and only if $syn2(g) = g$.

Obviously, all the timed versions of refinements, forward simulations, etc., that we defined above are synchronous. The following observation is more interesting.

Lemma 5.10

1. *Any refinement from $cl(A)$ to $cl(B)$ is synchronous.*
2. *If f is a forward simulation from $cl(A)$ to $cl(B)$, then $syn(f)$ is a synchronous forward simulation from $cl(A)$ to $cl(B)$.*
3. *Any backward simulation from $cl(A)$ to $cl(B)$ is synchronous.*
4. *If g is a forward-backward simulation from $cl(A)$ to $cl(B)$, then $syn1(g)$ is a synchronous forward-backward simulation from $cl(A)$ to $cl(B)$.*
5. *If g is a backward-forward simulation from $cl(A)$ to $cl(B)$, then $syn2(g)$ is a synchronous backward-forward simulation from $cl(A)$ to $cl(B)$.*
6. *Any history relation from $cl(A)$ to $cl(B)$ is synchronous.*
7. *Any prophecy relation from $cl(A)$ to $cl(B)$ is synchronous.*

Proof: For 1, suppose that r is a refinement from $cl(A)$ to $cl(B)$ and s is a state of A with $s.now_A = t$. By definition of the closure construction, $s \xrightarrow{(\nu,t)}_{cl(A)} s$. Thus, as r is a refinement, $r(s) \overset{(\nu,t)}{\Longrightarrow}_{cl(B)} r(s)$. From this it easily follows that $r(s).now = t = s.now$.

For 2, suppose f is a forward simulation from $cl(A)$ to $cl(B)$. By the definition of a forward simulation, if $s \in start(A)$, then there is a state $u \in f[s]$ that is in $start(B)$. Axiom **S1** implies that $s.now = u.now = 0$, and thus $u \in syn(f)[s]$.

Suppose $s' \xrightarrow{\tau}_{cl(A)} s$ and $u' \in syn(f)[s']$. Then $u'.now = s'.now = s.now$. Also, $u' \in f[s']$ and therefore there exists a state $u \in f[s]$ such that $u' \overset{\lambda}{\Longrightarrow}_{cl(B)} u$. It follows that $u.now = u'.now = s.now$. Hence $u \in syn(f)[s]$.

Now suppose $s' \xrightarrow{(a,t)}_{cl(A)} s$, for some $a \in ext(A)$, and $u' \in syn(f)[s']$. Then $s.now = t$. Also, $u' \in f[s']$ and therefore there exists a state $u \in f[s]$ such that $u' \overset{(a,t)}{\Longrightarrow}_{cl(B)} u$. As $u.now = t$, it follows that $u.now = s.now$. Hence $u \in syn(f)[s]$.

For 3, suppose that b is a backward simulation from $cl(A)$ to $cl(B)$, and suppose s is a state of A with $s.now = t$. Let $u \in b[s]$. Now we use that $cl(A)$ has an 'idling' transition $s \xrightarrow{(\nu,t)}_{cl(A)} s$. Thus, since b is a backward simulation there exists $u' \in b[s]$ with $u' \overset{(\nu,t)}{\Longrightarrow}_{cl(B)} u$. This implies $u.now = t = s.now$.

Next we prove 4. Suppose that g is a forward-backward simulation from $cl(A)$ to $cl(B)$ By definition of a forward-backward simulation, if $s \in start(A)$, then there is a set $S \in g[s]$ such that $S \subseteq start(B)$. By axiom **S1**, $u.now = 0$ for every $u \in S$. Therefore, $S \in syn1(g)[s]$, as needed.

Now suppose $s' \xrightarrow{(a,t)}_{cl(A)} s$ and $S' \in syn1(g)[s']$. Then for all $u' \in S'$, we have $u'.now = s'.now$. Also, $S' \in g[s']$ and therefore there exists a set $S \in g[s]$ such that for every $u \in S$ there exists $u' \in S'$ with $u' \overset{(a,t)}{\Longrightarrow}_{cl(B)} u$. We must show that $S \in syn1(g)[s]$, i.e., that $u.now =$

$s.now$ for every $u \in S$. So fix $u \in S$. There exists $u' \in S'$ such that $u' \stackrel{(a,t)}{\Longrightarrow}_{cl(B)} u$. Then it follows using axioms **S2** and **S3** that $u.now = s.now$. Therefore, $S \in syn1(g)[s]$.

The case where $s' \stackrel{\tau}{\longrightarrow}_{cl(A)} s$ and $S' \in syn1(g)[s']$ is similar.

It remains to show the nonemptiness property, i.e., that for every state $s \in states(A)$, every set $S \in syn1(g)[s]$ is nonempty. But this follows from the fact that $S \in g[s]$ and g is a forward-backward simulation.

For 5, suppose g is a backward-forward simulation from $cl(A)$ to $cl(B)$. Let $s \in start(A)$ and let $S \in syn2(g)[s]$. Then there exists a set $T \in g[s]$ such that $S = \{u \in T \mid u.now = s.now\}$. Since g is a backward-forward simulation, T contains an element $u \in start(B)$. By axiom **S1**, $u.now = s.now = 0$, so $u \in S$. This suffices for the start condition.

Now suppose $s' \stackrel{(a,t)}{\longrightarrow}_{cl(A)} s$ and $S \in syn2(g)[s]$. Then there is a set $T \in g[s]$ with $S = \{u \in T \mid u.now = s.now\}$. Since $T \in g[s]$, there exists a set $T' \in g[s']$ such that for every $u' \in T'$ there exists $u \in T$ with $u' \stackrel{(a,t)}{\Longrightarrow}_{cl(B)} u$. Let $S' = \{u' \in T' \mid u'.now = s'.now\}$. Then $S' \in syn2(g)[s']$, by definition. Now consider any $u' \in S'$. Since $u' \in T'$, there exists $u \in T$ with $u' \stackrel{(a,t)}{\Longrightarrow}_{cl(B)} u$. Then it follows by axioms **S2** and **S3** that $u.now = s.now$, which implies that $u \in S$.

The case where $s' \stackrel{\tau}{\longrightarrow}_{cl(A)} s$ and $S \in syn2(g)[s]$ is similar. This suffices for the second condition.

It remains to show that $syn2(g)$ is total, i.e., that for every state $s \in states(A)$, there is some set $S \in syn2(g)[s]$. But this follows from the fact that any backward-forward simulation, and hence g in particular, is total, and every set in $g[s]$ has a subset in $syn(g)[s]$.

Parts 6 and 7 are easy: the inverse of any history or prophecy relation is a refinement, and by Part 1 refinements are synchronous. ■

5.4 Relating Timed and Untimed Simulations

In Section 5.1 we showed that (under certain finiteness conditions) there is a one-to-one correspondence between inclusion of timed traces for timed automata, and inclusion of traces between the closures of these automata. In this subsection we observe that there is also a strong connection between timed simulations between timed automata, and the same functions viewed as untimed simulations between the closures of these automata. As an immediate consequence of this observation we obtain easy soundness proofs for all the timed simulations of Section 5.2, since soundness of the timed simulations reduces to the soundness of the corresponding untimed simulations. Moreover, we obtain a completeness result for timed forward-backward simulations.

Lemma 5.11 *A synchronous relation is a timed refinement from A to B if and only if it is a refinement from $cl(A)$ to $cl(B)$. Moreover, the above property also holds if both occurrences of the word "refinement" are replaced by "forward simulation", "backward simulation", "forward-backward simulation", "backward-forward simulation", "history relation" or "prophecy relation".*

Proof: Here we prove the case of refinements. The other mappings can be handled similarly.

First, suppose r is a timed refinement from A to B. We show that r is a refinement from $cl(A)$ to $cl(B)$, and the only thing nontrivial here is to demonstrate that r satisfies the second clause of the definition of a refinement. We distinguish between three possible cases.

1. $s' \xrightarrow{(a,t)}_{cl(A)} s$, for some $a \in vis(A)$. By definition of the closure construction, $s' \xrightarrow{a}_A s$. Thus, since r is a timed refinement from A to B, $r(s') \overset{((a,t),t)}{\rightsquigarrow}_B r(s)$. This means that B has an execution fragment from $r(s')$ to $r(s)$ consisting of τ steps and one single a step. There can be no ν steps in the execution since $r(s').now = s'.now = s.now = r(s).now = t$. But this implies that also $cl(B)$ has an execution fragment from $r(s')$ to $r(s)$ consisting of τ steps and one (a,t) step, that is $r(s') \overset{(a,t)}{\Longrightarrow}_{cl(B)} r(s)$.

2. $s' \xrightarrow{\tau}_{cl(A)} s$. By definition of the closure construction, $s' \xrightarrow{\tau}_A s$. Since r is a timed refinement, $r(s') \overset{(\lambda,t)}{\rightsquigarrow}_B r(s)$, where $t = s.now$. Since $r(s').now = s'.now = s.now = t = r(s).now$, this implies that B, and hence $cl(B)$, has an execution fragment from $r(s')$ to $r(s)$ consisting of τ steps only. Thus, $r(s') \overset{\lambda}{\Longrightarrow}_{cl(B)} r(s)$.

3. $s' \xrightarrow{(\nu,t)}_{cl(A)} s$. By definition of the closure construction, $s' \overset{(\lambda,t)}{\rightsquigarrow}_A s$. Thus, A has an execution fragment α from s' to s consisting of τ and ν steps only. Using that r is a timed refinement from A to B, we can construct a corresponding execution fragment in B from $r(s')$ to $r(s)$: for each τ step in α we construct an execution fragment in B consisting of τ steps only, and for each ν step in α we can find an execution fragment consisting of τ and ν steps only. If we glue all these execution fragments together we obtain an execution fragment in B from $r(s')$ to $r(s)$ consisting of τ and ν steps only. Thus, $r(s') \overset{(\lambda,t)}{\rightsquigarrow}_B r(s)$. By definition of closure, this implies $r(s') \xrightarrow{(\nu,t)}_{cl(B)} r(s)$, which in turn implies $r(s') \overset{(\nu,t)}{\Longrightarrow}_{cl(B)} r(s)$.

For the other direction, suppose r is a refinement from $cl(A)$ to $cl(B)$. By Lemma 5.10, we know that r is synchronous. We have to establish that r is a timed refinement from A to B, and for this again the only nontrivial part is the second clause in the definition of a timed refinement. So suppose $s' \xrightarrow{a}_A s$. Let $t = s.now$. Again we distinguish between three cases.

1. $a \in vis(A)$. Then $s' \xrightarrow{(a,t)}_{cl(A)} s$. Using that r is a refinement we get $r(s') \overset{(a,t)}{\Longrightarrow}_{cl(B)} r(s)$. This means that $cl(B)$ has an execution fragment from $r(s')$ to $r(s)$ consisting of τ steps and one single (a,t) step. Thus B has an execution fragment from $r(s')$ to $r(s)$ consisting of τ steps and one single a step. This implies $r(s') \overset{((a,t),t)}{\rightsquigarrow}_B r(s)$, as required.

2. $a = \tau$. Then $s' \xrightarrow{\tau}_{cl(A)} s$. Since r is a refinement, we get $r(s') \overset{\lambda}{\Longrightarrow}_{cl(B)} r(s)$. From this $r(s') \overset{(\lambda,t)}{\rightsquigarrow}_B r(s)$ follows via a simple argument, as in (1).

3. $a = \nu$. Then $s' \xrightarrow{(\nu,t)}_{cl(A)} s$. Since r is a refinement, we get $r(s') \overset{(\nu,t)}{\Longrightarrow}_{cl(B)} r(s)$. From this $r(s') \overset{(\lambda,t)}{\rightsquigarrow}_B r(s)$ follows via a simple argument, as in (1).

∎

Corollary 5.12 *Suppose $X \in \{R, F, (i)B, (i)FB, (i)BF, H, (i)P\}$. Then $A \leq^{\mathrm{t}}_{\mathrm{X}} B \Leftrightarrow cl(A) \leq_{\mathrm{X}} cl(B)$.*

Proof: We give the proof for the case $X = F$. The other cases are similar. By definition, $A \leq_{\mathrm{F}}^{\mathrm{t}} B$ iff there exists a timed forward simulation from A to B. By Lemma 5.11, this is the case iff there exists a synchronous simulation from $cl(A)$ to $cl(B)$. But by Lemma 5.10, this in turn is the case iff there exists a simulation from $cl(A)$ to $cl(B)$, that is, $cl(A) \leq_{\mathrm{F}} cl(B)$. ■

Proposition 5.13 *The relations $\leq_{\mathrm{R}}^{\mathrm{t}}$, $\leq_{\mathrm{F}}^{\mathrm{t}}$, $\leq_{\mathrm{B}}^{\mathrm{t}}$, $\leq_{\mathrm{iB}}^{\mathrm{t}}$, $\leq_{\mathrm{FB}}^{\mathrm{t}}$, $\leq_{\mathrm{iFB}}^{\mathrm{t}}$, $\leq_{\mathrm{BF}}^{\mathrm{t}}$, $\leq_{\mathrm{H}}^{\mathrm{t}}$, $\leq_{\mathrm{P}}^{\mathrm{t}}$ and $\leq_{\mathrm{iP}}^{\mathrm{t}}$ are all preorders. (However, $\leq_{\mathrm{iBF}}^{\mathrm{t}}$ is not a preorder.)*

Proof: By Corollary 5.12, since the corresponding untimed simulations are preorders. Essentially the same counterexample that we used to show that $\leq_{\mathrm{iBF}}$ is not a preorder (the automata I and J of Example 5.10), can be used again in the proof that $\leq_{\mathrm{iBF}}^{\mathrm{t}}$ is not a preorder. In order to turn the untimed automata into timed automata one only has to attach $.now$-value 0 to each state. ■

5.5 Reachability

Analogously to the way in which we defined, in Section 7 of Part I, weak versions of the various untimed simulations, we will now define weak versions of all the timed simulations.

For any timed automaton A, let the *reachable timed subautomaton* $R(A)$, be the timed automaton defined as follows.

- $states(R(A)) = rstates(A)$,
- $s.now_{R(A)} = s.now_A$,
- $start(R(A)) = start(A)$,
- $acts(R(A)) = acts(A)$, and
- $steps(R(A)) = steps(A) \cap (rstates(A) \times acts(A) \times rstates(A))$.

For $X \in \{$R, F, (i)B, (i)FB, (i)BF, H, (i)P$\}$, define $A \leq_{\mathrm{wX}}^{\mathrm{t}} B$ iff $R(A) \leq_{\mathrm{X}}^{\mathrm{t}} R(B)$.

Lemma 5.14 $R(cl(A)) = cl(R(A))$.

Proof: Straightforward from the definitions. ■

Proposition 5.15 *Suppose $X \in \{R, F, (i)B, (i)FB, (i)BF, H, (i)P\}$. Then $A \leq_{\mathrm{wX}}^{\mathrm{t}} B$ iff $cl(A) \leq_{\mathrm{wX}} cl(B)$.*

Proof:

$$\begin{array}{rcl}
A \leq_{\mathrm{wX}}^{\mathrm{t}} B & \Leftrightarrow & \text{(By definition)} \\
R(A) \leq_{\mathrm{X}}^{\mathrm{t}} R(B) & \Leftrightarrow & \text{(By Corollary 5.12)} \\
cl(R(A)) \leq_{\mathrm{X}} cl(R(B)) & \Leftrightarrow & \text{(By Lemma 5.14)} \\
R(cl(A)) \leq_{\mathrm{X}} R(cl(B)) & \Leftrightarrow & \text{(By definition)} \\
cl(A) \leq_{\mathrm{wX}} cl(B). & &
\end{array}$$

■

We leave it to the reader to give direct definitions of weak timed simulations that do not involve reachable timed subautomata. This can be done completely analogously to the untimed case.

5.6 Classification of Basic Relations Between Timed Automata

The classification of Section 8 of Part I carries over to the timed setting: Figure 1, except for the superscripts t, is the same as Figure 6 of Part I, which gives an overview of the relationships in the untimed case.

Theorem 5.16 *Suppose $X, Y \in \{T, *T, (w)R, (w)F, (w)(i)B, (w)(i)FB, (w)(i)BF, (w)H, (w)(i)P\}$. Then $A \leq^{\mathrm{t}}_{\mathrm{X}} B \Rightarrow A \leq^{\mathrm{t}}_{\mathrm{Y}} B$ for all timed automata A and B if and only if there is a path from $\leq^{\mathrm{t}}_{\mathrm{X}}$ to $\leq^{\mathrm{t}}_{\mathrm{Y}}$ in Figure 1 consisting of thin lines. If B has t-fin, then $A \leq^{\mathrm{t}}_{\mathrm{X}} B \Rightarrow A \leq^{\mathrm{t}}_{\mathrm{Y}} B$ for all automata A and B if and only if there is a path from $\leq^{\mathrm{t}}_{\mathrm{X}}$ to $\leq^{\mathrm{t}}_{\mathrm{Y}}$ consisting of thin lines and thick lines.*

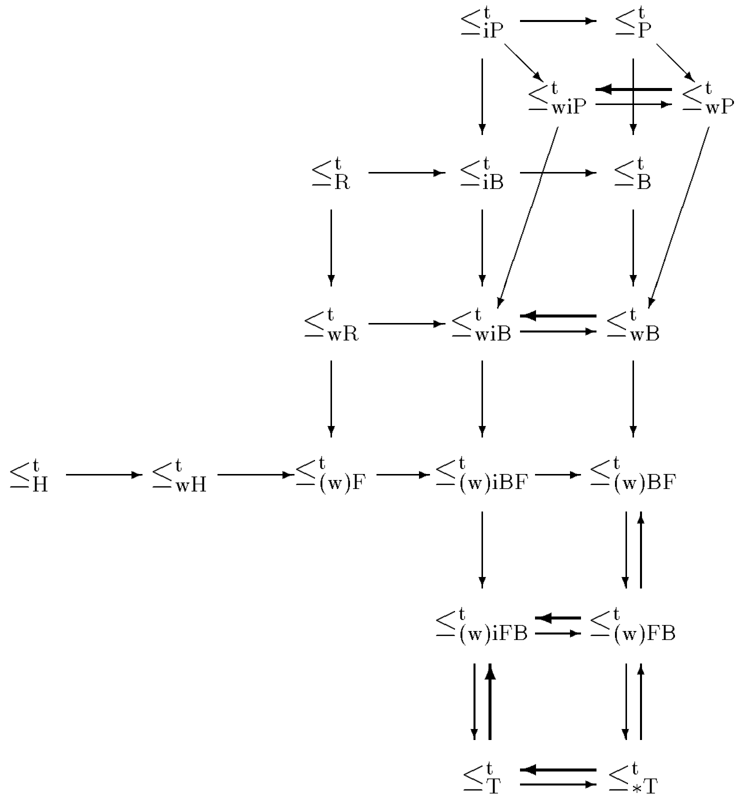

Figure 1: Classification of basic relations between timed automata.

Proof: Note that except for the superscripts t, Figure 1 is the same as Figure 6 in Part I, which gives an overview of the relationships in the untimed case. Using Corollary 5.12, Corollary 5.9, and Prop. 5.15 the inclusions for the timed case follow from the corresponding inclusions for the untimed case.

In order to show that all the inclusions are strict, one can use essentially the same counterexamples as in the untimed setting. In order to turn the untimed automata into timed automata one only has to attach *.now*-value 0 to each state. Only for establishing the difference between $\leq^{\mathrm{t}}_{*\mathrm{T}}$ and $\leq^{\mathrm{t}}_{\mathrm{T}}$ the examples of Part I, Section 3 are not adequate, and one has to use Example 4.5 instead. (If A^0 and B^0 denote the timed automata obtained by associating time 0 to all states of the automata A and B of Example 3.1 in Part I, respectively, then $A^0 \equiv^{\mathrm{t}}_{*\mathrm{T}} B^0$ but, since both timed automata have no admissible traces, also $A^0 \equiv^{\mathrm{t}}_{\mathrm{T}} B^0$.) ∎

Here are two more results that carry over because of the correspondence between the timed and the untimed case.

Theorem 5.17 *(Partial completeness of timed forward simulations)*
Suppose B is t-deterministic and $A \leq^{\mathrm{t}}_{*\mathrm{T}} B$. Then $A \leq^{\mathrm{t}}_{\mathrm{F}} B$.

Proof: By Lemma 5.4(1), $cl(B)$ is deterministic, and by Lemma 5.5, $cl(A) \leq_{*\mathrm{T}} cl(B)$. Thus by the partial completeness result for forward simulations (Theorem 4.11, Part I), $cl(A) \leq_{\mathrm{F}} cl(B)$. Now Corollary 5.12 allows us to conclude $A \leq^{\mathrm{t}}_{\mathrm{F}} B$, as required. ∎

Proposition 5.18 *Suppose all states of A are reachable, B is t-deterministic and $A \leq^{\mathrm{t}}_{\mathrm{B}} B$. Then $A \leq_{\mathrm{R}} B$.*

Proof: All states of $cl(A)$ are reachable, $cl(B)$ is deterministic and $cl(A) \leq_{\mathrm{B}} cl(B)$. By Prop. 4.19, the untimed version of the fact we are proving, $cl(A) \leq_{\mathrm{R}} cl(B)$. Hence $A \leq^{\mathrm{t}}_{\mathrm{R}} B$. ∎

6 Additional Results for Timed Automata

The previous sections show how some simple correspondences cause most of the results for untimed automata to carry over to the timed setting. There are some results about untimed automata that do not carry over because of these correspondences, but are nonetheless true. First, there are the partial completeness results that involve t-forests. These do not carry over since the closure construction does not map t-forests to forests. Also, the various results that require the construction of a timed automaton, such as the completeness result for forward and backward simulations and the Abadi-Lamport completeness result, do not carry over via the correspondence. In this section, we establish these remaining results in the setting of timed automata. In each case, the proof is analogous to the corresponding proof in Part I.

6.1 Partial Completeness Results

We begin with the partial completeness results for t-forests.

Theorem 6.1 *(Partial completeness of timed refinements) Suppose A is a t-forest, B is t-deterministic and $A \leq^{\mathrm{t}}_{*\mathrm{T}} B$. Then $A \leq^{\mathrm{t}}_{\mathrm{R}} B$.*

Proof: Analogous to the proof of Theorem 4.5 in Part I. Define $r \triangleq \textit{t-after}(B) \circ \textit{t-past}(A)$. We claim that r is a timed refinement from A to B. Conditions 1 and 2 are straightforward. For Condition 3, suppose that $s' \xrightarrow{a}_A s$. Let $q = \textit{t-trace}((s', a, s))$. We must show that $r(s') \overset{q}{\leadsto}_B r(s)$.

By Lemma 3.3, there is a unique timed trace leading from a start state of A to s', and also a unique timed trace leading from a start state of A to s. Let p' and p denote these respective timed traces; then $p' \leq q = p$. Let q' be such that $p'; q' = p$.

Since B is t-deterministic, Lemma 3.3 implies that B has a unique start state; call it s_0. By construction of r, we have that $s_0 \overset{p'}{\leadsto}_B r(s')$ and $s_0 \overset{p}{\leadsto}_B r(s)$. Since $s_0 \overset{p}{\leadsto}_B r(s)$, Lemma 2.3 implies that B has a state s'' such that $s_0 \overset{p'}{\leadsto}_B s''$ and $s'' \overset{q'}{\leadsto}_B r(s)$. But since B is t-deterministic, $s_0 \overset{p'}{\leadsto}_B r(s')$ and $s_0 \overset{p'}{\leadsto}_B s''$, implies that $s'' = r(s')$. Thus, $r(s') \overset{q}{\leadsto}_B r(s)$, as needed. ■

Theorem 6.2 *(Partial completeness of timed backward simulations) Suppose A is a t-forest and $A \leq^{\mathrm{t}}_{*\mathrm{T}} B$. Then*

1. *$A \leq^{\mathrm{t}}_{\mathrm{B}} B$, and*
2. *if B has t-fin then $A \leq^{\mathrm{t}}_{\mathrm{iB}} B$.*

Proof: Analogous to the proof of Theorem 4.18 in Part I. For a given state s of A, Lemma 3.3 implies that there is a unique timed trace leading to s, say p. Define

$$b[s] = \{u | \exists \Sigma \in \textit{t-execs}^*(B) : \textit{t-trace}(\Sigma) = p, \Sigma.\textit{fstate} = u, \text{ and}$$
$$[\Sigma' \prec \Sigma \rightarrow \textit{t-trace}(\Sigma') \neq p]\}.$$

Using $\textit{t-traces}^*(A) \subseteq \textit{t-traces}^*(B)$, it follows that relation b is total. By construction, the relation satisfies Condition 1. Condition 2 follows as in the proof of Theorem 4.18, and Condition 3 uses an argument similar to that in the proof of Prop. 6.1.

Clearly, if B had t-fin then relation b is image-finite. ■

6.2 Completeness of Timed Forward and Timed Backward Simulations

In this subsection, we give the completeness results for timed forward and timed backward simulations.

Lemma 6.3

1. $A \leq_{\mathrm{FB}}^{\mathrm{t}} B \Leftrightarrow (\exists C : A \leq_{\mathrm{F}}^{\mathrm{t}} C \leq_{\mathrm{B}}^{\mathrm{t}} B)$.

2. $A \leq_{\mathrm{iFB}}^{\mathrm{t}} B \Leftrightarrow (\exists C : A \leq_{\mathrm{F}}^{\mathrm{t}} C \leq_{\mathrm{iB}}^{\mathrm{t}} B)$.

3. $A \leq_{\mathrm{BF}}^{\mathrm{t}} B \Leftrightarrow (\exists C : A \leq_{\mathrm{B}}^{\mathrm{t}} C \leq_{\mathrm{F}}^{\mathrm{t}} B)$.

4. $A \leq_{\mathrm{iBF}}^{\mathrm{t}} B \Leftrightarrow (\exists C : A \leq_{\mathrm{iB}}^{\mathrm{t}} C \leq_{\mathrm{F}}^{\mathrm{t}} B)$.

Proof: Similar to the corresponding proofs in Part I (Theorems 5.1 and 5.7). We sketch the proof of the implication "$\Rightarrow$" in 3. Define the structure C by:

- $\mathit{states}(C) = \{(t, S) \in \mathsf{R}^{\geq 0} \times \mathbf{P}(\mathit{states}(B)) | \forall s \in S : s.\mathit{now}_B = t\}$,
- $(t, S).\mathit{now}_C = t$,
- $\mathit{start}(C) = \{(0, S) \in \mathit{states}(C) | S \cap \mathit{start}(B) \neq \emptyset\}$,
- $\mathit{acts}(C) = \mathit{acts}(B)$, and
- for $(t', S'), (t, S) \in \mathit{states}(C)$ and $a \in \mathit{acts}(C) - \{\nu\}$,

$$\begin{array}{lcl} (t', S') \xrightarrow{a}_C (t, S) & \Leftrightarrow & t' = t \text{ and } \forall u' \in S' \; \exists u \in S : u' \stackrel{\hat{a}}{\Longrightarrow}_B u, \text{ and} \\ (t', S') \xrightarrow{\nu}_C (t, S) & \Leftrightarrow & t' < t \text{ and } \forall u' \in S' \; \exists u \in S : u' \stackrel{(\lambda, t)}{\leadsto}_B u. \end{array}$$

Then C is a timed automaton. Let g be a timed backward-forward simulation from A to B. Then the relation g' given by

$$g' = \{(s, (t, S)) \in \mathit{states}(A) \times \mathit{states}(C) \mid t = s.\mathit{now}_A \text{ and } S \in g[s]\}$$

is a timed backward simulation from A to C. Also, the relation f given by

$$f = \{((t, S), u) \in \mathit{states}(C) \times \mathit{states}(B) \mid u \in S\}$$

is a timed forward simulation from C to B. ■

Theorem 6.4 *(Completeness of timed forward and timed backward simulations) Suppose $A \leq_{*\mathrm{T}}^{\mathrm{t}} B$. Then*

1. $\exists C : A \leq_{\mathrm{F}}^{\mathrm{t}} C \leq_{\mathrm{B}}^{\mathrm{t}} B$,
2. *if B has t-fin then* $\exists C : A \leq_{\mathrm{F}}^{\mathrm{t}} C \leq_{\mathrm{iB}}^{\mathrm{t}} B$, *and*
3. $\exists C : A \leq_{\mathrm{B}}^{\mathrm{t}} C \leq_{\mathrm{F}}^{\mathrm{t}} B$.

Proof: Immediate from Lemma 6.3 and Theorem 5.16.

An alternative proof of 1 and 2 can be obtained analogous to the proof of Theorem 4.22 of Part I. Let $C = \mathit{t\text{-}can}(\mathit{t\text{-}beh}(A))$. By Lemma 4.8, C is a t-deterministic t-forest and $A \equiv_{*\mathrm{T}}^{\mathrm{t}} C$. Since C is t-deterministic, $A \leq_{\mathrm{F}}^{\mathrm{t}} C$ by partial completeness of timed forward simulations (Theorem 5.17), and because C is a t-forest, $C \leq_{\mathrm{B}}^{\mathrm{t}} B$ follows by partial completeness of timed backward simulations (Theorem 6.2(1)). Similarly, if B has t-fin then $C \leq_{\mathrm{iB}}^{\mathrm{t}} B$ follows by Theorem 6.2(2). ■

6.3 Auxiliary Variable Constructions

In this subsection, we present results about timed auxiliary variable constructions.

6.3.1 Timed History Relations

Here we describe the results about timed history relations. We begin with a timed analogue to the unfolding construction of Part I.

The *timed unfolding* of A, notation $t\text{-}unfold(A)$, is the timed automaton B defined by

- $states(B) = t\text{-}execs^*(A)$,
- $\Sigma.now_B = \Sigma.ltime$,
- $start(B) = \{\{s\} | s \in start(A)\}$,
- $acts(B) = acts(A)$, and
- for $\Sigma' S', \Sigma S \in states(B)$,

$$\Sigma' S' \xrightarrow{\nu}_B \Sigma S \;\Leftrightarrow\; \Sigma' = \Sigma \wedge S' \subset S$$

and, for $\Sigma', \Sigma \in states(B)$ and $a \in acts(B) - \{\nu\}$,

$$\Sigma' \xrightarrow{a}_B \Sigma \Leftrightarrow \Sigma = \Sigma' \, a \, \{\Sigma.lstate\}.$$

We leave it to the reader to verify that B is a timed automaton.

Proposition 6.5 *$t\text{-}unfold(A)$ is a t-forest and $A \leq^{\mathrm{t}}_{\mathrm{H}} t\text{-}unfold(A)$.*

Proof: Using Lemma 3.2 it follows easily that $t\text{-}unfold(A)$ is a t-forest. The function $.lstate$, which maps each finite timed execution of A to its last state is a timed refinement from $t\text{-}unfold(A)$ to A, and the relation $.lstate^{-1}$ is a timed forward simulation from A to $t\text{-}unfold(A)$. Thus, $.lstate^{-1}$ is a timed history relation from A to $t\text{-}unfold(A)$. ■

We are now in a position to prove a timed version of Sistla's completeness result.

Theorem 6.6 *(Completeness of timed history relations and timed backward simulations) Suppose $A \leq^{\mathrm{t}}_{*\mathrm{T}} B$. Then*

1. *$\exists C : A \leq^{\mathrm{t}}_{\mathrm{H}} C \leq^{\mathrm{t}}_{\mathrm{B}} B$, and*
2. *If B has t-fin then $\exists C : A \leq^{\mathrm{t}}_{\mathrm{H}} C \leq^{\mathrm{t}}_{\mathrm{iB}} B$.*

Proof: Analogous to the proof of Theorem 6.6 in Part I. ■

We next define a notion of *timed superposition*, analogous to the notion of superposition in Part I. Suppose k is a synchronous relation over $states(A)$ and $states(B)$ satisfying $k \cap (start(A) \times start(B)) \neq \emptyset$. The *timed superposition* $t\text{-}sup(A, B, k)$ of B onto A via k is the timed automaton C given by

- $states(C) = k$,
- $(s, v).now_C = s.now_A$,
- $start(C) = k \cap (start(A) \times start(B))$,
- $acts(C) = acts(A) \cap acts(B)$, and
- for $(s', v'), (s, v) \in states(C)$ and $a \in vis(C)$,

$$(s', v') \xrightarrow{\nu}_C (s, v) \;\Leftrightarrow\; s'.now < s.now \wedge s' \overset{(\lambda, s.now)}{\rightsquigarrow}_A s \wedge v' \overset{(\lambda, v.now)}{\rightsquigarrow}_B v,$$

$$(s', v') \xrightarrow{\tau}_C (s, v) \;\Leftrightarrow\; s' \overset{\lambda}{\Longrightarrow}_A s \wedge v' \overset{\lambda}{\Longrightarrow}_B v,$$

$$(s', v') \xrightarrow{a}_C (s, v) \;\Leftrightarrow\; s' \overset{a}{\Longrightarrow}_A s \wedge v' \overset{a}{\Longrightarrow}_B v.$$

Again we leave it to the reader to check that C is indeed a timed automaton.

Theorem 6.7 $A \leq_{\mathrm{F}}^{\mathrm{t}} B \Leftrightarrow (\exists C : A \leq_{\mathrm{H}}^{\mathrm{t}} C \leq_{\mathrm{R}}^{\mathrm{t}} B)$.

Proof: Suppose $A \leq_{\mathrm{F}}^{\mathrm{t}} B$. Let f be a timed forward simulation from A to B, let $C = \textit{t-sup}(A, B, f)$ and let π_1 and π_2 be the projection functions that map states of C to their first and second components, respectively. Then it is easy to check that π_1^{-1} is a timed history relation from A to C and π_2 is a timed refinement from C to B.

The reverse implication also follows via a standard argument. ∎

6.3.2 Timed Prophecy Relations

Finally, we describe the results about timed prophecy relations. We give a timed analogue to the guess construction of Part I. This can be regarded as a dual to the timed unfolding construction of the previous subsection.

The *timed guess* of A, notation $\textit{t-guess}(A)$, is the timed automaton B defined by

- $states(B) = \textit{t-frag}^*(A)$,
- $\Sigma.now_B = \Sigma.\mathit{ftime}$,
- $start(B) = \textit{t-execs}^*(A)$,
- $acts(B) = acts(A)$, and
- for $S'\Sigma', S\Sigma \in states(B)$,

  $$S'\Sigma' \xrightarrow{\nu}_B S\Sigma \;\Leftrightarrow\; S' \supset S \wedge S'.\mathit{ltime} = S.\mathit{ltime} \wedge \Sigma' = \Sigma$$

  and, for $\Sigma', \Sigma \in states(B)$ and $a \in acts(B) - \{\nu\}$,

  $$\Sigma' \xrightarrow{a}_B \Sigma \Leftrightarrow \{\Sigma'.\mathit{fstate}\}\, a\, \Sigma = \Sigma'.$$

As usual, we leave it to the reader to verify that B is a timed automaton.

Proposition 6.8 $A \leq^{\mathrm{t}}_{\mathrm{P}}$ *t-guess*(A).

Proof: Similar to the proof of Prop. 6.5. ■

Theorem 6.9

1. $A \leq^{\mathrm{t}}_{\mathrm{B}} B \Leftrightarrow (\exists C : A \leq^{\mathrm{t}}_{\mathrm{P}} C \leq^{\mathrm{t}}_{\mathrm{R}} B)$.
2. $A \leq^{\mathrm{t}}_{\mathrm{iB}} B \Leftrightarrow (\exists C : A \leq^{\mathrm{t}}_{\mathrm{iP}} C \leq^{\mathrm{t}}_{\mathrm{R}} B)$.

Proof: Similar to the proof of Theorem 6.7, using timed backward simulations instead. ■

Theorem 6.10 *(Completeness of timed prophecy relations and timed forward simulations)* $A \leq^{\mathrm{t}}_{*\mathrm{T}} B \Rightarrow \exists C : A \leq^{\mathrm{t}}_{\mathrm{P}} C \leq^{\mathrm{t}}_{\mathrm{F}} B$.

6.3.3 Completeness of Timed History and Timed Prophecy Relations

Theorem 6.11 *(Completeness of timed history/prophecy relations and refinements) Suppose* $A \leq^{\mathrm{t}}_{*\mathrm{T}} B$*. Then*

1. $\exists C, D : A \leq^{\mathrm{t}}_{\mathrm{H}} C \leq^{\mathrm{t}}_{\mathrm{P}} D \leq^{\mathrm{t}}_{\mathrm{R}} B$.
2. *If* B *has t-fin then* $\exists C, D : A \leq^{\mathrm{t}}_{\mathrm{H}} C \leq^{\mathrm{t}}_{\mathrm{iP}} D \leq^{\mathrm{t}}_{\mathrm{R}} B$.
3. $\exists C, D : A \leq^{\mathrm{t}}_{\mathrm{P}} C \leq^{\mathrm{t}}_{\mathrm{H}} D \leq^{\mathrm{t}}_{\mathrm{R}} B$.

Proof: Completely analogous to the proofs of Theorems 6.18 and 6.19 in Part I. ■

7 Discussion

In this paper, we have presented an automata-theoretic model for timing-based systems, and have used it to develop a variety of simulation proof techniques for such systems. These include timed refinements, timed forward and backward simulations and combinations thereof, and timed history and prophecy relations. These techniques are analogous to those described in Part I, [15], for untimed systems. As in that paper, we present basic results for all of the simulations, including soundness and completeness results. The development is organized so that the proofs are based on the results of Part I. In fact, we have shown that all the results of Part I carry over to Part II, except for Prop. 4.12. At present we do not know whether the timed version of this result holds, i.e., whether if A is a t-forest and $A \leq^{\mathrm{t}}_{\mathrm{F}} B$, it is the case that $A \leq^{\mathrm{t}}_{\mathrm{R}} B$.

It remains to apply these methods to a wide range of practical verification examples, in order to determine their utility, to develop them further, and to exploit their power. Timed forward simulations have already been used in [12] to verify some simple toy example timed systems, and in [11] to verify more realistic algorithms. These uses already suggest that at least the timed forward simulations will prove to be very useful in practice, but more evidence is needed. Note that the results in [12] use a more restrictive model than the one in this paper, namely, that of [16]. The extra structure of that model supports development of specialized *progress measure* techniques not discussed in this paper. It remains to develop this and other specialized methods further.

It remains to develop other proof methods within the same general timed automaton model. In particular, we are interested in extending the methods of process algebra to our timed automaton model. Our recent paper [22] contains the beginning of such work, including definitions of interesting operators on timed automata, and proofs of substitutivity results for the timed trace semantics, but there is more to be done.

Finally, although the timed automaton model presented here is very general, there are at least two ways in which it could be extended: to include reasoning about liveness and about probabilities. It remains to extend the model in these ways, while preserving the ability to use the simpler model of this paper where appropriate. Some preliminary work on integrating liveness into the present model appears in [11]. Both liveness and probabilities introduce their own sets of additional proof methods, e.g., temporal logic and Markov analysis. Eventually, the entire collection of proof tools should be integrated into a sensibly coordinated whole.

Acknowledgements

We thank Albert Meyer, Jeff Sanders, Roberto Segala, Jørgen Søgaard-Andersen, Eugene Stark and George Varghese for their valuable criticism and useful comments on this paper and on [15]. We also thank the organizers of the 1991 REX Workshop for providing the environment for an active research interchange that led to many improvements in our work.

A The Trajectory Axiom

Of the five axioms we give for timed automata, the axiom

S5 If $s' \xrightarrow{\nu} s$ then there exists a trajectory from s' to s.

seems to us like the only one that might be controversial. In Wang [23] and elsewhere ([14, 20]), the following weaker axiom **S5′** occurs instead:

S5′ If $s' \xrightarrow{\nu} s$ and $s'.now < t < s.now$, then there is an s'' with $s''.now = t$ such that $s' \xrightarrow{\nu} s''$ and $s'' \xrightarrow{\nu} s$.

It is immediate from the definition of a trajectory that **S5** $\Rightarrow$ **S5′**. In this appendix we discuss the reverse implication. The relationship between the two axioms is also investigated in [10].

As the time domain for our timed automata we have chosen the set $\mathsf{R}^{\geq 0}$ of nonnegative real numbers. We could have chosen a different time domain though, or parametrized our automata with an arbitrary time domain as in [9, 20]. In order to state the axioms for timed automata, all we need is the presence of a set T of *points in time*, containing an *initial* point in time 0, and equipped with a binary relation $<$ expressing *precedence*. Thus we can generalize our notion of a timed automaton by parametrizing it with a *time domain* $(T, 0, <)$. For an overview of different types of time domains that have been proposed in the literature we refer to [4].

Define a *semi-timed automaton* to be a timed automaton, except that it does not have to satisfy **S5**, but only the weaker axiom **S5′**. Here we give three results:

1. Each time deterministic semi-timed automaton is a timed automaton.

2. Each semi-timed automaton is a timed automaton, if instead of $\mathsf{R}^{\geq 0}$ a countable, totally ordered time domain is used, like for instance the set $\mathsf{Q}^{\geq 0}$ of nonnegative rational numbers.

3. If the time domain consists of $\mathsf{R}^{\geq 0}$, then there exists a semi-timed automaton that is not a timed automaton.

The axiom of *time determinism* can be formulated in our setting as follows:

TD If $s \xrightarrow{\nu} s'$, $s \xrightarrow{\nu} s''$ and $s'.now = s''.now$, then $s' = s''$.

The axiom **TD** says that time is deterministic in the sense that, after a certain amount of time has elapsed since the system arrived in some state, the new state is uniquely determined provided no internal or visible action has taken place. The following theorem is easy to prove.

Theorem A.1 *Each time deterministic semi-timed automaton is a timed automaton.*

Theorem A.2 *Suppose A is a semi-timed automaton over a time domain $(T, 0, <)$. If T is countable and $<$ is a total order, then A satisfies* **S5** *(and is thus a timed automaton).*

Proof: Suppose T is countable, $<$ is a total order, and $s' \xrightarrow{\nu}_A s$. We have to prove that there exists a trajectory from s' to s. Let $t' = s'.now$ and $t = s.now$, and let $t_1, t_2, \ldots$ be some arbitrary enumeration of all the points in time in the interval (t', t). Inductively, we define a sequence $s_1, s_2, \ldots$ of states of A such that, for all i, (1) $s_i.now = t_i$, and (2) the set $S_i = \{s', s, s_1, \ldots, s_i\}$ is *time connected*, that is, for all $u', u \in S_i$, $u'.now < u.now \Rightarrow u' \xrightarrow{\nu} u$.

Suppose that, for some $n \geq 0$, we have defined states $s_1, \ldots, s_n$ that satisfy properties (1) and (2) above. Let u' be the state in S_n with the largest time that is smaller than t_{n+1}, and let u be the state in S_n with the smallest time that is larger than t_{n+1}. (The existence and uniqueness of u' and u are guaranteed since $.now$ is injective on S_n and $<$ is a total order.) Since $u' \xrightarrow{\nu} u$ and $u'.now < t_{n+1} < u.now$, there exists, by axiom **S5′**, a state s_{n+1} such that $u' \xrightarrow{\nu} s_{n+1}$ and $s_{n+1} \xrightarrow{\nu} u$. Let u'' be a state in $S_n - \{u', u\}$. Then either $u''.now < t_{n+1}$ or $t_{n+1} < u''.now$. If $u''.now < t_{n+1}$ then $u''.now < u'.now$, and thus, since S_n is time conected by induction hypothesis, $u'' \xrightarrow{\nu} u'$. From $u'' \xrightarrow{\nu} u'$ and $u' \xrightarrow{\nu} s_{n+1}$, $u'' \xrightarrow{\nu} s_{n+1}$ follows by axiom **S4**. If $t_{n+1} < u''.now$ then we can infer via a similar argument that $s_{n+1} \xrightarrow{\nu} u''$. Thus it follows that S_{n+1} is time connected, and we have proved the induction step.

Once we have constructed the sequence $s_1, s_2, \ldots$ as above, it is immediate that the function w defined by $w(t') = s'$, $w(t) = s$, and $w(t_i) = s_i$, for all i, is a trajectory from s' to s. ∎

The above proof relies heavily on the assumption that the time domain is countable: since the interval $[t', t]$ is countable we can construct a trajectory from s' to s in an inductive fashion, state by state. Such a construction is no longer possible if the time domain is uncountable, as in the case of $\mathsf{R}^{\geq 0}$. This is illustrated by the following counterexample.

Theorem A.3 (Schneider) *Let D be defined by*

- $states(D) = \mathsf{R}^{\geq 0} \times \mathsf{Q}^{\geq 0}$,
- $start(D) = \{(0, 0)\}$,
- $(t, q).now_D = t$,
- $acts(D) = \{\nu, \tau\}$,
- *$steps(D)$ is specified by* $(t', q') \xrightarrow{\nu}_D (t, q) \;\Leftrightarrow\; t' < t \wedge q' < q$.

Then automaton D is semi-timed, but not timed.

Proof: One can easily check that D is semi-timed. However, it is not timed: D does not satisfy the trajectory axiom **S5** because that would imply, for instance, that the interval $[0, 1]$ of reals can be injectively mapped into the rationals. ■

At the time we first defined axiom **S5**, we constructed a complex counterexample to show that it was stronger than **S5′**. The simpler counterexample described above was subsequently discovered by Steve Schneider.

References

[1] R. Alur and D.L. Dill. The theory of timed automata. In de Bakker et al. [6], pages 45–73.

[2] J.C.M. Baeten and J.A. Bergstra. Real time process algebra. *Journal of Formal Aspects of Computing Science*, 3(2):142–188, 1991.

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

[4] J. van Benthem. Time, logic and computation. In J.W. de Bakker, W.P. de Roever, and G. Rozenberg, editors, *REX School/Workshop on Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency,* Noordwijkerhout, volume 354 of *Lecture Notes in Computer Science*, pages 1–49. Springer-Verlag, 1989.

[5] G. Berry and L. Cosserat. The **Esterel** synchronous programming language and its mathematical semantics. In S.D. Brookes, A.W. Roscoe, and G. Winskel, editors, *Seminar on Concurrency*, volume 197 of *Lecture Notes in Computer Science*, pages 389–448. Springer-Verlag, 1984.

[6] 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, 1992.

[7] R. Gerber and I. Lee. The formal treatment of priorities in real-time computation. In *Proceedings 6th IEEE Workshop on Real-Time Software and Operating Systems*, 1989.

[8] J.F. Groote. Specification and verification of real time systems in ACP. Report CS-R9015, CWI, Amsterdam, 1990. An extended abstract appeared in L. Logrippo, R.L. Probert and H. Ural, editors, *Proceedings* 10^{th} *International Symposium on Protocol Specification, Testing and Verification*, Ottawa, pages 261–274, 1990.

[9] A. Jeffrey. A linear time process algebra. In K.G. Larsen and A. Skou, editors, *Proceedings of the Third Workshop on Computer Aided Verification,* Aalborg, Denmark, July 1991, volume 575 of *Lecture Notes in Computer Science*, pages 432–442. Springer-Verlag, 1992.

[10] A. Jeffrey, S. Schneider, and F.W. Vaandrager. A comparison of additivity axioms in timed transition systems, 1993. In preparation.

[11] B. Lampson, N. Lynch, and J. Søgaard-Andersen. Correctness of at-most-once message delivery protocols, 1993. Submitted for publication.

[12] N.A. Lynch and H. Attiya. Using mappings to prove timing properties. *Distributed Computing*, 6(2):121–139, 1992.

[13] 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, August 1987. A full version is available as MIT Technical Report MIT/LCS/TR-387.

[14] N.A. Lynch and F.W. Vaandrager. Forward and backward simulations for timing-based systems. In de Bakker et al. [6], pages 397–446.

[15] N.A. Lynch and F.W. Vaandrager. Forward and backward simulations – part I: Untimed systems, 1993. In preparation.

[16] M. Merritt, F. Modugno, and M. Tuttle. Time constrained automata. In J.C.M. Baeten and J.F. Groote, editors, *Proceedings CONCUR 91,* Amsterdam, volume 527 of *Lecture Notes in Computer Science*, pages 408–423. Springer-Verlag, 1991.

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

[18] F. Moller and C. Tofts. A temporal calculus of communicating systems. In Baeten and Klop [3], pages 401–415.

[19] X. Nicollin, J.-L. Richier, J. Sifakis, and J. Voiron. ATP: An algebra for timed processes. In M. Broy and C.B. Jones, editors, *Proceedings IFIP TC2 Working Conference on Programming Concepts and Methods,* Sea of Gallilea, Israel, pages 402–429, 1990.

[20] X. Nicollin, J. Sifakis, and S. Yovine. From ATP to timed graphs and hybrid systems. In de Bakker et al. [6], pages 549–572.

[21] G.M. Reed and A.W. Roscoe. A timed model for communicating sequential processes. *Theoretical Computer Science*, 58:249–261, 1988.

[22] F.W. Vaandrager and N.A. Lynch. Action transducers and timed automata. In W.R. Cleaveland, editor, *Proceedings CONCUR 92,* Stony Brook, NY, USA, volume 630 of *Lecture Notes in Computer Science*, pages 436–455. Springer-Verlag, 1992.

[23] Wang Yi. Real-time behaviour of asynchronous agents. In Baeten and Klop [3], pages 502–520.

[24] A. Zwarico. *Timed Acceptance: An Algebra of Time Dependent Computing.* PhD thesis, Department of Computer and Information Science, University of Pennsylvania, 1988.