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 94079, 1090 GB Amsterdam
The Netherlands
`fritsv@cwi.nl`

April 26, 1995

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) hybrid forward-backward and backward-forward simulations, and (4) history and prophecy relations. Relationships between the different types of simulations, as well as soundness and completeness results, are stated and proved. These results are (with one exception) analogous to the results for untimed systems in Part I of this paper. 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, D.2.4, F.1.1, F.3.1.
Keywords and Phrases: Timing-based systems, real-time, timed automata, trajectories, safety properties, verification, simulations, refinement mappings, forward simulations, backward simulations, forward-backward simulations, backward-forward simulations, history relations, prophecy relations, invariants.
Notes: This work was supported by ONR contracts N00014-85-K-0168 and N00014-91-J-1988, by AFOSR-ONR contract F49620-94-1-0199, by NSF grants CCR-8915206 and 9225124-CCR, and by ARPA 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 ESPRIT Basic Research Action 7166, CONCUR2. Earlier versions of this paper appeared as [42] (Part I+II) and as [43].

1 Introduction

Most of the existing semantic models, languages and logics for describing and reasoning about timing-based systems implicitly view an execution as an alternating sequence of instantaneous "discrete" actions and "continuous" phases during which time advances [2, 5, 7, 8, 9, 11, 14, 17, 20, 25, 26, 27, 48, 50, 52, 54, 61, 62]. To each system described in any of these formalisms one can associate a *transition system* or *automaton* consisting of (1) a set of states, (2) a set of initial states, (3) a set of discrete actions, (4) a set of discrete steps $s' \xrightarrow{a} s$ asserting that "from state s' the system can instantaneously move to state s via the occurrence of the discrete action a", and, finally, (5) a set of time-passage steps $s' \xrightarrow{d} s$ asserting that "from state s' the system can move to state s during a positive amount of time d in which no discrete action occurs".

These transition systems provide a very abstract view of the behavior of the original system in which many aspects, such as the number of parallel components, the communication between these components, the way in which a system evolves during the continuous phases, etc., are no longer represented. Also, they are in general highly infinite and may even have uncountable state spaces. Nevertheless, it is clear that these transition systems play a central role in the theory of timing-based systems:

- Many important behavioral preorders and equivalences, for instance those based on traces, failure pairs and bisimulations, can be defined in terms of states and transitions. Thus transition systems contain enough information to define what it means that one system *implements* or *is equivalent to* another system. Also, the transition systems still contain enough information to serve as models for many temporal and modal logics, i.e., they can be used to define what it means that a system *satisfies* a formula.

- Many simulation proof techniques for verification of implementation and equivalence relations between timing-based systems can be defined and studied at the level of transition systems.

- Transition systems provide an excellent framework for comparing and interrelating a wide variety of different formalisms for timing-based systems. Moreover, since they also play a central role in the "comparative semantics" of untimed discrete event systems [18], they provide a basis for comparing timed and untimed formalisms.

In this paper, we define a formal transition system model for timing-based systems and use it to develop a variety of simulation proof techniques. The key characteristic of the transition systems discussed above is the presence of time-passage steps and the specific interpretation of these steps. The transition systems always satisfy the following two properties. First, if time can advance by a particular amount d in two steps (with no intervening discrete steps), then it can also advance by d in a single step. And second, if time can advance by d in one step from state s' to state s, then there exists an assignment (a *trajectory*) that maps all times in the interval $[0, d]$ to automaton states in a "consistent" way to explain how the system evolves from s' to s. This motivates our formal definition of a *timed automaton* as an automaton (in the sense of Part I) whose set of actions includes the set R^+ of positive reals, and which satisfies the above two properties for time-passage. We believe that timed automata, defined in this way, provide an excellent basis for defining and studying behavioral preorders and simulation proof techniques for timing based systems. Since timed automata can be viewed as an underlying semantic domain for any of the models, languages and logics of [2, 5, 7, 8, 9, 11, 14, 17, 20, 25, 26, 27, 48, 50, 52, 54, 61, 62], all the results that we obtain for timed automata carry over directly to those settings.

For convenience, we use R^+ as our domain of times in this paper. The need for dense-time models has been well discussed in [4]. However, for the purpose of generality we could have parameterized our timed automata by an arbitrary (possibly discrete) *time domain* in the sense of [27, 53, 28]. We do not assume a general lower bound on the time between events, or an upper bound on the number of instantaneous actions; this choice is also made in e.g., [7, 2, 9, 25, 48, 53, 61], but still distinguishes our model from many others, e.g., [11, 17, 20, 50, 52, 55, 62]. The cost of this generality is that our timed automata may produce some annoying "Zeno executions", i.e., infinite executions in which the sum of the time-passage actions is bounded.

In order to define correctness for timed automata, we define two notions of external behavior. 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 visible actions together with a final time of observation. Second, as the infinite behaviors, we take the *admissible timed traces*, each of which consists of a sequence of timed visible actions that occurs in some execution in which the time grows unboundedly (i.e., a "non-Zeno" infinite execution). In [16] it is argued that inclusion of finite and admissible timed traces is a good notion of implementation, provided that the implementation automaton has a sufficiently rich collection of admissible executions.

Inclusion of finite and admissible timed traces is implied by inclusion of finite and infinite traces (if we consider the R^+ actions as external/visible). Consequently all the simulation proof techniques that we developed in Part I are still "sound" for proving inclusion of timed traces, in the sense that if one has established a simulation between timed automata A and B it follows that the timed traces of A are included in those of B. However, "completeness" is lost in the sense that it may occur that the timed traces of a timed automaton A are included in those of a timed automaton B, but that there exists no simulation from A to B, not even if it is allowed to use auxiliary intermediate timed automata. One reason for this is that several of the constructions that were used in the proofs of completeness results in Part I, such as the canonical automaton and the unfolding, do not yield timed automata in general. Also — and this is much more serious — inclusion of timed traces differs from inclusion of traces in the case of systems with internal actions.

Example 1.1 Let A be the timed automaton that performs no discrete actions but just lets time advance: the set of states of A is $\mathsf{R}^{\geq 0}$, with 0 the initial state, and there is a step $t \xrightarrow{d} t+d$, for each $t \in \mathsf{R}^{\geq 0}$ and $d \in \mathsf{R}^+$. Let B be the timed automaton that behaves exactly as A, except that it performs an internal τ-step at time 1: the set of states of B is $\mathsf{R}^{\geq 0} \times \{\mathsf{T}, \mathsf{F}\}$, with $(0, \mathsf{T})$ the initial state, and there are steps

- $(t, \mathsf{T}) \xrightarrow{d} (t + d, \mathsf{T})$, for each $t \in \mathsf{R}^{\geq 0}$ and $d \in \mathsf{R}^+$ with $t + d \leq 1$;
- $(1, \mathsf{T}) \xrightarrow{\tau} (1, \mathsf{F})$;
- $(t, \mathsf{F}) \xrightarrow{d} (t + d, \mathsf{F})$, for each $t \in \mathsf{R}^{\geq 0}$ and $d \in \mathsf{R}^+$.

Then A and B have different sets of traces since A has a trace consisting of the single (time-passage) action 2, which B does not have.

In our opinion, this example shows that traces are not the right notion of behavior for timed automata: through the absence of certain traces with large time-passage steps the presence of certain internal actions in the system is revealed, and thus internal actions are not truly invisible. Internal actions have received proper attention in the context of process algebras based on bisimulation or failures, and thus the two systems of Example 1.1 are identified in the approaches of (for instance) [30, 55, 14]. In models based on linear time semantics, however, internal (or stuttering) actions have largely been ignored. Abadi and Lamport [2] advocate the use of untimed trace inclusion (logical implication in TLA) as an implementation relation for timed systems. Although this "old-fashioned recipe" works in many practical cases, the two systems of Example 1.1, which can easily be translated to the state-based setting of [2], indicate that it cannot be used in general, and that a serious effort is required to fully adapt existing formalisms for untimed systems to the timed setting.

Simulation methods have long been used successfully for the verification of untimed concurrent systems. In Part I of this paper [44], we gave a unified, comprehensive presentation of simulation techniques for untimed systems, including refinements, forward simulations, backward simulations, forward-backward and backward-forward simulations, history and prophecy relations. We showed relationships among the different types of simulations and soundness and completeness theorems. Part I also contains pointers to examples of uses of simulation methods for verification.

Because simulations have been so successful for untimed systems, we believe that they will also prove to be successful for timed systems. (Considerable evidence for this is described below.) Thus, in writing Part II of this paper, our goal has been to define timed versions of *all* the simulations in Part I (timed refinements, timed forward simulations, etc.) in terms of timed automata, and to establish the timed versions of *all* the soundness, completeness and other results of Part I.

The definitions of all of our timed simulations are analogous to the definitions of the corresponding untimed simulations in Part I, but are based on our new notions of external behavior. It turns out that the results for timed simulations are *almost entirely* analogous to those for the untimed simulations (even though it requires considerable effort to prove this). In fact, in many cases, we are able to derive the results for timed simulations as consequences of the results for untimed simulations. In the remaining cases, new proofs analogous to those in Part I are presented. Our presentation highlights the adaptability of the various simulation techniques from the untimed to the timed setting. There is just one minor result from Part I, Proposition 3.12, that does not carry over to the timed setting. We remark that we found the definitions involving timed automata and their simulations quite difficult to get "right". These definitions involve many choices, most of which do either lead to longer proofs or do not yield all the properties in this paper. The problem to develop a theory of timed transition systems and timed simulations with analogues of *all* results of Part I is still open.

This paper does not contain examples of verifications carried out using timed simulations. However, our timed simulations have already been used extensively elsewhere [12, 23, 32, 34, 35, 36, 37, 38, 45, 58, 60]. The algorithms and systems verified in these papers include toy examples such as counters and process races, as well as substantial real examples such as a clock-based at-most-once message delivery protocol, a clock synchronization algorithm, two mutual exclusion algorithms, a leader election algorithm, and a communication protocol used in a consumer electronics system. They also include a toy process control example involving control of a railroad crossing gate. An interesting feature of these proofs is that the simulations have been used not only to prove "ordinary" safety properties, as in the untimed setting, but also to prove timing properties, e.g., upper and lower bounds on time. In this way, the power of simulation techniques seems to be much greater in the timed setting than in the untimed setting. Also, the systems verified are typically parameterized by arbitrary parameters representing process speeds, message delivery times, clock rates, etc., so that the results are very general. In [35, 19], three of the proofs are automated using the Larch Prover [22].

We consider the main contributions of this paper to be the following. (a) The definition of a timed automaton and of its external behavior. (b) The extension of simulation notions for untimed systems to the timed setting. (c) The unified presentation of all the simulation techniques together with their basic soundness and completeness properties. (d) The presentation of many auxiliary definitions and results, for instance about sampling of computations, timed forests, timed unfolding, a timed version of the historization construction of [29], etc. (e) The fact that our presentation parallels, and is based closely on, a similar development for untimed systems.

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 definitions of all the timed simulations. Sections 6 and 7 contain the major results of the paper – the relationships among the timed simulations and the soundness and completeness results. Section 6 contains those results that are derived from corresponding results for the untimed case, while Section 7 contains those results that require new proofs, in particular, the construction of auxiliary (intermediate) timed automata. Section 7 also contains the single example of a result from Part I that does not carry over to the timed setting. Section 8 describes how invariants can be included in the simulations. Finally, Section 9 contains some conclusions. Appendix A contains a discussion of some alternative axioms for timed automata, and Appendix B gives a glossary of notational conventions that we use. Because of the strong dependence of this paper on Part I [44], 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.

2 Timed Automata and Their Behaviors

In this section, we present the timed automaton model. We define "timed executions", which describe how timed automata operate, and "timed traces", which describe their externallyvisible behavior. A timed execution includes information about discrete changes to the automaton's state, plus information about the evolution of the state as time passes continuously.

Since timed automata are a special case of the (untimed) automata defined in Part I of this paper [44], the notions of "execution" and "trace" for untimed automata also make sense for timed automata. We relate the notions of execution and timed execution for a timed automaton: an execution can be regarded as "sampling" the state information of a timed execution at a countable number of points in time. Also, we relate the notion of trace and timed trace.

2.1 Timed Automata

A *timed automaton* (or *timed transition system*) A is an automaton (as defined in Part I) whose set of actions includes R^+, the set of positive reals.[1] Actions from R^+ are referred to as *time-passage actions*, while non-time-passage actions are referred to as *discrete actions*. We let $d, d', \ldots$ range over R^+ and more generally, $t, t', \ldots$ over the set $\mathsf{R}^{\geq 0} \cup \{\infty\}$ of nonnegative real numbers plus infinity. The set of *visible* actions is defined by $vis(A) \triangleq ext(A) - \mathsf{R}^+$. In this part of the paper, $A, B, \ldots$ will range over *timed* automata. We assume that a timed automaton satisfies two axioms.

S1 If $s' \xrightarrow{d} s''$ and $s'' \xrightarrow{d'} s$, then $s' \xrightarrow{d+d'} s$.

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

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

Thus, a trajectory assigns a state to each time in the interval I, in a "consistent" manner. We define $w.ltime$, the "last time" of w, to be the supremum of I. In particular, if I is an infinite interval then $w.ltime$ is ∞. We define $w.fstate$ to be $w(0)$, and if I is right-closed, we also define $w.lstate$ to be $w(w.ltime)$. A trajectory with a domain that is the single-point interval $[0, 0]$ is also called a *trivial* trajectory. A *trajectory* for a step $s' \xrightarrow{d} s$ is a $[0, d]$-trajectory such that $w.fstate = s'$ and $w.lstate = s$. Now we can state the second axiom.

S2 Each time-passage step $s' \xrightarrow{d} s$ has a trajectory.

Axiom **S1** allows repeated time-passage steps to be combined into one step. Axiom **S2** is a kind of converse to **S1**; it says that any time-passage step can be "filled in" with states for each intervening time, in a consistent way.

[1] The decision to use only positive reals as time-passage actions is a matter of taste. We could have allowed for a 0-action with as additional axiom

S0 $s' \xrightarrow{0} s$ if and only if $s' = s$.

However, we would like to distinguish the discrete action τ from the time-passage action 0, both for conceptual and technical reasons: the definitions of several process algebraic operations on timed automata, as discussed in [42], become much more involved if τ's are treated as time-passage actions.

In the modelling of hybrid systems, trajectories are often used to describe the evolution of physical parameters such as position, velocity, acceleration, temperature, and pressure. In such cases, each trajectory w is describable as a continuous function of time. Several models for hybrid systems [47, 6] include the assumption that trajectories are continuous. However, besides the model of this paper there are also models that do not include such an assumption [51], and in fact we do not need continuity of trajectories for our results.

Axiom **S2** is a strengthening of a similar axiom proposed by Wang [61] and used in [42, 53], which, rephrased in our terminology, reads:

S2′ If $s' \xrightarrow{d} s$ and $0 < d' < d$, then there is an s'' such that $s' \xrightarrow{d'} s''$ and $s'' \xrightarrow{d-d'} s$.

The stronger condition seems natural to us — for example, it provides a direct way of modelling changes in physical parameters in a hybrid system. Besides, we need it for some of our results, for instance, Lemma 3.4. In Appendix A, we discuss the relationship between axioms **S2** and **S2′** in more detail and show that **S2′** does not in general imply **S2**.

It is possible to combine two "compatible" trajectories of a timed automaton A into one: if w_1 is an I_1-trajectory, where I_1 is right-closed, if w_2 is an I_2-trajectory, if $w_1.\mathit{lstate} = w_2.\mathit{fstate}$, and if we let $l_1 = w_1.\mathit{ltime}$, then we can define $w_1 \cdot w_2$ to be the least function w such that: $w(t) = w_1(t)$ for $t \in I_1$, and $w(t + l_1) = w_2(t)$ for $t \in I_2$.

Lemma 2.1 *If $w = w_1 \cdot w_2$ then w is an I-trajectory, where $I = I_1 \cup \{t + l_1 \mid t \in I_2\}$.*

Proof: Choose $t, t' \in I$ with $t < t'$. We show that $w(t) \xrightarrow{t'-t} w(t')$. If $t' \leq l_1$, this follows from the fact that w_1 is an I_1-trajectory, while if $t \geq l_1$, this follows from the fact that w_2 is an I_2-trajectory.

The remaining case is where $t < l_1 < t'$. In this case, the fact that w_1 is an I_1-trajectory implies that $w_1(t) \xrightarrow{l_1-t} w_1.\mathit{lstate}$, which implies that $w(t) \xrightarrow{l_1-t} w_1.\mathit{lstate}$. Also, the fact that w_2 is an I_2-trajectory implies that $w_2.\mathit{fstate} \xrightarrow{t'-l_1} w_2(t'-l_1)$, which implies that $w_2.\mathit{fstate} \xrightarrow{t'-l_1} w(t')$. Since $w_1.\mathit{lstate} = w_2.\mathit{fstate}$, axiom **S1** implies that $w(t) \xrightarrow{t'-t} w(t')$, as needed. ∎

Likewise, we may combine a countable sequence of "compatible" trajectories into one: if w_i is an I_i-trajectory, for each positive integer i, where all I_i are right-closed, if $w_i.\mathit{lstate} = w_{i+1}.\mathit{fstate}$ and if we let $l_i = w_i.\mathit{ltime}$, for all i, then the infinite concatenation $w_1 \cdot w_2 \cdot w_3 \ldots$ is defined to be the least function w such that $w(t + \Sigma_{j<i} l_j) = w_i(t)$ for all $t \in I_i$.

Lemma 2.2 *If $w = w_1 \cdot w_2 \cdot w_3 \ldots$ then w is an I-trajectory, where $I = \bigcup_i \{t + \Sigma_{j<i} l_j \mid t \in I_i\}$.*

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; an execution is an alternating sequence of states and actions (including time-passage actions as a special case), subject to the natural consistency constraints. 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 the real times occurring during a time-passage step, we define a notion of *timed execution*, which includes such information. The usual kind of execution can be regarded as "sampling" a timed execution at countably many points in time, as we show in Section 2.4.2 below.

2.2.1 Basic Definitions

A *timed execution fragment* of a timed automaton A is a finite or infinite alternating sequence $W = w_0 a_1 w_1 a_2 w_2 \cdots$, where:

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

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

3. If w_i is not the last trajectory in W then its domain is a right-closed interval and $w_i.lstate \xrightarrow{a_{i+1}} w_{i+1}.fstate$.

An execution fragment describes all the discrete changes that occur, plus the evolution of the state during time-passage steps. The last property says that each pair (w_i, w_{i+1}) of successive trajectories in the fragment "matches up" properly, in that the intervening discrete action a_{i+1} spans properly between the last state of w_i and the first state of w_{i+1}.

Note that the definition of a timed execution fragment allows the modelling of consecutive discrete actions, without intervening time-passage. In this case, the trajectory between the two discrete actions is trivial.

If W is a timed execution fragment then we let $W.ltime$ denote $\Sigma_i w_i.ltime$. Note that we allow the case where the domain of the final trajectory is of the form $[0, \infty)$; in this case, $W.ltime = \infty$. We define the first state of W, $W.fstate$, to be $w_0.fstate$. A *timed execution* is a timed execution fragment W for which $W.fstate$ is a start state.

Note that the *super-dense* computations of [47] correspond closely to our timed executions.

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. The distinctions involve whether or not time passes to infinity, and whether an infinite or finite amount of activity occurs. Thus, we define a timed execution fragment W to be

1. *finite* if W is a finite sequence and the domain of its final trajectory is a right-closed interval,

2. *admissible* if $W.ltime = \infty$, and

3. *Zeno* if W is neither finite nor admissible.

If W is a finite timed execution fragment with final trajectory w_i, then $W.ltime$ is finite. In this case, we define $W.lstate$, the last state of α, to be $w_i.lstate$. We define a state s to be *t-reachable* in timed automaton A provided that there is a finite timed execution W such that $W.lstate = s$. The following fact follows directly by axiom **S2**.

Lemma 2.3 *A state s of a timed automaton A is t-reachable if and only if it is reachable, i.e., there is an ordinary finite execution of A that ends in s.*

An important implication of Lemma 2.3 is that any technique that can prove that a property holds for all final states of (ordinary) finite executions is a sound technique for proving that a property holds in all t-reachable states of a timed automaton. In particular, induction on the steps of ordinary executions is sound in this sense.

If W is a finite timed execution fragment with final trajectory w_i, W' is a timed execution fragment with initial trajectory w'_0, and $w_i.lstate = w'_0.fstate$ then we define $W \cdot W'$ to be the timed execution fragment obtained by concatenating the sequences W and W', except that the consecutive pair of trajectories w_i and w'_0 is replaced by $w_i \cdot w'_0$. Lemma 2.1 implies that $W \cdot W'$ is in fact a timed execution fragment. If W and W' are timed execution fragments, then define W' to be a *t-prefix* of W, denoted by $W' \preceq W$, if either $W' = W$, or else W' is finite and there exists a timed execution fragment W'' such that $W' \cdot W'' = W$. Relation $\preceq$ is a partial ordering on timed execution fragments.

The admissible timed execution fragments are those in which time passes without bound. Since (we believe) time does pass without bound in the real world, it is reasonable to restrict attention to the admissible timed executions when arguing the correctness of a system represented as a timed automaton. In this paper, we focus on the admissible and finite timed executions, and mostly ignore Zeno timed executions. We denote by $t\text{-}frag^*(A)$, $t\text{-}frag^\infty(A)$ and $t\text{-}frag(A)$ the sets of finite, admissible and all timed execution fragments of A. Similarly, we denote by $t\text{-}execs^*(A)$, $t\text{-}execs^\infty(A)$ and $t\text{-}execs(A)$ the sets of finite, admissible and all timed executions of A.

The notion of admissibility is the *only* notion of liveness that we include in our model. Many untimed automaton models (e.g., [40, 46, 31]) include facilities for describing rich classes of liveness properties, for example, various notions of fairness. In the timed setting, it is often possible to replace liveness notions with corresponding timing restrictions. These can be expressed by restrictions on time-passage steps, so they do not require any special machinery. The notion of admissibility is in some sense more tractable mathematically than some other liveness notions, e.g., the notion of a "fair execution" in the I/O automaton model [40]. This is because the admissible timed executions of a timed automaton can be expressed as the limits of infinite sequences of finite timed executions.

Proposition 2.4 *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 .ltime values approach ∞.*

The characterization in Proposition 2.4 permits the reduction of questions about infinite behaviors to questions about their finite prefixes. A similar reduction is not possible in untimed models that incorporate fairness.

One could extend the timed automaton model presented here by adding other liveness properties. Such an extended model is defined, and its properties explored, in [32, 58, 16]. In [32, 58], the extended model is also applied to substantial communication examples.

Zeno timed executions are a technical anomaly; they represent an infinite amount of activity occurring in a finite amount of time, which is (we believe) impossible in reality. Nevertheless, our definition of timed automata does admit Zeno executions. There are two types of Zeno timed executions in our model:

1. those containing infinitely many discrete actions, but for which *.ltime* is finite, and

2. those containing finitely many discrete actions, but for which the domain of the final trajectory is a right-open interval with a finite supremum.

For this second type of Zeno timed execution, the "infinite amount of activity occurring in a finite amount of time" corresponds to an infinite number of time-passage steps needed to span the final interval.

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). Our model does allow time deadlocks. However, in several of our theorems we will require that the timed automata be "feasible": a timed automaton is *feasible* provided that each finite timed execution is a t-prefix of some admissible timed execution.[2] A feasible timed automaton does not have time deadlocks, but it will have Zeno timed executions, simply because each feasible timed execution has t-prefixes that are Zeno timed excutions.

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 notion of external behavior for timed automata, because they do not reflect the invisible nature of time-passage actions (see Example 1.1 in the introduction). In this subsection, we define a new notion of external behavior for timed automata, which we call *timed traces*. These do not include explicit time-passage events, but do include information about the real time of visible events, as well as the final time up to which the observation is made.

We first define the auxiliary technical notion of a *timed sequence pair*, a general data type that is used in the definition of a timed trace.

2.3.1 Timed Sequence Pairs

Let K be any set with $K \cap \mathsf{R}^+ = \emptyset$. Then a *timed sequence* over 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., if (k,t) and (k',t') are consecutive elements in δ then $t \leq t'$. 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 than or equal to the limit of the time components in δ, and equal to this limit if δ is an infinite sequence. We write $p.seq$ and $p.ltime$ for the two respective components of p, and 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 with p finite. Then define $p \cdot p'$ to be the timed sequence pair $(p.seq\ \delta, p.ltime + p'.ltime)$, where δ is the modification of $p'.seq$ obtained

[2]This property is called *nonZenoness* in [2].

by adding $p.ltime$ to all the time components. If p and q are timed sequence pairs over K, then p is a *prefix* of q, denoted by $p \leq q$, if either $p = q$, or p is finite and there exists a timed sequence pair p' such that $p \cdot p' = q$. Relation $\leq$ is a partial ordering on the set of timed sequence pairs over K.

We describe how to translate from a sequence over $K \cup \mathsf{R}^+$ to a timed sequence pair over K and vice versa. First, if β is any sequence over $K \cup \mathsf{R}^+$, then we define the *time of occurrence* of any K-element in β to be the sum of all the reals that precede that element in β. We also define $\beta.ltime$ to be the sum of all the reals in β. In case β is the empty sequence, we define $\beta.ltime = 0$. Finally, we define $t\text{-}trace(\beta)$ to be the timed sequence pair $(\delta, \beta.ltime)$, where δ is the subsequence of β consisting of all the elements of K, each paired with its time of occurrence.

Conversely, if p is a timed sequence pair over K, then we define $trace(p)$, a corresponding sequence over $K \cup \mathsf{R}^+$. Namely, if $p.ltime$ is finite or $p.seq$ is infinite, then let $trace(p)$ be the unique sequence β over $K \cup \mathsf{R}^+$ such that $p = t\text{-}trace(\beta)$ and such that β does not contain two consecutive elements of R^+. On the other hand, if $p.ltime$ is infinite and $p.seq$ finite, then let $trace(p)$ be the unique sequence β over $K \cup \mathsf{R}^+$ such that $p = t\text{-}trace(\beta)$, such that β does not contain two consecutive elements of R^+ prior to the last K element, and such that the portion of β after the last K element is the default sequence $1\,1\,1\,\cdots$.

Thus by construction:

Lemma 2.5 *For any timed sequence pair p over K, $t\text{-}trace(trace(p)) = p$.*

Let β be a sequence over $K \cup \mathsf{R}^+$. Then we say that β is *admissible* if the sum of the positive reals in β is infinite.

Lemma 2.6 *β is admissible if and only if $t\text{-}trace(\beta)$ is admissible.*

It is not the case that β is finite if and only if $t\text{-}trace(\beta)$ is finite. A counterexample is provided by the infinite sequence $\frac{1}{2}\,\frac{1}{4}\,\frac{1}{8}\,\cdots$, of which the associated timed sequence pair $(\lambda, 1)$ is finite. (Recall that λ is the empty sequence.)

2.3.2 Timed Traces of Timed Automata

Suppose that $W = w_0 a_1 w_1 a_2 w_2 \cdots$ is a timed execution fragment of a timed automaton A. For each a_i, define the *time of occurrence* t_i to be $\Sigma_{j<i} w_j.ltime$, i.e., the sum of the lengths of all the trajectory intervals preceding a_i in W. Let $\delta = (a_1, t_1)(a_2, t_2)\cdots$ be the sequence consisting of the actions in W paired with their times of occurrence. Then $t\text{-}trace(W)$, the *timed trace* of W, is defined to be the pair[3]

$$t\text{-}trace(W) \;\triangleq\; (\delta\lceil(vis(A) \times \mathsf{R}^{\geq 0}), W.ltime).$$

Thus, $t\text{-}trace(W)$ records the occurrences of visible actions together with their times of occurrence, as well as the last time. Note that neither internal actions nor time-passage actions appear explicitly in the timed trace of W.

[3] Recall from Part I that the symbol $\lceil$ denotes the projection of a sequence on a subset of the domain of its elements.

Lemma 2.7 *If W is a timed execution fragment of A then $t\text{-}trace(W)$ is a timed sequence pair over $vis(A)$.*

Lemma 2.8 *If $W = W_1 \cdot W_2$ is a timed execution fragment of A then $t\text{-}trace(W) = t\text{-}trace(W_1) \cdot t\text{-}trace(W_2)$.*

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

Lemma 2.9 *The sets $t\text{-}traces^*(A)$ and $t\text{-}traces^\infty(A)$ consist of finite timed sequence pairs and admissible timed sequence pairs over $vis(A)$, respectively.*

These notions induce three natural preorders on timed automata. Namely, we define $A \leq^{\mathrm{t}}_{\mathrm{T}} B$ to mean that $t\text{-}traces(A) \subseteq t\text{-}traces(B)$, $A \leq^{\mathrm{t}}_{*\mathrm{T}} B$ to mean that $t\text{-}traces^*(A) \subseteq t\text{-}traces^*(B)$, and $A \leq^{\mathrm{t}}_{\infty\mathrm{T}} B$ to mean that $t\text{-}traces^\infty(A) \subseteq t\text{-}traces^\infty(B)$. 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

We include in this section one last definition, which is used in all the simulation definitions in Section 5.

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 W with $W.\mathit{fstate} = s'$, $t\text{-}trace(W) = p$ and $W.\mathit{lstate} = s$.

Lemma 2.10 *Suppose p, p_1 and p_2 are timed sequence pairs over $vis(A)$ and $p = p_1 \cdot p_2$.*

1. *If $s' \stackrel{p_1}{\leadsto}_A s''$ and $s'' \stackrel{p_2}{\leadsto}_A s$ then $s' \stackrel{p}{\leadsto}_A s$.*

2. *If $s' \stackrel{p}{\leadsto}_A s$ then there exists s'' such that $s' \stackrel{p_1}{\leadsto}_A s''$ and $s'' \stackrel{p_2}{\leadsto}_A s$.*

2.4 Relating Timed and Untimed Execution Fragments

In this subsection, we present some close connections between the timed execution fragments and the (ordinary) execution fragments of a timed automaton. Roughly speaking, an execution fragment can be regarded as "sampling" the state information in a timed execution fragment at a countable number of points in time. This close correspondence allows techniques for reasoning about ordinary execution fragments to be used for timed execution fragments (and vice versa).

2.4.1 Execution Fragments of Timed Automata

Suppose that α is an (ordinary) execution fragment of timed automaton A. We may define various timing notions for α simply, as follows.

$$\begin{aligned} t\text{-}trace(\alpha) &\triangleq t\text{-}trace(trace(\alpha)) \\ \alpha.ltime &\triangleq trace(\alpha).ltime \end{aligned}$$

As in Part I, α is defined to be *finite* if it is a finite sequence. We define α to be *admissible* if $\alpha.ltime = \infty$, and Zeno if it is neither finite nor admissible.

2.4.2 Sampling

To see the connections between the timing notions defined for (ordinary) executions and the corresponding ones for timed executions, we define a notion of "sampling".

Let $\alpha = s_0 a_1 s_1 \ldots$ be an execution fragment of A and $W = w_0 b_1 w_1 \ldots$ be a timed execution fragment of A. We define two auxiliary functions: f gives for each index i of α the number of discrete actions that precede s_i, and g gives for each index i of α the amount of time between s_i and the last discrete action preceding s_i. Formally, for all i,

$$f(0) = 0, \qquad f(i+1) = \begin{cases} f(i)+1 & \text{if } a_{i+1} \text{ discrete,} \\ f(i) & \text{otherwise.} \end{cases}$$

$$g(0) = 0, \qquad g(i+1) = \begin{cases} 0 & \text{if } a_{i+1} \text{ discrete,} \\ g(i) + a_{i+1} & \text{otherwise.} \end{cases}$$

We say that α *samples* W provided that the following conditions are satisfied.

1. f is a surjective mapping from indices of α to indices of W.
2. For all i, $s_i = w_{f(i)}(g(i))$.
3. For all $i > 0$ with a_i discrete, $a_i = b_{f(i)}$ and $g(i-1) = w_{f(i-1)}.ltime$.
4. $\alpha.ltime = W.ltime$.
5. α is finite if and only if W is finite.

The function f maps each state s_i in α to the trajectory of W to which it belongs. The first condition states that for each trajectory of W there should be at least one state of α that belongs to it. The second condition specifies how function g determines the position of s_i within the associated trajectory. The third condition guarantees that the discrete actions match up, and that the amount of idling in between discrete actions is the same for α and W. The last two conditions ensure that things match up properly at the end of the executions. The definition immediately implies that if α samples W then α is admissible if and only if W is admissible, and α is Zeno if and only if W is Zeno.

The following two lemmas show the close relationship between timed execution fragments and ordinary execution fragments. Note that these connections hold for finite, admissible and Zeno (timed) executions. The proofs are routine; the proof of Lemma 2.11 uses Lemmas 2.1 and 2.2.

Lemma 2.11 *If α is an execution fragment of A then there is a timed execution fragment W of A such that α samples W.*

Lemma 2.12 *If W is a timed execution fragment of A then there is an execution fragment α of A such that α samples W.*

Finally, we relate the definition of timed traces for execution fragments to the corresponding definition for timed execution fragments.

Lemma 2.13 *If α samples W then $t\text{-}trace(\alpha) = t\text{-}trace(W)$.*

3 Restricted Kinds of Timed Automata

In this section, paralleling our development in Part I, we define certain restricted kinds of timed automata that are useful in our proofs. Recall that in Part I, we defined what it meant for an untimed automaton to be *deterministic*, to have *finite invisible nondeterminism (fin)* and to be a *forest*. Now we define analogous notions of *t-deterministic*, *t-fin* and *t-forest*.

First, we say that timed automaton 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 $s' \overset{p}{\leadsto}_A s$. It turns out that this notion is equivalent to the original notion of determinism:

Lemma 3.1 *Timed automaton A is t-deterministic if and only if it is deterministic.*

Proof: Recall that the definition of determinism says that $|start(A)| = 1$ and that for any state s' and finite sequence β of actions in $ext(A)$, there is at most one state s such that $s' \overset{\beta}{\Longrightarrow} s$.

$\Rightarrow$: We suppose that A is t-deterministic and show that it is deterministic. The start condition is immediate. Suppose for the sake of contradiction that A is not deterministic; then there exist s', β, s_1 and s_2 such that $s' \overset{\beta}{\Longrightarrow} s_1$, $s' \overset{\beta}{\Longrightarrow} s_2$ and $s_1 \neq s_2$. This means that there are two execution fragments, α_1 and α_2, each starting with s' and having trace β, one of which ends in s_1 and the other in s_2. Then Lemma 2.11 implies that there are two timed execution fragments, W_1 and W_2, that are sampled by α_1 and α_2 respectively. By Lemma 2.13, W_1 and W_2 have the same timed trace, say p. It follows that $s' \overset{p}{\leadsto} s_1$ and $s' \overset{p}{\leadsto} s_2$, which violates t-determinism, yielding the needed contradiction.

$\Leftarrow$: We suppose that A is deterministic and show that it is t-deterministic. The start condition is immediate. Suppose for the sake of contradiction that A is not t-deterministic; then there exist s', p, s_1 and s_2 such that $s' \overset{p}{\leadsto} s_1$, $s' \overset{p}{\leadsto} s_2$ and $s_1 \neq s_2$. This means that there are two timed execution fragments, W_1 and W_2, each starting with s' and having timed trace p, one of which ends in s_1 and the other in s_2. Then Lemma 2.12 implies that there are two execution fragments, α_1 and α_2, that sample W_1 and W_2 respectively. By Lemma 2.13, α_1 and α_2 have the same timed trace, say p. By applying axiom **S2** to split time-passage actions, we may assume without loss of generality that α_1 and α_2 have the same trace, say β. It follows that $s' \overset{\beta}{\Longrightarrow} s_1$ and $s' \overset{\beta}{\Longrightarrow} s_2$, which violates determinism, yielding the needed contradiction. ∎

A simple characterization of t-determinism is then obtained from Lemma 3.1 and a characterization of determinism in Part I:

Lemma 3.2 *A timed automaton A is t-deterministic if and only if $|start(A)| = 1$, every τ transition is of the form (s, τ, s) for some s, and for any state s' and any action (either visible, internal or time-passage) a there is at most one state s such that $s' \xrightarrow{a} s$.*

Second, 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 $s' \overset{p}{\leadsto}_A s$. It is not hard to see that the analogous result to Lemma 3.1 for t-fin fails:

Example 3.3 Let A be the timed automaton with no visible actions that can do τ actions at any time and remembers the times at which it has done these internal actions. The states of A consist of components $now \in \mathsf{R}^{\geq 0}$, initially 0, and $tau\text{-}times \subseteq \mathsf{R}^{\geq 0}$, initially empty. The allowed steps are:

- $s' \xrightarrow{\tau} s$, where $s.now = s'.now$ and $s.tau\text{-}times = s'.tau\text{-}times \cup \{s'.now\}$, plus
- $s' \xrightarrow{d} s$, where $s.now = s'.now + d$ and $s.tau\text{-}times = s'.tau\text{-}times$.

Then A has fin but does not have t-fin.

Third and finally, we say that A is a *t-forest* if every state s has a unique timed execution W that leads to it, i.e., such that $W.lstate = s$. In the case of timed automata, the original definition of a forest is trivial: no timed automaton that contains a time-passage step can be a forest. This is because if a state s is reached by an execution that ends with a time-passage step, then axiom **S2** allows that time-passage step to be split in two, yielding a different execution leading to s. We can obtain a characterization of t-forests, analogous to the characterization in Part I for forests:

Lemma 3.4 *A timed automaton A is a t-forest if and only if 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 and a' are distinct time-passage actions, and either $r \xrightarrow{a-a'} r'$ or $r' \xrightarrow{a'-a} r$ (depending on whether $a > a'$ or $a' > a$).*

Proof: $\Rightarrow$: All states in a t-forest are reachable by Lemma 2.3. It is also easy to see that start states have no incoming steps. So suppose that $r \xrightarrow{a} s$ and $r' \xrightarrow{a'} s$, with $(r, a) \neq (r', a')$. Let W and W' be the unique timed executions leading to r and r', respectively.

We extend W to timed execution W_1 by adding the information contained in the step $r \xrightarrow{a} s$. Specifically, if a is a discrete action, we append a and a trivial trajectory with the single state s to W. On the other hand, if $a \in \mathsf{R}^+$, we use axiom **S2** to obtain a trajectory w for the step $r \xrightarrow{a} s$ and combine w with the final trajectory of W; Lemma 2.1 implies that the combination of the two trajectories is itself a trajectory. Likewise, we extend W' to timed execution W'_1 by adding the information contained in the step $r' \xrightarrow{a'} s$.

Since A is a t-forest and W_1 and W'_1 both lead to s, it must be that $W_1 = W'_1$. But since $(r, a) \neq (r', a')$, the only way this can happen is if a and a' are both time-passage actions and $a \neq a'$. In this case, the final trajectory w of $W_1 = W'_1$ ends with a trajectory of the step $r \xrightarrow{a} s$, and also ends with a trajectory of the step $r' \xrightarrow{a'} s$. In particular, if $w.ltime = t$, then $w(t - a') = r'$ and $w(t - a) = r$.

If $a < a'$, then $t - a' < t - a$, so the definition of a trajectory implies that $r' \xrightarrow{(t-a)-(t-a')} r$, i.e., $r' \xrightarrow{a'-a} r$. Symmetrically, if $a' < a$, we have $r \xrightarrow{a-a'} r'$. Either situation suffices.

$\Leftarrow$: Because all states of A are reachable, we know by Lemma 2.3 that for each state s there is at least one timed execution that leads to it. We show uniqueness. For any timed execution W, define $n(W)$ to be the sum of the number of nontrivial trajectories and the number of actions occurring in W. It suffices to prove the following claim for all $k \in \mathsf{N}$:

> If W and W' are two timed executions with $n(W) + n(W') \leq k$, and if W and W' lead to the same state s, then $W = W'$.

We prove this claim by induction on k.

Basis: $k = 0$.
Then each of W and W' consists of a trivial trajectory with the single state s, so $W = W'$.

Inductive step: $k > 0$.
If W consists of a single trivial trajectory, then s must be a start state. The fact that W' leads to s implies that the start state s has an incoming step, which is a contradiction. A similar contradiction is reached if W' consists of a single trivial trajectory. Thus, neither W nor W' consists of a single trivial trajectory.

If the last trajectory w of W is trivial, define a to be the last discrete action in W, and r the last state of the preceding trajectory. Thus, we have $r \xrightarrow{a} s$. Since each state can have at most one incoming discrete step, the last trajectory of W' must also be trivial, a must be the last discrete action in W', and r the last state of the preceding trajectory of W'. If W_1 and W_1' are the timed executions obtained from W and W', respectively, by omitting the $a\,w$ fragment at the end, the induction hypothesis gives $W_1 = W_1'$. This implies $W = W'$.

A similar proof can be given for case in which the last trajectory of W' is trivial. Thus we may assume that neither W nor W' end with a trivial trajectory.

Define $r = w(0)$ and $a = w.ltime$; the definition of a trajectory implies $r \xrightarrow{a} s$. Likewise, define r', a' and w' for W'.

If $a = a'$, then it is easy to prove that $w = w'$. In this case, let W_1 and W_1' be the results of removing the last trajectory w from W and W', respectively, replacing it with the trivial trajectory with state r. Application of the induction hypothesis gives $W_1 = W_1'$, and this implies $W = W'$.

Assume without loss of generality that $a' > a$. Since $r \xrightarrow{a} s$ and $r' \xrightarrow{a'} s$, we have by assumption $r' \xrightarrow{a'-a} r$. That is, both timed executions end with nontrivial trajectories, and W ends with the shorter one.

We claim that $w(a-t) = w'(a'-t)$ for all $t \in [0, a]$. For if not, then there are two distinct time-passage steps leading to s with the same amount of time-passage, namely, $w(a-t) \xrightarrow{t} s$ and $w'(a'-t) \xrightarrow{t} s$. In particular, $r = w(0) = w'(a'-a)$.

Now let W_1 be the result of removing the last trajectory w from W, replacing it with the trivial trajectory with state r. Also, let W_1' be the result of reducing the last trajectory w' of W' by removing the portion with domain $(a'-a, a']$. Then W_1 and W_1' are two timed executions, each of which leads to r, and such that $n(W_1) + n(W_1')$ is strictly less than $n(W) + n(W')$. By induction hypothesis, $W_1 = W_1'$. Since the removed portions of W and W' are identical, this implies that $W = W'$. ■

We define the relation $t\text{-}after(A)$ to consist of those pairs (p, s) for which there is a finite timed execution of A with timed trace p and last state s.

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

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

Lemma 3.5

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

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

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

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

Continuing the analogy with Part I, we define "timed trace properties", the structures that we consider as external behaviors for timed automata. We also prove some basic properties of timed trace properties and some lemmas relating timed trace properties to timed automata.

A set of timed sequence pairs is *prefix-closed* if, whenever a timed sequence pair is in the set all its prefixes (as defined in Section 2.3.1) are also. A *timed trace property* P is a pair (K, L) where K is a set 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.

A timed trace property 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 such that 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 $P \leq^{\mathrm{t}}_{\infty\mathrm{T}} Q \Leftrightarrow P \leq^{\mathrm{t}}_{\mathrm{T}} Q$.*

The *timed behavior* of a timed automaton A, $t\text{-}beh(A)$, is defined by

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

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: Part 1 follows directly from Lemma 2.9. Parts 3 and 4 are immediate from the definitions.

We sketch the proof of 2; it is analogous to that of Lemma 2.5 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 A.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 $p_{i+1} = p_i \cdot q$. Using Lemma 2.10, it is not difficult to show that G satisfies the hypotheses of Lemma A.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, B_1 and B_2, which are in a sense the timed analogues of the automata A_1 and A_2 of Example 2.1 of Part I. The example illustrates the necessity of the t-fin condition in Proposition 4.4(1). Timed automaton B_1 performs an a-action at each integer time. Each state of B_1 has components $now \in \mathsf{R}^{\geq 0}$ and $count \in \mathsf{N}$, both initially 0. B_1 has a single visible action a, and steps

- $s' \xrightarrow{d} s$, where $s.now = s'.now + d \leq s'.count$ and $s.count = s'.count$;
- $s' \xrightarrow{a} s$, where $s.now = s'.now = s'.count$ and $s.count = s'.count + 1$.

Timed automaton B_2 performs an a-action at each of finitely many integer times. Each state of B_2 has components $now \in \mathsf{R}^{\geq 0}$, initially 0, $count \in \mathsf{N}$, initially 0, and $total \in \mathsf{N}$, initially arbitrary. B_1 has a single visible action a and steps

- $s' \xrightarrow{d} s$, where $s.now = s'.now + d \leq s'.count$, $s.count = s'.count$, and $s.total = s'.total$;
- $s' \xrightarrow{a} s$, where $s.now = s'.now = s'.count \leq s'.total$, $s.count = s'.count{+}1$, and $s.total = s'.total$.

Then it is easy to see that B_1 has t-fin (in fact, it is t-deterministic). However, B_2 does not have t-fin: for instance, it has infinitely many start states. Also, in each finite timed trace of B_2, a occurs at every nonnegative integer time up to (and possibly including) the last time $total$, while in the unique admissible timed trace of B_1, a occurs at all nonnegative integer times. Then B_2 has the same finite timed traces as B_1 but no admissible timed traces. It follows that $B_1 \leq^{\mathrm{t}}_{*\mathrm{T}} B_2$ but $B_1 \not\leq^{\mathrm{t}}_{\mathrm{T}} B_2$.

Note that it is possible to modify B_2 so that it is feasible, yet still demonstrates the same point. Simply allow time to pass in B_2 after the last permitted a output.

Example 4.6 In order to see that the feasibility condition in Proposition 4.4(2) is needed, we consider a timed automaton Z with states drawn from the interval $[0,1)$, start state 0, no visible actions, and steps of the form $t' \xrightarrow{t-t'} t$ whenever $t' < t$. Since Z has no admissible timed traces, it is trivially the case that $Z \leq^{\mathrm{t}}_{\infty\mathrm{T}} B_1$. However, because B_1 does not allow initial time-passage steps, $Z \not\leq^{\mathrm{t}}_{\mathrm{T}} B_1$.

Again paralleling Part I, 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:

- $states(A) = t\text{-}traces^*(P)$,
- $start(A) = \{(\lambda, 0)\}$,
- $acts(A) = sort(P) \cup \{\tau\} \cup \mathsf{R}^+$, and
- for $p', p \in states(A)$ and $a \in acts(A)$,

$$p' \xrightarrow{a}_A p \quad \Leftrightarrow \quad a \neq \tau \wedge p' \cdot t\text{-}trace(a) = p.$$

It is not hard to check that $t\text{-}can(P)$ is in fact 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. *$P \leq^{\mathrm{t}}_{\mathrm{T}} t\text{-}beh(t\text{-}can(P))$.*
4. *If P is limit-closed then $t\text{-}beh(t\text{-}can(P)) \equiv^{\mathrm{t}}_{\mathrm{T}} P$.*
5. *If P is feasible then $t\text{-}can(P)$ is feasible.*

Proof: Part 1 follows easily using Lemmas 3.2 and 3.4. Part 2 follow from the definitions. Since $t\text{-}can(P)$ is t-deterministic it has t-fin, so it follows by Lemma 4.3 that $t\text{-}beh(t\text{-}can(P))$ is limit-closed. Now 3 and 4 follow by combination of 2 and Lemma 4.1. Part 5 is straightforward from the definitions. ∎

Lemma 4.8

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

Proof: By combining Lemmas 4.3 and 4.7. ∎

5 Simulations for Timed Automata

So far, we have presented the timed automaton model and its basic properties. In this section, we define simulation proof methods for timed automata. The properties of these relations are shown in the following two sections. In the definitions below, we require that an a step is simulated by a move $t\text{-}trace(\hat{a})$. This means that a τ step is simulated by the timed sequence pair $(\lambda, 0)$, a visible action a is simulated by the timed sequence pair $((a, 0), 0)$, and a time-passage step d is simulated by the timed sequence pair (λ, d).

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

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

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

A *timed backward simulation* from A to B is a total[4] relation b over $states(A)$ and $states(B)$ that satisfies:

1. If $s \in start(A)$ then $b[s] \subseteq start(B)$.

[4] For the definitions of "total", $\mathbf{N}()$, $\mathbf{P}()$, $()^{-1}$, etc., we refer the reader to Appendix A of Part I.

2. If $s' \xrightarrow{a}_A s$ and $u \in b[s]$, then there exists a state $u' \in b[s']$ such that $u' \overset{p}{\leadsto}_B u$, where $p = t\text{-}trace(\hat{a})$.

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

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

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

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

For each of the above simulations, we will refer to the first condition in the definition as the *start condition*, and to the second condition as the *transfer condition*.

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.

Analogously to Part I, we write $A \leq_{\mathrm{R}}^{\mathrm{t}} B$, $A \leq_{\mathrm{F}}^{\mathrm{t}} B$, etc., to indicate that there is a timed refinement, timed forward simulation, etc., from A to B.

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

We close this section with a technical lemma. The transfer condition of each simulation definition is stated for individual steps of A. It is straightforward to deduce a similar condition for moves rather than steps.

Lemma 5.1 *Suppose that A and B are timed automata and $s' \overset{p}{\leadsto}_A s$.*

1. *If r is a timed refinement from A to B then $r(s') \overset{p}{\leadsto}_B r(s)$.*

2. *If f is a timed forward simulation from A to B and $u' \in f[s']$, then there exists a state $u \in f[s]$ such that $u' \overset{p}{\leadsto}_B u$.*

3. *If b is a timed backward simulation from A to B and $u \in b[s]$, then there exists a state $u' \in b[s']$ such that $u' \overset{p}{\leadsto}_B u$.*

4. *If g is a timed forward-backward simulation from A to B and $S' \in g[s']$, then there exists a set $S \in g[s]$ such that for every $u \in S$ there exists $u' \in S'$ with $u' \overset{p}{\leadsto}_B u$.*

5. *If g is a timed backward-forward simulation from A to B and $S \in g[s]$, then there exists a set $S' \in g[s']$ such that for every $u' \in S'$ there exists $u \in S$ with $u' \overset{p}{\leadsto}_B u$.*

Proof: Let W be a timed execution fragment from A such that $s' = W.fstate$, $s = W.lstate$, and $p = t\text{-}trace(W)$. All parts are proved by induction on $k = n(W)$, where, as in the proof of Lemma 3.4, $n(W)$ is the sum of the number of nontrivial trajectories and the number of discrete actions occurring in W. As an example, we prove the result for timed refinements; the other cases are similar.

Basis: $k = 0$.

Then $s' = s$, W consists of the trivial trajectory containing the single state s, and $p = (\lambda, 0)$. Since $r(s) \overset{(\lambda,0)}{\leadsto}_B r(s)$, we have $r(s') \overset{p}{\leadsto}_B r(s)$.

Basis: $k = 1$.

This case follows easily from the transfer condition in the definition of a timed refinement.

Inductive step: $k > 1$.

Then W can be written as $W_1 \cdot W_2$, where $n(W_1) = k - 1$ and $n(W_2) = 1$. Let s'' denote $W_1.lstate$ $(= W_2.fstate)$. Let $p_1 = t\text{-}trace(W_1)$ and $p_2 = t\text{-}trace(W_2)$. Then $s' \overset{p_1}{\leadsto}_A s''$ and $s'' \overset{p_2}{\leadsto}_A s$. By inductive hypothesis, $r(s') \overset{p_1}{\leadsto}_B r(s'')$ and $r(s'') \overset{p_2}{\leadsto}_B r(s)$. By Lemma 2.8, $p = p_1 \cdot p_2$. Then Lemma 2.10(1) implies that $r(s') \overset{p}{\leadsto}_B r(s)$. ■

6 Timed Results from Untimed Results

In this and the next section we give soundness and completeness results for the various simulations defined in Section 5, as well as implication results among them. The distinction between the results in this section and those in Section 7 is that the ones given here are all derived from corresponding results for the untimed case. The statements of the results in Section 7 are also analogous to results of Part I, but these timed results are not derived from the untimed results, for instance because they require the construction of an intermediate timed automaton.

Most of the results in this section are presented in the form of a diagram, Figure 1. This is the same diagram that appears in Part I for the untimed setting, except for the t superscripts.

The machinery needed to prove the results in this section is developed in Section 6.1. In particular, we define an untimed automaton called the *closure automaton*, $cl(A)$, for every timed automaton A. We then show close correspondences between A and $cl(A)$, involving both external behavior notions and simulation relations. These correspondences allow us to derive the results in Section 6.2 from the corresponding results for untimed automata.

6.1 The Closure Automaton

In this section, we define the *closure* of a timed automaton, the basic technical device that we will used to derive results about timed automata from corresponding results about untimed automata. Section 6.1.1 contains the definition, Section 6.1.2 gives the relationships between timed traces of a timed automaton and traces of its closure, and Section 6.1.3 gives the relationships between timed simulations between timed automata and simulations between their closures.

6.1.1 Definition

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

- $states(B) = states(A)$,
- $start(B) = start(A)$,
- $acts(B) = acts(A)$, and
- $steps(B)$ consists of $steps(A)$ together with all steps $s' \xrightarrow{d}_B s$, such that $s' \overset{(\lambda,d)}{\rightsquigarrow}_A s$.

Thus, the closure construction augments A by adding new time-passage steps to short-circuit the effects of any number of τ and time-passage actions of A.

Proposition 6.1 *$cl(A)$ is a timed automaton.*

6.1.2 Relating Timed and Untimed Traces

In this section, we describe some close connections between A and $cl(A)$. We begin with a preliminary lemma showing the relationship between moves of A and of $cl(A)$.

Lemma 6.2 *Suppose s' and s are states of A.*

1. *If β is a finite sequence of actions in $ext(A)$ then*

$$s' \overset{\beta}{\Longrightarrow}_{cl(A)} s \text{ if and only if } s' \overset{t\text{-}trace(\beta)}{\rightsquigarrow}_A s.$$

2. *If p is a finite timed sequence pair over $vis(A)$ then*

$$s' \overset{trace(p)}{\Longrightarrow}_{cl(A)} s \text{ if and only if } s' \overset{p}{\rightsquigarrow}_A s.$$

Proof: Part 1 is straightforward. Part 2 follows from part 1 and Lemma 2.5. ∎

From this we can prove:

Lemma 6.3

1. *If β is a finite sequence of actions in $ext(A)$ then*

$$\beta \in traces^*(cl(A)) \text{ if and only if } t\text{-}trace(\beta) \in t\text{-}traces^*(A).$$

2. *If p is a finite timed sequence pair over $vis(A)$ then*

$$trace(p) \in traces^*(cl(A)) \text{ if and only if } p \in t\text{-}traces^*(A).$$

Proof: We show Part 1. Suppose that β is a finite sequence of actions in $ext(A)$, and let $p = t\text{-}trace(\beta)$.

$\Rightarrow$: Suppose that $\beta \in traces^*(cl(A))$. Then there exist $s' \in start(cl(A))$ and $s \in states(cl(A))$ such that $s' \stackrel{\beta}{\Longrightarrow}_{cl(A)} s$. Then Lemma 6.2 implies that $s' \stackrel{p}{\leadsto}_A s$. This implies that $p \in t\text{-}traces^*(A)$.

$\Leftarrow$: Suppose that $p \in t\text{-}traces^*(A)$. Then there exist $s' \in start(A)$ and $s \in states(A)$ such that $s' \stackrel{p}{\leadsto}_A s$. Then Lemma 6.2 implies that $s' \stackrel{\beta}{\Longrightarrow}_{cl(A)} s$. This implies that $\beta \in traces^*(cl(A))$.

Part 2 follows from part 1 and Lemma 2.5. ■

A similar result holds for admissible sequences:

Lemma 6.4

1. *If β is an admissible sequence of actions in $ext(A)$ then*

$$\beta \in traces^\omega(cl(A)) \textit{ if and only if } t\text{-}trace(\beta) \in t\text{-}traces^\infty(A).$$

2. *If p is an admissible timed sequence pair over $vis(A)$ then*

$$trace(p) \in traces^\omega(cl(A)) \textit{ if and only if } p \in t\text{-}traces^\infty(A).$$

We now show that t-determinism of A corresponds to determinism of $cl(A)$, and likewise for t-fin and fin.

Lemma 6.5

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 first prove part 1:

$\Rightarrow$: Suppose A is t-deterministic. Then, by Lemma 3.2, all τ steps of A are of the form $s \stackrel{\tau}{\longrightarrow} s$. But this means that $cl(A)$ and A are identical. And thus both A and $cl(A)$ are deterministic by Lemma 3.1.

$\Leftarrow$: Suppose $cl(A)$ is deterministic. Then all τ steps of $cl(A)$ are of the form $s \stackrel{\tau}{\longrightarrow} s$. But since $cl(A)$ is obtained from A by adding time-passage steps only, also all τ steps of A are of the form $s \stackrel{\tau}{\longrightarrow} s$. This again implies that $cl(A)$ and A are identical. And thus both A and $cl(A)$ are t-deterministic by Lemma 3.1.

Next we prove part 2:

$\Rightarrow$: Suppose A has t-fin. Then $start(A)$ is finite and hence $start(cl(A))$ is finite. Suppose s' is a state of $cl(A)$ and β is a finite sequence over $ext(cl(A))$. We show that the set $S = \{s \mid s' \stackrel{\beta}{\Longrightarrow}_{cl(A)} s\}$ is finite. Suppose $s \in S$. Then Lemma 6.2 implies that $s \in U$, where $U = \{u \mid s' \stackrel{t\text{-}trace(\beta)}{\leadsto}_A u\}$. Thus $S \subseteq U$. Since A has t-fin, U is finite. Thus S is finite, as required.

$\Leftarrow$: Suppose that $cl(A)$ has fin. Then $start(cl(A))$ is finite and hence $start(A)$ is finite. Suppose s' is a state of A and p is a finite timed sequence pair over $vis(A)$. We show that the set $S = \{s \mid s' \stackrel{p}{\leadsto}_A s\}$ is finite. Suppose $s \in S$. Then Lemma 6.2 implies that $s \in U$, where $U = \{u \mid s' \overset{trace(p)}{\Longrightarrow}_{cl(A)} u\}$. Since $cl(A)$ has fin, U is finite. Thus S is finite, as required. ∎

Now we relate finite timed trace inclusion for timed automata to ordinary finite trace inclusion for their closure automata.

Lemma 6.6 *$A \leq^{\mathrm{t}}_{*\mathrm{T}} B \Leftrightarrow cl(A) \leq_{*\mathrm{T}} cl(B)$.*

Proof:
$\Rightarrow$: Suppose that $\beta \in traces^*(cl(A))$. Then Lemma 6.3 implies that $p \in t\text{-}traces^*(A)$, where $p = t\text{-}trace(\beta)$. The hypothesis then implies that also $p \in t\text{-}traces^*(B)$. Again by Lemma 6.3, we have $\beta \in traces^*(cl(B))$.
$\Leftarrow$: Suppose that $p \in t\text{-}traces^*(A)$. Then Lemma 6.3 implies that $\beta \in traces^*(cl(A))$, where $\beta = trace(p)$. The hypothesis then implies that also $\beta \in traces^*(cl(B))$. Again by Lemma 6.3, we have $p \in t\text{-}traces^*(B)$. ∎

We can also obtain a one-way relationship between general timed trace inclusion for timed automaton and general trace inclusion for their closure automata.

Lemma 6.7 *If $cl(A) \leq_{\mathrm{T}} cl(B)$ then $A \leq^{\mathrm{t}}_{\mathrm{T}} B$.*

Proof: Suppose $cl(A) \leq_{\mathrm{T}} cl(B)$. Then certainly $cl(A) \leq_{*\mathrm{T}} cl(B)$, so by Lemma 6.6, $A \leq^{\mathrm{t}}_{*\mathrm{T}} B$. It remains to show that $A \leq^{\mathrm{t}}_{\infty\mathrm{T}} B$. For this, suppose that $p \in t\text{-}traces^{\infty}(A)$. Then Lemma 6.4 implies that $\beta \in traces^{\omega}(cl(A))$, where $\beta = trace(p)$. The hypothesis then implies that $\beta \in traces^{\omega}(cl(B))$. Again by Lemma 6.4, we have $p \in t\text{-}traces^{\infty}(B)$. ∎

> **Example 6.8** The converse of Lemma 6.7 does not hold in general. For a counterexample, let B be a timed automaton that nondeterministically chooses a positive natural number n, then performs action a at times $1-2^{-1}$, $1-2^{-2}$,..., $1-2^{-n}$, and then idles forever, allowing time to pass. Since each finite timed execution can be extended to an admissible one, B is feasible; since it has infinitely many start states B has 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 performs 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 which only a Zeno execution, and no admissible execution, is possible. Timed automata A and B have the same timed traces, but $cl(A)$ also 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 the converse of Lemma 6.7 does hold if B has t-fin.

Lemma 6.9 *Suppose B has t-fin. Then $cl(A) \leq_{\mathrm{T}} cl(B) \Leftrightarrow A \leq^{\mathrm{t}}_{\mathrm{T}} B$.*

Proof:
$$\begin{array}{lll} cl(A) \leq_{\mathrm{T}} cl(B) & \Leftrightarrow & \text{(by Lemma 6.5, and Proposition 2.6 of Part I)} \\ cl(A) \leq_{*\mathrm{T}} cl(B) & \Leftrightarrow & \text{(by Lemma 6.6)} \\ A \leq^{\mathrm{t}}_{*\mathrm{T}} B & \Leftrightarrow & \text{(by Proposition 4.4)} \\ A \leq^{\mathrm{t}}_{\mathrm{T}} B & & \end{array}$$
∎

Finally, we obtain a corollary that relates timed trace inclusions for timed automata to simulations for their closures.

Corollary 6.10 *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 to each other and to the three statements above.

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

Proof:

$$\begin{array}{lll} A \leq^{\mathrm{t}}_{*\mathrm{T}} B & \Leftrightarrow & \text{(by Lemma 6.6)} \\ cl(A) \leq_{*\mathrm{T}} cl(B) & \Leftrightarrow & \text{(by Theorems 4.5 and 4.6 of Part I)} \\ cl(A) \leq_{\mathrm{FB}} cl(B) & \Leftrightarrow & \text{(by Proposition 4.10 of Part I)} \\ cl(A) \leq_{\mathrm{BF}} cl(B) & & \end{array}$$

If B has t-fin then

$$\begin{array}{lll} A \leq^{\mathrm{t}}_{*\mathrm{T}} B & \Rightarrow & \text{(by Lemma 6.6)} \\ cl(A) \leq_{*\mathrm{T}} cl(B) & \Rightarrow & \text{(by Lemma 6.5, and Theorem 4.6 of Part I)} \\ cl(A) \leq_{\mathrm{iFB}} cl(B) & \Rightarrow & \text{(by Theorem 4.5 of Part I)} \\ cl(A) \leq_{\mathrm{T}} cl(B) & \Rightarrow & \text{(by Lemma 6.7)} \\ A \leq^{\mathrm{t}}_{\mathrm{T}} B & \Rightarrow & A \leq^{\mathrm{t}}_{*\mathrm{T}} B \end{array}$$

■

Corollary 6.10 already provides one method for proving that the finite timed traces of a timed automaton A are included among those of another timed automaton B: produce an ordinary forward-backward or a backward-forward simulation from $cl(A)$ to $cl(B)$. Of course, any simpler type of simulation from Part I, such as a forward or backward simulation, will do as well. Similarly, Corollary 6.10 provides a method for proving that all the timed traces of A are included among those of B, in case B has t-fin.

This approach is analogous to that followed for Milner's CCS [49] 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 [38], where the problem of showing inclusion 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.

However, this is not the approach we emphasize in this paper. Instead, we will use the closure automata as a technical device to help us prove soundness, completeness and implication results for the new timed simulations defined in Section 5. For this, we proceed in the next subsection to relate timed simulations to corresponding untimed simulations for closure automata.

6.1.3 Relating Timed and Untimed Simulations

In Section 6.1.2, we showed that (under certain finiteness conditions) inclusion of timed traces for timed automata is equivalent to inclusion of ordinary traces for the closures of these automata. Now we demonstrate strong relationships between between timed simulations for timed automata, and ordinary simulations for the closures of these automata.

Lemma 6.11 *A relation from $states(A)$ to $states(B)$ is a timed refinement from A to B if and only if it is a refinement from $cl(A)$ to $cl(B)$. Moreover, the same correspondence also holds for forward simulations, backward simulations, forward-backward simulations, backward-forward simulations, history relations and prophecy relations.*

Proof: We prove the prove the result for refinements.

$\Rightarrow$: Suppose that r is a timed refinement from A to B. We show that r is a refinement from $cl(A)$ to $cl(B)$. The start condition carries over immediately; we consider the step condition. Suppose that $s' \xrightarrow{a}_{cl(A)} s$. Then $s' \overset{\hat{a}}{\Longrightarrow}_{cl(A)} s$ and so Lemma 6.2 implies that $s' \overset{p}{\leadsto}_A s$, where $p = t\text{-}trace(\hat{a})$. Since r is a timed refinement, Lemma 5.1 implies that $r(s') \overset{p}{\leadsto}_B r(s)$. Then Lemma 6.2 implies that $r(s') \overset{trace(p)}{\Longrightarrow}_{cl(B)} r(s)$. But case analysis based on whether a is a visible, internal or time-passage action shows that $trace(p) = \hat{a}$, so this is as needed.

$\Leftarrow$: Suppose that r is a refinement from $cl(A)$ to $cl(B)$. We show that r is a timed refinement from A to B. The start condition carries over immediately; we consider the step condition. Suppose that $s' \xrightarrow{a}_A s$. Then $s' \xrightarrow{a}_{cl(A)} s$, by definition of $cl(A)$. Since r is a refinement, we have that $r(s') \overset{\hat{a}}{\Longrightarrow}_{cl(B)} r(s)$. Then Lemma 6.2 implies that $r(s') \overset{p}{\leadsto}_B r(s)$, where $p = t\text{-}trace(\hat{a})$, as needed.

The proofs for forward, backward, forward-backward and backward-forward simulations are entirely analogous, using the appropriate parts of Lemma 5.1. The results for history and prophecy relations follow from those for forward simulations, backward simulations and refinements. ∎

Therefore, we have:

Corollary 6.12 *Suppose X represents any of $\{R, F, B, iB, FB, iFB, BF, iBF, H, P, iP\}$. Then $A \leq^{\mathrm{t}}_{\mathrm{X}} B$ if and only if $cl(A) \leq_{\mathrm{X}} cl(B)$.*

Proposition 6.13 *The relations $\leq^{\mathrm{t}}_{\mathrm{R}}$, $\leq^{\mathrm{t}}_{\mathrm{F}}$, $\leq^{\mathrm{t}}_{\mathrm{B}}$, $\leq^{\mathrm{t}}_{\mathrm{iB}}$, $\leq^{\mathrm{t}}_{\mathrm{FB}}$, $\leq^{\mathrm{t}}_{\mathrm{iFB}}$, $\leq^{\mathrm{t}}_{\mathrm{BF}}$, $\leq^{\mathrm{t}}_{\mathrm{H}}$, $\leq^{\mathrm{t}}_{\mathrm{P}}$ and $\leq^{\mathrm{t}}_{\mathrm{iP}}$ are all preorders. (However, $\leq^{\mathrm{t}}_{\mathrm{iBF}}$ is not a preorder.)*

Proof: This follows from Corollary 6.12, since the corresponding untimed simulations are preorders. The same counterexample that we used to show that $\leq_{\mathrm{iBF}}$ is not a preorder (the automata A_{11} and A_{12} of Example 4.11 in Part I), can be used to show that $\leq^{\mathrm{t}}_{\mathrm{iBF}}$ is not a preorder. One can turn the automata from this counterexample into feasible timed automata via the *patient* construction of [41]. This construction introduces arbitrary time delays at each state by simply attaching, for each d, steps $s \xrightarrow{d} s$ to each state s. ∎

6.2 Soundness and Implication Results for Timed Automaton Simulation Relations

In this section, we give those results about timed automata that follow from corresponding results about untimed automata, using the results in the previous two sections. We present most of these results in a single theorem, which is entirely analogous to a classification given in Section 7 of Part I.

Theorem 6.14 *Suppose* $M, N \in \{T, *T, R, F, (i)B, (i)FB, (i)BF, H, (i)P\}$, *where the* (i) *indicates that* i *is optional.*

1. *If there is a path from* $\leq^{\mathrm{t}}_{\mathrm{M}}$ *to* $\leq^{\mathrm{t}}_{\mathrm{N}}$ *in Figure 1 consisting of thin arrows only, and if* $A \leq^{\mathrm{t}}_{\mathrm{M}} B$, *then* $A \leq^{\mathrm{t}}_{\mathrm{N}} B$.
2. *If there is a path from* $\leq^{\mathrm{t}}_{\mathrm{M}}$ *to* $\leq^{\mathrm{t}}_{\mathrm{N}}$ *consisting of thin and/or thick arrows, if* $A \leq^{\mathrm{t}}_{\mathrm{M}} B$ *and if* B *has t-fin, then* $A \leq^{\mathrm{t}}_{\mathrm{N}} B$.

$$
\begin{tikzcd}
 & & \leq^{\mathrm{t}}_{\mathrm{iP}} \arrow[r] \arrow[d] & \leq^{\mathrm{t}}_{\mathrm{P}} \arrow[d] \\
 & \leq^{\mathrm{t}}_{\mathrm{R}} \arrow[r] \arrow[d] & \leq^{\mathrm{t}}_{\mathrm{iB}} \arrow[r] \arrow[d] & \leq^{\mathrm{t}}_{\mathrm{B}} \arrow[d] \\
\leq^{\mathrm{t}}_{\mathrm{H}} \arrow[r] & \leq^{\mathrm{t}}_{\mathrm{F}} \arrow[r] & \leq^{\mathrm{t}}_{\mathrm{iBF}} \arrow[r] \arrow[d] & \leq^{\mathrm{t}}_{\mathrm{BF}} \arrow[d, shift right] \\
 & & \leq^{\mathrm{t}}_{\mathrm{iFB}} \arrow[r] \arrow[d, shift right] & \leq^{\mathrm{t}}_{\mathrm{FB}} \arrow[u, shift right] \arrow[d, shift right] \\
 & & \leq^{\mathrm{t}}_{\mathrm{T}} \arrow[u, shift right, line width=1.5pt] \arrow[r, shift right] & \leq^{\mathrm{t}}_{\mathrm{*T}} \arrow[u, shift right] \arrow[l, shift right, line width=1.5pt]
\end{tikzcd}
$$

Figure 1: Classification of basic relations between timed automata

Proof: Note that Figure 1 is identical to Figure 6 of Part I, which gives an overview of the relationships in the untimed case, except for the superscripts t. It is enough to prove:

1. If there is a thin arrow from $\leq^{\mathrm{t}}_{\mathrm{M}}$ to $\leq^{\mathrm{t}}_{\mathrm{N}}$ and if $A \leq^{\mathrm{t}}_{\mathrm{M}} B$, then $A \leq^{\mathrm{t}}_{\mathrm{N}} B$.
2. If there is a thick arrow from $\leq^{\mathrm{t}}_{\mathrm{M}}$ to $\leq^{\mathrm{t}}_{\mathrm{N}}$, if $A \leq^{\mathrm{t}}_{\mathrm{M}} B$ and if B has t-fin, then $A \leq^{\mathrm{t}}_{\mathrm{N}} B$.

For part 1, suppose that there is a thin arrow from $\leq^{\mathrm{t}}_{\mathrm{M}}$ to $\leq^{\mathrm{t}}_{\mathrm{N}}$ and that $A \leq^{\mathrm{t}}_{\mathrm{M}} B$. If $\{\mathrm{M}, \mathrm{N}\} \cap \{\mathrm{T}, *\mathrm{T}\} = \emptyset$, then Corollary 6.12 implies that $cl(A) \leq_{\mathrm{M}} cl(B)$. Then the corresponding result for the untimed case implies that $cl(A) \leq_{\mathrm{N}} cl(B)$, which implies by Corollary 6.12 that $A \leq^{\mathrm{t}}_{\mathrm{N}} B$, as needed. There are four remaining thin arrows to consider.

1. $\mathrm{M} = \mathrm{iFB}$ and $\mathrm{N} = \mathrm{T}$. Corollary 6.12 implies that $cl(A) \leq_{\mathrm{iFB}} cl(B)$. The untimed result implies that $cl(A) \leq_{\mathrm{T}} cl(B)$, which implies by Lemma 6.7 that $A \leq^{\mathrm{t}}_{\mathrm{T}} B$.

2. $\mathrm{M} = \mathrm{T}$ and $\mathrm{N} = *\mathrm{T}$. This is immediate from the definitions.

3. $\mathrm{M} = *\mathrm{T}$ and $\mathrm{N} = \mathrm{FB}$. Corollary 6.10 implies that $cl(A) \leq_{\mathrm{FB}} cl(B)$, which implies by Corollary 6.12 that $A \leq^{\mathrm{t}}_{\mathrm{FB}} B$.

4. $\mathrm{M} = \mathrm{FB}$ and $\mathrm{N} = *\mathrm{T}$. Corollary 6.12 implies that $cl(A) \leq_{\mathrm{FB}} cl(B)$, which implies by Corollary 6.10 that $A \leq^{\mathrm{t}}_{*\mathrm{T}} B$.

For part 2, suppose that there is a thick arrow from $\leq^{\mathrm{t}}_{\mathrm{M}}$ to $\leq^{\mathrm{t}}_{\mathrm{N}}$, that $A \leq^{\mathrm{t}}_{\mathrm{M}} B$ and that B has t-fin. There are only two thick arrows to consider:

1. $\mathrm{M} = *\mathrm{T}$ and $\mathrm{N} = \mathrm{T}$. This follows from Proposition 4.4.

2. $\mathrm{M} = \mathrm{T}$ and $\mathrm{N} = \mathrm{iFB}$. Corollary 6.10 implies that $cl(A) \leq_{\mathrm{iFB}} cl(B)$, which implies by Corollary 6.12 that $A \leq^{\mathrm{t}}_{\mathrm{iFB}} B$.

■

In order to show that all the inclusions are strict, one can use essentially the same counterexamples as in the untimed setting. Again one can turn these untimed counterexamples into feasible timed automata via the *patient* construction of [41], i.e., by introducing arbitrary time delays at each state by attaching, for each d, steps $s \xrightarrow{d} s$ to each state s.

We close this section with three more results that are derived from the analogous results for the untimed case using the correspondences.

Theorem 6.15 *(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 6.5(1), $cl(B)$ is deterministic, and by Lemma 6.6, $cl(A) \leq_{*\mathrm{T}} cl(B)$. Thus by the partial completeness result for forward simulations (Theorem 3.11, Part I), $cl(A) \leq_{\mathrm{F}} cl(B)$. Then Corollary 6.12 allows us to conclude that $A \leq^{\mathrm{t}}_{\mathrm{F}} B$, as required. ■

Proposition 6.16 *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: Lemma 6.2 implies that all states of $cl(A)$ are reachable, Lemma 6.5 implies that $cl(B)$ is deterministic, and Corollary 6.12 implies that $cl(A) \leq_{\mathrm{B}} cl(B)$. By Proposition 3.19 of Part I, the untimed version of the fact we are proving, $cl(A) \leq_{\mathrm{R}} cl(B)$. Then Corollary 6.12 allows us to conclude that $A \leq^{\mathrm{t}}_{\mathrm{R}} B$, as required. ■

Proposition 6.17 *Suppose all states of A are reachable, B has t-fin and $A \leq^{\mathrm{t}}_{\mathrm{B}} B$. Then $A \leq^{\mathrm{t}}_{\mathrm{iB}} B$.*

Proof: Similar to the proof of Proposition 6.16. ■

7 Remaining Results for Timed Automata

In Section 6, we showed how some simple correspondences enable most of the results for untimed automata to be extended to timed automata. In this section, we consider what happens to all the other results of Part I. We begin with the results about untimed automata that do not extend in this way but are nonetheless true. In Section 7.1 we present partial completeness results that involve t-forests. These do not carry over using the correspondences because the closure of a t-forest need not be a forest: in a t-forest (and hence also in its closure) a state may have multiple incoming time-passage steps, something which is not allowed in a forest. In Sections 7.2 and 7.3, we present results that assert the existence of timed automata with particular properties, including the completeness results for the combination of timed forward and timed backward simulations and the Abadi-Lamport completeness result. We prove all of these results directly for timed automata. In most cases, the proof is analogous to the corresponding proof in Part I. Finally, in Section 7.4, we demonstrate that the one remaining result of Part I, Proposition 3.12, is not true in the timed setting.

7.1 Partial Completeness Results for t-Forests

Theorem 7.1 *(Partial completeness of timed refinements) Suppose A is a t-forest, B is t-deterministic and $A \leq_{*\mathrm{T}}^{\mathrm{t}} B$. Then $A \leq_{\mathrm{R}}^{\mathrm{t}} B$.*

Proof: Analogous to the proof of Theorem 3.5 in Part I. Define $r \triangleq \textit{t-after}(B) \circ \textit{t-past}(A)$. Lemma 3.5 and the fact that $\textit{t-traces}^*(A) \subseteq \textit{t-traces}^*(B)$ together imply that r is a function from $states(A)$ to $states(B)$. We claim that r is a timed refinement from A to B.

The start condition is straightforward.

For the transfer condition, suppose that $s' \xrightarrow{a}_A s$. Let $p = \textit{t-trace}(\hat{a})$; then $s' \overset{p}{\leadsto}_A s$. We must show that $r(s') \overset{p}{\leadsto}_B r(s)$. Since A is a forest, there exist timed traces q' and q leading to s' and s respectively. Lemma 2.10 implies that $q' \cdot p$ leads from a start state of A to s. Since A is a forest and q and $q' \cdot p$ both lead to s, it must be that $q' \cdot p = q$.

By definition of r, we have we have $u_0 \overset{q}{\leadsto}_B r(s)$ for some start state u_0 of B. Then Lemma 2.10 implies that there is a state u of B such that $u_0 \overset{q'}{\leadsto}_B u$ and $u \overset{p}{\leadsto}_B r(s)$. Since q' leads from a start state of A to s', the definition of r then implies that $u = r(s')$. Thus, $r(s') \overset{p}{\leadsto}_B r(s)$, as needed. ■

Theorem 7.2 *(Partial completeness of timed backward simulations) Suppose A is a t-forest and $A \leq_{*\mathrm{T}}^{\mathrm{t}} B$. Then*

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

Proof: Analogous to the proof of Theorem 3.18 in Part I. We define a relation b over $states(A)$ and $states(B)$. For a given state s of A, Lemma 3.5 implies that there is a unique timed trace leading to s, say p. Define

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

Lemma 3.5 and the fact that $\textit{t-traces}^*(A) \subseteq \textit{t-traces}^*(B)$ implies that relation b is total. The start condition follows as in the proof of Theorem 3.18 in Part I.

For the transfer condition, suppose that $s' \xrightarrow{a}_A s$, $u \in b[s]$, and $p = \textit{t-trace}(\hat{a})$; then $s' \overset{p}{\leadsto}_A s$. We define $u' \in b[s']$ such that $u' \overset{p}{\leadsto}_B u$. As in the proof of Proposition 7.1, we obtain timed traces q' and q leading to s' and s respectively, and conclude that $q' \cdot p = q$. Since $u \in b[s]$, we have $u_0 \overset{q}{\leadsto}_B u$ for some start state u_0 of B. Then Lemma 2.10 implies that there is a state u' of B such that $u_0 \overset{q'}{\leadsto}_B u'$ and $u' \overset{p}{\leadsto}_B u$. Moreover, it is possible to select u' in a 'minimal' way so that there is an execution from u_0 to u' with timed trace q' that does not end with a τ step. Since q' leads from a start state of A to s', the definition of b implies that $u' \in b[s']$. This suffices.

Lemma 3.5 implies that if B has t-fin then relation b is image-finite. ■

7.2 Combined Timed Forward and Backward Simulations

In this subsection, we give the completeness results for the combination of timed forward and timed backward simulations. In order to prove these results, we use variants of the classic subset construction from automata theory, and a variant of the dual historization construction of Klarlund and Schneider [29].

The *backward power* of a timed automaton A, notation $\textit{b-power}(A)$, is the automaton B defined by

- $\textit{states}(B) = \mathbf{N}(\textit{states}(A))$,
- $\textit{start}(B) = \mathbf{N}(\textit{start}(A))$,
- $\textit{acts}(B) = \textit{acts}(A)$, and
- for $S', S \in \textit{states}(B)$ and $a \in \textit{acts}(B)$,

$$S' \xrightarrow{a}_B S \;\Leftrightarrow\; \forall s \in S\ \exists s' \in S' : s' \overset{\textit{t-trace}(\hat{a})}{\leadsto}_A s.$$

The *finitary backward power* of A, notation $\textit{fin-b-power}(A)$, is defined in exactly the same way, except that instead of all non-empty subsets of $\textit{states}(A)$ and $\textit{start}(A)$ only the finite non-empty subsets are used. The *forward power* or *historization* of A, notation $\textit{f-power}(A)$, is the automaton F defined by

- $\textit{states}(F) = \mathbf{P}(\textit{states}(A))$,
- $\textit{start}(F) = \{S \subseteq \textit{states}(A) \mid S \cap \textit{start}(A) \neq \emptyset\}$,
- $\textit{acts}(F) = \textit{acts}(A)$, and

- for $S', S \in states(F)$ and $a \in acts(F)$,

$$S' \xrightarrow{a}_F S \quad \Leftrightarrow \quad \forall s' \in S' \; \exists s \in S : s' \overset{t\text{-}trace(\hat{a})}{\leadsto}_A s.$$

Lemma 7.3 *Suppose $B = b\text{-}power(A)$, $I = fin\text{-}b\text{-}power(A)$ and $F = f\text{-}power(A)$. Then B, I and F are timed automata and*

1. $A \leq^{\mathrm{t}}_{\mathrm{R}} B$ *and* $B \leq^{\mathrm{t}}_{\mathrm{B}} A$,
2. $A \leq^{\mathrm{t}}_{\mathrm{R}} I$ *and* $I \leq^{\mathrm{t}}_{\mathrm{iB}} A$,
3. $A \leq^{\mathrm{t}}_{\mathrm{R}} F$ *and* $F \leq^{\mathrm{t}}_{\mathrm{F}} A$.

Proof: First we show that B satisfies axioms **S1** and **S2**. For **S1**, suppose that $S' \xrightarrow{d}_B S''$ and $S'' \xrightarrow{d'}_B S$. Then

$$\forall s'' \in S'' \; \exists s' \in S' : s' \overset{(\lambda,d)}{\leadsto}_A s'', \text{ and}$$

$$\forall s \in S \; \exists s'' \in S'' : s'' \overset{(\lambda,d')}{\leadsto}_A s.$$

It follows, using Lemma 2.10, that

$$\forall s \in S \; \exists s' \in S' : s' \overset{(\lambda,d+d')}{\leadsto}_A s,$$

i.e., that $S' \xrightarrow{d+d'}_B S$, as needed for **S1**.

For **S2**, suppose that $S' \xrightarrow{d}_B S$. Define $w : [0,d] \rightarrow states(B)$ as follows: let $w(0) = S'$, $w(d) = S$, and for any t, $0 < t < d$, let $w(t) = \{u \in states(A) \mid \exists s' \in S' : s' \overset{(\lambda,t)}{\leadsto}_A u\}$. Suppose $0 \leq t_1 < t_2 \leq d$; we must show that $w(t_1) \xrightarrow{t_2-t_1}_B w(t_2)$. There are three nontrivial cases:

1. $0 = t_1 < t_2 < d$.
   We must show that $S' \xrightarrow{t_2}_B w(t_2)$, that is, that

   $$\forall u \in w(t_2) \; \exists s' \in S' : s' \overset{(\lambda,t_2)}{\leadsto}_A u.$$

   But this is immediate from the definition of $w(t_2)$.

2. $0 < t_1 < t_2 = d$.
   We must show that $w(t_1) \xrightarrow{d-t_1}_B S$, that is, that

   $$\forall s \in S \; \exists u \in w(t_1) : u \overset{(\lambda,d-t_1)}{\leadsto}_A s.$$

   So suppose that $s \in S$. Since $S' \xrightarrow{d}_B S$, there exists a state $s' \in S'$ such that $s' \overset{(\lambda,d)}{\leadsto}_A s$. Then Lemma 2.10 implies that there exists u such that $s' \overset{(\lambda,t_1)}{\leadsto}_A u$ and $u \overset{(\lambda,d-t_1)}{\leadsto}_A s$. This u satisfies all our requirements.

3. $0 < t_1 < t_2 < d$.
   The argument to similar to that for case 2.

The mapping that relates to each state s of A the state $\{s\}$ of B is a timed refinement from A to B; hence $A \leq_{\mathrm{R}}^{\mathrm{t}} B$. The mapping that relates each state S of B to all its elements is a timed backward simulation from B to A; hence $B \leq_{\mathrm{B}}^{\mathrm{t}} A$.

The proofs for I and F are similar to those for B, except for the proof that I satisfies axiom **S2**. Suppose that $S' \xrightarrow{d}_I S$. Then there exists, for each $s \in S$, a finite timed execution fragment W_s of A with $W_s.\mathit{fstate} \in S'$, $t\text{-}trace(W_s) = (\lambda, d)$ and $W_s.\mathit{lstate} = s$. Define $w : [0, d] \to \mathit{states}(I)$ as follows: let $w(0) = S'$, $w(d) = S$, and for any t, $0 < t < d$, let $w(t)$ be the finite set which, for each $s \in S$, contains the last state of the shortest prefix of W_s with limit time t. Then it is routine to prove that w is a trajectory for $S' \xrightarrow{d}_I S$. ■

Theorem 7.4

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: The proof of the implications "$\Leftarrow$" is easy. We sketch the proof of "$\Rightarrow$" in 3 and 4. The proofs of "$\Rightarrow$" in 1 and 2 are similar.

Let g be a timed backward-forward simulation from A to B, which is image finite if $A \leq_{\mathrm{iBF}}^{\mathrm{t}} B$. Let $C = f\text{-}power(B)$. Then it is straightforward to check that g is also a timed backward simulation from A to C (and is image-finite if $A \leq_{\mathrm{iBF}}^{\mathrm{t}} B$). Moreover, Lemma 7.3 gives $C \leq_{\mathrm{F}}^{\mathrm{t}} B$. ■

It is interesting to note the difference between the above proof of Theorem 7.4 and the corresponding proofs of Theorems 4.1 and 4.8 in Part I. In those proofs the intermediate automata are "smaller" than the power constructions that we use here, since as states they only contain those sets of states of B that are in the range of g. It is not possible to use the constructions from Part I here because in general the resulting automata do not satisfy the trajectory axiom **S2**. However, we could have used the power constructions in Part I as well. In fact, one can even argue that in some sense this would have been less ad-hoc.

Theorem 7.5 *(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 Theorems 6.14 and 7.4.

Parts 1 and 2 can alternatively be shown using a proof analogous to that of Theorem 3.22 of Part I. Let $C = t\text{-}can(t\text{-}beh(A))$. 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 6.15), and because C is a t-forest, $C \leq^{\mathrm{t}}_{\mathrm{B}} B$ follows by partial completeness of timed backward simulations (Theorem 7.2(1)). Similarly, if B has t-fin then $C \leq^{\mathrm{t}}_{\mathrm{iB}} B$ follows by Theorem 7.2(2). ■

7.3 Timed History and Prophecy Relations

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

7.3.1 Timed History Relations

We begin with a timed analogue to the unfolding construction of Part I.

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

- $\textit{states}(B) = \textit{t-execs}^*(A)$,
- $\textit{start}(B) = [0,0] \rightarrow \textit{start}(A)$,
- $\textit{acts}(B) = \textit{acts}(A)$, and
- for $W', W \in \textit{states}(B)$, $d \in \mathsf{R}^+$ and $a \in \textit{acts}(B) - \mathsf{R}^+$,

$$W' \xrightarrow{d}_B W \quad \Leftrightarrow \quad \exists w : W' \cdot w = W \wedge w.\textit{ltime} = d$$
$$W' \xrightarrow{a}_B W \quad \Leftrightarrow \quad W' \, a \, w' = W,$$

  where w' is the trivial trajectory that maps 0 to $W.\textit{lstate}$.

We leave it to the reader to verify that $\textit{t-unfold}(A)$ is a timed automaton.

Proposition 7.6 *$\textit{t-unfold}(A)$ is a t-forest and $A \leq^{\mathrm{t}}_{\mathrm{H}} \textit{t-unfold}(A)$.*

Proof: Using Lemma 3.4 it follows easily that $\textit{t-unfold}(A)$ is a t-forest. The function $.\textit{lstate}$, which maps each finite timed execution of A to its last state, is a timed refinement from $\textit{t-unfold}(A)$ to A, and the relation $.\textit{lstate}^{-1}$ is a timed forward simulation from A to $\textit{t-unfold}(A)$. Thus, $.\textit{lstate}^{-1}$ is a timed history relation from A to $\textit{t-unfold}(A)$. ■

We are now in a position to prove a timed version of Sistla's [57] completeness result.

Theorem 7.7 *(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 5.6 in Part I; choose $C = \textit{t-unfold}(A)$. ■

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

- $states(C) = R$,
- $start(C) = R \cap (start(A) \times start(B))$,
- $acts(C) = acts(A) \cap acts(B)$, and
- for $(s', u'), (s, u) \in states(C)$ and $a \in acts(C)$,

$$(s', u') \xrightarrow{a}_C (s, u) \quad \Leftrightarrow \quad s' \overset{p}{\rightsquigarrow}_A s \wedge u' \overset{p}{\rightsquigarrow}_B u, \text{ where } p = t\text{-}trace(\hat{a}).$$

Again we leave it to the reader to check that $t\text{-}sup(A, B, R)$ is a timed automaton.

Theorem 7.8 $A \leq_{\mathrm{F}}^{\mathrm{t}} B \Leftrightarrow (\exists C : A \leq_{\mathrm{H}}^{\mathrm{t}} C \leq_{\mathrm{R}}^{\mathrm{t}} B)$.

Proof: Suppose $A \leq_{\mathrm{F}}^{\mathrm{t}} B$. Let f be a timed forward simulation from A to B, let $C = t\text{-}sup(A, B, f)$ 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. ■

7.3.2 Timed Prophecy Relations

Finally, we describe the additional 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 $t\text{-}guess(A)$, is the timed automaton B defined by

- $states(B) = t\text{-}frag^*(A)$,
- $start(B) = t\text{-}execs^*(A)$,
- $acts(B) = acts(A)$, and
- for $W', W \in states(B)$, $d \in \mathsf{R}^+$ and $a \in acts(B) - \mathsf{R}^+$,

$$W' \xrightarrow{d}_B W \quad \Leftrightarrow \quad \exists w : W' = w \cdot W \wedge w.ltime = d$$
$$W' \xrightarrow{a}_B W \quad \Leftrightarrow \quad W' = w'aW,$$

  where w' is the trivial trajectory that maps 0 to $W'.fstate$.

As before, we leave it to the reader to verify that $t\text{-}guess(A)$ is a timed automaton.

Proposition 7.9 $A \leq_{\mathrm{P}}^{\mathrm{t}} t\text{-}guess(A)$.

Proof: Similar to the proof of Proposition 7.6. ■

Theorem 7.10

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

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 7.8, using timed backward simulations instead of timed forward simulations. ■

We finish this subsection with a dual version of Sistla's completeness result [57] and variants of the completeness results of Abadi and Lamport [1].

Theorem 7.11 *(Completeness of timed prophecy relations and timed forward simulations)* $A \leq^{\mathrm{t}}_{*\mathrm{T}} B \Rightarrow \exists C : A \leq^{\mathrm{t}}_{\mathrm{P}} C \leq^{\mathrm{t}}_{\mathrm{F}} B$.

Proof: Analogous to the proof of Theorem 5.17 in Part I. ■

Theorem 7.12 *(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: Analogous to the proofs of Theorems 5.18 and 5.19 in Part I. ■

7.4 A Result That Does Not Carry Over

Proposition 3.12 of Part I does not carry over to our timed setting, i.e., there exist timed automata A and B such that A is a t-forest and $A \leq^{\mathrm{t}}_{\mathrm{F}} B$ but *not* $A \leq^{\mathrm{t}}_{\mathrm{R}} B$.

Example 7.13 Timed automaton A may perform a single visible action a at any rational time, and then stops. Timed automaton B may only perform a single action a at integer times. However, whereas A measures time with a 'perfect clock', B measures time with a clock that may run either too slow or too fast, in an arbitrary fashion. The set of states of A is $\mathsf{R}^{\geq 0} \times \{\mathsf{T}, \mathsf{F}\}$, with $(0, \mathsf{T})$ the initial state, and there are steps

- $(t, \mathsf{T}) \xrightarrow{d} (t + d, \mathsf{T})$, for each $t \in \mathsf{R}^{\geq 0}$ and $d \in \mathsf{R}^{+}$;
- $(t, \mathsf{T}) \xrightarrow{a} (t, \mathsf{F})$, for each $t \in \mathsf{Q}^{\geq 0}$.

The set of states of B is also $\mathsf{R}^{\geq 0} \times \{\mathsf{T}, \mathsf{F}\}$, with $(0, \mathsf{T})$ the initial state. The steps of B are

- $(t, \mathsf{T}) \xrightarrow{d} (t', \mathsf{T})$, for all $t, t' \in \mathsf{R}^{\geq 0}$ with $t < t'$ and all $d \in \mathsf{R}^{+}$;
- $(t, \mathsf{T}) \xrightarrow{a} (t, \mathsf{F})$, for each $t \in \mathsf{N}$.

Using Lemma 3.4 it is easy to see that A is a t-forest. Also, it is easy to check that the relation f given by

$$f \quad \triangleq \quad \{((t, b), (t', b')) \mid t \in \mathsf{R}^{\geq 0},\ t' \in \mathsf{N} \text{ and } b = b'\}$$

is a timed forward simulation from A to B. However, there does not exist a timed refinement from A to B. The proof is by contradiction. Suppose that r is a timed refinement. Then, by the start condition of a timed refinement, r maps the start state $(0, \mathsf{T})$ of A to the start state $(0, \mathsf{T})$ of B. The state $(1, \mathsf{T})$ of A has an outgoing a step, so it must be mapped to a state of B which has also an outgoing a step, i.e., a state (n, T) for some $n \in \mathsf{N}$. Since A has a step $(0, \mathsf{T}) \xrightarrow{1} (1, \mathsf{T})$, but B does not have a step $(0, \mathsf{T}) \xrightarrow{1} (0, \mathsf{T})$, it follows using the transfer condition of a timed refinement that $n > 0$. Let, for $0 \leq i \leq 2n$, s_i be the image under r of state $(\frac{i}{2n}, \mathsf{T})$ of A. By definition of A and by the transfer condition of a timed refinement, $s_i \xrightarrow{1/2n} s_{i+1}$, for all $i < 2n$. Further all s_i must be of the form (m_i, T), for some $m_i \in \mathsf{N}$. By definition of B, this means that $0 = m_0 < m_1 < m_2 < \cdots < m_{2n-1} < m_{2n} = n$. This is a contradiction, as there are only $n-1$ naturals strictly in between 0 and n, and not $2n-1$.

An interesting question (wide open to us) is to come up with some plausible additional axioms for timed automata, such that in the resulting setting all the results on simulations that we proved in Part I of this paper do carry over.

8 Including Invariants

We show how to introduce invariants into the timed simulations, just as we introduced them into the untimed simulations in Section 6 of Part I. An *invariant* of a timed automaton A is defined to be superset of the set of reachable states of A, i.e., a property that is true of all the reachable states of A. Let A and B be timed automata with invariants I_A and I_B, respectively.

A *weak timed refinement* from A to B, with respect to I_A and I_B, is a function r : $states(A) \rightarrow states(B)$ that satisfies:

1. If $s \in start(A)$ then $r(s) \in start(B)$.
2. If $s' \xrightarrow{a}_A s$, $s', s \in I_A$, and $r(s') \in I_B$, then $r(s') \overset{p}{\leadsto}_B r(s)$, where $p = t\text{-}trace(\hat{a})$.

A *weak timed forward simulation* from A to B, with respect to I_A and I_B, is a relation f over $states(A)$ and $states(B)$ that satisfies:

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

A *weak timed backward simulation* from A to B, with respect to I_A and I_B, is a relation b over $states(A)$ and $states(B)$ that satisfies:

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

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

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

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

3. If $s \in I_A$ and $S \in g[s]$ then $S \cap I_B \neq \emptyset$.

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

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

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

3. If $s \in I_A$ then $g[s] \neq \emptyset$.

A relation h over $states(A)$ and $states(B)$ is a *weak timed history relation* from A to B, with respect to I_A and I_B, provided that h is a weak timed forward simulation from A to B, with respect to I_A and I_B, and h^{-1} is a weak timed refinement from B to A, with respect to I_B and I_A.

A relation p over $states(A)$ and $states(B)$ is a *weak timed prophecy relation* from A to B, with respect to I_A and I_B, provided that p is a weak timed backward simulation from A to B, with respect to I_A and I_B, and p^{-1} is a weak timed refinement from B to A, with respect to I_B and I_A.

We write $A \leq^{\mathrm{t}}_{\mathrm{wR}} B$, $A \leq^{\mathrm{t}}_{\mathrm{wF}} B$, $A \leq^{\mathrm{t}}_{\mathrm{wB}} B$, $A \leq^{\mathrm{t}}_{\mathrm{wiB}} B$, $A \leq^{\mathrm{t}}_{\mathrm{wFB}} B$, $A \leq^{\mathrm{t}}_{\mathrm{wiFB}} B$, $A \leq^{\mathrm{t}}_{\mathrm{wBF}} B$, $A \leq^{\mathrm{t}}_{\mathrm{wiBF}} B$, $A \leq^{\mathrm{t}}_{\mathrm{wH}} B$, $A \leq^{\mathrm{t}}_{\mathrm{wP}} B$ and $A \leq^{\mathrm{t}}_{\mathrm{wiP}} B$ to denote the existence of a weak refinement, weak forward simulation, weak backward simulation, weak image-finite backward simulation, etc., from A to B, with respect to some invariants I_A and I_B.

Proposition 8.1 *The relations $\leq^{\mathrm{t}}_{\mathrm{wR}}$, $\leq^{\mathrm{t}}_{\mathrm{wF}}$, $\leq^{\mathrm{t}}_{\mathrm{wB}}$, $\leq^{\mathrm{t}}_{\mathrm{wiB}}$, $\leq^{\mathrm{t}}_{\mathrm{wFB}}$, $\leq^{\mathrm{t}}_{\mathrm{wiFB}}$, $\leq^{\mathrm{t}}_{\mathrm{wBF}}$, $\leq^{\mathrm{t}}_{\mathrm{wH}}$, $\leq^{\mathrm{t}}_{\mathrm{wP}}$ and $\leq^{\mathrm{t}}_{\mathrm{wiP}}$ are all preorders. (However, $\leq^{\mathrm{t}}_{\mathrm{wiBF}}$ is not a preorder.)*

Theorem 8.2 *(Soundness of weak simulations)*

1. *If $A \leq^{\mathrm{t}}_{\mathrm{wR}} B$, $A \leq^{\mathrm{t}}_{\mathrm{wF}} B$, $A \leq^{\mathrm{t}}_{\mathrm{wiB}} B$, $A \leq^{\mathrm{t}}_{\mathrm{wiFB}} B$, $A \leq^{\mathrm{t}}_{\mathrm{wiBF}} B$, $A \leq^{\mathrm{t}}_{\mathrm{wH}} B$, or $A \leq^{\mathrm{t}}_{\mathrm{wiP}} B$, then $A \leq^{\mathrm{t}}_{\mathrm{T}} B$.*

2. *If $A \leq^{\mathrm{t}}_{\mathrm{wB}} B$, $A \leq^{\mathrm{t}}_{\mathrm{wFB}} B$, $A \leq^{\mathrm{t}}_{\mathrm{wBF}} B$, or $A \leq^{\mathrm{t}}_{\mathrm{wP}} B$, then $A \leq^{\mathrm{t}}_{*\mathrm{T}} B$.*

9 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, [44], 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 Proposition 3.12.

The definitions of timed automata and their simulations involve many choices, such as the choice of the basic axioms for time-passage steps, whether non-time-passage steps have nonzero duration or are instantaneous, whether instantaneous time-passage steps are allowed, whether or not automata are required to have finitely many (or countably many) states, whether time-passage should be represented absolutely or incrementally, what the notion of external behavior should be, whether the simulations should require state reachability, etc. Most choices either lead to longer proofs (see for instance an earlier version of this paper [43] in which time-passage was represented absolutely) or do not yield all the properties in this paper.

Our notion of timed automaton is related to the models of Merritt, Modugno and Tuttle [48] and of Lynch and Attiya [38]. However, these models have more structure than ours, since they assume that the system being modelled is describable in terms of a collection of separate tasks, each with associated upper and lower bounds on its speed. Also, the model of [48] includes treatment of liveness, whereas our model does not. The absence of liveness considerations makes our model simpler; moreover, we do not lose much power because many properties of practical interest for timing-based systems can be expressed as safety properties, given the admissibility assumption that time increases without bound (cf. [24]). Lynch and Attiya [38] also extend simulation techniques to timing-based systems. That work, however, only considers forward simulations. The extra task structure of the model of Lynch and Attiya 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 [38] are stated in a setting that has more structure than is really necessary for those theorems.

Lynch and Vaandrager [41] show how a whole class of process algebraic operators can be defined on timed automata using the general notion of action transducers. Bosscher, Polak and Vaandrager [12] define a language of linear hybrid systems, inspired by the work of [5, 8], and provide it with a semantics in terms of timed automata. Our timed automata can also be used to define the semantics of the timed safety automata of Alur and Dill [7, 26]. In the latter model a finite state restriction is used in order to enable the use of effective model-checking methods, something which is of course not possible in our much more general model.

By using our timed automata model as a common semantic basis for several other models for timing-based systems, we get into a situation where we can easily use a variety of formal proof methods, including assertional methods, algebraic methods, and finite-state state exploration ("model-checking") methods. These methods are usable individually or in combination. It remains to further develop the various proof methods for timed automata. In particular, we are interested in extending the methods of process algebra to our timed automaton model. Our paper [41] 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 it remains to provide useful algebraic laws for reasoning about the operators.

Our timed simulations have already been used extensively elsewhere [12, 23, 32, 34, 35, 36, 37, 38, 45, 58, 60] for verification of timed algorithms and systems. More work is needed in applying timed simulations to additional practical verification examples. In particular, nearly all of the examples that have been carried out so far involve refinements, forward simulations and history variables. Only [58, 32] involve backward simulations and combinations of forward and backward simulations.

Finally, although the timed automaton model presented here is very general, it has become clear that there are at least three ways in which it can be extended: to include treatment of liveness properties, to include probabilistic transitions, and to include treatment of hybrid systems, including continuously-communicating components. Some work on integrating liveness into the present model appears in [16], and work on integrating probabilistic transitions appears in [39, 3, 56]. Both liveness and probabilities introduce their own sets of additional proof methods, e.g., temporal logic and Markov analysis. In [12], it has been shown how linear hybrid systems can be defined in terms of our timed automata. It remains to develop the treatment of general hybrid systems, and to integrate all three extensions, with their proof tools, into a sensibly coordinated whole.

Acknowledgements

We thank the referees, Alan Jeffrey, David Griffioen, Albert Meyer, Jeff Sanders, Roberto Segala, Steve Schneider, Jørgen Søgaard-Andersen, Eugene Stark and George Varghese for their valuable criticism and useful comments on this paper and on [44]. 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 Other Axioms for Timed Automata

We consider the relationship between axioms **S2** and **S2**$'$, as defined in Section 2.1. The relationship between the two axioms is also investigated in [28]. Define a *semi-timed automaton* to be a timed automaton, except that it does not have to satisfy **S2**, but only the weaker (and simpler) axiom **S2**$'$. It is immediate from the definition of a trajectory that each timed automaton is semi-timed. In this appendix, we consider the reverse implication.

A.1 Time Determinism

In the original paper [61] of Wang in which the axiom **S2**$'$ is proposed, also the axiom of *time determinacy* is introduced. In our setting this axiom can be formulated as follows:

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

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. We say that a semi-timed automaton is *time deterministic* if it satisfies axiom **TD**. The following theorem is easy to prove.

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

Thus, Wang's axiom **S2**$'$ is equivalent to the trajectory axiom **S2** in a context where the time determinacy axiom **TD** is assumed. In our timed automaton model we do not require the axiom **TD**: we find it unnatural to allow for nondeterminism for discrete actions but not for time-passage actions. As pointed out in [12], time nondeterministic timed automata arise naturally in the semantics of linear hybrid systems, for instance in the modelling of drifting clocks. Also, several of the constructions in this paper, like the *f-power*, *b-power* and *superposition* construction, introduce time nondeterminism.

A.2 Countable Time Domains

One way to obtain equivalence between timed and semi-timed automata is to change the underlying time domain. In this paper, we have chosen elements of the set $\mathsf{R}^{\geq 0}$ of nonnegative real numbers as time-passage actions for timed automata. Instead, we could have proved all our results for automata parametrized with an arbitrary *time domain* as in [27, 53, 28]. A *time domain* $\mathcal{D} = (T, +, 0)$ consists of a set T of *points in time*, equipped with a binary operator $+$ and constant 0 such that, for all $t, u, v \in T$,

T1 $t + 0 = 0 + t = t$

T2 $t + (u + v) = (t + u) + v$

T3 $t + u = t + v \quad \Rightarrow \quad u = v$

T4 $t + u = 0 \quad \Rightarrow \quad t = u = 0$

T5 $u \leq t \wedge v \leq t \quad \Rightarrow \quad u \leq v \vee v \leq u$

where $\leq$ is the *precedence* relation $\leq$ on T defined by $t \leq u \Leftrightarrow \exists v : t + v = u$. Axioms **T1** and **T2** say that $\mathcal{D}$ is a *monoid*. Axiom **T3** states that $\mathcal{D}$ is *left-cancellative*, axiom **T4** that $\mathcal{D}$ is *anti-symmetric*, and axiom **T5** that $\mathcal{D}$ is *locally linear*. It follows from axioms **T1-T4** that $\leq$ is a partial ordering with a unique minimal element 0. Axiom **T3** allows us to define the substraction operator that is required for the trajectory axiom: if $u \leq t$ then $t - u$ is defined to be the unique v with $u + v = t$. Axiom **T5** implies that $\leq$ is total on each interval. This last axiom does not occur in [27, 53, 28], but we fail to have a clear intuition about trajectories without it. Examples of time domains are the nonnegative reals, rationals and integers with addition and 0, but also the sets of finite sequences with concatenation and the empty sequence.

Theorem A.2 *Suppose A is a semi-timed automaton over a countable time domain. Then A is a timed automaton.*

Proof: Suppose that $s' \xrightarrow{d}_A s$. We construct a trajectory w from s' to s. As required, $w(0) = s'$ and $w(d) = s$. Let $t_1, t_2, \ldots$ be some arbitrary enumeration of all the times in the interval $(0, d)$. We define w on elements of this sequence, in order. Let I_n be the set $\{0, d, t_1, \ldots, t_n\}$. We will inductively construct w so that after w has been defined on I_n, we will have that $w(t') \xrightarrow{t-t'} w(t)$ for all $t', t \in I_n$, $t' < t$. This is enough to show that w is a trajectory from s' to s.

So suppose that, for some $n \geq 0$, w has been defined on I_n, and that $w(t') \xrightarrow{t-t'} w(t)$ for all $t', t \in I_n$, $t' < t$. Let u' be the largest time in I_n that is smaller than t_{n+1}, and let u be the smallest time in I_n that is larger than t_{n+1}. By the hypothesis about I_n, we have that $w(u') \xrightarrow{u'-u} w(u)$. Since $u' < t_{n+1} < u$, axiom **S2**$'$ implies that there exists a state s such that $w(u') \xrightarrow{t_{n+1}-u'} s$ and $s \xrightarrow{u-t_{n+1}} w(u)$. Define $w(t_{n+1}) = s$.

We claim that with this definition of $w(t_{n+1})$, we have $w(t') \xrightarrow{t-t'} w(t)$ for all $t', t \in I_{n+1}$, $t' < t$. Since we already know this for $t', t \in I_n$, it is enough to consider the case where one of t', t is equal to t_{n+1}. We give the argument for $t = t_{n+1}$; the argument for $t' = t_{n+1}$ is analogous.

So suppose $t = t_{n+1}$. If $t' = u'$ then we already have the needed claim, $w(u') \xrightarrow{t_{n+1}-u'} w(t_{n+1})$. The other possibility is that $t' < u'$. But then the claim for I_n implies that $w(t') \xrightarrow{u'-t'} w(u')$. Since also $w(u') \xrightarrow{t_{n+1}-u'} w(t_{n+1})$, axiom **S1** implies that $w(t') \xrightarrow{t_{n+1}-t'} w(t_{n+1})$, as needed. ■

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 $\mathsf{R}^{\geq 0}$.

A.3 A Counterexample

At the time we first defined axiom **S2**, we constructed a complex counterexample to show that it was stronger than **S2**$'$. The simpler counterexample described below was subsequently discovered by Steve Schneider.

Theorem A.3 *Let automaton D be defined by*

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

Then 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 **S2** because that would imply, for instance, that the interval $[0, 1]$ of reals can be injectively mapped into the rationals. ■

In the context of the present paper, there is no compelling technical reason why one should use **S2** instead of **S2**$'$. In fact, in an earlier version of this paper ([42]) we have developed a theory of simulations for semi-timed automata. However, we find the theory for semi-timed automata less natural. For instance, the semi-timed automaton D of Theorem A.3 is a t-forest according to the definitions of [42], which is strange since an execution that ends in $(1,1)$ may pass through state $(\frac{1}{2},\frac{1}{3})$ or through state $(\frac{1}{2},\frac{2}{3})$, but not through both. Also, the appealing local characterisation of t-forests of Lemma 3.4 does not hold for t-forests as defined in [42]. Trajectories play a vital role in the theory of hybrid systems [21]. Since we would like to view our timed automata as an underlying semantic domain for both timed and hybrid systems, this provides additional motivation for our choice for the axiom **S2**.

B Glossary of Conventions

a	Actions
b	Backward simulations
c	Choice functions
d	Positive real numbers
f	Forward simulations
g	Forward-backward and backward-forward simulations
h	History relations
i	Indices
k	Symbols
n	Natural numbers
p	Timed sequence pairs and prophecy relations
r	Refinements
s	States
t	Real numbers plus infinity
u	States
w	Trajectories
A, B	Timed automata
G	Digraphs
I	Internals (and also invariants)
K	Sets of symbols
L	Sets of sequences
M, N	Types of timed simulation mappings
P, Q	Timed trace properties
R	Relations
S, U	Sets of states
W	Timed execution fragments
X, Y, Z	Sets
α	Execution fragments
β	Sequences of external actions (traces)
γ	Sequences of actions
δ	Timed sequence
λ	The empty sequence
π	Projections
σ, ρ	Sequences
τ	The internal action

References

[1] M. Abadi and L. Lamport. The existence of refinement mappings. *Theoretical Computer Science*, 82(2):253–284, 1991.

[2] M. Abadi and L. Lamport. An old-fashioned recipe for real time. *ACM Transactions on Programming Languages and Systems*, 16(5):1543–1571, September 1994.

[3] S. Aggarwal. Time optimal self-stabilizing spanning tree algorithms. Master's thesis, MIT Electrical Engineering and Computer Science, May 1994.

[4] R. Alur. *Techniques for Automatic Verification of Real-time Systems*. PhD thesis, Dept. of Computer Science, Stanford University, 1991.

[5] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J.Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

[6] R. Alur, C. Courcoubetis, T.A. Henzinger, and P.-H. Ho. Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems. In Grossman et al. [21], pages 209–229.

[7] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[8] R. Alur, T.A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. In *Proceedings of the 14th Annual IEEE Real-time Systems Symposium*, 1993.

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

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

[11] 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.

[12] D.J.B. Bosscher, I. Polak, and F.W. Vaandrager. Verification of an audio control protocol. In Langmaack et al. [33], pages 170–192. Full version available as Report CS-R9445, CWI, Amsterdam, July 1994.

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

[14] J. Davies and S. Schneider. A brief history of Timed CSP. *Theoretical Computer Science*, 138:243–271, 1995.

[15] J.W. de Bakker, C. Huizing, W.P. de Roever, and G. Rozenberg, editors. *Proceedings REX Workshop on Real-Time: Theory in Practice,* Mook, The Netherlands, June 1991, volume 600 of *Lecture Notes in Computer Science*. Springer-Verlag, 1992.

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

[17] 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.

[18] R.J. van Glabbeek. *Comparative Concurrency Semantics and Refinement of Actions.* PhD thesis, Free University, Amsterdam, 1990.

[19] W.O.D. Griffioen. Proof-checking an audio control protocol with LP. Report CS-R95XX, CWI, Amsterdam, 1995. To appear.

[20] 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, *Protocol Specification, Testing and Verification, X*, Ottawa, Canada pages 261–274, 1990.

[21] R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors. *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.

[22] J.V. Guttag and J.J. Horning. *Larch: Languages and Tools for Formal Specification.* Springer-Verlag, 1993.

[23] C. Heitmeyer and N.A. Lynch. The generalized railroad crossing — a case study in formal verification of real-time systems. In *Proceedings 15th IEEE Real-Time Systems Symposium,* San Juan, Puerto Rico, pages 120–131, December 1994.

[24] T.A. Henzinger. Sooner is safer than later. *Information Processing Letters*, 43:135–141, 1992.

[25] T.A. Henzinger, Z. Manna, and A. Pnueli. Timed transition systems. In de Bakker et al. [15], pages 226–251.

[26] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111:193–244, 1994.

[27] A. Jeffrey. A linear time process algebra. In K.G. Larsen and A. Skou, editors, *Proceedings of the 3rd International Workshop on Computer Aided Verification,* Aalborg, Denmark, volume 575 of *Lecture Notes in Computer Science*, pages 432–442. Springer-Verlag, 1992.

[28] A.S.A. Jeffrey, S.A. Schneider, and F.W. Vaandrager. A comparison of additivity axioms in timed transition systems. Report CS-R9366, CWI, Amsterdam, November 1993.

[29] N. Klarlund and F.B. Schneider. Proving nondeterministically specified safety properties using progress measures. *Information and Computation*, 107(1):151–170, November 1993.

[30] A.S. Klusener. The silent step in time. In Cleaveland [13], pages 421–435.

[31] L. Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872–923, March 1994.

[32] B.W. Lampson, N.A. Lynch, and J.F. Søgaard-Andersen. Correctness of at-most-once message delivery protocols. In *FORTE'93 - Sixth International Conference on Formal Description Techniques,* Boston, MA, October 1993, pages 387–402, 1993.

[33] H. Langmaack, W.-P. de Roever, and J. Vytopil, editors. *Proceedings of the Third International School and Symposium on Formal Techniques in Real Time and Fault Tolerant Systems,* Lübeck, Germany, September 1994, volume 863 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[34] V. Luchangco. Using simulation techiniques to prove timing properties. Master's thesis, MIT Electrical Engineering and Computer Science, 1994.

[35] V. Luchangco, E. Söylemez, S. Garland, and N.A. Lynch. Verifying timing properties of concurrent algorithms. In *Proceedings of the Seventh International Conference on Formal Description Techniques for Distributed Systems and Communications Protocols*, pages 239–259, Berne, Switzerland, October 1994. IFIP WG6.1, Elsevier Science Publishers B. V. (North Holland). Preliminary version. Final version to be published by Chapman and Hall.

[36] N.A. Lynch. Simulation techniques for proving properties of real-time systems. In J.W. de Bakker, W.P. de Roever, and G. Rozenberg, editors, *Proceedings REX School/Symposium: A Decade of Concurrency,* Noordwijkerhout, The Netherlands, June 1993, volume 803 of *Lecture Notes in Computer Science*, pages 375–424. Springer-Verlag, 1994.

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

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

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

[40] 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.

[41] N.A. Lynch and F.W. Vaandrager. Action transducers and timed automata. In Cleaveland [13], pages 436–455. Full version available as CWI Report CS-R9460, Amsterdam, November 1994, and as Technical Memo MIT/LCS/TM-480.b, MIT LCS, Cambridge, MA, October 1994.

[42] N.A. Lynch and F.W. Vaandrager. Forward and backward simulations for timing-based systems. In de Bakker et al. [15], pages 397–446.

[43] N.A. Lynch and F.W. Vaandrager. Forward and backward simulations – part II: Timing-based systems. Report CS-R9314, CWI, Amsterdam, March 1993. Also, MIT/LCS/TM-487.b, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA.

[44] N.A. Lynch and F.W. Vaandrager. Forward and backward simulations – part I: Untimed systems. Technical Memo MIT/LCS/TM-486.b (new version of TM-486), Laboratory for Computer Science, Massachusetts Institute of Technolog, Cambridge, MA, August 1994. To appear in *Information and Computation*.

[45] N.A. Lynch and H.B. Weinberg. Proving correctness of a vehicle maneuver: Deceleration, June 1995. In *Proceedings Second European Workshop on Real-Time and Hybrid Systems*, Grenoble, France. To appear.

[46] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, 1992.

[47] Z. Manna and A. Pnueli. Verifying hybrid systems. In Grossman et al. [21], pages 4–35.

[48] 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.

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

[50] F. Moller and C. Tofts. A temporal calculus of communicating systems. In Baeten and Klop [10], pages 401–415.

[51] X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. An approach to the description and analysis of hybrid systems. In Grossman et al. [21], pages 149–178.

[52] 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.

[53] X. Nicollin, J. Sifakis, and S. Yovine. From ATP to timed graphs and hybrid systems. *Acta Informatica*, 30(2):181–202, 1993.

[54] A. Pnueli. Development of hybrid systems. In Langmaack et al. [33], pages 77–85.

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

[56] R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1995. In progress.

[57] A.P. Sistla. Proving correctness with respect to nondeterministic safety specifications. *Information Processing Letters*, 39(1):45–49, July 1991.

[58] J. Søgaard-Andersen. *Correctness of Protocols in Distributed Systems.* PhD thesis, Technical University of Denmark, Lyngby, Denmark, December 1993. ID-TR: 1993-131. Also, [59].

[59] J.F. Søgaard-Andersen, B.W. Lampson, and N.A. Lynch. Correctness of communication protocols – a case study. Technical Report MIT/LCS/TR-589, Laboratory for Computer Science, MIT, Cambridge, MA, November 1993.

[60] E. Söylemez. Automatic verification of the timing properties of MMT automata. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, February 1994.

[61] Wang Yi. Real-time behaviour of asynchronous agents. In Baeten and Klop [10], pages 502–520.

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