Forward and Backward Simulations for Timing-Based Systems*

Nancy Lynch and Frits Vaandrager
MIT Laboratory for Computer Science
Cambridge, MA 02139, USA

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. As a first step, a comprehensive overview of simulation techniques for simple untimed automata is given. In particular, soundness and completeness results for (1) refinements, (2) forward and backward simulations, (3) forward-backward and backward-forward simulations, and (4) history and prophecy relations are given. History and prophecy relations are new and are abstractions of the history variables of Owicki and Gries and the prophecy variables of Abadi and Lamport, respectively. As a subsequent step, it is shown how most of the results for untimed automata can be carried over to the setting of timed automata. In fact, many of the results for the timed case are obtained as consequences of the analogous results for the untimed case.

Keywords: Simulations, timing-based systems, real-time, timed automata, refinement mappings, forward simulations, backward simulations, forward-backward simulations, backward-forward simulations, history variables, prophecy variables, history relations, prophecy relations.

Contents

*This work was supported by ONR contracts N00014-85-K-0168 and N00014-91-J-1988, by NSF grant CCR-8915206, and by DARPA contract N00014-89-J-1988.

1 Introduction

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

In this paper, we describe a candidate for such a model, and use it to express some powerful simulation techniques for proving correctness of timing-based systems. The style of the model we define is *automata-theoretic*, which is the natural style for expressing assertional methods. However, we expect that the model can also serve as a semantic model for interesting algebraic languages, and thus that process algebraic methods can also be employed in the same framework. We define several kinds of simulations including *refinements*, *forward simulations*, *backward simulations*, and hybrid versions that we call *forward-backward* and *backward-forward simulations*. We prove basic results for these kinds of simulations, in particular, soundness and completeness theorems. We also define *history relations* and *prophecy relations*, which are abstract versions of the history and prophecy variables of Abadi and Lamport [1]. We prove theorems describing the properties of these various kinds of simulations and relating the different kinds of simulations to each other.

The goal of extending simulation techniques to timing-based systems is also the motivation for the work of Lynch and Attiya in [19]. That work, however, only explores forward simulations. Also, the model used in [19] has considerably more structure than the very general model proposed here; it is based closely on the *timed automaton* model of Merritt, Modugno and Tuttle [22], which assumes that the system being modeled is describable in terms of a collection of separate tasks, each with associated upper and lower bounds on its speed. This extra structure supports the development of some useful progress measure proof methods, which we do not develop here. On the other hand, the basic theorems about forward simulations that appear in [19] are stated in a setting that has more structure than is really necessary for those theorems. In this paper, we make only those assumptions that are needed for the basic results about simulation proof techniques.

We propose a notion of *timed automaton*, which is just an automaton (or labeled transition system) equipped with some additional structure. Specifically, each state of the automaton has an associated *time*, which indicates the current time. (Thus we use *absolute* time in the sense of [2].) The actions of the automaton are of three kinds: *visible actions*, *time-passage actions*, and a special *internal action* τ. As in many other formalisms for real-time, see for instance [2, 3, 7, 22, 24, 25, 32], all actions except for the time-passage actions are modeled as occurring instantaneously, i.e., they do not change the time component of the state.

To specify times, we use a *dense* time domain, specifically, the nonnegative reals (starting with time 0 in the initial state), and we impose no lower bound on the time between events. This choice distinguishes our work from many others', e.g., [4, 7, 24, 25, 29, 33], in which discrete time values or universal positive lower bounds on step time are used. Use of real-valued time is less restrictive, and we believe that the extra flexibility will be useful in the design and analysis of timing-based distributed algorithms. The penalty we pay for this flexibility is that our automata may admit some "Zeno executions", i.e., infinite executions in which the time component is bounded.

Timed automata are required to satisfy a small set of basic axioms which express natural properties of time. For instance, there is an axiom saying that time-passage actions may not decrease time, and another saying that all the other actions are instantaneous.

Also, time can advance by a particular amount in one time-passage step if and only if it can also advance by the same amount in two steps. (This property is called *continuity* in [32] and, more appropriately, *time additivity* in [26].) We attempt to use as few axioms as possible to obtain the results about simulations. Later, as we try to express different proof methods in terms of this model, we expect to have to add additional requirements to obtain the desired properties. A typical axiom we may have to add at some point is the axiom of *time determinism* [32, 26], which says that if from a given state s there are time-passage actions leading to states s' and s'', which both have the same time, s' and s'' must be equal.

In order to define correctness for timed automata, we require notions of external behavior. We emphasize two notions. First, as the finite behaviors of a timed automaton, we take the *finite timed traces*, each of which consists of a finite sequence of timed external actions together with a final time. Second, as the infinite behaviors, we take the *admissible timed traces*, each of which consists of a sequence of timed external actions that can occur in an execution in which the time grows unboundedly (i.e., a "non-Zeno" infinite execution). In a *feasible* timed automaton, i.e., a timed automaton in which each finite execution can be extended to an execution in which the time is unbounded, the finite timed traces are determined by the admissible ones. For this type of automaton, inclusion of sets of admissible timed traces appears to be a good notion of implementation. One of the main objectives of this paper is to develop proof techniques to show that one automaton implements another in this sense.

Even though our notion of timed automata has less structure than those of [22] and [19], it is closely related to those models. Ours can be regarded as a generalization of the model in [19], in which the notion of separate tasks is removed. (There are some minor distinctions; for instance, we do not include names for internal actions, but label them all by the special symbol τ. This distinction is unimportant in a setting without separate tasks.) On the other hand, the model of [22] includes treatment of fairness and liveness, whereas our model does not. (The model in [22] was originally designed as an extension of the non-timing-based input/output automaton model of [20], which emphasizes the notion of *fair execution*.) The reason we have not equipped our model with facilities for handling fairness and liveness is that we believe that in the setting of timing-based systems, all properties of practical importance can be expressed as safety properties, given the admissibility assumption that time increases without bound. The absence of fairness and liveness considerations in our model seems to remove various technical and philosophical complications, and to lead to simpler and more systematic proof techniques.

The simulations we consider are derived from simulations studied in many places in the research literature. The simplest kind of simulation we consider is a *refinement*, which is a functional simulation similar to those studied in [17] and very similar to a homomorphism between automata in the sense of classical automata theory [6]. A refinement from a timed automaton A to another timed automaton B is a time-preserving function from states of A to states of B such that (a) the image of every start state of A is a start state of B, and (b) every step of A has a corresponding sequence of steps of B that begins and ends with the images of the respective beginning and ending states of the given step, and that has the same visible actions. In the untimed setting, it is well known that the corresponding untimed notion of refinement implies an implementation relation between A and B; we give the analogous soundness result, as well as a partial completeness result, for the timed setting.

We then consider *forward simulations* and *backward simulations*, which are generalizations of refinements that allow a set of states of B to correspond to a single state of A. Forward simulations are similar to the the simulations of [28, 10], the possibilities mappings of [20], the downward simulations of [9, 14, 8], and the forward simulations of [13]. The correspondence conditions (a) and (b) above are generalized so that (a) every start state of A has *some* image that is a start state of B, and (b) every step of A and every state of B corresponding to the *beginning* state of the step yield a corresponding sequence of steps of B ending with the image of the ending state of the given step. The usefulness of such simulations in proving correctness in the untimed setting has been well demonstrated. (See, e.g., [18] for some examples.) Again, we give soundness and partial completeness results for the timed setting. Backward simulations occurred first in [9] under the name of *upward simulations* and were used later in the setting of CSP in [14, 8]. In [21, 12] it is observed that they are closely related to the prophecy variables first defined in [1]. In the case of a backward simulation, conditions (a) and (b) are generalized so that (a) *all* images of every start state of A are start states of B, and (b) every step of A and every state of B corresponding to the *ending* state of the step yield a corresponding sequence of steps of B beginning with the image of the beginning state of the given step. Abadi and Lamport [1] demonstrate the usefulness of prophecy variables (and hence backward simulations) in the untimed setting, with some simple examples. Again, we give soundness and partial completeness results for the timed setting.

We also consider *forward-backward* and *backward-forward simulations*, which are essentially compositions of one forward and one backward simulation, in the two possible orders. The definition of a forward-backward simulation has been inspired by the work of Klarlund and Schneider [15] for the untimed setting (with no internal actions) again, we extend these ideas to the timed setting (with internal actions). The notion of a backward-forward simulation is suggested by symmetry with forward-backward simulations. While some of the results for this case are symmetric with the forward-backward case, others (notably, certain completeness results) do not hold.

We also provide redefinitions of the *history variable* notion of [27] and the *prophecy variable* notions of [1], in terms of timed automata, and prove equivalence results between these explicit definitions and our more abstract simulation definitions.

In order to present our results for timed automata, we find it convenient first to describe corresponding results for the simpler untimed setting. Therefore, we first define a simple untimed automaton model corresponding to the timed automaton model, and explore all the types of simulations described above in terms of this model. The definitions and results for timed automata are given in a subsequent step. The results for the timed setting are completely analogous to those for the untimed setting; in fact, in many cases, our results for the timed setting are derived directly from those for the untimed setting. An advantage of this two-phase approach is that it highlights the adaptability of the various verification techniques from the untimed to the timed setting.

As far as the classification of simulations is concerned, our work is closely related to and extends that of Jonsson [13]. However, whereas we focus on real-time issues, Jonsson addresses fairness instead. Also, Jonsson has more powerful notion of backward simulation, which we prefer not to use since it fails to reduce global reasoning about infinite behaviors to local reasoning about states and actions.

We consider the main contributions of the paper to be the following. First, we give an organized presentation, in terms of a very simple and abstract model, of a wide range of important simulation techniques, together with their basic soundness and completeness properties. We present the various simulation techniques in a "bottom-up" fashion, starting with simple ones such as forward and backward simulations and building up to more complicated simulations such as forward-backward simulations and history relations. We give elegant and short proofs of soundness and completeness results for complicated simulations in terms of soundness and (partial) completeness results for simple simulations. Second, we introduce the notions of a timed automaton and its behavior, and extend existing simulation notions to this new setting. Third, there are several specific new definitions and results, notably: (1) The definition of a notion of composition of forward-backward simulations. This allows us to prove that image-finite forward-backward simulations induce a preorder on the domain of general automata. (2) The introduction of backward-forward simulations. Although these simulations do not lead to a complete proof method, they are sound and possibly useful in practice. They arise naturally as the dual notion of forward-backward simulations. (3) The notions of history and prophecy relations. Fourth and finally, our presentation style, which bases the timed case on the untimed case, explains the connections between these two settings.

In what follows, some of the proofs have been omitted because of length restrictions.

2 Preliminaries

2.1 Sequences

Let K be any set. The sets of finite and infinite sequences of elements of K are denoted by K^* and K^ω, respectively. Concatenation of a finite sequence with a finite or infinite sequence is denoted by juxtaposition; λ denotes the empty sequence and the sequence containing one element $a \in K$ is denoted a. We say that a sequence σ is a *prefix* of a sequence ρ, notation $\sigma \leq \rho$, if either $\sigma = \rho$, or σ is finite and $\rho = \sigma\sigma'$ for some sequence σ'. A set S of sequences is *prefix closed* if, whenever some sequence is in S all its prefixes are also. If σ is a nonempty sequence then *first*(σ) returns the first element of σ, and *tail*(σ) returns σ with its first element removed. Moreover, if σ is finite, then *last*(σ) returns the last element of σ. If σ is a sequence over K and $L \subseteq K$, then $\sigma\lceil L$ denotes the sequence obtained by projecting σ on L. If S is a set of sequences, $S\lceil L$ is defined as $\{\sigma\lceil L \mid \sigma \in S\}$.

2.2 Sets, Relations and Functions

A *relation* over sets X and Y is defined to be any subset of $X \times Y$. If f is a relation over X and Y, then we define the *domain* of f to be $\mathit{domain}(f) \triangleq \{x \in X \mid (x, y) \in f \text{ for some } y \in Y\}$, and the *range* of f to be $\mathit{range}(f) \triangleq \{y \in Y \mid (x, y) \in f \text{ for some } x \in X\}$. A *total* relation over X and Y is a relation f over X and Y with $\mathit{domain}(f) = X$. If X is any set, we let $\mathit{id}(X)$ denote the identity relation over X and X, i.e., $\{(x, x) \mid x \in X\}$. We define *composition* of relations in the usual way, i.e., if f and g are relations over X and Y and over Y and Z, respectively, then $g \circ f$ denotes the relation over X and Z consisting of all pairs (x, z) such that there exists $y \in Y$ with $(x, y) \in f$ and $(y, z) \in g$. For all relations f, g and h, $f \circ (g \circ h) = (f \circ g) \circ h$. Also, for $X \supseteq \mathit{domain}(f)$ and $Y \supseteq \mathit{range}(f)$,

$id(X) \circ f = f \circ id(Y) = f$. If f is a relation over X and Y, then the *inverse* of f, written f^{-1}, is defined to be the relation over Y and X consisting of those pairs (y, x) such that $(x, y) \in f$. Recall that for any pair of relations f and g, $(g \circ f)^{-1} = f^{-1} \circ g^{-1}$. If f is a relation over X and Y, and Z is a set, then $f\lceil Z$ is the relation over $X \cap Z$ and Y given by $f\lceil Z \triangleq f \cap (Z \times Y)$. If f is a relation over X and Y and $x \in X$, we define $f[x] = \{y \in Y \mid (x, y) \in f\}$. We say that a relation f over X and Y is a *function from X to Y*, and write $f : X \rightarrow Y$, if $|f[x]| = 1$ for all $x \in X$; in this case, we write $f(x)$ to denote the unique element of $f[x]$. A function c from X to Y is a *choice function* for a relation f over X to Y provided that $c \subseteq f$ (i.e., $c(x) \in f[x]$ for all $x \in X$). If X is a set, $\mathbf{P}(X)$ denotes the powerset of X, i.e., the set of subsets of X, and $\mathbf{PN}(X)$ the set of nonempty subsets of X, i.e., the set $\mathbf{P}(X) - \{\emptyset\}$. We say that a relation f over X and Y is *image-finite* if $f[x]$ is finite for all x in X. If f is a relation over X and $\mathbf{P}(Y)$, then we say that f is *image-2-finite* if every set in the range of f is finite.

2.3 A Basic Graph Lemma

We require the following lemma, a generalization of König's Lemma [16]. If G is a digraph, then a *root* of G is defined to be a node with no incoming edges.

Lemma 2.1 *Let G be an infinite digraph that satisfies the following properties.*

1. *G has finitely many roots.*

2. *Each node of G has finite outdegree.*

3. *Each node of G is reachable from some root of G.*

Then there is an infinite path in G starting from some root.

Proof: The usual proof for König's Lemma extends to this case. ■

3 Untimed Automata and Their Behaviors

This section presents the basic definitions and results for untimed automata.

3.1 Automata

We begin with the definition of an (untimed) automaton. An *automaton* A consists of:

- a set $states(A)$ of states,

- a nonempty set $start(A) \subseteq states(A)$ of start states,

- a set $acts(A)$ of actions that includes a special element τ, and

- a set $steps(A) \subseteq states(A) \times acts(A) \times states(A)$ of steps.

We let $s, s', u, u', ..$ range over states, and $a, ..$ over actions. We let $ext(A)$, the *external actions*, denote $acts(A) - \{\tau\}$. We call τ the *internal action*. We use the term *event* to refer to an occurrence of an action in a sequence. If σ is a sequence of actions then $\widehat{\sigma}$ is the sequence gained by deleting all τ events from σ. We write $s' \xrightarrow{a}_A s$, or just $s' \xrightarrow{a} s$ if A is clear from the context, as a shorthand for $(s', a, s) \in steps(A)$. In this section as well as the next one, A, B,.. range over automata. Later, however, we will use these symbols to range over timed automata.

An *execution fragment* of A is a finite or infinite alternating sequence $s_0 a_1 s_1 a_2 s_2 \cdots$ of states and actions of A, beginning with a state, and if it is finite also ending with a state, such that for all i, $s_i \xrightarrow{a_{i+1}} s_{i+1}$. We denote by $frag^*(A)$, $frag^\omega(A)$ and $frag(A)$ the sets of finite, infinite, and all execution fragments of A, respectively. An *execution* of A is an execution fragment that begins with a start state. We denote by $execs^*(A)$, $execs^\omega(A)$ and $execs(A)$ the sets of finite, infinite, and all executions of A, respectively. A state s of A is *reachable* if $s = last(\alpha)$ for some finite execution α of A.

Suppose $\alpha = s_0 a_1 s_1 a_2 s_2 \cdots$ is an execution fragment of A. Let γ be the sequence consisting of the actions in α: $\gamma = a_1 a_2 \ldots$. Then $trace(\alpha)$ is defined to be the sequence $\hat{\gamma}$. A finite or infinite sequence β of actions is a *trace* of A if A has an execution α with $\beta = trace(\alpha)$. We write $traces^*(A)$, $traces^\omega(A)$ and $traces(A)$ for the sets of finite, infinite and all traces of A, respectively. These notions induce three *preorders* (i.e., reflexive and transitive relations). For A and B automata, we define $A \leq_{*\mathrm{T}} B \triangleq traces^*(A) \subseteq traces^*(B)$, $A \leq_{\omega\mathrm{T}} B \triangleq traces^\omega(A) \subseteq traces^\omega(B)$, and $A \leq_{\mathrm{T}} B \triangleq traces(A) \subseteq traces(B)$. Recall that the *kernel* of a preorder $\sqsubseteq$ is the equivalence $\equiv$ defined by $x \equiv y \triangleq x \sqsubseteq y \wedge y \sqsubseteq x$. We denote by $\equiv_{*\mathrm{T}}$, $\equiv_{\omega\mathrm{T}}$ and $\equiv_{\mathrm{T}}$, the respective kernels of the preorders $\leq_{*\mathrm{T}}$, $\leq_{\omega\mathrm{T}}$ and $\leq_{\mathrm{T}}$.

Suppose A is an automaton, s' and s are states of A, and β is a finite sequence over $ext(A)$. We say that (s', β, s) is a *move* of A, and write $s' \overset{\beta}{\Longrightarrow}_A s$, or just $s' \overset{\beta}{\Longrightarrow} s$ when A is clear, if A has a finite execution fragment α with $first(\alpha) = s'$, $trace(\alpha) = \beta$ and $last(\alpha) = s$.

3.2 Restricted Kinds of Automata

Automaton A is *deterministic* if $|start(A)| = 1$, $steps(A)$ contains no τ steps, and for all states s' and all external actions a there is at most one state s such that $s' \xrightarrow{a}_A s$.

A has *finite invisible nondeterminism (fin)* if $start(A)$ is finite, and for any state s' and any finite sequence β over $ext(A)$, there are only finitely many states s such that $s' \overset{\beta}{\Longrightarrow}_A s$.

A is a *forest* if for each state of A there is a unique execution that leads to it. Recall that a forest is characterized uniquely by the property that all states of A are reachable, start states have no incoming steps and each of the other states has exactly one incoming step.

The relation $after(A)$ consists of the pairs $(\beta, s) \in (ext(A))^* \times states(A)$ for which there is a finite execution of A with trace β and last state s.

$$after(A) \triangleq \{(\beta, s) \mid \exists \alpha \in execs^*(A) : trace(\alpha) = \beta \text{ and } last(\alpha) = s\}.$$

The relation $past(A) \triangleq after(A)^{-1}$ relates a state s of A to the traces of finite executions of A that lead to s. Also, define $before(A)$ to be the relation that relates a finite sequence

β to those states of A from where an execution with trace β is possible.

$$\mathit{before}(A) \triangleq \{(\beta, s) \mid \exists \alpha \in \mathit{frag}^*(A) : \mathit{trace}(\alpha) = \beta \text{ and } \mathit{first}(\alpha) = s\}.$$

We write $\mathit{future}(A)$ for $\mathit{before}(A)^{-1}$.

Lemma 3.1

1. *If A is deterministic then $\mathit{after}(A)$ is a function from $\mathit{traces}^*(A)$ to $\mathit{states}(A)$.*
2. *If A has fin then $\mathit{after}(A)$ is image-finite.*
3. *If A is a forest then $\mathit{past}(A)$ is a function from $\mathit{states}(A)$ to $\mathit{traces}^*(A)$.*

3.3 Trace Properties

For A an automaton, its *behavior*, $\mathit{beh}(A)$, is defined by $\mathit{beh}(A) \triangleq (\mathit{ext}(A), \mathit{traces}(A))$. In this subsection, we characterize the structures that can be obtained as the behavior $\mathit{beh}(A)$ for some automaton A as *trace properties*.

A *trace property* P is a pair (K, L) with K a set and L a nonempty, prefix closed set of (finite or infinite) sequences over K. We will refer to the constituents of P as $\mathit{sort}(P)$ and $\mathit{traces}(P)$, respectively. Also, we write $\mathit{traces}^*(P) \triangleq K^* \cap L$ and $\mathit{traces}^\omega(P) \triangleq K^\omega \cap L$. For P and Q trace properties, we define $P \leq_{*\mathrm{T}} Q \triangleq \mathit{traces}^*(P) \subseteq \mathit{traces}^*(Q)$, $P \leq_{\omega\mathrm{T}} Q \triangleq \mathit{traces}^\omega(P) \subseteq \mathit{traces}^\omega(Q)$, and $P \leq_{\mathrm{T}} Q \triangleq \mathit{traces}(P) \subseteq \mathit{traces}(Q)$. With $\equiv_{*\mathrm{T}}$, $\equiv_{\omega\mathrm{T}}$ and $\equiv_{\mathrm{T}}$, we denote the kernels of the preorders $\leq_{*\mathrm{T}}$, $\leq_{\omega\mathrm{T}}$ and $\leq_{\mathrm{T}}$, respectively. A trace property P is *limit-closed* if an infinite sequence is in $\mathit{traces}(P)$ whenever all its finite prefixes are.

Lemma 3.2 *Suppose P and Q are trace properties with Q limit-closed. Then $P \leq_{*\mathrm{T}} Q \Leftrightarrow P \leq_{\mathrm{T}} Q$.*

Lemma 3.3

1. *$\mathit{beh}(A)$ is a trace property.*
2. *If A has fin then $\mathit{beh}(A)$ is limit-closed.*
3. *$A \leq_{*\mathrm{T}} B \Leftrightarrow \mathit{beh}(A) \leq_{*\mathrm{T}} \mathit{beh}(B)$, $A \leq_{\omega\mathrm{T}} B \Leftrightarrow \mathit{beh}(A) \leq_{\omega\mathrm{T}} \mathit{beh}(B)$, and $A \leq_{\mathrm{T}} B \Leftrightarrow \mathit{beh}(A) \leq_{\mathrm{T}} \mathit{beh}(B)$.*

Proof: It is easy to see that $\mathit{beh}(A)$ is a trace property.

For Part 2, suppose A has fin. We use Lemma 2.1 to show that $\mathit{beh}(A)$ is limit-closed. Suppose β is an infinite sequence over $\mathit{ext}(A)$ such that all finite prefixes of β are in $\mathit{traces}(A)$. Consider the digraph G whose nodes are pairs (γ, s), where γ is a finite prefix of β and s is a state of A, and where there exists an execution α of A that ends with state s and such that $\gamma = \mathit{trace}(\alpha)$; there is an edge from node (γ', s') to node (γ, s) exactly if γ is of the form $\gamma' a$, where $a \in \mathit{ext}(A)$, and where $s' \overset{a}{\Longrightarrow}_A s$. Then G satisfies the hypotheses of Lemma 2.1, which implies that there is an infinite path in G starting at a root. This corresponds directly to an execution α having $\mathit{trace}(\alpha) = \beta$. Hence, $\beta \in \mathit{traces}(A)$.

Part 3 is immediate from the definitions. ■

Proposition 3.4 *If B has fin then $A \leq_{*\mathrm{T}} B \Leftrightarrow A \leq_{\mathrm{T}} B$.*

Proof: Immediate from Lemmas 3.2 and 3.3. ■

Example 3.1 The automata A and B of Figure 1 illustrate the difference between $\leq_{*\mathrm{T}}$ and $\leq_{\mathrm{T}}$. Note that automaton B does not have fin.

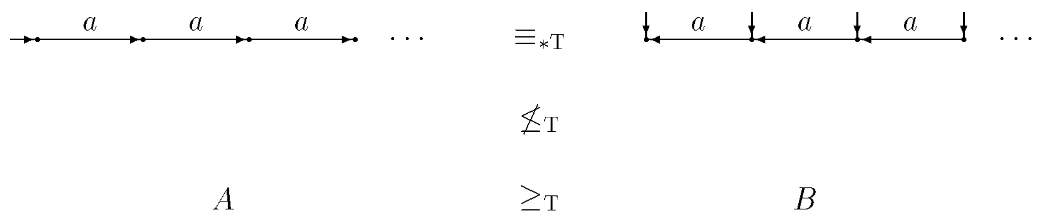

Figure 1: $\leq_{*\mathrm{T}}$ versus $\leq_{\mathrm{T}}$.

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

Definition 3.1 For P a trace property, the associated *canonical* automaton $can(P)$ is the structure A given by

- $states(A) = traces^*(P)$,
- $start(A) = \{\lambda\}$,
- $acts(A) = sort(P) \cup \{\tau\}$, and
- for $\beta', \beta \in states(A)$ and $a \in acts(A)$, $\beta' \xrightarrow{a}_A \beta \quad \Leftrightarrow \quad a \in ext(A) \wedge \beta' a = \beta$.

Lemma 3.5

1. *$can(P)$ is a deterministic forest,*
2. *$beh(can(P)) \equiv_{*\mathrm{T}} P$,*
3. *$P \leq_{\mathrm{T}} beh(can(P))$, and*
4. *if P is limit-closed then $beh(can(P)) \equiv_{\mathrm{T}} P$.*

Proof: Parts 1 and 2 follow easily from the definitions. Since $can(P)$ is deterministic it certainly has fin, so it follows by Lemma 3.3 that $beh(can(P))$ is limit-closed. Now 3 and 4 follow by combination of 2 and Lemma 3.2. ■

Lemma 3.6

1. *$can(beh(A))$ is a deterministic forest,*
2. *$can(beh(A)) \equiv_{*\mathrm{T}} A$,*
3. *$A \leq_{\mathrm{T}} can(beh(A))$, and*
4. *if A has fin then $can(beh(A)) \equiv_{\mathrm{T}} A$.*

Proof: By combining Lemmas 3.3 and 3.5. ■

4 Simulations for Untimed Automata

In this section, we develop simulation techniques for untimed automata.

4.1 Refinements

The simplest type of simulation we consider is a *refinement*. A *refinement* from A to B is a function r from states of A to states of B that satisfies the following two conditions:

1. If $s \in \mathit{start}(A)$ then $r(s) \in \mathit{start}(B)$.
2. If $s' \xrightarrow{a}_A s$ then $r(s') \overset{\hat{a}}{\Longrightarrow}_B r(s)$.

We write $A \leq_{\mathrm{R}} B$ if there exists a refinement from A to B.

This notion is similar to that of a *homomorphism* in classical automata theory; see for instance Ginzberg [6]. Besides our additional treatment of internal actions, a difference between the two notions is that the classical notion involves a mapping between the action sets of the automata, whereas our refinements do not.

Example 4.1 Figure 2 presents some canonical examples of $\leq_{\mathrm{R}}$.

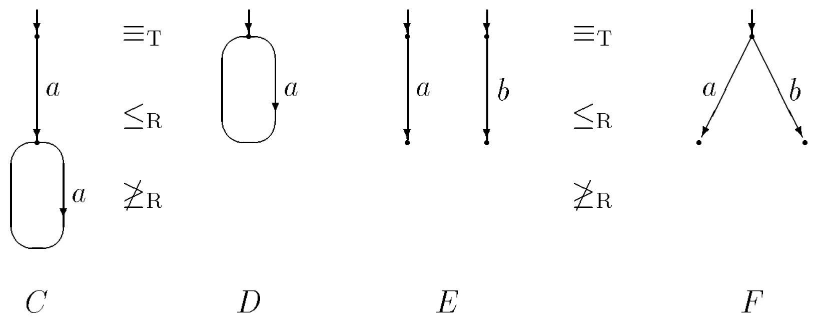

Figure 2: Refinements.

The following technical lemma is a straightforward consequence of the definition of a refinement.

Lemma 4.1 *Suppose r is a refinement from A to B and $s' \overset{\beta}{\Longrightarrow}_A s$. Then $r(s') \overset{\beta}{\Longrightarrow}_B r(s)$.*

Proposition 4.2 *$\leq_{\mathrm{R}}$ is a preorder (i.e., is transitive and reflexive).*

Proof: The identity function $\mathit{id}(\mathit{states}(A))$ is a refinement from A to itself. This implies that $\leq_{\mathrm{R}}$ is reflexive. Using Lemma 4.1, transitivity follows from the observation that if r is a refinement from A to B and r' is a refinement from B to C, $r' \circ r$ is a refinement from A to C. ■

Theorem 4.3 *(Soundness of refinements) $A \leq_{\mathrm{R}} B \;\Rightarrow\; A \leq_{\mathrm{T}} B$.*

Proof: Suppose $A \leq_{\mathrm{R}} B$. Let r be a refinement from A to B, and let e be a function that maps each move (s', β, s) of B to a finite execution fragment of B from s' to s with trace β. Suppose $\beta \in \mathit{traces}(A)$. Then there exists an execution $\alpha = s_0 a_1 s_1 a_2 s_2 \cdots$ of A with $\beta = \mathit{trace}(\alpha)$. By the first condition in the definition of a refinement, $r(s_0)$ is a start state of B, and by the second condition, $r(s_i) \overset{\widehat{a_{i+1}}}{\Longrightarrow}_B r(s_{i+1})$ for all i. For $i \geq 0$, define $\alpha_i = e((r(s_i), \widehat{a_{i+1}}, r(s_{i+1})))$. Next define sequence α' to be the (infinitary) concatenation $\alpha_0 \mathit{tail}(\alpha_1)\mathit{tail}(\alpha_2)\cdots$. By construction, α' is an execution of B with $\mathit{trace}(\alpha') = \mathit{trace}(\alpha) = \beta \in \mathit{traces}(B)$. ■

Theorem 4.4 *(Partial completeness of refinements) Suppose A is a forest, B is deterministic and $A \leq_{*\mathrm{T}} B$. Then $A \leq_{\mathrm{R}} B$.*

Proof: The relation $r \triangleq \mathit{after}(B) \circ \mathit{past}(A)$ is a refinement from A to B. ■

4.2 Forward and Backward Simulations

4.2.1 Forward Simulations

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

1. If $s \in \mathit{start}(A)$ then $f[s] \cap \mathit{start}(B) \neq \emptyset$.
2. If $s' \xrightarrow{a}_A s$ and $u' \in f[s']$, then there exists a state $u \in f[s]$ such that $u' \overset{\hat{a}}{\Longrightarrow}_B u$.

We write $A \leq_{\mathrm{F}} B$ if there exists a forward simulation from A to B.

Example 4.2 Let C, D, E, F be as in Figure 2. Then $D \leq_{\mathrm{F}} C$ and $F \not\leq_{\mathrm{F}} E$.

Proposition 4.5 *$A \leq_{\mathrm{R}} B \Rightarrow A \leq_{\mathrm{F}} B$.*

Proof: Any refinement relation is a forward simulation. ■

The following lemma is the analogue for forward simulations of Lemma 4.1.

Lemma 4.6 *Suppose f is a forward simulation from A to B and $s' \overset{\beta}{\Longrightarrow}_A s$. If $u' \in f[s']$, then there exists a state $u \in f[s]$ such that $u' \overset{\beta}{\Longrightarrow}_B u$.*

Proposition 4.7 *$\leq_{\mathrm{F}}$ is a preorder.*

Proof: For reflexivity, observe that the identity function $\mathit{id}(\mathit{states}(A))$ is a forward simulation from A to itself. For transitivity, use Lemma 4.6 to show that if f and f' are forward simulations from A to B and from B to C, respectively, $f' \circ f$ is a forward simulation from A to C. ■

Theorem 4.8 *(Soundness of forward simulations, [20, 11, 30]) $A \leq_{\mathrm{F}} B \Rightarrow A \leq_{\mathrm{T}} B$.*

Proof: Versions of this proof appears in the cited papers. The proof is similar to that of Theorem 4.3. ■

Theorem 4.9 *(Partial completeness of forward simulations) Suppose B is deterministic and $A \leq_{*\mathrm{T}} B$. Then $A \leq_{\mathrm{F}} B$.*

Proof: The relation $f \triangleq \mathit{after}(B) \circ \mathit{past}(A)$ is a forward simulation from A to B. ■

4.2.2 Backward Simulations

In many respects, backward simulations are the dual of forward simulations. Whereas a forward simulation requires that *some* state in the image of each start state should be a start state, a backward simulation requires that *all* states in the image of a start state be start states. Also, a forward simulation requires that forward steps in the source automaton can be simulated from related states in the target automaton, whereas the corresponding condition for a backward simulations requires that backward steps can be simulated. However, the two notions are not completely dual: the definition of a backward simulation contains a nonemptiness condition, and also, in order to imply soundness in general, backward simulations also require a finite image condition. The mismatch is due to the asymmetry in our automata between future and past: from any given state, all the possible histories are finite executions, whereas the possible futures can be infinite.

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

1. If $s \in start(A)$ then $b[s] \subseteq start(B)$.
2. If $s' \xrightarrow{a}_A s$ and $u \in b[s]$, then there exists a state $u' \in b[s']$ such that $u' \overset{\hat{a}}{\Longrightarrow}_B u$.

We write $A \leq_{\mathrm{B}} B$ if there exists a backward simulation from A to B, and $A \leq_{\mathrm{iB}} B$ if there exists an image-finite backward simulation from A to B.

Example 4.3 Let A, B be as in Figure 1. Then $A \leq_{\mathrm{B}} B$ but $A \not\leq_{\mathrm{iB}} B$. If C, D, E, F are as in Figure 2, then $D \not\leq_{\mathrm{B}} C$ and $F \leq_{\mathrm{iB}} E$.

Proposition 4.10 *$A \leq_{\mathrm{R}} B \Rightarrow A \leq_{\mathrm{iB}} B$.*

The following lemma is useful in the proofs of the preorder properties and of soundness.

Lemma 4.11 *Suppose b is a backward simulation from A to B and $s' \overset{\beta}{\Longrightarrow}_A s$. If $u \in b[s]$, then there exists a state $u' \in b[s']$ such that $u' \overset{\beta}{\Longrightarrow}_B u$.*

Proposition 4.12 *$\leq_{\mathrm{B}}$ and $\leq_{\mathrm{iB}}$ are preorders.*

Proof: The identity function $id(states(A))$ is a backward simulation from A to itself. Using Lemma 4.11 one can easily show that if b is backward simulation from A to B and b' is a backward simulation from B to C, $b' \circ b$ is a backward simulation from A to C. Moreover, if both b and b' are image-finite, then $b' \circ b$ is image-finite too. ∎

Theorem 4.13 *(Soundness of backward simulations)*

1. *$A \leq_{\mathrm{B}} B \Rightarrow A \leq_{*\mathrm{T}} B$, and*
2. *$A \leq_{\mathrm{iB}} B \Rightarrow A \leq_{\mathrm{T}} B$.*

Proof: Suppose b is a backward simulation from A to B and suppose $\beta \in traces^*(A)$. Then there is a move $s' \overset{\beta}{\Longrightarrow}_A s$, where s' is a start state of A. Since b is a backward simulation it is a total relation, so there exists a state $u \in b[s]$. By Lemma 4.11, there exists $u' \in b[s']$ with $u' \overset{\beta}{\Longrightarrow}_B u$. By the first condition of the definition of a backward simulation, $u' \in start(B)$. Therefore, $\beta \in traces^*(B)$, which shows the first part of the proposition.

For the second part, suppose that b is image-finite. We have already established $A \leq_{*\mathrm{T}} B$, so it is sufficient to show $A \leq_{\omega\mathrm{T}} B$. Suppose that $\beta \in traces^\omega(A)$, and let $\alpha = s_0 a_1 s_1 a_2 \cdots$ be an infinite execution of A with $trace(\alpha) = \beta$.

Consider the digraph G whose nodes are pairs (u, i) such that $(s_i, u) \in b$ and in which there is an edge from (u', i') to (u, i) exactly if $i = i'+1$ and $u' \overset{\widehat{a_i}}{\Longrightarrow}_B u$. Then G satisfies the hypotheses of Lemma 2.1, which implies that there is an infinite path in G starting at a root. This corresponds directly to an execution α' of B having $trace(\alpha') = trace(\alpha) = \beta$. Hence, $\beta \in traces(B)$. ■

In a recent paper, Jonsson [13] considers a weaker image-finiteness condition for backward simulations. Translated into our setting, the key observation of Jonsson is that in order to prove $A \leq_{\mathrm{T}} B$, it is enough to give a backward simulation b from A to B with the property that each infinite execution of A contains infinitely many states s with $b[s]$ finite. We do not explore this extension in this paper, primarily because it lacks a key feature of simulation techniques. Namely, it fails to reduce global reasoning about infinite behaviors to local reasoning about states and actions.

The following partial completeness result slightly generalizes a similar result of Jonsson [12] in that it also alllows for τ-steps in the B automaton.

Theorem 4.14 *(Partial completeness of backward simulations) Suppose A is a forest and $A \leq_{*\mathrm{T}} B$. Then*

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

Proof: We define a relation b over $states(A)$ and $states(B)$. Suppose s is a state of A. Since A is a forest there is a unique trace leading up to s, say β. Now define

$$b[s] = \{u \mid \exists \alpha \in execs^*(B) \colon trace(\alpha) = \beta,\ last(\alpha) = u \wedge [\alpha' < \alpha \Rightarrow trace(\alpha') \neq \beta]\}.$$

By letting $b[s]$ consist only of those states of B which can be reached via a *minimal* execution with trace β, we achieve that, if s is a start state, all the states in $b[s]$ are start states of B. It is also the case that b satisfies the other conditions in the definition of a backward simulation.

Lemma 3.1 implies that b is image-finite if B has fin. ■

Proposition 4.15 *Suppose all states of A are reachable, B has fin and $A \leq_{\mathrm{B}} B$. Then $A \leq_{\mathrm{iB}} B$.*

Proof: Let b be a backward simulation from A to B and let s be a state of A. Since s is reachable we can find a trace $\beta \in past(A)[s]$. From the fact that b is a backward simulation it follows that $b[s] \subseteq after(B)[\beta]$. But since B has fin, $after(B)[\beta]$ is finite by Lemma 3.1. This implies that b is image-finite. ■

Example 4.4 Figure 3 shows that the reachability assumption in Proposition 4.15 is essential. There is a backward simulation from G to H, but even though H is deterministic there is no image-finite backward simulation.

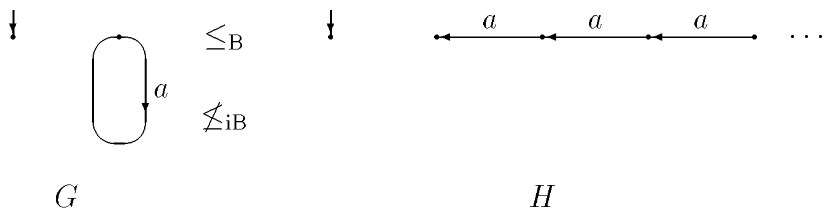

Figure 3: $\leq_{\mathrm{B}}$ and $\leq_{\mathrm{iB}}$ are different, even for automata with fin.

4.2.3 Combined Forward and Backward Simulations

Several authors have observed that forward and backward simulations together give a complete proof method (see [9, 8, 14, 12]): if $A \leq_{*\mathrm{T}} B$ then there exists an intermediate automaton C with a forward simulation from A to C and a backward simulation from C to B. We prove this below.

Theorem 4.16 *(Completeness of forward and backward simulations) If $A \leq_{*\mathrm{T}} B$ then the following are true.*

1. *$\exists C : A \leq_{\mathrm{F}} C \leq_{\mathrm{B}} B$.*
2. *If B has fin then $\exists C : A \leq_{\mathrm{F}} C \leq_{\mathrm{iB}} B$.*

Proof: Let $C = \mathit{can}(\mathit{beh}(A))$. By Lemma 3.6, C is a deterministic forest and $A \equiv_{*\mathrm{T}} C$. Since C is deterministic, $A \leq_{\mathrm{F}} C$ by Theorem 4.9, and because C is a forest, $C \leq_{\mathrm{B}} B$ follows by Theorem 4.14(1). If B has fin then $C \leq_{\mathrm{iB}} B$ follows by Theorem 4.14(2). ■

4.3 Forward-Backward and Backward-Forward Simulations

4.3.1 Forward-Backward Simulations

Forward-backward simulations were introduced by Klarlund and Schneider [15] who call them *invariants*. They also occur in the work of Jonsson [13] under the name *subset simulations*, and are related to the *failure simulations* of Gerth [5]. Forward-backward simulations combine in a single relation both a forward and a backward simulation. Below we present simple proofs of their soundness and completeness by making this connection explicit.

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

1. If $s \in \mathit{start}(A)$ then there exists $S \in g[s]$ such that $S \subseteq \mathit{start}(B)$.
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 $u' \overset{\hat{a}}{\Longrightarrow}_B u$.

We write $A \leq_{\mathrm{FB}} B$ if there exists a forward-backward simulation from A to B, and $A \leq_{\mathrm{iFB}} B$ if there exists an image-2-finite forward-backward simulation from A to B.

The following theorem says that a forward-backward simulation can be obtained by combining a forward and a backward simulation.

Theorem 4.17

1. $A \leq_{\mathrm{F}} C \leq_{\mathrm{B}} B \Rightarrow A \leq_{\mathrm{FB}} B$.

2. $A \leq_{\mathrm{F}} C \leq_{\mathrm{iB}} B \Rightarrow A \leq_{\mathrm{iFB}} B$.

Proof: Suppose f is a forward simulation from A to C, and b is a backward simulation from C to B. Then the relation g over $states(A)$ and $\mathbf{PN}(states(B))$ defined by $g = \{(s, b[u]) \mid (s,u) \in f\}$ is a forward-backward simulation from A to B. If b is image-finite then g is image-2-finite. ■

Proposition 4.18

1. $A \leq_{\mathrm{F}} B \Rightarrow A \leq_{\mathrm{iFB}} B$.

2. $A \leq_{\mathrm{B}} B \Rightarrow A \leq_{\mathrm{FB}} B$.

3. $A \leq_{\mathrm{iB}} B \Rightarrow A \leq_{\mathrm{iFB}} B$.

Proof: Immediate from Theorem 4.17, using that $\leq_{\mathrm{iB}}$ and $\leq_{\mathrm{F}}$ are reflexive. ■

In combination with Theorem 4.17, the following theorem tells us that forward-backward simulations are equivalent to forward simulations followed by backward simulations.

Theorem 4.19

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

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

Proof: Let g be a forward-backward simulation from A to B, which is image-2-finite if $A \leq_{\mathrm{iFB}} B$. Define C to be the automaton given by:

- $states(C) = range(g)$,
- $start(C) = range(g) \cap \mathbf{P}(start(B))$,
- $acts(C) = acts(B)$, and
- for $S', S \in states(C)$ and $a \in acts(C)$, $S' \xrightarrow{a}_C S \quad \Leftrightarrow \quad \forall u \in S \; \exists u' \in S' : u' \overset{\hat{a}}{\Longrightarrow}_B u$.

Then g is a forward simulation from A to C. Also, $\{(S, u) \mid S \in states(C) \text{ and } u \in S\}$ is a backward simulation from C to B, which is image finite if g is image-2-finite. ■

In order to show that $\leq_{\mathrm{FB}}$ and $\leq_{\mathrm{iFB}}$ are preorders, we require a definition of composition for forward-backward simulations, and a transitivity lemma.

Definition 4.1 If g is a relation over X and $\mathbf{PN}(Y)$ and g' is a relation over Y and $\mathbf{PN}(Z)$ then the composition $g' \bullet g$ is a relation over X and $\mathbf{PN}(Z)$ defined as follows.

$$(x, S') \in g' \bullet g \Leftrightarrow \exists S \in g[x], \exists c, \text{ a choice function for } g' \lceil S : S' = \bigcup\{c(y) : y \in S\}.$$

(The nonemptiness assumptions for g and g' immediately imply the nonemptiness assumption for $g' \bullet g$.)

Lemma 4.20 *Suppose g is a forward-backward simulation from A to B, and g' is a forward-backward simulation from B to C. Then $g' \bullet g$ is a forward-backward simulation from A to C. Moreover, if g and g' are image-2-finite then $g' \bullet g$ is also image-2-finite.*

Proof: For Condition 1 of the definition of a forward-backward simulation, suppose $s \in start(A)$. Because g is a forward-backward simulation, there is a set $S \in g[s]$ with $S \subseteq start(B)$. Since g' is a forward-backward simulation, it is possible to find, for each $u \in S$, a set $S_u \in g'[u]$ with $S_u \subseteq start(C)$. Hence all states in the set $S' = \bigcup\{S_u \mid u \in S\}$ are start states of C. Now let c be the function with domain S given by $c(u) = S_u$. Then c is a choice function for $g' \lceil S$. From the definition of $\bullet$ it now follows that $(s, S') \in g' \bullet g$. This shows that $g' \bullet g$ satisfies Condition 1.

Now we show Condition 2 of the definition of a forward-backward simulation. Suppose $s' \xrightarrow{a}_A s$ and $(s', S') \in g' \bullet g$. By definition of $g' \bullet g$, there exist $T' \in g[s']$ and a choice function c' for $g' \lceil T'$ such that $S' = \bigcup\{c'(u') : u' \in T'\}$. Because g is a forward-backward simulation from A to B, there is a set $T \in g[s]$ such that for each $u \in T$ there exists $u' \in T'$ with $u' \overset{\hat{a}}{\Longrightarrow}_B u$. Consider any particular $u \in T$. Choose $u' \in T'$ with $u' \overset{\hat{a}}{\Longrightarrow}_B u$. Because g' is a forward-backward simulation, there exists a set $S_u \in g'[u]$ such that for every $v \in S_u$ there exists a $v' \in c'(u')$ with $v' \overset{\hat{a}}{\Longrightarrow}_C v$. Define a choice function c for $g' \lceil T$ by taking $c(u)$ to be the set S_u.

Now consider the set $S = \bigcup\{c(u) : u \in T\}$. Then $(s, S) \in g' \bullet g$ by definition. By construction, we can find, for each $v \in S$, a state $v' \in S'$ with $v' \overset{\hat{a}}{\Longrightarrow}_C v$. Thus S has the required property to show Condition 2.

Finally, it is immediate from the definitions that, if g and g' are image-2 finite, $g' \bullet g$ is also image-2-finite. ■

Proposition 4.21 *$\leq_{\mathrm{FB}}$ and $\leq_{\mathrm{iFB}}$ are preorders.*

Proof: By Lemma 4.20. ■

Theorem 4.22 *(Soundness of forward-backward simulations, [15])*

1. *$A \leq_{\mathrm{FB}} B \Rightarrow A \leq_{*\mathrm{T}} B$, and*
2. *$A \leq_{\mathrm{iFB}} B \Rightarrow A \leq_{\mathrm{T}} B$.*

Proof: For part 1, suppose $A \leq_{\mathrm{FB}} B$. By Theorem 4.19, there exists an automaton C with $A \leq_{\mathrm{F}} C \leq_{\mathrm{B}} B$. By soundness of forward simulations, Theorem 4.8, $A \leq_{\mathrm{T}} C$, and by soundness of backward simulations, Theorem 4.13, $C \leq_{*\mathrm{T}} B$. This implies $A \leq_{*\mathrm{T}} B$. Part 2 is similar. ■

Theorem 4.23 *(Completeness of forward-backward simulations, [15]) Suppose $A \leq_{*\mathrm{T}} B$. Then*

1. *$A \leq_{\mathrm{FB}} B$.*
2. *If B has fin then $A \leq_{\mathrm{iFB}} B$.*

Proof: By Theorem 4.16, there exists an automaton C with $A \leq_{\mathrm{F}} C \leq_{\mathrm{B}} B$. Moreover, if B has fin then $A \leq_{\mathrm{F}} C \leq_{\mathrm{iB}} B$. Then Theorem 4.17 implies the needed conclusions. ■

4.3.2 Backward-Forward Simulations

Having studied forward-backward simulations, we find it natural to define and study a dual notion of backward-formulation simulation.

A *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 a $u \in S$ with $u' \overset{\hat{a}}{\Longrightarrow}_B u$.

We write $A \leq_{\mathrm{BF}} B$ if there exists a backward-forward simulation from A to B, and $A \leq_{\mathrm{iBF}} B$ if there exists an image-finite backward-forward simulation from A to B.

As for forward-backward simulations, backward-forward simulations can be characterized as combinations of forward and backward simulations.

Theorem 4.24

1. $A \leq_{\mathrm{B}} C \leq_{\mathrm{F}} B \Rightarrow A \leq_{\mathrm{BF}} B$.

2. $A \leq_{\mathrm{iB}} C \leq_{\mathrm{F}} B \Rightarrow A \leq_{\mathrm{iBF}} B$.

Proposition 4.25

1. $A \leq_{\mathrm{F}} B \Rightarrow A \leq_{\mathrm{iBF}} B$.

2. $A \leq_{\mathrm{B}} B \Rightarrow A \leq_{\mathrm{BF}} B$.

3. $A \leq_{\mathrm{iB}} B \Rightarrow A \leq_{\mathrm{iBF}} B$.

Theorem 4.26

1. $A \leq_{\mathrm{BF}} B \Rightarrow (\exists C : A \leq_{\mathrm{B}} C \leq_{\mathrm{F}} B)$.

2. $A \leq_{\mathrm{iBF}} B \Rightarrow (\exists C : A \leq_{\mathrm{iB}} C \leq_{\mathrm{F}} B)$.

Proof: Let g be a backward-forward simulation from A to B, which is image-finite if $A \leq_{\mathrm{iBF}} B$. Define C to be the automaton given by:

- $states(C) = range(g)$,
- $start(C) = range(g \lceil start(A))$,
- $acts(C) = acts(B)$, and
- for $S', S \in states(C)$ and $a \in acts(C)$, $S' \xrightarrow{a}_C S \quad \Leftrightarrow \quad \forall u' \in S' \; \exists u \in S : u' \overset{\hat{a}}{\Longrightarrow}_B u$.

Then g is a backward simulation from A to C (and image-finiteness carries over). Also, the relation $\{(S, u) \mid S \in states(C) \text{ and } u \in S\}$ is a forward simulation from C to B. ■

In order to show the properties of backward-forward simulations, it is useful to relate them to forward-backward simulations.

Theorem 4.27

1. $A \leq_{\mathrm{BF}} B \Leftrightarrow A \leq_{\mathrm{FB}} B$.

2. $A \leq_{\mathrm{iBF}} B \Rightarrow A \leq_{\mathrm{iFB}} B$.

Proof: For one direction of 1, suppose that $A \leq_{\mathrm{BF}} B$. Then by Theorem 4.26, there exists an automaton C with $A \leq_{\mathrm{B}} C \leq_{\mathrm{F}} B$. By Proposition 4.18, $A \leq_{\mathrm{FB}} C$ and $C \leq_{\mathrm{FB}} B$. Now $A \leq_{\mathrm{FB}} B$ follows by Proposition 4.21. The proof of 2 is similar.

For the other direction of 1, suppose that f is a forward-backward simulation from A to B. Given a state s of A, we define $g[s]$ to be exactly the set of subsets S of $\mathit{states}(B)$ such that S intersects each set in $f[s]$ in at least one element. Then g is a backward-forward simulation. ∎

Example 4.5 In general it is not the case that $A \leq_{\mathrm{iFB}} B$ implies $A \leq_{\mathrm{iBF}} B$. A counterexample is presented in Figure 4. The diagram shows two automata I and J. In the diagram a label $> i$ next to an arc means that in fact there are infinitely many steps, labeled $i+1$, $i+2$, $i+3$, etc..

Figure 4: $I \leq_{\mathrm{iFB}} J$ but $I \not\leq_{\mathrm{iBF}} J$.

We claim that the relation g given by

$$\begin{aligned} g[0] &= \{\{0\}, \{0', 1\}, \{0', 1', 2\}, \ldots\} \\ g[n] &= \{\{\omega\}, \{\omega'\}\} \quad \text{for } n > 0 \end{aligned}$$

is an image-finite forward-backward simulation from I to J.

However, there is no image-finite backward-forward simulation from I to J. We see this as follows. Suppose g is an image-finite backward-forward simulation from I to J. In order to prove that this assumption leads to a contradiction, we first establish that $g[0]$ does not contain a finite subset X of N. First note that by the first condition in the definition of a backward-forward simulation, all sets in $g[0]$ are nonempty. The proof proceeds by induction on the maximal element of X. For the induction base, observe that $\{0\} \notin g[0]$, since 0 has an incoming 0-step in I but not in J. For the induction step, suppose that we have established that $g[0]$ contains no finite subset of N with a maximum less than n, and suppose $X \in g[0]$ with X a finite subset of N with maximum n. Using that 0 has an incoming 0-step in I, the second condition in the definition of a backward-forward simulation gives that $g[0]$ contains an element of $g[0]$ which is a subset of N with a maximum less than n. This contradicts the induction hypothesis.

Pick some state $n > 0$ of I and a set $S' \in g[n]$. Since $0 \xrightarrow{n}_I n$, there exists a set $S \in g[0]$ such that every state in S has an outgoing n-step. Then S must be a subset of $\{0, \ldots, n-1, (n-1)'\}$. Since $g[0]$ does not contain the empty set or a finite subset of N, it follows that $(n-1)' \in S$. But since n was chosen arbitrarily (besides being positive) it follows that $g[0]$ has an infinite number of elements. This gives a contradiction with the assumption that g is image-finite.

Proposition 4.28 $\leq_{\mathrm{BF}}$ *is a preorder.*

Proof: Trivially implied by Theorem 4.27 and Proposition 4.21. ■

However, the counterexample of Figure 4 tells us that $\leq_{\mathrm{iBF}}$ is not a preorder in general. If we take the two automata I and J from the example, then we can find an automaton C with $I \leq_{\mathrm{F}} C \leq_{\mathrm{iB}} J$, using Theorem 4.16. By Proposition 4.25, $I \leq_{\mathrm{iBF}} C$ and $C \leq_{\mathrm{iBF}} J$. Hence it cannot be that $\leq_{\mathrm{iBF}}$ is transitive, because this would imply $I \leq_{\mathrm{iBF}} J$.

Soundness and completeness results for backward-forward simulations now follow from those for forward-backward simulations.

Theorem 4.29 *(Soundness of backward-forward simulations)*

1. $A \leq_{\mathrm{BF}} B \Rightarrow A \leq_{*\mathrm{T}} B$.

2. $A \leq_{\mathrm{iBF}} B \Rightarrow A \leq_{\mathrm{T}} B$.

Proof: By Theorems 4.27 and 4.22. ■

Theorem 4.30 *(Completeness of backward-forward simulations)* $A \leq_{*\mathrm{T}} B \Rightarrow A \leq_{\mathrm{BF}} B$.

Proof: By Theorems 4.23 and 4.27. ■

Example 4.5 falsifies the completeness result that one might expect here. That is, Theorem 4.30 does not have a second case saying that if B has fin and $A \leq_{*\mathrm{T}} B$, then $A \leq_{\mathrm{iBF}} B$.

4.4 Auxiliary Variable Constructions

In this subsection, we present two new types of relations, history relations and prophecy relations, which correspond to the notions of history variable and prophecy variable of Abadi and Lamport [1]. We show that there exists a close connection between history relations and forward simulations, and also between prophecy relations and backward simulations. Using these connections together with the earlier results of this section, we can easily derive a completeness theorem for refinements similar to the one of Abadi and Lamport [1]. In fact, in the setting of this paper the combination of history and prophecy relations and refinements gives exactly the same verification power as the combination of forward and backward simulations.

4.4.1 History Relations

A relation h over $states(A)$ and $states(B)$ is a *history relation* from A to B if h is a forward simulation from A to B and h^{-1} is a refinement from B to A. We write $A \leq_{\mathrm{H}} B$ if there exists a history relation from A to B.

We give an example of a history relation, using the construction of the "unfolding" of an automaton; the unfolding of an automaton augments the automaton by remembering information about the past.

Definition 4.2 The *unfolding* of an automaton A, notation $unfold(A)$, is the automaton B defined by

- $states(B) = execs^*(A)$,
- $start(B) =$ the set of finite executions of A that consist of a single start state,
- $acts(B) = acts(A)$, and
- for $\alpha', \alpha \in states(B)$ and $a \in acts(B)$, $\alpha' \xrightarrow{a}_B \alpha \quad \Leftrightarrow \quad \alpha = \alpha' \, a \, last(\alpha)$.

Proposition 4.31 *$unfold(A)$ is a forest and $A \leq_{\mathrm{H}} unfold(A)$.*

Proof: Clearly, $unfold(A)$ is a forest. The function $last$ which maps each finite execution of A to its last state is a refinement from $unfold(A)$ to A, and the relation $last^{-1}$ is a forward simulation from A to $unfold(A)$. ■

Example 4.6 For C, D, E, F as in Example 4.1, $C \not\leq_{\mathrm{H}} D$, $D \leq_{\mathrm{H}} C$, $E \not\leq_{\mathrm{H}} F$ and $F \not\leq_{\mathrm{H}} E$.

Proposition 4.32 *$\leq_{\mathrm{H}}$ is a preorder.*

Proof: Reflexivity is trivial. For transitivity, suppose h is a history relation from A to B and h' is a history relation from B to C. Then h is a forward simulation from A to B and h' is a forward simulation from B to C, so $h' \circ h$ is a forward simulation from A to C, by Proposition 4.7. Also, since h'^{-1} is a refinement from C to B and h^{-1} is a refinement from B to A, $(h' \circ h)^{-1} = h^{-1} \circ h'^{-1}$ is a refinement from C to A by Proposition 4.2. It now follows that $h' \circ h$ is a history relation from A to C. ■

The notion of a history relation is a new contribution of this paper. It provides a simple and abstract view of the *history variables* of Abadi and Lamport [1]. Translated to the setting of this paper history variables can be simply defined in terms of history relations, as follows.

Definition 4.3 An automaton B is obtained from an automaton A by *adding a history variable* if there exists a set V such that

- $states(B) \subseteq states(A) \times V$, and
- the relation $\{(s, (s, v)) \mid (s, v) \in states(B)\}$ is a history relation from A to B.

Whenever B is obtained from A by adding a history variable, then $A \leq_{\mathrm{H}} B$ by definition. The following proposition states that the converse is also true if one is willing to consider automata up to isomorphism.

Proposition 4.33 *Suppose $A \leq_{\mathrm{H}} B$. Then there exists an automaton C that is isomorphic to B and obtained from A by adding a history variable.*

Proof: Let h be a history relation from A to B. Define automaton C by

- $states(C) = h$,
- $(s, u) \in start(C) \quad \Leftrightarrow \quad u \in start(B)$,

- $acts(C) = acts(B)$, and
- for $(s', u'), (s, u) \in states(C)$ and $a \in acts(C)$, $(s', u') \xrightarrow{a}_C (s, u) \;\Leftrightarrow\; u' \xrightarrow{a}_B u$.

Clearly, the projection function π_2 that maps a state (s, u) of C to the state u of B is an isomorphism between C and B.

In order to show that C is obtained from A by adding a history variable, let $states(B)$ play the role of the set V required in the definition of a history variable. It is easy to check that relation $\{(s, (s, v)) \mid (s, v) \in states(C)\}$ is a history relation from A to C. ■

Proposition 4.33 shows that history relations already capture the essence of history variables. For this reason and also because history relations have nicer theoretical properties, we will state all our results in this subsection in terms of relations, and will not mention the auxiliary variables any further.

Theorem 4.34 *(Soundness of history relations)* $A \leq_{\mathrm{H}} B \Rightarrow A \equiv_{\mathrm{T}} B$.

Proof: Immediate from the soundness of refinements and forward simulations. ■

In fact, a history relation from A to B is just a functional *bisimulation* between A and B in the sense of Park [28] and Milner [23]. This implies that if there exists a history relation from A to B, both automata are *bisimulation equivalent*. Hence, history relations preserve the behavior of automata in a very strong sense.

Definition 4.4 Suppose k is a relation over $states(A)$ and $states(B)$ satisfying $k \cap (start(A) \times start(B)) \neq \emptyset$. (Typically, k will be a forward or a backward simulation.) The *superposition* $sup(A, B, k)$ of B onto A via k is the automaton C given by

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

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

Lemma 4.35 *Suppose f is a forward simulation from A to B. Let $C = sup(A, B, f)$ and let π_1 and π_2 be the projection functions that map states of C to their first and second components, respectively. Then π_1^{-1} is a history relation from A to C and π_2 is a refinement from C to B.*

Theorem 4.36 $A \leq_{\mathrm{F}} B \Leftrightarrow (\exists C : A \leq_{\mathrm{H}} C \leq_{\mathrm{R}} B)$.

Proof: For the implication "$\Rightarrow$", suppose $A \leq_{\mathrm{F}} B$. Let f be a forward simulation from A to B. Take $C = sup(A, B, f)$. The result follows by Lemma 4.35. For the implication "$\Leftarrow$", suppose that $A \leq_{\mathrm{H}} C \leq_{\mathrm{R}} B$. Then $A \leq_{\mathrm{F}} C$ by the definition of history relations, and $C \leq_{\mathrm{F}} B$ because any refinement is a forward simulation. Now $A \leq_{\mathrm{F}} B$ follows by the fact that $\leq_{\mathrm{F}}$ is a preorder. ■

4.4.2 Prophecy Relations

Now we will present prophecy relations and show that they correspond to backward simulations, very similarly to the way in which history relations correspond to forward simulations.

A relation p over $states(A)$ and $states(B)$ is a *prophecy relation* from A to B if p is a backward simulation from A to B and p^{-1} is a refinement from B to A. We write $A \leq_{\mathrm{P}} B$ if there exists a prophecy relation from A to B, and $A \leq_{\mathrm{iP}} B$ if there is an image-finite prophecy relation from A to B. We give an example of a prophecy relation, using the construction of the "guess" of an automaton. This construction is a kind of dual to the unfolding construction of the previous subsection in that the states contain information about the future rather than about the past.

Definition 4.5 The *guess* of an automaton A, notation $guess(A)$, is the automaton B defined by

- $states(B) = frag^*(A)$,
- $start(B) = execs^*(A)$,
- $acts(B) = acts(A)$, and
- for $\alpha', \alpha \in states(B)$ and $a \in acts(B)$, $\alpha' \xrightarrow{a}_B \alpha \;\Leftrightarrow\; first(\alpha')\, a\, \alpha = \alpha'$.

Proposition 4.37 *$A \leq_{\mathrm{P}} guess(A)$.*

Proof: The function $first$ which maps each execution fragment of A to its first state is a refinement from $guess(A)$ to A, and the relation $first^{-1}$ is a backward simulation from A to $guess(A)$. ■

Example 4.7 For the automata of Figure 2 we have $C \not\leq_{\mathrm{P}} D$, $D \not\leq_{\mathrm{P}} C$, $E \not\leq_{\mathrm{P}} F$ and $F \leq_{\mathrm{iP}} E$. The difference between $\leq_{\mathrm{P}}$ and $\leq_{\mathrm{iP}}$ is illustrated by the automata of Figure 3: $G \leq_{\mathrm{P}} H$ but $G \not\leq_{\mathrm{iP}} H$.

Proposition 4.38 *$\leq_{\mathrm{P}}$ and $\leq_{\mathrm{iP}}$ are preorders.*

Just as history relations capture the essence of history variables, prophecy relations capture the essence of prophecy variables:

Definition 4.6 An automaton B is obtained from an automaton A by *adding a prophecy variable* if there exists a set V such that

- $states(B) \subseteq states(A) \times V$, and
- the relation $\{(s,(s,v)) \mid (s,v) \in states(B)\}$ is a prophecy relation from A to B.

A prophecy variable is *bounded* if the underlying prophecy relation is image-finite.

Proposition 4.39 *Suppose $A \leq_{\mathrm{P}} B$. Then there exists an automaton C that is isomorphic to B and obtained from A by adding a prophecy variable, which is bounded if $A \leq_{\mathrm{iP}} B$.*

Again, we will state all further results in this subsection in terms of relations, and not mention the auxiliary variables any further.

Theorem 4.40 *(Soundness of prophecy relations)*

1. *$A \leq_{\mathrm{P}} B \Rightarrow A \equiv_{*\mathrm{T}} B$, and*
2. *$A \leq_{\mathrm{iP}} B \Rightarrow A \equiv_{\mathrm{T}} B$.*

Proof: Immediate from the soundness of refinements and backward simulations. ■

Lemma 4.41 *Suppose b is a backward simulation from A to B. Let $C = sup(A, B, b)$ and let π_1 and π_2 be the projection functions that map states of C to their first and second components, respectively. Then π_1^{-1} is a prophecy relation from A to C and π_2 is a refinement from C to B. If b is image-finite then so is π_1^{-1}.*

Theorem 4.42

1. *$A \leq_{\mathrm{B}} B \Leftrightarrow (\exists C : A \leq_{\mathrm{P}} C \leq_{\mathrm{R}} B)$,*
2. *$A \leq_{\mathrm{iB}} B \Leftrightarrow (\exists C : A \leq_{\mathrm{iP}} C \leq_{\mathrm{R}} B)$.*

Proof: The proof of 1 is analogous to that of Theorem 4.36, using Lemma 4.41. 2 can be proved similarly. ■

We finish this section with versions of the completeness results of [1].

Theorem 4.43 *(Completeness of history relations, prophecy relations and refinements, [1]) Suppose $A \leq_{*\mathrm{T}} B$. Then*

1. *$\exists C, D : A \leq_{\mathrm{H}} C \leq_{\mathrm{P}} D \leq_{\mathrm{R}} B$, and*
2. *if B has fin then $\exists C, D : A \leq_{\mathrm{H}} C \leq_{\mathrm{iP}} D \leq_{\mathrm{R}} B$.*

Proof: By Theorem 4.16, there exists an automaton E with $A \leq_{\mathrm{F}} E \leq_{\mathrm{B}} B$. Hence, by Theorem 4.36, there is an automaton C with $A \leq_{\mathrm{H}} C \leq_{\mathrm{R}} E$. Combining $C \leq_{\mathrm{R}} E$ and $E \leq_{\mathrm{B}} B$ yields $C \leq_{\mathrm{B}} B$. Theorem 4.42 yields an automaton D with $C \leq_{\mathrm{P}} D \leq_{\mathrm{R}} B$, which proves 1. Now statement 2 is routine. ■

Similarly, we obtain:

Theorem 4.44 *$A \leq_{*\mathrm{T}} B \Rightarrow \exists C, D : A \leq_{\mathrm{P}} C \leq_{\mathrm{H}} D \leq_{\mathrm{R}} B$.*

$$\begin{tikzcd}
 & & \le_{\mathrm{iP}} \arrow[r] \arrow[d] & \le_{\mathrm{P}} \arrow[d] \\
 & \le_{\mathrm{R}} \arrow[r] \arrow[d] & \le_{\mathrm{iB}} \arrow[r] \arrow[d] & \le_{\mathrm{B}} \arrow[d] \\
\le_{\mathrm{H}} \arrow[r] & \le_{\mathrm{F}} \arrow[r] & \le_{\mathrm{iBF}} \arrow[r] \arrow[d] & \le_{\mathrm{BF}} \arrow[d, shift right] \\
 & & \le_{\mathrm{iFB}} \arrow[r] \arrow[d, shift right] & \le_{\mathrm{FB}} \arrow[u, shift right, line width=1pt] \arrow[d, shift right] \\
 & & \le_{\mathrm{T}} \arrow[u, shift right, line width=1pt] \arrow[r, shift right] & \le_{*\mathrm{T}} \arrow[u, shift right, line width=1pt] \arrow[l, shift right, line width=1pt]
\end{tikzcd}$$

Figure 5: Classification of basic relations between automata.

4.5 Classification of Basic Relations Between Automata

We can summarize the basic implications between the various simulation techniques of this section as follows. Suppose $X, Y \in \{\mathrm{T}, *\mathrm{T}, \mathrm{R}, \mathrm{F}, \mathrm{iB}, \mathrm{B}, \mathrm{iFB}, \mathrm{FB}, \mathrm{iBF}, \mathrm{BF}, \mathrm{H}, \mathrm{iP}, \mathrm{P}\}$. Then $A \le_{\mathrm{X}} B \Rightarrow A \le_{\mathrm{Y}} B$ for all automata A and B if and only if there is a path from $\le_{\mathrm{X}}$ to $\le_{\mathrm{Y}}$ in Figure 5 consisting of thin lines only. If B has fin, then $A \le_{\mathrm{X}} B \Rightarrow A \le_{\mathrm{Y}} B$ for all automata A and B if and only if there is a path from $\le_{\mathrm{X}}$ to $\le_{\mathrm{Y}}$ consisting of thin lines and thick lines.

5 Timed Automata and Their Behaviors

This section presents the basic definitions and results for timed automata. The development is generally parallel to that in Section 3.

5.1 Timed Automata

A *timed automaton* A is an automaton whose set of actions is a superset of $\{\tau\} \cup \mathsf{R}^{\geq 0}$, and whose step relation satisfies a number of axioms that will be presented below. The actions in $\mathsf{R}^{\geq 0}$ are referred to as the *time-passage* actions. (Each action $t \in \mathsf{R}^{\geq 0}$ represents the passage of time exactly up to real time t.) The set of *visible* actions is defined by $\mathit{vis}(A) \triangleq \mathit{acts}(A) - (\{\tau\} \cup \mathsf{R}^{\geq 0})$.

The first axiom a timed automaton A has to satisfy is:

T1 For each state s there is a unique $t \in \mathsf{R}^{\geq 0}$ such that $s \xrightarrow{t} s$.

We call the steps introduced by axiom **T1** *idling* steps. The intended meaning of an idling step $s \xrightarrow{t} s$ is that the current time in state s is t. In this case we write $s.\mathit{time}_A = t$, or $s.time = t$ if A is clear from the context. Instead of the idling steps, we could have included the mapping $.time_A$ as a basic component of a timed automaton. Formally this would have been equivalent, but we prefer the present formulation for technical reasons.

We assume further that a timed automaton A satisfies the following restrictions on individual steps.

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

S2 If $s' \xrightarrow{a} s$ and $a \notin \mathsf{R}^{\geq 0}$, then $s'.time = s.time$.

S3 If $s' \xrightarrow{a} s$ and $a \in \mathsf{R}^{\geq 0}$, then $s'.time \leq a = s.time$.

S4 If $s' \xrightarrow{a} s$ and $a = s'.time$, then $s' = s$.

Axiom **S1** says that the time is always 0 in a start state. Axiom **S2** says that non-time-passage actions do not change the time; that is, they occur "instantaneously", at a single point in time. Axiom **S3** says that time-passage actions may not cause the time to decrease; the label of the transition refers to the time in the final state. Axiom **S4** implies that the only time-passage steps that does not cause the time to increase are the idling steps.

We also require that A include a sufficiently rich collection of time-passage steps:

T2 If $s' \xrightarrow{t} s''$ and $s'' \xrightarrow{t'} s$, then $s' \xrightarrow{t'} s$.

T3 If $s' \xrightarrow{t} s$ and $s'.time < t' < t$, then there is an s'' with $s''.time = t'$ such that $s' \xrightarrow{t'} s''$ and $s'' \xrightarrow{t} s$.

Axiom **T2** allows repeated time-passage steps to be combined into one step, and axiom **T3** says that if time can pass to some time t, it can also pass to t in two steps, via any intermediate time t'.

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

Suppose α is an execution fragment of A. Then $\alpha.ftime$ denotes the time component of the first state in α, and $\alpha.ltime$ denotes the smallest element of $\mathsf{R}^{\geq 0} \cup \{\infty\}$ larger or equal than (i.e., the supremum of) the time components of all states in α. In particular, if α is an execution, then $\alpha.ftime = 0$, and if α is a finite execution fragment, then $\alpha.ltime = last(\alpha).time$.

5.2 Admissible Executions and Feasibility

Timed automata do not include any features for describing liveness or fairness (such as the class structure of I/O automata). We believe that such features are not so important in the timed setting as they are in the untimed setting. In fact, we think that by simply requiring that time grow unboundedly in infinite executions, we will be able to handle the liveness properties that arise in practice. Thus, in our study of timed automata, we concentrate on the *admissible* executions and execution fragments, i.e., those in which the time components of the states increase without bound. So α is an admissible execution fragment iff $\alpha.ltime = \infty$.

The notion of an admissible execution is more tractable mathematically than the notion of a fair execution in the I/O automaton model; this is because the admissible executions of a timed automaton are exactly the limits of the infinite sequences of finite executions, where each execution in the sequence is a prefix of the next and the time components of the states go to ∞. This characterization permits the reduction of questions about infinite behaviors to questions about finite behaviors. A similar reduction is not possible in untimed models that incorporate fairness.

The idea behind the notion of admissible executions is that time is an independent force, beyond the control of any automaton, which happens to grow unboundedly. We note that, according to our definitions, there are timed automata in which from some (or even all) states no admissible execution fragment is possible. This can either be because from these states onwards time cannot advance at all (that is, a *time deadlock* occurs), or because time can continue advancing, but not beyond a certain point (that is, all executions are so-called *Zeno executions*). The possibility of time deadlocks occurs in several process algebraic models ([2, 7, 24]) but we have no intuition whatsoever about what it means to "stop time". Zeno executions arise due to the inability of automaton models to deal with an infinite amount of activity within a bounded period of time. Some models of real-time computation, for instance the model of real-time CSP [29], exclude Zeno executions altogether. As a result of our attempt to make our results as general as possible, our model does allow for both time deadlocks and Zeno executions. However, in several of our theorems we will require that the automata be *feasible*. A timed automaton A is *feasible* provided that each finite execution is a prefix of some admissible execution. Thus, a feasible timed automaton does not have time deadlocks, but may have Zeno executions.

5.3 Timed Traces

The traces of timed automata do not provide a sufficiently abstract view of their behavior, because they do not reflect the invisible nature of time-passage actions. We illustrate this via the following key example.

Example 5.1 Consider two timed automata, *Idle* and *Idle′*. Automaton *Idle* does nothing except that it lets time pass. Its state set is just $\mathsf{R}^{\geq 0}$, the start state is 0, the set of actions is $\{\tau\} \cup \mathsf{R}^{\geq 0}$, and there is a transition $t' \xrightarrow{t} t$ whenever $t' \leq t$. Automaton *Idle′* is also rather boring: it idles all the time except that it does a τ-step at time 37. In fact, we like to argue that from an observational point of view *Idle′* is just as boring as *Idle* since both automata do not engage in any interaction with their environment at all. Formally, timed automaton *Idle′* is defined as follows:

- $states(Idle') = \mathsf{R}^{\geq 0} \times \{\mathsf{T}, \mathsf{F}\}$,
- $start(Idle') = \{(0, \mathsf{T})\}$,
- $acts(Idle') = \{\tau\} \cup \mathsf{R}^{\geq 0}$, and
- $steps(Idle')$ is specified by:

$$(t', b) \xrightarrow{t} (t, b) \qquad \text{if } t' = t \vee (t' < t \wedge (b = \mathsf{T} \Rightarrow t \leq 37))$$

$$(37, \mathsf{T}) \xrightarrow{\tau} (37, \mathsf{F})$$

Although $Idle' \leq_{\mathrm{T}} Idle$, it is not the case that $Idle \leq_{\mathrm{T}} Idle'$. This is because $Idle$ has a trace consisting of 38 only, which $Idle'$ does not have. (Note that $Idle'$ *does* have a trace 37 38.) So if we would use $\leq_{\mathrm{T}}$ as an implementation relation it would not be allowed to implement a specification that only requires an internal (unobservable) step at time 37 by a device that does nothing at all. It is for this reason that we consider $\leq_{\mathrm{T}}$ not to be a good implementation relation.

In this subsection, we define an alternative notion of external behavior for timed automata that does not include explicit individual time-passage actions. We describe the external behavior of timed automata in terms of observations that we call *timed traces*; these contain information about the visible actions that occur, together with their time of occurrence, and also about the final time up to which the observation is made. Along the way to the definition of a *timed trace*, it is helpful to define the basic notion of a *timed sequence pair*.

5.3.1 Timed Sequence Pairs

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

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

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

Lemma 5.1 *$\leq$ is a partial ordering on timed sequence pairs over K.*

5.3.2 Timed Traces of Timed Automata

Suppose $\alpha = s_0 a_1 s_1 a_2 s_2 \cdots$ is an execution fragment of a timed automaton A. Let γ be the sequence consisting of the actions in α paired with the time of their occurrence:

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

Then $t\text{-}trace(\alpha)$ is defined to be the pair

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

So $t\text{-}trace(\alpha)$ records the occurrences of external actions together with their time of occurrence, and the limit time of the execution fragment. We call $t\text{-}trace(\alpha)$ the *timed trace* *of* α. Thus, the timed trace of an execution fragment suppresses both τ and time-passage actions. It is easily checked that $\textit{t-trace}(\alpha)$ is a timed sequence pair over $\textit{vis}(A)$.

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

Proposition 5.2 *The sets $\textit{t-traces}^*(A)$ and $\textit{t-traces}^\infty(A)$ consist of finite timed sequence pairs and admissible timed sequence pairs over $\textit{vis}(A)$, respectively.*

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

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

Lemma 5.3 *Suppose $s' \overset{p}{\leadsto}_A s$ and $p = q; r$. Then there exists s'' such that $s' \overset{q}{\leadsto}_A s''$ and $s'' \overset{r}{\leadsto}_A s$.*

5.3.3 From Traces to Timed Traces

A trace preserves more information about an execution than a timed trace. Below we show how the timed trace of an execution fragment can be reconstructed from its starting time and its trace. This reconstruction will allow us to relate the untimed and the timed trace preorders.

Let K be some set with $\mathsf{R}^{\geq 0} \subseteq K$, and let $\beta = a_1 a_2 a_3 \cdots$ be a sequence over K. We say β is *monotonic* if the time-passage events contained in it increase monotonically, i.e., for all $i < j$, $a_i, a_j \in \mathsf{R}^{\geq 0} \Rightarrow a_i \leq a_j$. Clearly, each trace of a timed automaton A is a monotonic sequence over $\textit{ext}(A)$.

Let $\beta = a_1 a_2 a_3 \cdots$ be monotonic sequence over K, and let $t \in \mathsf{R}^{\geq 0}$ be less than or equal to all time-passage events in β. We define a timed sequence pair $\textit{t-pair}(t, \beta)$ in two steps. First, define for $i \in \mathsf{N}$, $t_i \in \mathsf{R}^{\geq 0}$ inductively by: $t_0 \triangleq t$ and if $a_i \in \mathsf{R}^{\geq 0}$ then $t_{i+1} \triangleq a_i$ else $t_{i+1} \triangleq t_i$. Let γ be the sequence $(a_1, t_1)(a_2, t_2)(a_3, t_3) \cdots$, and let t' be the supremum, in $\mathsf{R}^{\geq 0} \cup \{\infty\}$, of all the t_i's. Then

$$\textit{t-pair}(t, \beta) \triangleq (\gamma \lceil ((K - \mathsf{R}^{\geq 0}) \times \mathsf{R}^{\geq 0}), t').$$

We write $\textit{t-pair}(\beta)$ for $\textit{t-pair}(0, \beta)$. The following lemma relates the usual notion of trace to the new notion of timed trace.

Lemma 5.4 *Suppose $\alpha \in \textit{frag}(A)$. Then $\textit{t-trace}(\alpha) = \textit{t-pair}(\alpha.\textit{ftime}, \textit{trace}(\alpha))$.*

Corollary 5.5 *Suppose $\beta \in \textit{traces}^*(A)$. Then $\textit{t-pair}(\beta) \in \textit{t-traces}^*(A)$.*

Lemma 5.6

1. $A \leq_{*\mathrm{T}} B \Rightarrow A \leq^{\mathrm{t}}_{*\mathrm{T}} B$.

2. $A \leq_{\omega\mathrm{T}} B \Rightarrow A \leq^{\mathrm{t}}_{\infty\mathrm{T}} B$.

3. $A \leq_{\mathrm{T}} B \Rightarrow A \leq^{\mathrm{t}}_{\mathrm{T}} B$.

Proof: For 1, suppose that $A \leq_{*\mathrm{T}} B$ and $p \in \textit{t-traces}^*(A)$. Then A has a finite execution α with $p = \textit{t-trace}(\alpha)$. Let $\beta = \textit{trace}(\alpha)$. Then $\beta \in \textit{traces}^*(A)$ and, using $A \leq_{*\mathrm{T}} B$, also $\beta \in \textit{traces}^*(B)$. By Corollary 5.5, $\textit{t-pair}(\beta) \in \textit{t-traces}^*(B)$. But Lemma 5.4 yields $\textit{t-pair}(\beta) = p$. Thus $p \in \textit{t-traces}^*(B)$, and $A \leq^{\mathrm{t}}_{*\mathrm{T}} B$ follows as required.

For 2, suppose that $A \leq_{\omega\mathrm{T}} B$ and $p \in \textit{t-traces}^\infty(A)$. Then there is an admissible execution α of A such that $p = \textit{t-trace}(\alpha)$. Let $\beta = \textit{trace}(\alpha)$. Since $\beta \in \textit{traces}^\omega(A)$ and $A \leq_{\omega\mathrm{T}} B$, also $\beta \in \textit{traces}^\omega(B)$. Thus B has an infinite execution α' with $\textit{trace}(\alpha') = \beta$. Since α is admissible the time-passage actions contained in β grow unboundedly. But since $\textit{trace}(\alpha') = \beta$, this means α' is also admissible. Using Lemma 5.4 we derive:

$$\textit{t-trace}(\alpha) = \textit{t-pair}(\textit{trace}(\alpha)) = \textit{t-pair}(\beta) = \textit{t-pair}(\textit{trace}(\alpha')) = \textit{t-trace}(\alpha').$$

Thus $p = \textit{t-trace}(\alpha') \in \textit{t-traces}^\infty(B)$. It follows that $A \leq^{\mathrm{t}}_{\infty\mathrm{T}} B$.

Finally, 3 follows from the first two parts. ■

The automata *Idle* and *Idle'* of Example 5.1 illustrate that the reverse implications of the statements in the above lemma are not valid.

5.4 Restricted Kinds of Timed Automata

A has *t-finite invisible nondeterminism (t-fin)* if $\textit{start}(A)$ is finite, and for any state s' and any finite timed sequence pair p over $\textit{vis}(A)$, there are only finitely many states s such that $s' \overset{p}{\leadsto}_A s$.

Example 5.2 In order to illustrate the difference between fin and t-fin we define a timed automaton *Idle''*. Basically, automaton *Idle''* does nothing except that it lets time pass. However, at one nondeterministically chosen time $t > 0$, *Idle''* makes a τ-step, and then it subsequently remembers t. Formally, *Idle''* is defined as follows:

- $\textit{states}(\textit{Idle}'') = \mathsf{R}^{\geq 0} \times \mathsf{R}^{\geq 0}$,
- $\textit{start}(\textit{Idle}'') = \{(0,0)\}$,
- $\textit{acts}(\textit{Idle}'') = \{\tau\} \cup \mathsf{R}^{\geq 0}$, and
- $\textit{steps}(\textit{Idle}'')$ is specified by:

$$(t'', t) \xrightarrow{t'} (t', t) \quad \text{if } t'' \leq t',$$

$$(t, 0) \xrightarrow{\tau} (t, t) \quad \text{if } t > 0.$$

By induction on the length of the sequence of time-passage actions one can easily establish that $Idle''$ has fin. However, $Idle''$ does not have t-fin since for any $t \in \mathsf{R}^{+}$ all the states from the uncountable set $\{(t,t')|0 < t' \leq t\}$ can be reached with timed sequence pair (λ, t) from the initial state.

The requirement that a timed automaton be a forest is not a very interesting one because if a state has one incoming time-passage step from a state with a smaller time component, then it must have an infinite number of them (as a consequence of axiom **T3**) so that the timed automaton cannot be a forest. Now a forest is characterized by the property that for each state there is a unique execution that leads to it. In analogy we will define below the notion of a t-forest. This is a timed automaton with the property that for each state there is an execution that leads to it, which is unique "modulo" axioms **T1**, **T2** and **T3**.

Call a finite execution fragment of A *fat* if it contains no idling steps and no pair of consecutive time-passage steps. Then A is defined to be a *t-forest* if for each state of A there is a unique fat execution that leads to it. The following lemma gives a sufficient condition for a timed automaton to be a t-forest.

Lemma 5.7 *Suppose that all states of A are reachable, the only incoming steps of start states are idling steps, and for every state s, if there are two distinct non-idling steps leading to s, $r \xrightarrow{a} s$ and $r' \xrightarrow{a'} s$, then $a = a' \in \mathsf{R}^{\geq 0}$ and either $r \xrightarrow{t'} r'$ or $r' \xrightarrow{t} r$, where $t = r.time$ and $t' = r'.time$. Then A is a t-forest.*

Proof: Suppose A satisfies the conditions of the lemma. Because all states of A are reachable we know that for each state s there is at least one execution that leads to it. Since we can remove idling steps from an execution, and since by repeated application of axiom **T2** we can contract successive time-passage steps, there is at least one fat execution that leads to s. In order to show uniqueness, suppose that A has two fat executions, α and α' that lead to s. By induction on the sum of the lengths of α and α', using the fact that A is a t-forest, we prove that $\alpha = \alpha'$.

If α consists of state s only, then so does α', and vice versa. We see this as follows. If α consists of s only, then s is a start state. Because A is a t-forest, there are no incoming non-idling steps to s. But since α' contains no idling steps and ends with s, α' must also consist of s only. So if either α or α' contains no steps, then $\alpha = \alpha'$.

Now suppose that $\alpha = \gamma a s$ and $\alpha' = \gamma' a' s$. Let $last(\gamma) = r$, $last(\gamma') = r'$, $r.time = t$ and $r'.time = t'$. If $r = r'$, then $\gamma = \gamma'$ by induction hypothesis, $a = a'$ since A is a t-forest, and hence $\alpha = \alpha'$. So assume that $r \neq r'$. We derive a contradiction. In this case $r \xrightarrow{a} s$ and $r' \xrightarrow{a'} s$ are two distinct steps ending with s, so since A is a t-forest, it must be that a and a' are both time-passage actions and either $r \xrightarrow{t'} r'$ or $r' \xrightarrow{t} r$. Without loss of generality, we may assume that the former holds.

Now consider the execution $\gamma'' = \gamma t' r'$. Since α is fat, γ'' must be fat too. Thus γ'' and γ' are two fat executions leading to r', and by induction hypothesis we obtain $\gamma'' = \gamma'$. But since γ'' ends with a time-passage step, this means that α' ends with two time-passage steps, which is in contradiction with the fact that this execution is fat.

Thus we have shown that for each state of A there is a unique fat execution that leads to it, which means that A is a t-forest. ■

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

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

The relation *t-past*$(A) \triangleq$ *t-after*$(A)^{-1}$ relates a state s of A to the timed traces of executions that lead to s. Also, define *t-before*(A) to be the relation that relates a timed sequence pair p to those states of A from where an execution with timed trace p is possible.

$$\textit{t-before}(A) \triangleq \{(p,s) \mid \exists \alpha \in frag^*(A) : \textit{t-trace}(\alpha) = p \text{ and } first(\alpha) = s\}.$$

We write *t-future*(A) for *t-before*$(A)^{-1}$.

Lemma 5.8

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

2. *If A has t-fin then t-after(A) is image-finite.*

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

Proof: For 1, suppose A is deterministic. By definition, *t-after*$(A)[p]$ contains at least one element for each $p \in$ *t-traces*$^*(A)$. Suppose that for some $p \in$ *t-traces*$^*(A)$, both s and s' are in *t-after*$(A)[p]$. Then A has finite executions α and α' with *t-trace*$(\alpha) =$ *t-trace*$(\alpha') = p$, $last(\alpha) = s$ and $last(\alpha') = s'$. Without loss of generality we may assume that both α and α' contain no idling steps. Since α and α' have the same timed trace, $s.time = s'.time$. Since A is deterministic, there are no τ events in either execution.

By induction on the sum of the lengths of α and α' we prove that $s = s'$. If α consists of state s only, then s is the unique start state of A and therefore also the first state of α'. As noted above, α' does not contain τ events. Moreover, α' does not contain any events in $ext(A)$ because that would violate the condition that *t-trace*$(\alpha) =$ *t-trace*(α'). Thus α' contains no events at all and consists of state s only. But this implies $s = s'$, as required. By a symmetric argument it also follows that $s = s'$ if we start from the assumption that α' consists of s' only.

Now suppose $\alpha = \gamma a s$ and $\alpha' = \gamma' a' s'$, with $last(\gamma) = r$ and $last(\gamma') = r'$. Let $t = r.time$ and $t' = r'.time$. Since a and a' are either time-passage actions or visible actions, and α and α' have the same timed trace and end at the same time, it cannot be that one of a and a' is an external action and the other is a time-passage action. In fact, it must be the case that $a = a'$. If $t = t'$, then *t-trace*$(\gamma) =$ *t-trace*(γ'). This means we can apply the induction hypothesis to obtain $r = r'$. Then in combination with the fact that A is deterministic, this gives $s = s'$. Otherwise, we can assume without loss of generality that $t < t'$. (The other case is symmetric.) In this case, it must be that a and a' are both time-passage actions, and so $t' < s.time$. By axiom **T2**, there is an r'' such that $r \xrightarrow{t'} r''$ and $r'' \xrightarrow{s.time} s$. Since *t-trace*$(\gamma t' r'') =$ *t-trace*(γ'), we can apply the induction hypothesis to obtain $r'' = r'$. Now $s = s'$ follows by the fact that A is deterministic.

Parts 2 is immediate 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-past*$(A)[s]$ contains at least one element. Suppose that both

p and p' are in *t-past*$(A)[s]$ for some $s \in$ *states*(A). Then there are finite executions α and α' of A with *t-trace*$(\alpha) = p$, *t-trace*$(\alpha') = p'$ and *last*$(\alpha) =$ *last*$(\alpha') = s$. Without loss of generality we can assume that both α and α' are fat, since we can always remove idling steps, and by repeatedly applying axiom **T1** we can eliminate successive time-passage steps, and this does not influence the timed traces of the executions. But now the assumption that A is a t-forest gives $\alpha = \alpha'$. This immediately implies $p = p'$. ■

5.5 Timed Trace Properties

For each timed automaton A, its *timed behavior*, *t-beh*(A), is defined by *t-beh*$(A) \triangleq$ (*vis*(A), *t-traces*(A)). Completely analogous to the way in which we characterized, in Section 3.3, the behaviors of automata in terms of trace properties, we will now characterize the timed behaviors of timed automata in terms of *timed trace properties*.

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

Lemma 5.9 *Suppose P and Q are timed trace properties with Q limit-closed. Then $P \leq^{\rm t}_{*\rm T} Q \Leftrightarrow P \leq^{\rm t}_{\rm T} Q$.*

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

Lemma 5.10 *Suppose P and Q are timed trace properties such that P is feasible. Then $P \leq^{\rm t}_{\infty\rm T} Q \Leftrightarrow P \leq^{\rm t}_{\rm T} Q$.*

Lemma 5.11

1. *t-beh(A) is a timed trace property.*

2. *If A has t-fin then t-beh(A) is limit-closed.*

3. *If A is feasible then t-beh(A) is feasible.*

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

Proof: We sketch the proof of 2; it is analogous to that of Lemma 3.3. Suppose A has t-fin and $p_1 \leq p_2 \leq \ldots$ is an infinite chain of timed sequence pairs in *t-traces*$^*(A)$ such that the limits of the time components of the p_i's is ∞. Assume without loss of generality that $p_i.ltime < p_{i+1}.ltime$ for all $i \geq 1$. Let p be the limit of the p_i's. We must show that $p \in$ *t-traces*$^\infty(A)$.

We use Lemma 2.1. 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 there exists an execution α of A that ends with state s and such that $p_i =$ *t-trace*(α). There is an edge from node (p_i, s') to node (p_{i+1}, s) exactly if $s' \overset{q}{\rightsquigarrow}_A s$, where $p_{i+1} = p_i; q$. Using Lemma 5.3, it is not difficult to show that G satisfies the hypotheses of Lemma 2.1. Then that lemma implies the existence of an infinite path in G starting at a root; this yields an admissible execution of A having p as its timed trace. ■

Proposition 5.12

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 5.9 and 5.11. Part 2 is a corollary of Lemmas 5.10 and 5.11. ■

Example 5.3 We present two timed automata, TA and TB, which are in a sense the timed analogues of the automata A and B of Example 3.1, that illustrate the necessity of the t-fin condition in Proposition 5.12. Automaton TA does an a-action at integer times:

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

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

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

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

- $states(TB) = \mathsf{R}^{\geq 0} \times \mathsf{N} \times \mathsf{N}$,
- $start(TB) = \{(0, 0, m) \mid m \in \mathsf{N}\}$,
- $acts(TB) = \{\tau, a\} \cup \mathsf{R}^{\geq 0}$, and
- $steps(TB)$ is specified by:

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

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

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

In order to see that the feasibility condition in Proposition 5.12(2) is actually needed, we consider a Zeno machine: a timed automaton *Zeno* with states drawn from the interval $[0, 1)$, start state 0, actions from $\mathsf{R}^{\geq 0} \cup \{\tau\}$, and a step $t' \xrightarrow{t} t$ whenever $t' \leq t$. Since *Zeno* has no admissible timed traces, $Zeno \leq^{\mathrm{t}}_{\infty\mathrm{T}} AT$. However, because AT does not allow for initial (nontrivial) time-passage steps, $Zeno \not\leq^{\mathrm{t}}_{\mathrm{T}} AT$.

Definition 5.1 For P a timed trace property, the associated *canonical timed automaton* $t\text{-}can(P)$ is the structure A given by

- $states(A) = t\text{-}traces^*(P)$,
- $start(A) = \{(\lambda, 0)\}$,
- $acts(A) = sort(P) \cup (\{\tau\} \cup \mathsf{R}^{\geq 0})$, and
- $steps(A)$ consists of all triples of the form $(\delta', t') \xrightarrow{a}_A (\delta, t)$, where $(\delta, t'), (\delta, t) \in states(A)$, $a \in ext(A)$, and where the following hold. If $a \in \mathsf{R}^{\geq 0}$ then $t' \leq t$ and $\delta' = \delta$, and if $a \in vis(A)$ then $t' = t$ and $\delta' \, (a, t) = \delta$.

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

Lemma 5.13 *Suppose P is a timed trace property. Then*

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

Proof: 1 and 2 follow easily from the definitions. Since $t\text{-}can(P)$ has t-fin, it follows by Lemma 5.11 that $t\text{-}beh(t\text{-}can(P))$ is limit-closed. Now 3 and 4 follow by combination of 2 and Lemma 5.9. ■

Lemma 5.14 *Suppose A and B are timed automata. Then*

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

Proof: By combining Lemmas 5.11 and 5.13. ■

6 Simulations for Timed Automata

Our aim is to develop proof techniques for showing inclusion between the sets of timed traces of timed automata. In order to do this, we show how this problem can be reduced to the problem of proving inclusion between the sets of traces of certain derived automata. This reduction solves our problem, in a sense, since it allows us to use the various simulation techniques in Section 4 to prove inclusion results for timed automata. The approach is analogous to that followed for Milner's CCS [23] where the problem of deciding weak observation equivalence is reduced to the problem of deciding strong bisimulation. A key role in our reduction is played by the construction of the closure of a timed automaton.

6.1 t-Closed Timed Automata

In the previous section we have shown that for timed automata the traces contain (in general) more information than the timed traces. That is, from the traces of a timed automaton we can retrieve the timed traces (Lemma 5.4), but the reverse is not always possible (Example 5.1). However, there exist certain classes of timed automata for which the traces *can* be retrieved from the timed traces. In this subsection, we will identify one such a class, namely the *t-closed* timed automata.

A timed automaton A is said to be *t-closed* provided that it satisfies the following closure condition:

T4 If $s' \xrightarrow{t} s''$, $s'.time < s''.time$ and $s'' \xrightarrow{\tau} s$, then $s' \xrightarrow{t} s$.

The timed automaton *Idle* of Example 5.1 is t-closed. The timed automata *Idle′* and *Idle″* of Example 5.1 and Example 5.2, respectively, are not t-closed.

In order to show that the (finite) traces of a t-closed timed automaton can be retrieved from its (finite) timed traces, we proceed in two steps. First we define an operation *prune* that associates to each montonic sequence a normal form. We then show (Lemma 6.1) that a sequence is a trace of a t-closed automaton if and only if its normal form is. Next, we define an operation *monot* that takes a timed sequence pair and transforms it into a monotonic sequence in normal form. We show (Lemma 6.2) that *prune* is nothing but the composition of *t-pair*, which takes a trace to a timed trace, and *monot*. In the same lemma we also prove that if p is a timed trace of a t-closed automaton, $monot(p)$ is a trace of that automaton. From these results it follows that $traces^*(A)$ can be retrieved from $t\text{-}traces^*(A)$: $traces^*(A)$ consists of all the monotonic sequences whose normal form equals $monot(p)$ for some timed trace p of A.

Let β be a finite monotonic sequence over some set K, and let $t \in \mathsf{R}^{\geq 0}$ be less than or equal to all time-passage actions in β. Then $prune(t, \beta)$ is the monotonic sequence obtained from β by (1) removing all time-passage events that are either t or preceded (not necessarily directly) by a time-passage event with the same value, and (2) removing all time-passage events that are immediately followed by a time-passage event with a higher value. We write $prune(\beta)$ for $prune(0, \beta)$.

Lemma 6.1 *Suppose A is t-closed, $s', s \in states(A)$, $t = s'.time$, β is a finite monotonic sequence over $ext(A)$ with t less or equal than all time-passage actions in β, and $\beta' = prune(t, \beta)$. Then $s' \overset{\beta}{\Longrightarrow} s$ iff $s' \overset{\beta'}{\Longrightarrow} s$.*

Proof: Suppose $s' \overset{\beta}{\Longrightarrow} s$. Then there exists a finite execution fragment, α, of A with $\mathit{first}(\alpha) = s'$, $\mathit{trace}(\alpha) = \beta$ and $\mathit{last}(\alpha) = s$. Let α_1 be the execution obtained from α by removing all idling steps, and let $\mathit{trace}(\alpha_1) = \beta_1$. By axiom **S4** all instanteneous time-passage steps are idling steps. Using this fact, it is easy to see that the idling steps in α are in one-one correspondence with the time-passage events in β that are either t or preceded by a time-passage event with the same value. This means that β_1 is the sequence obtained from β by applying pruning step (1). Next, let α_2 be the execution fragment obtained from α_1 by eliminating, through application of axiom **T4**, all τ-steps that are immediately preceded by a time-passage step. This transformation does not affect the trace of the execution fragment: $\mathit{trace}(\alpha_2) = \beta_1$. Also both α_1 and α_2 start in s' and end in s. Finally, let α_3 be the execution fragment obtained from α_2 by contracting all successive time-passage steps through application of axiom **T2**. Let $\beta_3 = \mathit{trace}(\alpha_3)$. Since β_3 does not contain successive time-passage steps, it is obtained from β_1 by applying pruning operation (2). Thus in fact we have $\beta_3 = \mathit{prune}(t, \beta) = \beta'$. Since α_3 also leads from s' to s, this implies $s' \overset{\beta'}{\Longrightarrow} s$.

Conversely, suppose $s' \overset{\beta'}{\Longrightarrow} s$. Then there exists a finite execution fragment, α, of A with $\mathit{first}(\alpha) = s'$, $\mathit{trace}(\alpha) = \mathit{prune}(t, \beta)$ and $\mathit{last}(\alpha) = s$. Using axioms **T1** and **T3**, we can simply insert additional time-passage steps in α to obtain another finite execution, α', of A with $\mathit{first}(\alpha') = s'$, $\mathit{trace}(\alpha') = \beta$ and $\mathit{last}(\alpha') = s$. Therefore, $s' \overset{\beta}{\Longrightarrow} s$. ■

Let $t \in \mathsf{R}^{\geq 0}$ and let $p = ((a_1, t_1) \cdots (a_n, t_n), t_{n+1})$ be a finite timed sequence pair over K with $t \leq p.\mathit{ftime}$. Then the monotonic sequence $\mathit{monot}(t, p)$ is obtained by taking the sequence $t_1 a_1 \cdots t_n a_n t_{n+1}$ and removing from it all t_i events that are either t or preceded by an event t_j that has the same value. We write $\mathit{monot}(p)$ for $\mathit{monot}(0, p)$.

Lemma 6.2

1. *Suppose $t \in \mathsf{R}^{\geq 0}$ and β is a finite monotonic sequence with t less or equal than all time-passage actions in β. Then $\mathit{prune}(t, \beta) = \mathit{monot}(t, \mathit{t\text{-}pair}(t, \beta))$.*

2. *Suppose A is t-closed, $s', s \in \mathit{states}(A)$, $t = s'.\mathit{time}$, p is a finite timed sequence pair over $\mathit{vis}(A)$, and $\beta = \mathit{monot}(t, p)$. Then $s' \overset{p}{\rightsquigarrow} s$ implies $s' \overset{\beta}{\Longrightarrow} s$.*

Proof: 1 easily follows from the definitions. For 2, suppose $s' \overset{p}{\rightsquigarrow} s$. Then A has an execution fragment α from s' to s with $\mathit{t\text{-}trace}(\alpha) = p$. Let $\beta' = \mathit{trace}(\alpha)$ and $\beta'' = \mathit{prune}(t, \beta')$. Then $s' \overset{\beta''}{\Longrightarrow} s$, by Lemma 6.1. Using 1 and Lemma 5.4, we derive $\beta'' = \mathit{prune}(t, \beta') = \mathit{monot}(t, \mathit{t\text{-}pair}(t, \beta')) = \mathit{monot}(t, p) = \beta$. Thus $s' \overset{\beta}{\Longrightarrow} s$. ■

Corollary 6.3 *Suppose B is t-closed. Then $A \leq^{\mathrm{t}}_{*\mathrm{T}} B \Leftrightarrow A \leq_{*\mathrm{T}} B$.*

Proof: Suppose $A \leq^{\mathrm{t}}_{*\mathrm{T}} B$ and $\beta \in \mathit{traces}^*(A)$. By Corollary 5.5, $\mathit{t\text{-}pair}(\beta) \in \mathit{t\text{-}traces}^*(A)$. Using $A \leq^{\mathrm{t}}_{*\mathrm{T}} B$, we get $\mathit{t\text{-}pair}(\beta) \in \mathit{t\text{-}traces}^*(B)$. By Lemma 6.2(2), $\mathit{monot}(\mathit{t\text{-}pair}(\beta)) \in \mathit{traces}^*(B)$ and by Lemma 6.2(1), $\mathit{monot}(\mathit{t\text{-}pair}(\beta)) = \mathit{prune}(\beta)$. Now $\beta \in \mathit{traces}^*(B)$ is a consequence of Lemma 6.1. It follows that $A \leq_{*\mathrm{T}} B$.

The converse direction follows from Lemma 5.6. ■

We also have the following property involving fin.

Lemma 6.4 *Suppose A is t-closed. Then A has t-fin if and only if A has fin.*

Proof: Suppose A does not have fin but does have t-fin. Then A has a state s' and a sequence β such that for infinitely many states s, $s' \overset{\beta}{\Longrightarrow} s$. Let $p = \textit{t-pair}(s'.\textit{time}, \beta)$. Then, by Lemma 5.4, $s' \overset{p}{\rightsquigarrow} s$ for infinitely many s. Thus A does not have t-fin, which is a contradiction.

For the other direction, assume that A does not have t-fin but does have fin. Then A has a state s' and a timed sequence pair p such that for infinitely many states s, $s' \overset{p}{\rightsquigarrow} s$. Let $\beta = \textit{monot}(s'.\textit{time}, p)$. Then it follows by Lemma 6.2(2) that $s' \overset{\beta}{\Longrightarrow} s$ for infinitely many s. Thus A does not have fin, which is again a contradiction. ■

6.2 Closure of a Timed Automaton

In this subsection, we give a useful construction to extend an arbitrary timed automaton to a t-closed timed automaton.

Let A be a timed automaton. The *closure* of A, denoted by $\textit{cl}(A)$, is the structure B which is exactly the same as A, except that the relation $\textit{steps}(A)$ is augmented by closing it under the closure condition given in **T4** (simultaneously with **T2** to let the result be a timed automaton again).

Lemma 6.5 *$\textit{cl}(A)$ is a t-closed timed automaton, $\textit{cl}(\textit{cl}(A)) = \textit{cl}(A)$ and $\textit{cl}(A) \equiv^{\mathrm{t}}_{\mathrm{T}} A$.*

Lemma 6.6

1. *A is deterministic if and only if $\textit{cl}(A)$ is deterministic.*

2. *A has t-fin if and only if $\textit{cl}(A)$ has fin.*

Proof: 1 is trivial. Since $\textit{cl}(A)$ is closed, $\textit{cl}(A)$ has fin if and only if it has t-fin (Lemma 6.4). But $\textit{cl}(A)$ has t-fin if and only if A has t-fin, since it is obvious from the definition of the closure operation that $s' \overset{p}{\rightsquigarrow}_{\textit{cl}(A)} s \Leftrightarrow s' \overset{p}{\rightsquigarrow}_A s$. ■

The importance of the closure construction is a consequence of the following lemmas.

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

Proof: By combination of Lemma 6.5 and Corollary 6.3. ■

Lemma 6.8 *$\textit{cl}(A) \leq_{\mathrm{T}} \textit{cl}(B) \Rightarrow A \leq^{\mathrm{t}}_{\mathrm{T}} B$.*

Proof: Suppose $\textit{cl}(A) \leq_{\mathrm{T}} \textit{cl}(B)$. Lemma 5.6 implies that $\textit{cl}(A) \leq^{\mathrm{t}}_{\mathrm{T}} \textit{cl}(B)$. Then Lemma 6.5 implies that $A \leq^{\mathrm{t}}_{\mathrm{T}} B$. ■

Example 6.1 The reverse implication of Lemma 6.8 does not hold in general. To obtain a counterexample: take B to be a machine, modeled as a timed automaton, which nondeterministically chooses a positive natural number n, then does action a at times $1 - 2^{-1}$, $1 - 2^{-2}$,..., $1 - 2^{-n}$, and then idles for ever. B is a feasible timed automaton with infinite invisible nondeterminism. Let A be the same as B, except that it may also choose ω at the beginning, in which case it subsequently does action a at times $1 - 2^{-1}$,

$1-2^{-2}$,..., $1-2^{-n}$,... Timed automaton A is not feasible because by choosing ω it reaches a state from where only a Zeno execution is possible and no admissible execution. Timed automata A and B have the same timed traces, but $cl(A)$ has an infinite trace $(a, 1-2^{-1})$, $(a, 1-2^{-2})$,..., $(a, 1-2^{-n})$,... which $cl(B)$ does not have.

It turns out that we *do* have the reverse of Lemma 6.8 in case 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: By Lemma 6.6 and Proposition 3.4, $cl(A) \leq_{\mathrm{T}} cl(B)$ iff $cl(A) \leq_{*\mathrm{T}} cl(B)$. By Lemma 6.7, $cl(A) \leq_{*\mathrm{T}} cl(B)$ is in turn equivalent to $A \leq_{*\mathrm{T}}^{\mathrm{t}} B$. Proposition 5.12 gives the equivalence of $A \leq_{*\mathrm{T}}^{\mathrm{t}} B$ and $A \leq_{\mathrm{T}}^{\mathrm{t}} B$. ■

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 the following statements are equivalent.

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

Proof: The equivalence of the first three statements follows by combining Lemma 6.7 with the soundness and completeness results for $\leq_{\mathrm{FB}}$ and $\leq_{\mathrm{BF}}$ (Theorems 4.22, 4.23, 4.29 and 4.30).

The equivalence of the last two statements follows by combining Lemma 6.9 and Lemma 6.6 with the soundness and completeness result for $\leq_{\mathrm{iFB}}$ (Theorems 4.22 and 4.23). ■

In a sense, we have solved our problem now: we have found a way to prove inclusion of the sets of timed traces of timed automata A and B, under the reasonable assumption that B has t-fin. All we have to do is to establish an image-2-finite forward-backward simulation between two closely related timed automata, $cl(A)$ and $cl(B)$. The timed automata $cl(A)$ and $cl(B)$ are really very similar to A and B: they are the same except for their step relations, which are just a kind of transitive closure of the step relations of A and B. Still, it would be more elegant to define the various simulations directly on the timed automata themselves. This will be done in the next section. A simple lemma will subsequently relate the new simulations to the simulations between the closures of the automata.

6.3 Direct Simulations Between Timed Automata

We require two auxiliary definition. First, if A is a timed automaton, s' and s are states of A, and β is a sequence of elements of $vis(A)$, then we write $s' \overset{\beta}{\hookrightarrow}_A s$, or just $s' \overset{\beta}{\hookrightarrow} s$ when A is clear, if A has a finite execution fragment α with $first(\alpha) = s'$, $trace(\alpha)\lceil vis(A) = \beta$ and $last(\alpha) = s$. Second, if σ is any sequence then $\widetilde{\sigma}$ is the sequence obtained by removing all internal and time-passage actions from σ.

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

1. $r(s).time = s.time$.
2. If $s \in start(A)$ then $r(s) \in start(B)$.
3. If $s' \xrightarrow{a}_A s$ then $r(s') \overset{\widetilde{a}}{\hookrightarrow}_B r(s)$.

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

1. If $u \in f[s]$ then $u.time = s.time$.
2. If $s \in start(A)$ then $f[s] \cap start(B) \neq \emptyset$.
3. If $s' \xrightarrow{a}_A s$ and $u' \in f[s']$, then there exists $u \in f[s]$ such that $u' \overset{\widetilde{a}}{\hookrightarrow}_B u$.

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

1. If $u \in b[s]$ then $u.time = s.time$.
2. If $s \in start(A)$ then $b[s] \subseteq start(B)$.
3. If $s' \xrightarrow{a}_A s$ and $u \in b[s]$, then there exists $u' \in b[s']$ such that $u' \overset{\widetilde{a}}{\hookrightarrow}_B u$.

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

1. If u is an element of any set in $g[s]$ then $u.time = s.time$.
2. If $s \in start(A)$ then there exists $S \in g[s]$ such that $S \subseteq start(B)$.
3. If $s' \xrightarrow{a}_A s$ and $S' \in g[s']$, then there exists $S \in g[s]$ such that for every $u \in S$ there exists $u' \in S'$ such that $u' \overset{\widetilde{a}}{\hookrightarrow}_B u$.

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

1. If u is an element of any set in $g[s]$ then $u.time = s.time$.
2. If $s \in start(A)$ then for all $S \in g[s]$, $S \cap start(B) \neq \emptyset$.
3. If $s' \xrightarrow{a}_A s$ and $S \in g[s]$, then there exists $S' \in g[s']$ such that for every $u' \in S'$ there exists $u \in S$ such that $u' \overset{\widetilde{a}}{\hookrightarrow}_B u$.

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.

We write $A \leq_{\mathrm{R}}^{\mathrm{t}} B$, $A \leq_{\mathrm{F}}^{\mathrm{t}} B$, etc. in case there exists a timed refinement, timed forward simulation, etc., from A to B.

6.4 Synchronicity

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

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

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

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

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

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

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

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

$$\{(s, S) \mid \exists (s, T) \in g : S = T \cap \{u \mid u.time = s.time\}\}$$

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

Obviously, all the timed versions of refinements, forward simulations, etc., that we defined above are synchronous. The following observation is more interesting. Note that in the proof below the idling steps play a key role. In fact, the result would not be correct without them.

Lemma 6.11

1. *Any refinement from A to B is synchronous.*
2. *If f is a forward simulation from A to B, then $syn(f)$ is a synchronous forward simulation from A to B.*
3. *Any backward simulation from A to B is synchronous.*
4. *If g is a forward-backward simulation from A to B, then $syn1(g)$ is a synchronous forward-backward simulation from A to B.*

5. *If g is a backward-forward simulation from A to B, then $syn2(g)$ is a synchronous backward-forward simulation from A to B.*

6. *Any history relation from A to B is synchronous.*

7. *Any prophecy relation from A to B is synchronous.*

Proof: For 1, suppose that r is a refinement from A to B and s is a state of A with $s.time_A = t$. By axiom **T1**, $s \xrightarrow{t}_A s$. Thus, since r is a refinement, $r(s) \overset{t}{\Longrightarrow}_B r(s)$. From this it folows, by axioms **S2** and **S3**, that $r(s).time = t = s.time$.

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

Now suppose $s' \xrightarrow{a}_A s$ and $u' \in syn(f)[s']$. Then $u'.time = s'.time$. Also, $u' \in f[s']$ and therefore there exists a state $u \in f[s]$ such that $u' \overset{\widehat{a}}{\Longrightarrow}_B u$. Then it follows by axioms **S2** and **S3** that $s.time = u.time$. Hence $u \in syn(f)[s]$.

For 3, suppose that b is a backward simulation from A to B, and suppose s is a state of A with $s.time_A = t$. Let $u \in b[s]$. By axiom **T1** $s \xrightarrow{t}_A s$. Thus, since b is a backward simulation there exists $u' \in b[s]$ with $u' \overset{t}{\Longrightarrow}_B u$. By axioms **S2** and **S3**, this implies $u.time = t = s.time$.

Parts 4-7 are similar. ■

6.5 Relating Timed and Untimed Simulations

In Section 6.2, we showed that (under certain finiteness conditions) there is a one-to-one correspondence between inclusion of timed traces on the level of timed automata, and inclusion of traces between the closures of these automata. In this subsection we observe that there is also a strong connection between timed simulations between timed automata, and the same functions viewed as untimed simulations between the closures of these automata. As an immediate consequence of this observation we obtain easy soundness proofs for all the timed simulations of Section 6.3, since soundness of the timed simulations reduces to the soundness of the corresponding untimed simulations. Moreover we obtain "for free" a completeness result for timed forward-backward simulations.

Lemma 6.12 *Suppose $s, s' \in states(A)$ and $a \in acts(A)$ such that if $a \in \mathsf{R}^{\geq 0}$ then $s.time = a$ else $s.time = s'.time$. Then $s' \overset{\widetilde{a}}{\hookrightarrow}_A s \Leftrightarrow s' \overset{\widehat{a}}{\Longrightarrow}_{cl(A)} s$.*

Proof: Easy from the definitions. ■

Lemma 6.13 *A synchronous relation is a timed refinement from A to B if and only if it is a refinement from $cl(A)$ to $cl(B)$. Moreover, the above property also holds if both occurrences of the word "refinement" are replaced by "forward simulation", "backward simulation", "forward-backward simulation", "backward-forward simulation", "history relation" or "prophecy relation".*

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

Suppose r is a timed refinement from A to B. We have to show that r is a refinement from $cl(A)$ to $cl(B)$, and the only thing nontrivial here is to demonstrate that r satsifies the second clause from the definition of a refinement. For this, suppose $s' \xrightarrow{a}_{cl(A)} s$. Then certainly $s' \overset{\widehat{a}}{\Longrightarrow}_{cl(A)} s$, and thus, by Lemma 6.12, $s' \overset{\widetilde{a}}{\hookrightarrow}_A s$. Since r is a timed refinement, we can use this fact to infer $r(s') \overset{\widetilde{a}}{\hookrightarrow}_B r(s)$. Now $r(s') \overset{\widehat{a}}{\Longrightarrow}_{cl(B)} r(s)$ follows by another application of Lemma 6.12.

For the other direction, suppose r is a refinement from $cl(A)$ to $cl(B)$. We have to establish that r is a timed refinement from A to B, and for this again the only nontrivial part is the second clause in the definition of a timed refinement. So suppose $s' \xrightarrow{a}_A s$. Since the closure construction only adds transitions, this trivially implies $s' \xrightarrow{a}_{cl(A)} s$. Now we use the fact that r is a refinement from $cl(A)$ to $cl(B)$ to obtain $r(s') \overset{\widehat{a}}{\Longrightarrow}_{cl(B)} r(s)$. From this $r(s') \overset{\widetilde{a}}{\hookrightarrow}_B r(s)$ follows by Lemma 6.12. ■

Corollary 6.14 *Suppose $X \in \{R, F, iB, B, iFB, FB, iBF, BF, H, iP, P\}$. Then $A \leq^{\mathrm{t}}_{\mathrm{X}} B$ iff $cl(A) \leq_{\mathrm{X}} cl(B)$.*

Proof: Immediate from Lemmas 6.11 and 6.13. ■

Proposition 6.15 *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: By Lemma 6.14, using that the corresponding untimed simulations are preorders. ■

Also the classification of Section 4.5 carries over to the timed setting:

Theorem 6.16 *Suppose $X, Y \in \{T, *T, R, F, iB, B, iFB, FB, iBF, BF, H, iP, P\}$. Then $A \leq^{\mathrm{t}}_{\mathrm{X}} B \Rightarrow A \leq^{\mathrm{t}}_{\mathrm{Y}} B$ for all timed automata A and B if and only if there is a path from $\leq^{\mathrm{t}}_{\mathrm{X}}$ to $\leq^{\mathrm{t}}_{\mathrm{Y}}$ in Figure 6 consisting of thin lines. If B has t-fin, then $A \leq^{\mathrm{t}}_{\mathrm{X}} B \Rightarrow A \leq^{\mathrm{t}}_{\mathrm{Y}} B$ for all automata A and B if and only if there is a path from $\leq^{\mathrm{t}}_{\mathrm{X}}$ to $\leq^{\mathrm{t}}_{\mathrm{Y}}$ consisting of thin lines and thick lines.*

Proof: Note that except for the superscripts t, Figure 6 is the same as Figure 5, which gives an overview of the relationships in the untimed case. Using Corollary 6.14 and Lemmas 6.7 and 6.8, the "thin line" inclusions for the timed case follow from the corresponding inclusions for the untimed case. For the "thick" line inclusions one needs in addition Lemmas 6.6 and 6.9.

In order to show that all the inclusions are strict, one can basically just use the same counterexamples as in the untimed setting. In order to turn the untimed automata into timed automata one only has to attach a 0-loop to each state. Only for establishing the difference between $\leq^{\mathrm{t}}_{*\mathrm{T}}$ and $\leq^{\mathrm{t}}_{\mathrm{T}}$ the examples of Section 4 are not adequate, and one has to use Example 5.3 instead. (If A^0 and B^0 denote the timed automata obtained by adding 0-loops to all states of the automata A and B of Example 3.1, respectively, then $A^0 \equiv^{\mathrm{t}}_{*\mathrm{T}} B^0$ but, since both timed automata have no admissible traces, also $A^0 \equiv^{\mathrm{t}}_{\mathrm{T}} B^0$.) ■

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

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

Figure 6: Classification of basic relations between timed automata

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: From the definition of the closure construction it is immediate that all states of $cl(A)$ are reachable. By Lemma 6.6(2), $cl(B)$ has fin, and by Lemma 6.13, $cl(A) \leq_{\mathrm{B}} cl(B)$. Now we can apply Proposition 4.15, the untimed version of the fact we are proving, to obtain $cl(A) \leq_{\mathrm{iB}} cl(B)$. By Lemma 6.11 any backward simulation is synchronous, which means that we can apply Lemma 6.13 in the other direction to conclude $A \leq^{\mathrm{t}}_{\mathrm{iB}} B$. ■

Theorem 6.18 *(Partial completeness of timed forward simulations) Suppose B is deterministic and $A \leq^{\mathrm{t}}_{*\mathrm{T}} B$. Then $A \leq^{\mathrm{t}}_{\mathrm{F}} B$.*

Proof: By Lemma 6.6, $cl(B)$ is deterministic, and by Lemma 6.7, $cl(A) \leq_{*\mathrm{T}} cl(B)$. Thus by the partial completeness result for forward simulations (Theorem 4.9), $cl(A) \leq_{\mathrm{F}} cl(B)$. Now Lemmas 6.11 and 6.13 allow us to conclude $A \leq^{\mathrm{t}}_{\mathrm{F}} B$, as required. ■

6.6 Additional Results for Timed Automata

The previous sections show how some simple correspondences cause most of the results for untimed automata to carry over to the timed setting. There are some results about untimed automata that do not carry over because of these correspondences, but are nonetheless true. Firstly, there are the partial completeness results that involve t-forests. These do not carry over since the closure construction does not map t-forests to forests. Secondly, the various results that require the construction of some timed automaton (for instance the timed version of the Abadi-Lamport result) do not carry over via the correspondence. This subsection is devoted to establishing these remaining results in the setting of timed automata.

6.6.1 Partial Completeness Results

Theorem 6.19 *(Partial completeness of timed refinements) Suppose A is a t-forest, B is deterministic and $A \leq^{\mathrm{t}}_{*\mathrm{T}} B$. Then $A \leq^{\mathrm{t}}_{\mathrm{R}} B$.*

Proof: Analogous to the proof of Theorem 4.4. If $r \triangleq \textit{t-after}(B) \circ \textit{t-past}(A)$, then r can be shown to be a timed refinement from A to B. The proof uses Lemmas 5.8 and 5.3. ■

Theorem 6.20 *(Partial completeness of timed backward simulations) Suppose A is a t-forest and $A \leq^{\mathrm{t}}_{*\mathrm{T}} B$. Then*

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

6.6.2 Results Involving an Intermediate Timed Automaton

Theorem 6.21 *(Completeness of timed forward and timed backward simulations) Suppose $A \leq^{\mathrm{t}}_{*\mathrm{T}} B$. Then*

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

Proof: The proof is essentially the same as the proof of Theorem 4.16.

Let $C = \textit{t-can}(\textit{t-beh}(A))$. By Lemma 5.14, C is a deterministic t-forest and $A \equiv^{\mathrm{t}}_{*\mathrm{T}} C$. Since C is deterministic, $A \leq^{\mathrm{t}}_{\mathrm{F}} C$ by partial completeness of timed forward simulations (Theorem 6.18), and because C is a t-forest, $C \leq^{\mathrm{t}}_{\mathrm{B}} B$ follows by partial completeness of timed backward simulations (Theorem 6.20(1)). Similarly, if B has t-fin then $C \leq^{\mathrm{t}}_{\mathrm{iB}} B$ follows by Theorem 6.20(2). ■

Definition 6.1 Suppose k is a synchronous relation over $\textit{states}(A)$ and $\textit{states}(B)$ satisfying $k \cap (\textit{start}(A) \times \textit{start}(B)) \neq \emptyset$. The *timed superposition* $\textit{t-sup}(A, B, k)$ of B onto A via k is the timed automaton C given by

- $\textit{states}(C) = k$,
- $\textit{start}(C) = k \cap (\textit{start}(A) \times \textit{start}(B))$,
- $\textit{acts}(C) = \textit{acts}(A) \cap \textit{acts}(B)$, and
- for $(s', v'), (s, v) \in \textit{states}(C)$ and $a \in \textit{acts}(C)$,

  $$(s', v') \xrightarrow{a}_C (s, v) \quad \Leftrightarrow \quad s' \overset{\widetilde{a}}{\hookrightarrow}_A s \;\wedge\; v' \overset{\widetilde{a}}{\hookrightarrow}_B v \;\wedge\; S_2 \;\wedge\; S_3 \;\wedge\; S_4,$$

  where $S_2 \triangleq a \notin \mathsf{R}^{\geq 0} \Rightarrow s'.\textit{time} = s.\textit{time}$, $S_3 \triangleq a \in \mathsf{R}^{\geq 0} \Rightarrow s'.\textit{time} \leq a = s.\textit{time}$, and $S_4 \triangleq a = s'.\textit{time} \Rightarrow (s' = s \wedge v' = v)$.

Theorem 6.22 $A \leq^{\mathrm{t}}_{\mathrm{F}} B \Leftrightarrow (\exists C : A \leq^{\mathrm{t}}_{\mathrm{H}} C \leq^{\mathrm{t}}_{\mathrm{R}} B)$.

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

The reverse implication also follows via a standard argument. ■

Theorem 6.23

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

Proof: Similar to the proof of Theorem 6.22, using timed backward simulations instead. ■

Theorem 6.24 *(Completeness of timed history/prophecy relations and refinements) If $A \leq_{*\mathrm{T}}^{\mathrm{t}} B$ then the following are true.*

1. $\exists C, D : A \leq_{\mathrm{H}}^{\mathrm{t}} C \leq_{\mathrm{P}}^{\mathrm{t}} D \leq_{\mathrm{R}}^{\mathrm{t}} B$.

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

3. $\exists C, D : A \leq_{\mathrm{P}}^{\mathrm{t}} C \leq_{\mathrm{H}}^{\mathrm{t}} D \leq_{\mathrm{R}}^{\mathrm{t}} B$.

Proof: Completely analogous to the proofs of Theorem 4.43 and Theorem 4.44. ■

6.6.3 Unfold and Guess Constructions

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

- $\textit{states}(B)$ = the set of fat executions of A,
- $\textit{start}(B)$ = the executions of A that consist of a single start state,
- $\textit{acts}(B) = \textit{acts}(A)$, and
- for $\alpha', \alpha \in \textit{states}(B)$ and $t \in \mathsf{R}^{\geq 0}$,

  $$\alpha' \xrightarrow{t}_B \alpha \quad \Leftrightarrow \quad \textit{strip-end}(\alpha') = \textit{strip-end}(\alpha) \wedge \textit{last}(\alpha') \xrightarrow{t}_A \textit{last}(\alpha)$$

  and, for $a \in \textit{acts}(B) - \mathsf{R}^{\geq 0}$,

  $$\alpha' \xrightarrow{a}_B \alpha \;\Leftrightarrow\; \alpha = \alpha' \, a \, \textit{last}(\alpha),$$

  where $\textit{strip-end}(\alpha)$ is obtained from α by removing the time-passage step (if present) at the end of α.

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

Proposition 6.25 *$\textit{t-unfold}(A)$ is a t-forest and $A \leq_{\mathrm{H}}^{\mathrm{t}} \textit{t-unfold}(A)$.*

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

Dual to the timed unfolding is the following timed guess construction. The *timed guess* of A, notation *t-guess*(A), is the timed automaton B defined by

- $states(B)$ = the set of fat execution fragments of A,
- $start(B)$ = the set of fat executions of A,
- $acts(B) = acts(A)$, and
- for $\alpha', \alpha \in states(B)$ and $t \in \mathsf{R}^{\geq 0}$,

  $$\alpha' \xrightarrow{t}_B \alpha \quad \Leftrightarrow \quad strip\text{-}begin(\alpha') = strip\text{-}begin(\alpha) \wedge first(\alpha') \xrightarrow{t}_A first(\alpha)$$

  and, for $a \in acts(B) - \mathsf{R}^{\geq 0}$,

  $$\alpha' \xrightarrow{a}_B \alpha \;\Leftrightarrow\; first(\alpha')\, a\, \alpha = \alpha',$$

  where $strip\text{-}begin(\alpha)$ is obtained from α by removing the time-passage step (if present) at the begin of α.

(Again we leave it to the reader to verify that B is a timed automaton.)

Proposition 6.26 *$A \leq_{\mathrm{P}}^{\mathrm{t}}$ t-guess(A).*

Proof: Similar to the proof of Proposition 6.25. ■

7 Discussion

In this paper, we have presented an automata-theoretic model for timing-based systems, and have used it to develop a variety of proof techniques for such systems. A considerable amount of further work remains to be done.

First, there is a technical issue. Some of the other work on simulations (e.g., [21]) includes reachability restrictions in the step correspondence conditions; we would like theorems justifying the soundness of those simulations in terms of the soundness of our simulations (without reachability hypotheses).

Refinements and forward simulations have already been used extensively and successfully for verifying concurrent algorithms, and backward simulations (in the form of prophecy variables) have also been shown to be of practical value in some cases. Additional work remains to determine the practical utility of the various kinds of simulations studied in this paper, particularly in the case of timing-based systems. This will involve applying the simulation techniques to a wide range of examples. It may also involve development of techniques analogous to the progress measure techniques in [19], based on extra structure to be added to our timed automaton model.

Finally, it remains to carry out process algebraic work using the same timed automaton model. A paper in progress [31] contains the beginning of such work, including definitions of interesting operators on timed automata, and proofs of substitutivity results for the timed trace semantics. However, much remains to be done.

Acknowledgements

We thank Albert Meyer, Jeff Sanders, Roberto Segala, Eugene Stark and George Varghese for their valuable criticism and useful comments. 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.

References

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

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

[3] G. Berry and L. Cosserat. The ESTEREL synchronous programming language and its mathematical semantics. In A.W. Roscoe & G. Winskel S.D. Brookes, editor, *Seminar on Concurrency*, pages 389–448, Springer-Verlag, 1984.

[4] 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.

[5] R. Gerth. Foundations of compositional program refinement (first version). In J.W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *REX Workshop on Stepwise Refinement of Distributed Systems: Models, Formalism, Correctness,* Mook, The Netherlands 1989, pages 777–560, Springer-Verlag, 1990.

[6] A. Ginzburg. *Algebraic Theory of Automata.* Academic Press, New York – London, 1968.

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

[8] J. He. Process simulation and refinement. *Journal of Formal Aspects of Computing Science*, 1:229–241, 1989.

[9] C.A.R. Hoare, J. He, and J.W. Sanders. Prespecification in data refinement. *Information Processing Letters*, 25:71–76, 1987.

[10] B. Jonsson. *Compositional Verification of Distributed Systems.* PhD thesis, Department of Computer Systems, Uppsala University, 1987. DoCS 87/09.

[11] B. Jonsson. Modular verification of asynchronous networks. In *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing,* Vancouver, Canada, pages 152–166, 1987.

[12] B. Jonsson. On decomposing and refining specifications of distributed systems. In J.W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *REX Workshop on Stepwise Refinement of Distributed Systems: Models, Formalism, Correctness,* Mook, The Netherlands 1989, pages 361–387, Springer-Verlag, 1990.

[13] B. Jonsson. Simulations between specifications of distributed systems. In J.C.M. Baeten and J.F. Groote, editors, *Proceedings CONCUR 91,* Amsterdam, pages 346–360, Springer-Verlag, 1991.

[14] M.B. Josephs. A state-based approach to communicating processes. *Distributed Computing*, 3:9–18, 1988.

[15] N. Klarlund and F.B. Schneider. *Verifying Safety Properties Using Infinite-state Automata.* Technical Report 89-1039, Department of Computer Science, Cornell University, Ithaca, New York, 1989.

[16] D.E. Knuth. *Fundamental Algorithms.* Volume 1 of *The Art of Computer Programming*, Addison-Wesley, Reading, Massachusetts, 1973. Second edition.

[17] L. Lamport. Specifying concurrent program modules. *ACM Transactions on Programming Languages and Systems*, 5(2):190–222, 1983.

[18] N.A. Lynch. Multivalued possibilities mappings. In J.W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *REX Workshop on Stepwise Refinement of Distributed Systems: Models, Formalism, Correctness,* Mook, The Netherlands 1989, pages 519–543, Springer-Verlag, 1990.

[19] N.A. Lynch and H. Attiya. Using mappings to prove timing properties. In *Proceedings of the 9th Annual ACM Symposium on Principles of Distributed Computing,* Quebec, Canada, August 1990. Expanded version: Technical Memo MIT/LCS/TM-412.c, Laboratory for Computer Science, MIT, March 1991. Submitted for publication.

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

[21] M. Merritt. Completeness theorems for automata. In J.W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *REX Workshop on Stepwise Refinement of Distributed Systems: Models, Formalism, Correctness,* Mook, The Netherlands 1989, pages 544–560, Springer-Verlag, 1990.

[22] M. Merritt, F. Modugno, and M. Tuttle. Time constrained automata. In J.C.M. Baeten and J.F. Groote, editors, *Proceedings CONCUR 91,* Amsterdam, pages 408–423, Springer-Verlag, 1991.

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

[24] F. Moller and C. Tofts. A temporal calculus of communicating systems. In J.C.M. Baeten and J.W. Klop, editors, *Proceedings CONCUR 90,* Amsterdam, pages 401–415, Springer-Verlag, 1990.

[25] 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.

[26] X. Nicollin, J. Sifakis, and S. Yovine. From ATP to timed graphs and hybrid systems. 1991. This volume.

[27] S. Owicki and D. Gries. An axiomatic proof technique for parallel programs. *Acta Informatica*, 6(4):319–340, 1976.

[28] D.M.R. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, 5th *GI Conference*, pages 167–183, Springer-Verlag, 1981.

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

[30] E. W. Stark. Proving entailment between conceptual state specifications. *Theoretical Computer Science*, 56:135–154, 1988.

[31] F.W. Vaandrager and N.A. Lynch. Process algebras for timed automata. 1991. In preparation.

[32] Wang Yi. Real-time behaviour of asynchronous agents. In J.C.M. Baeten and J.W. Klop, editors, *Proceedings CONCUR 90,* Amsterdam, pages 502–520, Springer-Verlag, 1990.

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