Technical Report 2009-003
A Testing Theory for Real-Time Systems*

Stefan D. Bruda and Chun Dai
Department of Computer Science, Bishop's University
Sherbrooke, Quebec J1M 1Z7, Canada
email: {bruda|cdai}@cs.ubishops.ca
30 October 2009

Abstract

We develop a testing theory for real-time systems. We keep the usual notion of success or failure (based on finite runs) but we also provide a mechanism of determining the success or failure of infinite runs, using a formalism similar to the acceptance in Büchi automata. We present two refinement timed preorders similar to De Nicola and Hennessy's may and must testing. We then provide alternative, behavioural and language-based characterizations for these relations to show that the new preorders are extensions of the traditional preorders. Finally we focus on test generation, showing how tests can be automatically generated out of a timed variant of linear-time logic formulae.

Keywords: formal methods, real-time systems, model-based testing, may testing, must testing, testing preorders, test generation, timed temporal logic, TPTL.

1 Introduction

The development of hardware and software is getting more and more complex. How to guarantee validity and reliability is one of the most pressing problems nowadays [5, 13]. Among many theoretical methods for this, *conformance testing* [20] is the most notable one for its succinctness and high automatization. Its aim is to check whether an implementation conforms to a given specification.

Formal system specifications [17], together with the implementations, can be mainly classified into two kinds: algebraic and logic. The first favors refinement, when a single algebraic formalism is equipped with a refinement relation to represent a system's specification and implementation [21]. An implementation is validated correct if it refines its specification. Since it often defines the system transitionally, process algebra [18], labelled transition systems [7], and finite automata are commonly used in this classification, with traditional refinement relations being either behavioural equivalences or preorders [7, 14]. A typical example is model-based testing [7]. The second approach to conformance testing prefers assertive constructs; different formalisms are used to describe the properties of the system specifications and implementations [10, 11]. Specifications are usually defined in a logical language while implementations are given in an operational notation. The semantics of assertions is to determine whether an implementation satisfies its specification. A typical example is model checking [11].

The domain of conformance testing consists in reactive systems, which interact with their environment (also regarded as a reactive system). Often such systems are required to be *real time*, meaning that in addition to the correct order of events, they must satisfy constraints on delays separating certain events. A system that does not respond within our lifetime is obviously not useful, but many times we require a more precise time-wise characterization. *Real-time specifications* are then used as the basis of conformance testing for such systems [17].

The aim of this paper is to develop a semantic theory for real-time system specification featuring real-time transition systems modeling the behaviour of timed processes. Using a theory of timed ω-final

states as well as a timed testing framework based on De Nicola and Hennessy's may and must testing [14], we develop our timed may and must preorders that relate timed processes on the basis of their responses to timed tests. We also provide concise alternative characterizations of these two preorders. Our framework is as close to the original framework of (untimed) testing as possible, and is also as general as possible. While studies of real-time testing exist, they have mostly restricted the real-time domain to make it tractable; by contrast, our theory is general. As a consequence, it may be regarded as not applicable in practice. To address this perception, we also tackle automatic test generation. More specifically, we show how to algorithmically build equivalent tests starting from timed propositional temporal logic (TPTL) [3] formulae.

Our theory considers the whole domain with all its particularities. We believe that starting from a general theory is more productive than starting directly from some practically feasible (and thus restricted) subset of the issue. On the other hand, our theory is more practical than one might expect. Indeed, the characterization of the timed may and must preorders uses a surprisingly concise set of timed tests. In addition, we are able to find an algorithm for automatic test generation starting from TPTL formulae. This algorithm is also a first (but significant) step toward an integration of operational and assertive specification styles in the area of real-time systems, to obtain heterogeneous (algebraic and logic) real-time specifications and tools.

The remainder of this paper is structured as follows. Preliminaries such as preorders, timed automata, and timed transition systems are presented in the next section. We formalize the notion of timed processes and timed tests in Section 3, where we also introduce and characterize our timed preorders. Section 4 presents our conversion of TPTL formulae into equivalent timed tests. We conclude in Section 5.

2 Preliminaries and Notations

Preorders are reflexive and transitive relations. They are widely used as implementation relations comparing specifications and implementations. Preorders are easier to construct and analyze compared to equivalence relations, and once a preorder is established, an associated equivalence relation is immediate. The cardinality of $\mathbb{N}$ is denoted by ω.

All our constructions are based on some *action alphabet* A representing a set of actions excluding the internal action τ and on a *time alphabet* L which contains some kind of positive numbers (such as $\mathbb{N}$ or $\mathbb{R}^+$). A set of *time clocks* C is a set of clocks (i.e, variables over L) associated to states. We denote by $\mathbb{T}(\mathsf{C})$ the set of time constraints over a set C of clocks. A *clock interpretation* for a set C is a mapping $\mathsf{C} \rightarrow \mathsf{L}$. *Clock progress* denotes the effect of time sequences that increase clock values. If $t > 0$ and c is a clock interpretation over C, in the clock interpretation $c' = c + t$ we have $c'(x) = c(x) + t$ for all clocks $x \in \mathsf{C}$. Clocks can be *reset* to zero.

A *time constraint* is also called *clock constraint* here, since we constrain the clocks of the states to constrain time between states. If x is a clock and r is a real number, then $x \sim r$ is a clock constraint, where $\sim \in \{\leq, <, =, \neq, >, \geq\}$. Clock constraints can be joined together either in conjunctions ($\wedge$) or disjunctions ($\vee$). That is, if x is a clock, the following is an admissible clock constraint: $x < 3 \vee x > 5$ ($\wedge$ is for multiple clocks).

2.1 Timed Transition Systems, Traces, and Languages

Labelled transition systems [7] are used to model the behaviour of various processes; they serve as a semantic model for formal specification languages. A timed transition system is essentially a labelled transition system extended with time values associated to actions. Timed automata [2, 16] are based on the automata theory and introduce the notion of time constraints over their transitions. In general labelled

transition systems model the execution of a process, while timed automata are suitable for specifying processes or defining tests upon processes. We find convenient to combine the two concepts to obtain a unified model for real time, which we call by abuse of terminology (and for lack of a better term) timed transition system; we do not introduce any new concept, instead we unify existing constructions into a convenient single construction. A timed transition system is essentially a timed automaton (or more precisely a timed transition table, since final states will be introduced later) without the restriction of the number of states being finite.

Definition 1. TIMED TRANSITION SYSTEM. *For a set A of observable actions ($\tau \notin \mathsf{A}$), a set L of times values, and a set C of clocks with an associated set $\mathbb{T}(\mathsf{C})$ of time constraints, a timed transition system is a tuple $((\mathsf{A} \times \mathsf{L}) \cup \{\tau\}, \mathsf{C}, S, \rightarrow, p_0)$, where: S is a countable set of states; every state $p \in S$ has an associated clock interpretation $c_p : \mathsf{C} \rightarrow \mathsf{L}$; $\rightarrow \subseteq (S \times (\mathsf{A} \times \mathsf{L}) \times S \times \mathbb{T}(\mathsf{C}) \times 2^{\mathsf{C}}) \cup (S \times \{\tau\} \times S)$ is the transition relation (commonly, we use $p \xrightarrow[\mathbb{t},C]{(a,\delta)} p'$ instead of $(p, (a, \delta), p', \mathbb{t}, C) \in \rightarrow$, further omitting C whenever $C = \emptyset$); p_0 is the initial state[1].*

A timed transition system picks its way from one state to the next state according to the transition relation. Whenever $p \xrightarrow[\mathbb{t},C]{(a,\delta)} p'$, the transition system performs a with delay δ; the delay causes the clocks to progress so that $c_{p'}(x) = c_p(x) + \delta$ whenever $x \notin C$ and $c_{p'}(x) = 0$ otherwise; the transition is enabled only if $\mathbb{t}$ holds under the interpretation c_p; τ transitions change the state but do not affect clock interpretations and cannot be time constrained.

Timed transition systems are different from labelled transition systems in their treatment of traces (by associating time information with them). Normally a trace is described as a sequence of the events or states (but not the delays between them). To add time to a trace, we add time information to the usual notion of trace (that contains actions only).

Definition 2. TIMED TRACE. *A timed trace over A, L, and C is a member of $(\mathsf{A} \times \mathsf{L} \times \mathbb{T}(\mathsf{C}) \times 2^{\mathsf{C}})^* \cup (\mathsf{A} \times \mathsf{L} \times \mathbb{T}(\mathsf{C}) \times 2^{\mathsf{C}})^\omega$, where A is a finite set of events, L is a set of time actions, C is a set of clocks, and $\mathbb{T}(\mathsf{C})$ is a set of time constraints over C.*

If both L and $\mathbb{T}(\mathsf{C})$ (or equivalently C) are empty, the timed transition system is the same as a labelled transition system, and the timed trace becomes a normal trace. However, one of L or $\mathbb{T}(\mathsf{C})$ could be empty and we still obtain a timed trace; this will be used later to differentiate between processes and specifications.

We will use the following relation: $p \xRightarrow[\mathbb{t},C]{(a,\delta)} p'$ iff $p = p_0 \xrightarrow{\tau} p_1 \xrightarrow{\tau} p_2 \xrightarrow{\tau} \cdots \xrightarrow{\tau} p_n \xrightarrow[\mathbb{t},C]{(a,\delta)} p'$ for some $n \geq 0$, and $p \xRightarrow{\varepsilon} p'$ iff $p = p_0 \xrightarrow{\tau} p_1 \xrightarrow{\tau} p_2 \xrightarrow{\tau} \cdots \xrightarrow{\tau} p_n = p'$ for some $n \geq 0$. By abuse of notation, we also write $p \xRightarrow{w} q$ whenever $w = (a_i, \delta_i, \mathbb{t}_i, C_i)_{0 < i \leq k}$ and $p \xRightarrow[\mathbb{t}_1, C_1]{(a_1, \delta_1)} p_1 \xRightarrow[\mathbb{t}_2, C_2]{(a_2, \delta_2)} p_2 \cdots \xRightarrow[\mathbb{t}_k, C_k]{(a_k, \delta_k)} p_k = p'$.

Definition 3. TIMED PATH. *Let $M = ((\mathsf{A} \times \mathsf{L}) \cup \{\tau\}, \mathsf{C}, S, \rightarrow, p_0)$ be a timed transition system. A timed path π of M is a potentially infinite sequence $(p_{i-1}, (a_i, \delta_i, \mathbb{t}_i, C_i), p_i)_{0 < i \leq k}$, where $p_{i-1} \xrightarrow[\mathbb{t}_i, C_i]{(a_i, \delta_i)} p_i$, for all $0 < i \leq k$ with $a_i \in \mathsf{A}$, $\delta_i \in \mathsf{L}$, $\mathbb{t}_i \in \mathbb{T}(\mathsf{C})$, and $C_i \subseteq \mathsf{C}$.*

We use $|\pi|$ to refer to k, the length of π. If $|\pi| = \omega$, we say that π is *infinite*; otherwise, π is *finite*. A *deadlock* occurs when the transition system cannot move to another state. If $|\pi| \in \mathbb{N}$ and $p_{|\pi|} \nrightarrow$ (that is, $p_{|\pi|}$ is a deadlock state), then the timed path π is called maximal. $\text{trace}(\pi)$, the (timed) trace of π is defined as the sequence $(a_i, \delta_i, \mathbb{t}_i, C_i)_{0 \leq i \leq |\pi|} \in (\mathsf{A} \times \mathsf{L} \times \mathbb{T}(\mathsf{C}) \times 2^{\mathsf{C}})^*$.

---

[1] Whenever the transition relation is global and understood we can regard a state as the timed transition system whose initial state is the given state.

We use $\Pi_f(p')$, $\Pi_m(p')$, $\Pi_I(p')$ to denote the sets of all finite timed paths, all maximal timed paths, and all infinite timed paths starting from state $p' \in S$, respectively. We also put $\Pi(p') = \Pi_f(p') \cup \Pi_m(p') \cup \Pi_I(p')$. The empty timed path π with $|\pi| = 0$ is symbolized by () and its (always empty) trace by ε.

Divergence, the special notion of partially defined states (that may engage in infinite internal computations), is important for the testing theory of reactive systems. State p' of transition system p is *timed divergent*, denoted by $p' \Uparrow_p$ (or just $p' \Uparrow$ when there is no ambiguity), if $\exists \pi \in \Pi_I(p') : \text{trace}(\pi) = \varepsilon$. State p' is called timed w-divergent (denoted by $p' \Uparrow_p w$) for some $w = (a_i, \delta_i, \mathbb{t}_i, C_i)_{0 < i < k} \in (\mathsf{A} \times \mathsf{L} \times \mathbb{T}(\mathsf{C}) \times 2^{\mathsf{C}})^* \cup (\mathsf{A} \times \mathsf{L} \times \mathbb{T}(\mathsf{C}) \times 2^{\mathsf{C}})^\omega$ if one can reach a divergent state starting from p' when executing a finite prefix of w, that is, if $\exists l \in \mathbb{N}, p'' \in S : l \le k, p' \xRightarrow{w'} p'', p'' \Uparrow_p$, with $w' = (a_i, \delta_i, \mathbb{t}_i, C_i)_{0 < i < l}$. For convenience we denote by $L_D(p')$ the divergence language of p'. Conversely, state p' is timed convergent or timed w-convergent (denoted $p' \Downarrow_p$ and $p \Downarrow_p w$, respectively, with the subscript omitted when there is no ambiguity) if it is not the case that $p' \Uparrow_p$ and $p' \Uparrow_p w$, respectively.

The set of initial actions of state p' if defined as follows: $\text{init}_p(p') = \{(a, \delta, \mathbb{t}, C) \in \mathsf{A} \times \mathsf{L} \times 2^{\mathsf{C}} : \exists p'' : p' \xRightarrow[\mathbb{t}, C]{(a, \delta)} p''\}$.

We finally introduce different languages for a timed transition system (state) p.

Definition 4. TIMED LANGUAGES. *The timed finite-trace language $L_f(p)$, timed maximal-trace (complete-trace) language $L_m(p)$, and timed infinite-trace language $L_I(p)$ of p are*

$$
\begin{aligned}
L_f(p) &= \{\text{trace}(\pi) : \pi \in \Pi_f(p)\} \subseteq (\mathsf{A} \times \mathsf{L} \times \mathbb{T}(\mathsf{C}) \times 2^{\mathsf{C}})^* \\
L_m(p) &= \{\text{trace}(\pi) : \pi \in \Pi_m(p)\} \subseteq (\mathsf{A} \times \mathsf{L} \times \mathbb{T}(\mathsf{C}) \times 2^{\mathsf{C}})^* \\
L_I(p) &= \{\text{trace}(\pi) : \pi \in \Pi_I(p)\} \subseteq (\mathsf{A} \times \mathsf{L} \times \mathbb{T}(\mathsf{C}) \times 2^{\mathsf{C}})^\omega.
\end{aligned}
$$

The divergence language of p is $L_D(p) = \{w \in (\mathsf{A} \times \mathsf{L} \times \mathbb{T}(\mathsf{C}) \times 2^{\mathsf{C}})^* \cup (\mathsf{A} \times \mathsf{L} \times \mathbb{T}(\mathsf{C}) \times 2^{\mathsf{C}})^\omega : p \Uparrow w\}$.

2.2 Timed ω-Languages and ω-Final States

Originally [2], a timed word over an alphabet Σ was defined as a pair (σ, t), where σ is an infinite word over Σ and t is a time sequence (an infinite sequence of time values). A timed language over Σ is then a set of timed words over Σ. We defined timed languages slightly differently, in order to reflect the use of such languages for system specification (so that we also include time constraints) and also to simplify the presentation. We note however that if we omit the clocks and their constraints (which we will do when it comes to processes) there is a natural bijection between our definition and the original.

We now come to the characterization of infinite timed traces. Once more similar to the theory of timed automata [2] we introduce for this purpose a set of timed ω-final states, that contains exactly all the states which allow time-event sequences to be accepted by appearing infinitely often in the corresponding timed path. We then define timed ω-regular trace languages as follows:

Definition 5. TIMED ω-REGULAR TRACE LANGUAGE. *The timed ω-regular trace language of some timed transition system p is $L_\omega(p) = \{\text{trace}(\pi) : \pi \in \Pi_\omega(p)\} \subseteq (\mathsf{A} \times \mathsf{L} \times \mathbb{T}(\mathsf{C}) \times 2^{\mathsf{C}})^\omega$, where $\Pi_\omega(p)$ contains exactly all the ω-regular timed paths; that is, ω-final states must occur infinitely often in any $\pi \in \Pi_\omega(p)$.*

In addition we exclude henceforth Zeno behaviours from all the languages that we consider: no trace is allowed to show Zeno behaviour. In other words, time progresses and must eventually grow past any constant value (this property is also called progress [2, 9]).

2.3   Timed Propositional Temporal Logic

Timed Propositional Temporal Logic (TPTL) [3] is one of the most general temporal logics with time constraints [4]. TPTL extends traditional linear-time temporal logic (LTL) [4, 11] by adding time constraints, so that its semantics is given with respect to infinite and finite timed traces[2], that is, timed words in $(A \times L)^* \cup (A \times L)^\omega$. This allows formulae to constrain ongoing as well as deadlocking behavior for both actions and time.

We will define and use a slightly modified form of TPTL without congruence. However, it is immediate that our form is equivalent to the original, for indeed we add negation and conjunction, which can all be expressed using $\bot$ and logical implication (which exist in the original definition of TPTL). Similarly, implication can be expressed using negation and conjunction. The constant $\top$ is introduced only for notational convenience. For this reason, we will continue to call our temporal logic TPTL without congruence—we will in fact shorten this to just TPTL, the lack of congruence being henceforth implied.

We assume a set of clocks C ranged over by x. With ϕ, ϕ_1, ϕ_2 ranging over TPTL formulae, a ranging over A, and c ranging over positive constants, the syntax of the term θ and the TPTL formula ϕ is the following:

$$
\begin{aligned}
\theta &= x + c \mid c \\
\phi &= \theta_1 \leq \theta_2 \mid \top \mid \bot \mid a \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid X\,\phi \mid \phi_1\,U\,\phi_2 \mid x.\phi
\end{aligned}
$$

Let the set of all TPTL formulae be $\mathscr{F}$. We say that a timed trace $w = (a_i, \delta_i)_{0 < i \leq k} \in (A \times L)^* \cup (A \times L)^\omega$ satisfies ϕ iff $w \vDash_\gamma \phi$ holds ($k = |w|$ is the length of w). The relation $\vDash_\gamma \subseteq ((A \times L)^* \cup (A \times L)^\omega) \times \mathscr{F}$ is the least relation satisfying the conditions in the semantics of TPTL formulae shown below, with w_j standing for $(a_i, \delta_i)_{j \leq i \leq k}$ for any $1 \leq j \leq k$, and $\gamma : C \to L$ being some clock interpretation.

- $\theta_1 \leq \theta_2$ iff $\gamma(\theta_1) \leq \gamma(\theta_2)$,

- $w \vDash_\gamma \top$ and $w \nvDash_\gamma \bot$ for any w,

- $w \vDash_\gamma a$ iff $w \neq \varepsilon$ and $a_1 = a$,

- $w \vDash_\gamma \neg\phi$ iff $\neg w \vDash_\gamma \phi$,

- $w \vDash_\gamma \phi_1 \wedge \phi_2$ iff $w \vDash_\gamma \phi_1$ and $w \vDash_\gamma \phi_2$,

- $w \vDash_\gamma X\,\phi$ iff $w_2 \vDash_\gamma \phi$,

- $w \vDash_\gamma \phi_1\,U\,\phi_2$ if $\exists 0 < j \leq k: \forall i \leq r \leq k: w_r \vDash_\gamma \phi_2, \forall 0 < s < i: w_s \vDash_\gamma \phi_1$,

- $w \vDash_\gamma x.\phi$ iff $w \vDash_{\gamma[0/x]} \phi$.

The clock interpretation γ is arbitrary, except that $\gamma(x + c) = \gamma(x) + c$ and $\gamma(c) = c$; $\gamma[t/x]$ is the clock interpretation that agrees with γ on all clocks except x, which is mapped to $t \in L$.

The occurrence of a free time variable x in a formula "freezes" the moment in time, which can be checked later by using x in various expressions. There is no other concept of time restriction, but these restrictions are sufficient to model most phenomenae from other timed temporal logics [3].

As usual one can also introduce the derived operators G ("globally") and F ("eventually") as $G\,\phi = \bot\,R\,\phi$ and $F\,\phi = \top\,U\,\phi$, respectively. The operator R ("releases") is defined as usual as the dual of the operator U.

---

[2]Time traces as presented in the previous section also contain clock constraints; however, as we will see in Section 3, the clock constraints appear only in tests and so the trace of processes are over $A \times L$ only.

We also say that a timed process[3] p satisfies the TPTL formula ϕ, written $p \vDash_\gamma \phi$, iff $\forall w \in L_f(p) \cup L_m(p) \cup L_\omega(p) \cup L_D(p) : w \vDash_\gamma \phi$.

3   A Testing Theory

We are now ready to extend the testing theory of De Nicola and Hennessy [14] in two ways. For one thing, we adapt this testing theory to timed testing. In addition, we are also introducing the concept of Büchi acceptance to tests (or Büchi success), so that the properties of infinite runs of a process can be readily identified by tests. Timed testing has been studied before in many contexts [6, 16, 19] but to our knowledge never in such a general setting and never including Büchi success. In addition, timed testing has never been considered in conjunction with test generation from temporal logic formulae. We note however that a somehow incipient consideration of Büchi success for tests and also of temporal logic formulae as test generators for untimed tests exists [12], though to our knowledge this has not been pursued any further.

The traditional testing framework defines behavioural preorders that relate labelled transition systems according to their responses to tests [1]. Tests are used to verify the external interactions between a system and its environment. We use timed transition systems as the formalism for both processes and tests.

3.1   Timed Tests and Timed Testing Preorders

In our framework a test is a timed transition system where certain states are considered to be success states. In order to determine whether a system passes a test, we run the test in parallel with the system under test and examine the resulting finite or infinite computations until the test runs into a success state[4] (pass) or a deadlock state (fail). In addition, a set of ω-final states is used to compartmentalize infinite runs into successful and unsuccessful.

Definition 6. TIMED PROCESSES AND TIMED TESTS. *A timed process* $((A \times L) \cup \{\tau\}, S, \rightarrow, p_0)$ *is a timed transition system* $((A \times L) \cup \{\tau\}, \emptyset, S, \rightarrow, p_0)$ *with an empty set of clocks (and thus with no clock constraints). It follows that all the traces of any timed process are in the set* $(A \times L)^* \cup (A \times L)^\omega$.

A timed test $(A \cup \{\tau\}, C, T, \rightarrow_t, \Sigma, \Omega, t_0)$ *is a timed transition system* $((A \times \emptyset) \cup \{\tau\}, C, T, \rightarrow, t_0)$ *with the addition of a set* $\Sigma \subseteq T$ *of success states and a set* $\Omega \subseteq T$ *of* ω-final states. Note that $L = \emptyset$ for tests and therefore $\rightarrow_t \subseteq (T \times A \times T \times \mathbb{T}(C) \times 2^C) \cup (T \times \{\tau\} \times T)$.

The transition relation of a process and a test are restricted (in different manners) because the test runs in parallel with the process under the test[5]. This latter process (called the implementation) features time sequences but no time constraints, while the test features only time constraints. It is meaningless to run the test by itself. If $\mathbb{T}(C) = \emptyset$ which means there is no time constraint in the test, we call the test *classical*. The set of all timed tests is denoted by $\mathscr{T}$.

Definition 7. PARTIAL COMPUTATION. *A partial computation c with respect to a timed process p and a timed test t is a potentially infinite sequence* $(\langle p_{i-1}, t_{i-1} \rangle \xrightarrow[\mathbb{t}_i, C_i]{(a_i, \delta_i)}_R \langle p_i, t_i \rangle)_{0 < i \leq k}$, *where* $k \in \mathbb{N} \cup \{\omega\}$, *such that*

    1. $p_i \in P$ and $t_i \in T$ for all $0 < i \leq k$, and

---

[3]A timed process is a timed transition system without time constraints, as detailed in Section 3

[4]Success states are deadlock states too, but we distinguish then as special deadlock states.

[5]Note however that the difference is syntactical only, for indeed the transition relation for a timed process allows for an empty set L.

2. $\delta_i \in \mathsf{L}$ *is taken from* p, $\mathfrak{t}_i$ *and* C_i *are taken from* t, *and* $R \in \{1,2,3\}$ *for all* $0 < i \leq k$.

The relation $\mapsto$ *is defined by the following rules:*

- $\langle p_{i-1}, t_{i-1} \rangle \overset{\tau}{\longmapsto}_1 \langle p_i, t_i \rangle$ *if* $a_i = \tau$, $p_{i-1} \overset{\tau}{\longrightarrow}_p p_i$, $t_{i-1} = t_i$, *and* $t_{i-1} \notin \Sigma$,

- $\langle p_{i-1}, t_{i-1} \rangle \overset{\tau}{\longmapsto}_2 \langle p_i, t_i \rangle$ *if* $a_i = \tau$, $p_{i-1} = p_i$, $t_{i-1} \overset{\tau}{\longrightarrow}_t t_i$, *and* $t_{i-1} \notin \Sigma$,

- $\langle p_{i-1}, t_{i-1} \rangle \overset{(a_i,\delta_i)}{\underset{\mathfrak{t}_i,C_i}{\longmapsto}}_3 \langle p_i, t_i \rangle$ *if* $(a_i, \delta_i) \in \mathsf{A} \times \mathsf{L}$, $p_{i-1} \overset{(a_i,\delta_i)}{\longrightarrow}_p p_i$, $t_{i-1} \overset{(a_i,\delta_i)}{\underset{\mathfrak{t}_i,C_i}{\longrightarrow}}_t t_i$, *and* $t_{i-1} \notin \Sigma$.

The first expression in the definition of $\mapsto$ indicates that when the process under the test is executing an internal action from p_{i-1} to p_i, the test keeps its state. The second expression indicates that when the test is executing an internal action from t_{i-1} to t_i, the process under test keeps its state. The third expression indicates that when the action is not internal, the test and the process under test execute their respective action in parallel, and spend the same time while doing so. Moreover, the test also needs to check its time constraint.

If $k \in \mathbb{N}$ then c is finite, denoted by $|c| < \omega$; otherwise, it is infinite, that is, $|c| = \omega$. The projection $\text{proj}_p(c)$ of c on p is defined as $(p_{i-1}, (a_i, \delta_i), p_i)_{I_p^c} \in \Pi(p)$, where $I_p^c = \{0 < i \leq k : R_i \in \{1,3\}\}$. Similarly, the projection $proj_t(c)$ of c on t if defined as $(t_{i-1}, (a_i, \delta_i, \mathfrak{t}_i, C_i), t_i)_{i \in I_t^c} \in \Pi(t)$, where $I_t^c = \{0 < i \leq k : R_i \in \{2,3\}\}$.

Definition 8. COMPUTATIONS AND SUCCESSFUL COMPUTATIONS. *A partial computation* c *is called computation if it satisfies the following properties:*

1. $k \in \mathbb{N}$ *implies that* c *is maximal, that is,* $p_k \overset{\tau}{\not\longrightarrow}_p$, $t_k \overset{\tau}{\not\longrightarrow}_t$, *and* $\text{init}_p(p_k) \cap \text{init}_t(t_k) = \emptyset$ *(* p_k *and* t_k *cannot execute the same action) or the time delay of* p_k *does not satisfy the time constraint of* t_k; *and*

2. $k = \omega$ *implies* $\text{proj}_p(c) \in \Pi_I(p)$.

The set of all computations of p *and* t *is denoted by* $C(p,t)$.
  Computation c *is* successful *if* $t_{|c|} \in \Sigma$ *whenever* $|c| \in \mathbb{N}$, *and* $\text{proj}_t(c) \in \Pi_\omega(t)$ *whenever* $|c| = \omega$.

Definition 9. TIMED MAY AND MUST. p *may pass* t, *denoted by* p $\text{may}_\mathbb{T}$ t, *iff there exists at least one successful computation* $c \in C(p,t)$. *Analogously,* p *must pass* t, *denoted by* p $\text{must}_\mathbb{T}$ t *iff every computation* $c \in C(p,t)$ *is successful.*

Intuitively, an infinite computation of process p and test t is successful if the test passes through a set of ω-final states infinitely often. Hence some infinite computations can be successful in our setting. Since timed processes and timed tests potentially exhibit nondeterministic behaviour, one distinguishes between the possibility and inevitability of success. This is captured in the following definition of the timed may and must preorders.

Definition 10. TIMED MAY AND MUST PREORDERS. *Let* p *and* q *be timed processes. Then,* $p \sqsubseteq_\mathbb{T}^{may} q$ *iff* $\forall t \in \mathscr{T} : p\ \text{may}_\mathbb{T}\ t \implies q\ \text{may}_\mathbb{T}\ t$; *and* $p \sqsubseteq_\mathbb{T}^{must} q$ *iff* $\forall t \in \mathscr{T} : p\ \text{must}_\mathbb{T}\ t \implies q\ \text{must}_\mathbb{T}\ t$.

It is immediate that the relations $\sqsubseteq_\mathbb{T}^{may}$ and $\sqsubseteq_\mathbb{T}^{must}$ are preorders. They are defined analogously to the classical may and must preorders (which are based on labelled transition systems and restrict $\mathscr{T}$ to classical tests).

3.2 Alternative Characterizations

We now present alternative characterizations of the timed may and must preorders. The characterizations are similar in style to other characterizations and provide the basis for comparing the existing testing theory to our timed testing. The first characterization is similar to the characterization of other preorders [1, 8] and relates timed testing directly with the behaviour of processes.

Theorem 1. *1. $p \sqsubseteq_{\mathbb{T}}^{may} q$ iff $L_f(p) \subseteq L_f(q)$ and $L_\omega(p) \subseteq L_\omega(q)$.*

2. $p \sqsubseteq_{\mathbb{T}}^{must} q$ iff for all $w \in (\mathsf{A} \times \mathsf{L})^* \cup (\mathsf{A} \times \mathsf{L})^\omega$ such that $p \Downarrow w$ it holds that: (a) $q \Downarrow w$, (b) if $|w| < \omega$ then $\forall q' : q \overset{w}{\Longrightarrow} q'$ implies $\exists p' : p \overset{w}{\Longrightarrow} p'$ and $\text{init}_p(p') \subseteq \text{init}_q(q')$, and (c) if $|w| = \omega$ then $w \in L_\omega(p)$ implies $w \in L_\omega(q)$.

The second characterization is given in terms of timed trace inclusions, once more similarly to the characterization of other preorders [8, 21]. Note that we are now concerned with $\sqsubseteq_{\mathbb{T}}^{must}$ only, as the simplest $\sqsubseteq_{\mathbb{T}}^{may}$ is already characterized in terms of timed traces in Theorem 1.

To state this result we need to introduce the notion of *pure nondeterminism*. We call a timed process p purely nondeterministic, if for all states p' of p, (a) $p' \overset{\tau}{\longrightarrow}_p$ implies $p' \overset{(q,\delta)}{\not\longrightarrow}_p$ and $|\{((a,\delta),p'') : p' \overset{(a,\delta)}{\longrightarrow}_p p''\}| = 1$. Note that every timed process p can be transformed to a purely nondeterministic timed process p', such that $L_f(p) = L_f(p')$, $L_D(p) = L_D(p')$, $L_m(p) = L_m(p')$, and $L_\omega(p) = L_\omega(p')$ by splitting every transition $p' \overset{(a,\delta)}{\longrightarrow}_p p''$ into two transitions $p' \overset{\tau}{\longrightarrow}_p p_{\langle p',(a,\delta),p''\rangle}$ and $p_{\langle p',(a,\delta),p''\rangle} \overset{(a,\delta)}{\longrightarrow}_p p''$, where $p_{\langle p',(a,\delta),p''\rangle}$ is a new, distinct state.

Theorem 2. *Let p and q be timed processes such that p is purely nondeterministic. Then $p \sqsubseteq_{\mathbb{T}}^{must} q$ iff all of the following hold:*

$$
\begin{align}
L_D(q) &\subseteq L_D(p) \tag{1}\\
L_f(q) \setminus L_D(q) &\subseteq L_f(p) \tag{2}\\
L_m(q) \setminus L_D(q) &\subseteq L_m(p) \tag{3}\\
L_\omega(q) \setminus L_D(q) &\subseteq L_\omega(p) \tag{4}
\end{align}
$$

With respect to finite traces, the characterizations of timed tests differ from the ones of classical preorders by the addition of time variables. We also need to refine the classical characterizations so as to capture the behaviour of timed may and must testing with respect to infinite traces. The proofs of the two characterization theorems rely on the properties of the following specific timed tests.

- For $w = (a_i, \delta_i)_{0 < i \leq k} \in (\mathsf{A} \times \mathsf{L})^*$, let $t_w^{May,*} = (\mathsf{A} \cup \{\tau\}, \{x\}, T, \rightarrow, \{k\}, \emptyset, 0)$, where $T = \{0, 1, \ldots, k\}$ and $\rightarrow = \{(i-1, a_i, i, x = \delta_i, \{x\}) : 0 < i \leq k\}$.

- For $w = (a_i, \delta_i)_{0 < i \leq k} \in (\mathsf{A} \times \mathsf{L},)^\omega$, let $t_w^{May,\omega} = (\mathsf{A} \cup \{\tau\}, \{x\}, T, \rightarrow, \emptyset, T, 0)$, where $T = \mathbb{N}$, $\rightarrow = \{(i-1, a_i, i, x = \delta_i, \{x\}) : i > 0\}$.

- For $w = (a_i, \delta_i)_{0 < i \leq k} \in (\mathsf{A} \times \mathsf{L})^*$, let $t_w^{May,div} = (\mathsf{A} \cup \{\tau\}, \{x\}, T, \rightarrow, \emptyset, \{k\}, 0)$, where $T = \{0, 1, \ldots, k\}$, $\rightarrow = \{(i-1, a_i, i, x = \delta_i, \{x\}) : 0 < i \leq k\} \cup \{(k, \tau, k, \top)\}$.

- For $w = (a_i, \delta_i)_{0 < i \leq k} \in (\mathsf{A} \times \mathsf{L})^*$, let $t_w^{\Downarrow,*} = (\mathsf{A} \cup \{\tau\}, \{x\}, T, \rightarrow, \{s\}, \emptyset, 0)$, where $T = \{0, 1, \ldots, k\} \cup \{s\}$, $\rightarrow = \{(i-1, a_i, i, x = \delta_i, \{x\}) : 0 < i \leq k\} \cup \{(i, \tau, s, \top) : 0 < i \leq k\}$.

- For $w = (a_i, \delta_i)_{0 < i \leq k} \in (\mathsf{A} \times \mathsf{L})^\omega$, let $t_w^{\Downarrow,\omega} = (\mathsf{A} \cup \{\tau\}, \{x\}, T, \rightarrow, \{s\}, \mathbb{N}, 0)$, where $T = \mathbb{N} \cup \{s\}$, $\rightarrow = \{(i-1, a_i, i, x = \delta_i, \{x\}) : i > 0\} \cup \{(i, \tau, s, \top) : i > 0\}$.
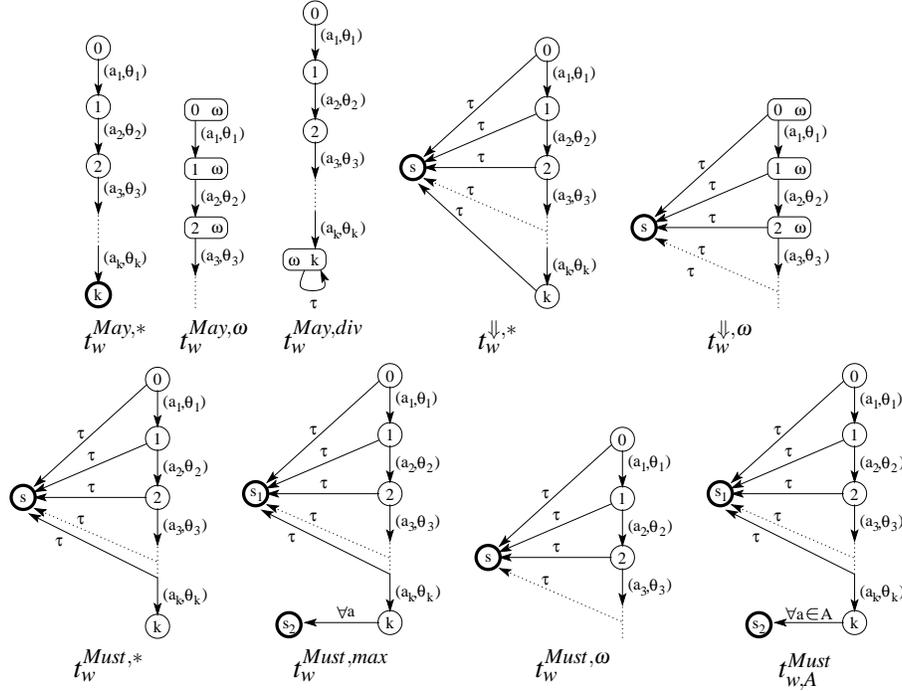
Figure 1: Timed tests used for the characterization of timed may and must preorders; ω-final states are marked by the symbol ω; success states are distinguished from regular states by thick borders.

- For $w = (a_i, \delta_i)_{0 < i \leq k} \in (A \times L)^*$, let $t_w^{Must,*} = (A \cup \{\tau\}, \{x\}, T, \rightarrow, \{s\}, \emptyset, 0)$, where $T = \{0, 1, \ldots, k\} \cup \{s\}$, $\rightarrow = \{(i-1, a_i, i, x = \delta_i, \{x\}) : 0 < i \leq k\} \cup \{(i, \tau, s, \top) : 0 \leq i < k\}$

- For $w = (a_i, \delta_i)_{0 < i \leq k} \in (A \times L)^*$, let $t_w^{Must,max} = (A \cup \{\tau\}, \{x\}, T, \rightarrow, \{s_1, s_2\}, \emptyset, 0)$, where $T = \{0, 1, \ldots, k\} \cup \{s_1, s_2\}$, $\rightarrow = \{(i-1, a_i, i, x = \delta_i, \{x\}) : 0 < i \leq k\} \cup \{(i, \tau, s_1, \top) : 0 \leq i < k\} \cup \{(k, a, s_2, \top) : (a, \mathfrak{t}) \in A \times \mathbb{T}(C)\}$.

- For $w = (a_i, \delta_i)_{0 < i \leq k} \in (A \times L)^\omega$, let $t_w^{Must,\omega} = (A \cup \{\tau\}, \{x\}, T, \rightarrow, \{s\}\emptyset, 0)$, where $T = \mathbb{N} \cup \{s\}$, $\rightarrow = \{(i-1, a_i, i, x = \delta_i, \{x\}) : i > 0\} \cup \{(i, \tau, s, \top) : i \in \mathbb{N}\}$.

- For $w = (a_i, \delta_i)_{0 < i \leq k} \in (A \times L)^*$ and $A \subseteq A$, let $t_{w,A}^{Must} = (A \cup \{\tau\}, \{x\}, T, \rightarrow, \{s_1, s_2\}, \emptyset, 0)$, where $T = \{0, 1, \ldots, k\} \cup \{s_1, s_2\}$, $\rightarrow = \{(i-1, a_i, i, x = \delta_i, \{x\}) : 0 < i \leq k\} \cup \{(i, \tau, s_1, \top) : 0 \leq i < k\} \cup \{(k, a, s_2, \top) : a \in A\}$.

The states of the tests are identified by numbers, but whenever we need to avoid ambiguity we will also refer to state i of some test as t_i instead of just i. The tests are depicted graphically in Figure 1.

Intuitively, while timed tests $t_w^{May,*}$ and $t_w^{May,\omega}$ test for the presence of a finite and infinite trace w, respectively, timed tests $t_w^{May,div}$, $t_w^{\Downarrow,*}$, and $t_w^{\Downarrow,\omega}$ are capable of detecting divergent behaviour when executing trace w. These are "presence" tests, that check whether a trace w (finite or infinite) exists in the implementation. Timed tests $t_w^{Must,*}$, $t_w^{Must,max}$, and $t_w^{Must,\omega}$ test for the absence of the finite trace, maximal trace, and ω-final trace (that is, trace that goes through infinite occurrences of ω-final states) w, respectively. Timed must testing is a bit trickier, since we cannot feasibly check all the possible traces or computations exhaustively (as we need to do according to the definition of must testing). So we think the other way around: We assume one "failure trace," which does not satisfy the test and leads to failure. If there exists at least one such failure trace, then the test fails. On the other hand, if we cannot find the failure trace in the implementation, the test succeeds. We then test the absence of this trace. Finally, timed test $t_{w,A}^{Must}$

is capable of comparing the initial action sets of states reached when executing trace w against a subset $A \subseteq \mathsf{A}$. Note that we use the tightest time constraint possible in our test. We denote $x = \delta_i, \{x\}$ by θ_i in what follows (and also in Figure 1).

Our specific timed tests satisfy the following desired properties:

Lemma 3.   *1. Let $w \in (\mathsf{A} \times \mathsf{L})^*$. Then, $w \in L_f(p)$ iff p may$_{\mathbb{T}}$ $t_w^{May,*}$.*

2. Let $w \in (\mathsf{A} \times \mathsf{L})^\omega$. Then $w \in L_\omega(p)$ iff p may$_{\mathbb{T}}$ $t_w^{May,\omega}$.

3. Let $w \in (\mathsf{A} \times \mathsf{L})^*$. Then, $w \in L_D(p)$ iff p may$_{\mathbb{T}}$ $t_w^{May,div}$.

4. Let $w \in (\mathsf{A} \times \mathsf{L})^*$. Then, $p \Downarrow w$ iff p must$_{\mathbb{T}}$ $t_w^{\Downarrow,*}$.

5. Let $w \in (\mathsf{A} \times \mathsf{L})^* \cup (\mathsf{A} \times \mathsf{L})^\omega$. Then, $p \Downarrow w$ iff p must$_{\mathbb{T}}$ $t_w^{\Downarrow,\omega}$.

6. Let $w \in (\mathsf{A} \times \mathsf{L})^*$ such that $p \Downarrow w$. Then, $w \notin L_f(p)$ iff p must$_{\mathbb{T}}$ $t_w^{Must,*}$.

7. Let $w \in (\mathsf{A} \times \mathsf{L})^*$ such that $p \Downarrow w$. Then, $w \notin L_m(p)$ iff p must$_{\mathbb{T}}$ $t_w^{Must,max}$.

8. Let $w \in (\mathsf{A} \times \mathsf{L})^\omega$ such that $p \Downarrow w$. Then, $w \notin L_\omega(p)$ iff p must$_{\mathbb{T}}$ $t_w^{Must,\omega}$.

Proof. The proofs are simple analyses of the potential computations arising when running the timed tests in lock-step (to a deadlock or successful state) with arbitrary timed processes. Let $w = ((a_i, \delta_i))_{0 < i \leq k}$ for some $k \in \mathbb{N} \cup \{\omega\}$.

- Item 1, $\Rightarrow$: $w \in L_f(p)$, and thus $p_0 \stackrel{(a_1,\delta_1)}{\Longrightarrow} p_1 \stackrel{(a_2,\delta_2)}{\Longrightarrow} \cdots \stackrel{(a_k,\delta_k)}{\Longrightarrow} p_k$ (Definition 4). On the other hand, $t_0^{May,*} \stackrel{(a_1,\delta_1)}{\underset{\mathfrak{t}_1}{\Longrightarrow}} t_1^{May,*} \stackrel{(a_2,\delta_2)}{\underset{\mathfrak{t}_2}{\Longrightarrow}} \cdots \stackrel{(a_k,\delta_k)}{\underset{\mathfrak{t}_k}{\Longrightarrow}} t_k^{May,*}$ (definition of $t_w^{May,*}$ including the form of $\mathfrak{t}_i$). Therefore, $(\langle p_{i-1}, t_{i-1}^{May,*}\rangle \stackrel{(a_i,\delta_i)}{\underset{\mathfrak{t}_i}{\longmapsto}}_R \langle p_i, t_i^{May,*}\rangle)_{0<i\leq k}$, so w is the trace of a potential computation c for both p and $t_w^{May,*}$. In fact w is even the trace of a computation of p and $t_w^{May,*}$ (indeed, $t_k^{May,*} \stackrel{\tau}{\not\longrightarrow}$ and $\mathrm{init}_p(p_k) \cap \mathrm{init}_t(t_k^{May,*}) = \emptyset$), and is further the trace of a successful computation (since $t_k^{May,*} \in \Sigma$). It then follows that p may$_{\mathbb{T}}$ $t_w^{May,*}$.

  $\Leftarrow$: Given that p may$_{\mathbb{T}}$ $t_w^{May,*}$, we have a successful computation c of p and $t_w^{May,*}$. That is, $(\langle p_{i-1}, t_{i-1}^{May,*}\rangle \stackrel{(a_i,\delta_i)}{\underset{\mathfrak{t}_i}{\longmapsto}}_R \langle p_i, t_i^{May,*}\rangle)_{0<i\leq k}$, $t_k^{May,*} \stackrel{\tau}{\not\longrightarrow}$, $\mathrm{init}_p(p_k) \cap \mathrm{init}_t(t_k^{May,*}) = \emptyset$, and $t_w^{May,*} = t_k^{May,*} \in \Sigma$. By a reverse argument we conclude then that $w \in L_f(p)$ ($t_0^{May,*} \stackrel{(a_1,\delta_1)}{\underset{\mathfrak{t}_1}{\Longrightarrow}} t_1^{May,*} \stackrel{(a_2,\delta_2)}{\underset{\mathfrak{t}_2}{\Longrightarrow}} \cdots \stackrel{(a_k,\delta_k)}{\underset{\mathfrak{t}_k}{\Longrightarrow}} t_k^{May,*}$, then $p_0 \stackrel{(a_1,\delta_1)}{\Longrightarrow} p_1 \stackrel{(a_2,\delta_2)}{\Longrightarrow} \cdots \stackrel{(a_k,\delta_k)}{\Longrightarrow} p_k$, and thus $w \in L_f(p)$). Items 2 and 3 are proven similarly.

- Item 4, $\Rightarrow$: Assume that p must$_{\mathbb{T}}$ $t_w^{\Downarrow,*}$ does not hold. However, the trace w passes $t_w^{\Downarrow,*}$ (by the definition of $t_w^{\Downarrow,*}$), only divergence can cause the test to fail. So for some $0 < l \leq k$ there exists one trace $p_0 \stackrel{(a_1,\delta_1)}{\Longrightarrow} p_1 \stackrel{(a_2,\delta_2)}{\Longrightarrow} \cdots \stackrel{(a_l,\delta_l)}{\Longrightarrow} p_l \stackrel{\tau}{\longrightarrow} p_l \cdots$ which means that $p \Uparrow w$, a contradiction. So it must be that p must$_{\mathbb{T}}$ $t_w^{\Downarrow,*}$.

  $\Leftarrow$: Assume that $p \Uparrow w$. Then for some $0 < l \leq k$ there exists one trace $p_0 \stackrel{(a_1,\delta_1)}{\Longrightarrow} p_1 \stackrel{(a_2,\delta_2)}{\Longrightarrow} \cdots \stackrel{(a_l,\delta_l)}{\Longrightarrow} p_l \stackrel{\tau}{\longrightarrow} p_l \cdots$ which fails the test $t_w^{\Downarrow,*}$. This contradicts the condition p must$_{\mathbb{T}}$ $t_w^{\Downarrow,*}$ and so it must be that $p \Downarrow w$. Item 5 is proven similarly.

- Item 6, $\Rightarrow$: Assume that p must$_{\mathbb{T}}$ $t_w^{Must,*}$ does not hold. According to the definition of $t_w^{Must,*}$, there are two ways for p to fail the test: Either $p_0 \stackrel{(a_1,\delta_1)}{\Longrightarrow} p_1 \stackrel{(a_2,\delta_2)}{\Longrightarrow} \cdots \stackrel{(a_i,\delta_i)}{\Longrightarrow} p_i \stackrel{\tau}{\longrightarrow} p_i \cdots$, or $p_0 \stackrel{(a_1,\delta_1)}{\Longrightarrow} p_1 \stackrel{(a_2,\delta_2)}{\Longrightarrow} \cdots \stackrel{(a_k,\delta_k)}{\Longrightarrow} p_k$. These contradict the conditions $p \Downarrow w$ or $w \notin L_f(p)$, respectively.

$\Leftarrow$: Assume that $w \in L_f(p)$. By the definition of $t_w^{Must,*}$, w fails to pass this test. This contradicts the condition that p must$_\mathbb{T}$ $t_w^{Must,*}$. Items 7 and 8 are proven similarly.

$\square$

The proof of Theorem 1 relies extensively on these intuitive properties of timed tests. Notice that the usage of ω-state tests (that is, tests that accept based on an acceptance family, not only on Σ)—even when discussing finite-state timed processes—is justified by our view that timed tests represent the arbitrary, potentially irregular behaviour of the unknown real-time environment.

Proof of Theorem 1 Item 1 of the theorem is fairly immediate. For the $\Rightarrow$ direction we distinguish the following cases: $w \in L_f(p)$ implies that p may$_\mathbb{T}$ $t_w^{May,*}$. Since $p \sqsubseteq_\mathbb{T}^{may} q$ it follows that q may$_\mathbb{T}$ $t_w^{May,*}$ and thus $w \in L_f(q)$. $w \in L_\omega(p)$ has two sub-cases: (a) If $|w| = \omega$, then p may$_\mathbb{T}$ $t_w^{May,\omega}$. Since $p \sqsubseteq_\mathbb{T}^{may} q$ it follows that q may$_\mathbb{T}$ $t_w^{May,\omega}$ and thus $w \in L_\omega(q)$. (b) If $|w| < \omega$, then p may$_\mathbb{T}$ $t_w^{May,div}$. Since $p \sqsubseteq_\mathbb{T}^{may} q$ it follows that q may$_\mathbb{T}$ $t_w^{May,div}$ and thus $w \in L_\omega(q)$.

We go now to the $\Leftarrow$ direction for Item 1. Let t be any timed test such that p may$_\mathbb{T}$ t, that is, there exists a successful computation $c \in C(p,t)$ with $w = \text{trace}(\text{proj}_p(c)) = \text{trace}(\text{proj}_t(c))$. If $|w| = \omega$, then $w \in L_\omega(p)$ and thus $w \in L_\omega(q)$ (since $L_\omega(p) \subseteq L_\omega(q)$). It follows that we can construct a successful computation $c' \in C(q,t)$ such that $w = \text{trace}(\text{proj}_q(c')) = \text{trace}(\text{proj}_t(c'))$ and $\text{proj}_t(c') = \text{proj}_t(c)$. It follows that q may$_\mathbb{T}$ t and therefore $q \sqsubseteq_\mathbb{T}^{may} q$. If $|w| < \omega$, we can split the proof into two cases: either $w \in L_f(p)$ or $w \in L_\omega(p)$. We an then establish that q may$_\mathbb{T}$ t as above.

On to Item 2 now. For the $\Rightarrow$ direction we have that $p \sqsubseteq_\mathbb{T}^{must} q$, $w \in (A \times L)^* \cup (A \times L)^\omega$ such that $p \Downarrow w$. Then p must$_\mathbb{T}$ $t_w^{\Downarrow,*}$ or p must$_\mathbb{T}$ $t_w^{\Downarrow,\omega}$ (Lemma 3), then q must$_\mathbb{T}$ $t_w^{\Downarrow,*}$ or q must$_\mathbb{T}$ $t_w^{\Downarrow,\omega}$ (Definition 10), thus $q \Downarrow w$ (Lemma 3). We further distinguish two cases, depending on whether $|w| = \omega$ or not:

If $|w| < \omega$, let $q \overset{w}{\Longrightarrow} q'$ for some q', that is, $w \in L_f(q)$. Assume that there is no p' such that $p \overset{w}{\Longrightarrow} p'$ and $\text{init}_p(p') \subseteq \text{init}_q(q')$. Suppose that $p \overset{w}{\nLongrightarrow}$ that is, $w \notin L_f(p)$. Then p must$_\mathbb{T}$ $t_w^{Must,*}$ (Lemma 3), so q must$_\mathbb{T}$ $t_w^{Must,*}$ (Definition 10). However, q must$_\mathbb{T}$ $t_w^{Must,*}$ does not hold (contrapositive of Lemma 3), a contradiction. Suppose now that $p \overset{w}{\Longrightarrow}$. Let then $X = \{(a,\delta) \in \text{init}_p(p') : p \overset{w}{\Longrightarrow} p'\} \neq \emptyset$. Since $\text{init}_p(p') \not\subseteq \text{init}_q(q')$ (assumption), for every $A \in X$ there exists an $(a,\delta) \in A \setminus \text{init}_q(q')$. Let B be the set of all such actions a (ignoring the time actions). It is then immediate then that p must$_\mathbb{T}$ $t_{w,B}^{Must}$ (by the construction of $t_{w,B}^{Must}$); however, it is not the case that q must$_\mathbb{T}$ $t_{w,B}^{Must}$ (since $q' \overset{(a,\delta)}{\nLongrightarrow}$ for any $(a,\delta) \in (B,L)$). This contradicts the assumption that $p \sqsubseteq_\mathbb{T}^{must} q$.

If on the other hand $|w| = \omega$, assume that $w \notin L_\omega(p)$. Then p must$_\mathbb{T}$ $t_w^{Must,\omega}$ (Definition 10) and thus $w \notin L_\omega(q)$ (Lemma 3). This contradicts with $w \in L_\omega(q)$ (given).

Finally, for the $\Leftarrow$ direction of Item 2, let t be any timed test such that q must$_\mathbb{T}$ t does not hold, that is, there exists an unsuccessful computation $c = (\langle q_{i-1}, t_{i-1}\rangle, (a_i, \delta_i, \mathbb{t}_i), \langle q_i, t_i\rangle)_{0 < i \leq k} \in C(q,t)$ (Definition 9). Let $w = \text{trace}(\text{proj}_p(c)) = \text{trace}(\text{proj}_t(c))$.

Assume that $p \Uparrow w$. We can then construct an unsuccessful, infinite computation c' which resembles c until p can engage in its timed divergent computation and then we force t not to contribute anymore. Then $\text{proj}_p(c') \in \Pi_\omega(p)$ and $\text{proj}_t(c') \notin \Pi_I(t)$ (because $|proj_p(c')| < \omega$).This implies that p must$_\mathbb{T}$ t does not hold (Definition 9) and thus $p \sqsubseteq_\mathbb{T}^{must} q$ (since q must$_\mathbb{T}$ t does not hold, by the contrapositive of Definition 10).

Assume now that $p \Downarrow w$, that is, $w \notin L_D(p)$. We have again two cases depending on whether $|c| < \omega$ or not.

Whenever $|c| < \omega$, (a) $w \in L_f(q)$, $q \overset{w}{\Longrightarrow} q'$ for some q' and $t_k \neq \Sigma$ by definition of $t_w^{Must,*}$, (b) $q_k \overset{\tau}{\nrightarrow}$, $t_k \overset{\tau}{\nrightarrow}$, $\text{init}_q^c(q_k) \cap \text{init}^c(t_k) = \emptyset$ by definition of $t_w^{Must,max}$; and (c) $\exists p' : p \overset{w}{\Longrightarrow} p', \text{init}_p^c(p') \subseteq \text{init}_q^c(q')$ by condition 2(b). By observations (a)–(c) we have a finite computation $c' = (\langle p_{i-1}, t'_{i-1}\rangle, (a_i, \delta_i, \mathbb{t}_i),$

$\langle p_i, t_i' \rangle)_{0 < i \le l} \in C(p,t)$ with $\mathrm{proj}_t(c') = \mathrm{proj}_t(c)$ and $\langle p_l, t_l \rangle = \langle p'', t_k \rangle$, where $p' \stackrel{\varepsilon}{\Longrightarrow} p''$ for some $p'' \stackrel{\tau}{\nrightarrow}_p$. Note that such a p'' must exist since $p \Downarrow w$. Then $\mathrm{init}_p{}^c(p'') \subseteq \mathrm{init}_p{}^c(p')$, definition of c' and p'', and observations (a) and (b) above imply that $\mathrm{init}_p{}^c(p'') \cap \mathrm{init}_t{}^c(t_k) \subseteq \mathrm{init}_q{}^c(q') \cap \mathrm{init}_t{}^c(t_l') \subseteq \mathrm{init}_q{}^c(q_k) \cap \mathrm{init}_t{}^c(t_l)$; thus c' cannot be extended. Since $t_l' = t_k \notin \Sigma$, c' is unsuccessful, so p must$_\mathbb{T}$ t does not hold.

Whenever $|c| = \omega$, q must$_\mathbb{T}$ $t_w^{Must,\omega}$ does not hold. It follows that $w \in L_\omega(q)$, and thus $w \in L_\omega(p)$ (given). So p must$_\mathbb{T}$ $t_w^{Must,\omega}$ does not hold either (contrapositive of Lemma 3). In all, $p \sqsubseteq_\mathbb{T}^{must} q$, as desired. □

The proof of Theorem 2 also relies on the properties of the timed tests introduced in Lemma 3.

Proof of Theorem 2 For the $\Rightarrow$ direction, assume that $p \sqsubseteq_\mathbb{T}^{must} q$ and let $w \in (\mathsf{A} \times \mathsf{L})^* \cup (\mathsf{A} \times \mathsf{L})^\omega$. Then,

(1) $w \in L_D(q)$ implies $q \Uparrow w$, so it is not the case that q must$_\mathbb{T}$ $t_w^{\Downarrow,\omega}$ (Lemma 3(5)). Therefore it is not the case that p must$_\mathbb{T}$ $t_w^{\Downarrow,\omega}$ (since $p \sqsubseteq_\mathbb{T}^{must} q$), so $p \Uparrow w$, or $w \in L_D(p)$, as desired.

(2) $w \in L_f(q) \setminus L_D(q)$ implies $q \Downarrow w$ and thus $p \Downarrow w$ (same as (1) but using Lemma 3(4)). In addition, it is not the case that q must$_\mathbb{T}$ $t_w^{Must,*}$ (Lemma 3(6)) and thus p must$_\mathbb{T}$ $t_w^{Must,*}$ does not hold (since $p \sqsubseteq_\mathbb{T}^{must} q$). Therefore, $w \in L_f(p)$, again as desired.

The proofs of (3) and (4) are the same as the proof of (2) using Lemma 3(7) and Lemma 3(8), respectively.

On to the $\Leftarrow$ direction now. We assume that (1), (2), (3), and (4) hold. We further assume that there exists a timed test t such that q must$_\mathbb{T}$ t does not hold (if such a test does not exist then $p \sqsubseteq_\mathbb{T}^{must} q$ for any process p). Thus there exists an unsuccessful computation $c = (\langle q_{i-1}, t_{i-1} \rangle (a_i, \delta_i) \langle q_i, t_i \rangle)_{0 < i \le k} \in C(q,t)$, with $w = \mathrm{trace}(\mathrm{proj}_q(c)) = trace(\mathrm{proj}_t(c))$.

If $p \Uparrow w$ then construct an unsuccessful, infinite computation c' which resembles c until p can engage in its divergent computation, at which point t can be forced to stop contributing to c'. Thus $q \Uparrow w$ and it is not the case that p must$_\mathbb{T}$ t.

If $p \Downarrow w$, $|c| < \omega$, and $t_k \notin \Sigma$ we distinguish two cases:

Let $w \in L_f(q) \setminus L_m(q)$. Then there exists some $(a,\delta) \in \mathsf{A} \times \mathsf{L}$ such that $q_k \stackrel{(a,\delta)}{\longrightarrow}_q$ but $t_k \stackrel{(q,\delta)}{\nrightarrow}_t$. That is, $w \cdot (a,\delta) \in L_m(q)$ and so (by (3)) $w \cdot (a,\delta) \in L_m(p)$. Since p is purely nondeterministic, we can construct a finite computation $c' = (\langle q_{i-1}, t_{i-1}' \rangle (a_i, \delta_i) \langle q_i, t_i' \rangle)_{0 < i \le l} \in C(q,t)$ where $\mathrm{proj}_t(c) = \mathrm{proj}_t(c')$, $t_l' = t_k$, and $p_l \stackrel{(a,\delta)}{\longrightarrow}_p$. The computation is maximal (since $t_k = t_j' \stackrel{(q,\delta)}{\nrightarrow}_{t'}$) and unsuccessful (since $|c'| < \omega$ and $t_l' \notin \Sigma$). Therefore, p must$_\mathbb{T}$ t does not hold.

Let now $w \in L_m(q)$ (and thus $w \in L_m(p)$). We can then construct a maximal computation c' as above and then p must$_\mathbb{T}$ t does not hold given that q must$_\mathbb{T}$ t does not hold.

Finally, if $p \Downarrow w$ and $|c| = \omega$, since $\mathrm{proj}_t(c) \notin \Pi_\omega(t)$, $\mathrm{proj}_t(c) \in \Pi_\omega(q)$, and $w \in L_\omega(p)$, we can construct an infinite computation $c' \in C(q,t)$ such that $\mathrm{proj}_t(c) = \mathrm{proj}_t(c')$. Similar to the above, c' is unsuccessful and so p must$_\mathbb{T}$ t does not hold. All the cases lead to $p \sqsubseteq_\mathbb{T}^{must} q$, as desired. □

4 Timed Test Generation

We now establish an algorithm that generates equivalent timed tests starting from any TPTL formula. This can also be regarded as a relation between TPTL and timed must testing. Our result builds on timed Büchi automata [2] approaches to LTL model checking [11, 14, 15, 22, 23].

Theorem 4. *Given a TPTL formula ϕ there exists a test T_ϕ such that $p \vDash_\gamma \phi$ for any suitable γ if and only if p must$_\mathbb{T}$ T_ϕ for any timed process p. Furthermore T_ϕ can be algorithmically constructed starting from ϕ.*

Proof. The test T_ϕ will be first constructed to consider only infinite computations, and will be then modified to consider maximal traces. In the following p is an arbitrary timed process. A sub-formula of any TPTL formula ϕ is defined inductively as follows:

1. ϕ is a sub-formula of ϕ,

2. any formula t involving only terms of form θ and relational and boolean operators (henceforth called "time formula") occurring in ϕ is a sub-formula of ϕ, but no sub-formula of t is a sub-formula of ϕ (any time formula is indivisibly a sub-formula of ϕ),

3. if $\neg\xi$ is a sub-formula of ϕ, then so is ξ,

4. if $O\,\xi$ is a sub-formula of ϕ then so is ξ, $O \in \{\mathsf{X}, x.\}$,

5. if $\xi_1\, O\, \xi_2$ is a sub-formula of ϕ then so are ξ_1 and ξ_2, $O \in \{\wedge, \vee, \mathsf{U}\}$.

To consider infinite traces we build on the existing work relating Büchi automata and LTL [15, 22]. Let C_ϕ be the set that contains exactly all the clocks that occur in a TPTL formula ϕ, and let $\mathscr{C}(\phi)$ be the closure of ϕ, that is, the set containing exactly all the sub-formulae of ϕ. Furthermore, let $\Theta(\phi) \subseteq \mathscr{C}(\phi)$ contain exactly all the sub-formulae of ϕ that are time formulae.

The construction of our test T_ϕ is then based on the untimed construction developed by Vardi and Wolper [22]. Specifically, we add timing constraints to their automaton.

We first consider the "local" automaton $L_\phi = (2^{\mathscr{C}(\phi)}, \mathsf{C}_\phi, N_L, \rightarrow_L, \emptyset, N_L, s_0)$. The set of states contain all the subsets of $\mathscr{C}(\phi)$ that have no internal inconsistency, plus one designated initial state. The local automaton does not impose any acceptance condition (for indeed all of its states are ω-final). A state s has no internal inconsistency iff it satisfies the following conditions:

1. $\psi \in s$ iff $\neg\psi \notin s$ for all $\psi \in \mathscr{C}(\phi)$,

2. $\xi \wedge \psi \in s$ iff $\xi \in s$ and $\psi \in s$ for all $\xi \wedge \psi \in \mathscr{C}(\phi)$,

3. $x.\psi \in s$ implies $\psi \in s$.

The transition relation is defined as $s \xrightarrow[\mathbb{t},C]{a} {}_L t$ iff $a = t$, $C = \{x \in \mathsf{C}_\phi : x.\psi \in s \wedge \psi \in t\}$, $\mathbb{t} = \bigwedge(\Theta(\phi) \cap s)$, $\psi \in t \wedge x.\psi \in s$ implies $x.\psi \notin t$, and either

1. $s = s_0$ and $\phi \in a$, or

2. $s \neq s_0$ and

    (a) for all $\psi \in \mathscr{C}(\phi)$, $\mathsf{X}\,\psi \in s$ iff $\psi \in t$,
    (b) for all $\xi\, \mathsf{U}\, \phi \in \mathscr{C}(\phi)$ either $\psi \in s$, or $\xi \in s$ and $\xi\, \mathsf{U}\, \psi \in t$.

Note that L_ϕ does not impose any acceptance conditions (as mentioned before), but enforces all the time constraints present in the original formula ϕ. Indeed, we note that at every moment frozen in time by a construction $x.\psi$ we reset the respective clock in the local automaton (indeed, the set C of clocks reset by a transition out of s contains exactly all the sets of clocks x reset by an $x.$ construction in s). Later, whenever a time formula is encountered, that formula is added to the time constraints that enable the transition. Clearly checking the time formula to determine that the transition is enabled has the intended effect; indeed, the time formula needs to be true for the whole formula ϕ to be true, and the semantics of time in a timed transition system ensures that every clock measures the time from when it was reset in the transition system (that is, frozen in the formula) to the current moment in time, as desired.

The acceptance conditions will be imposed by the "eventuality" automaton $E_\phi = (2^{\mathscr{C}(\phi)}, \emptyset, 2^{\mathscr{E}(\phi)}, \rightarrow_E , \emptyset, \{\emptyset\}, \emptyset)$, with $\mathscr{E}(\phi) = \{\xi\, \mathsf{U}\, \psi \in \mathscr{C}(\phi)\}$ and $s \xrightarrow[\top,\emptyset]{a} {}_E t$ iff

1. $s = \emptyset$ and $\xi \cup \psi \in t$ iff $\psi \notin a$ for all $\xi \cup \psi \in a$, and

2. $s \neq \emptyset$ and $\xi \cup \psi \in t$ iff $\psi \notin a$ for all $\xi \cup \psi \in s$.

The eventuality automaton is in fact identical to the one developed elsewhere [22]. This automaton has in particular no time constraints. It tries to satisfy the eventualities of the formula (with no regard of time constraints). When the current state is the initial state $\emptyset$, it looks to see which eventualities must be satisfied; afterward the current state keeps track of which eventualities have yet to be satisfied.

The test T_ϕ is obtained by taking the cross-product of L_ϕ and E_ϕ. The cross-product is taken using the usual (untimed) construction [22], noting that only the transitions in $\rightarrow_L$ contain time constraints and/or clock resets (and these go into the composite automaton together with the actions that accompany them in L_ϕ). This test characterizes traces over $2^{\mathscr{C}(\phi)} \times \mathsf{L}$; in order to switch to $\mathsf{A} \times \mathsf{L}$ we project over A the action labels of all the transitions, as done previously [22].

The construction of T_ϕ follows carefully the construction for the untimed case. It is then immediate that T_ϕ is correct as far as untimed words are concerned, in the sense that $\mathrm{trace}(\mathrm{proj}_p(c)) \vDash \phi$ for exactly all the untimed, successful infinite computations $c \in C(p, T_\phi)$. The timing information is added (via L_ϕ), as detailed above. In all, for all the infinite computations c,

$$\mathrm{trace}(\mathrm{proj}_p(c)) \vDash_\gamma \phi \text{ for exactly all the successful infinite computations } c \in C(p, T_\phi) \qquad (5)$$

We now enhance T_ϕ so that it also accepts finite maximal traces. To do this, for every state s in T_ϕ, we check if all the formulae contained in s are satisfied by the trace ε. Checking for acceptance of the trace ε (like for any trace of fixed length) can be done algorithmically along the structure of the formula ϕ. Then, for every state s in T_ϕ such that each TPTL formula ϕ labeling s is satisfied by ε, we add a transition $s \xrightarrow{\tau} \Delta$, where Δ is a fresh new state. We use such a state to distinguish from other states also having no outgoing transitions; these states represent deadlock due to inconsistent sub-formulae of ϕ. Note that Δ is a deadlock state, but it differs from the some other deadlock states to allow the identification of maximal time traces. The final states of T_ϕ will then be the set containing only the state Δ thus introduced. The other deadlocking states need not concern us any further since they are not success states. For all the finite computations c we then have:

$$\mathrm{trace}(\mathrm{proj}_p(c)) \vDash_\gamma \phi \text{ for exactly all the successful maximal computations } c \in C(p, T_\phi) \qquad (6)$$

Indeed, according to the algorithm outlined above, if $\chi(p) \vDash_\gamma \phi$ (where $\chi(x)$ stands for $\mathrm{trace}(\mathrm{proj}_x(c))$), then $s_0 \xRightarrow{\chi(T_\phi)} s \xrightarrow{\tau} \Delta$ holds (with s_0 the initial state of T_ϕ). That is $s_0 \xRightarrow{\chi(T_\phi)} s \not\rightarrow$. Thus, c is a maximal (and also successful) computation. Conversely, if c is a successful maximal computation, then there exits $s_0 \xRightarrow{\chi(T_\phi)} s \not\rightarrow$ in T_ϕ. According to the algorithm of T_ϕ, there exits then $s_0 \xRightarrow{\chi(T_\phi)} s \xrightarrow{\tau} \Delta$. Thus, c is maximal.

That $p \vDash_\gamma \phi$ if and only if $p \ \mathbf{must}_{\mathbb{T}} \ T_\phi$ now follows immediately from Properties (5) and (6) (and Definition 9), as desired. $\square$

5   Conclusion

We proposed in this paper a model of timed tests based on timed transition systems. We addressed the problem of characterizing infinite behaviours of timed processes by developing a theory of timed ω-final states. This theory is inspired by the acceptance family of Büchi automata. We also extended the testing theory of De Nicola and Hennessy to timed testing. We then studied the derived timed may and must preorders and developed an alternative characterization for them. This characterization is very similar to the characterization of De Nicola and Hennessy's testing preorders, which shows that our preorders are

fully back compatible: they extend the existing preorders as mentioned, but they do not take anything away. Further into the characterization process we also showed that the timed must preorder is equivalent to a variant of reverse timed trace inclusion when its first argument is purely nondeterministic.

We then presented an algorithm for test generation out of TPTL formulae. We note that both processes and tests are represented by timed transition systems instead of automata (that is, their number of states is not necessarily finite). This is consistent with the huge body of similar constructs in the untimed domain. However, the timed tests produced out of TPTL formulae (Theorem 4) are always finite automata (meaning that their set of states is always finite). This is quite nice to have for a very practical reason, as infinite-state tests must be further refined to become practical characterization tools.

One significance of our results stems from the fact that while algorithms and techniques for real-time testing have been studied actively [6, 19], the domain still lacks solid techniques and theories. Our paper attempts to present a general theoretical framework for real-time testing, in order to facilitate the subsequent evolution of the area. To serve such a purpose our framework is as close as possible to the original framework of (untimed) testing, as shown in our characterization theorems. In addition, our characterization is surprisingly concise in terms of the test cases needed.

Beside the obvious use of the algorithm for test generation, the algorithm also relates the satisfaction relation of TPTL to the $\text{must}_\mathbb{T}$ operator. We therefore note that the algebraic and logic specification techniques attempt to achieve the same thing (conformance testing) in two different ways. Each of them is more convenient for certain systems, as they both have advantages and disadvantages. Indeed, logic approaches allow loose specifications (and therefore greater latitude in their implementations) but lack compositionality, while algebraic specifications are compositional by definition but are often seen as too detailed (and therefore too constraining). Therefore our test generation algorithm also forms the basis of bringing logic and algebraic specifications together, thus obtaining heterogeneous specifications for real-time systems that combine the advantages of the two paradigms. This has the potential of providing a uniform basis for analyzing heterogeneous real-time system specifications with a mixture of timed transition systems and timed logic formulae.

We consider TPTL without congruence because the theory of timed transition systems does not offer a congruence mechanism. Such a mechanism could likely be introduced without much difficulty, but to our knowledge none of the temporal logics (timed of not) used in practical settings take congruence into consideration, so we preferred to leave time transition systems intact and exclude congruence from TPTL instead. This does diminish the expressiveness of TPTL [3], but we are still significantly above the expressiveness of most real-time temporal logics [3, 4].

We have on purpose avoided the discussion of discrete versus continuous time. All the results and definitions from this paper are oblivious to whether time is considered discrete or continuous. We therefore leave the decision of discreteness to the future uses of this work.

This paper is only a first step in the direction of combining operational and assertional styles of timed specifications; the studying of techniques mixing operators from timed process algebras and TPTL is a widely open area. Indeed, we establish algorithms for constructing timed tests from TPTL formulae; how to go the other way around is still open for research. The timed preorder testing developed from De Nicola and Hennessy's preorder testing is not the only testing relation; other testing relations with the addition of time constraints will also be exciting to investigate.

References

[1] S. ABRAMSKY, *Observation equivalence as a testing equivalence*, Theoretical Computer Science, 53 (1987), pp. 225–241.

[2] R. ALUR AND D. L. DILL, *A theory of timed automata*, Theoretical Computer Science, 126 (1994), pp. 183–235.

[3] R. ALUR AND T. A. HENZINGER, *Real-time logics: Complexity and expressiveness*, Information and Computation, 104 (1993), pp. 35–77.

[4] P. BELLINI, R. MATTOLINI, AND P. NESI, *Temporal logics for real-time system specification*, ACM Computing Surveys, 32 (2000), pp. 12–42.

[5] A. BERTOLINO, *Software testing research: Achievements, challenges, dreams*, in Future of Software Engineering, IEEE, 2007, pp. 85–103.

[6] L. B. BRIONES AND E. BRINKSMA, *A test generation framework for quiescent real-time systems*, in Formal Approaches to Testing of Software, 2004, pp. 71–85.

[7] M. BROY, B. JONSSON, J.-P. KATOEN, M. LEUCKER, AND A. PRETSCHNER, eds., *Model-Based Testing of Reactive Systems: Advanced Lectures*, vol. 3472 of Lecture Notes in Computer Science, Springer, 2005.

[8] S. D. BRUDA, *Preorder relations*, in Model-Based Testing of Reactive Systems: Advanced Lectures, M. Broy, B. Jonsson, J.-P. Katoen, M. Leucker, and A. Pretschner, eds., vol. 3472 of Lecture Notes in Computer Science, Springer, 2005, pp. 117–149.

[9] S. D. BRUDA AND S. G. AKL, *Real-time computation: A formal definition and its applications*, International Journal of Computers and Applications, 25 (2003), pp. 247–257.

[10] E. M. CLARKE, E. A. EMERSON, AND A. P. SISTLA, *Automatic verification of finite state concurrent systems using temporal logic specification*, ACM Transactions on Programming Languages and Systems, 8 (1986), pp. 244–263.

[11] E. M. CLARKE, O. GRUMBERG, AND D. A. PELED, *Model Checking*, MIT Press, 1999.

[12] R. CLEAVELAND AND G. LÜTTGEN, *Model checking is refinement—Relating Büchi testing and linear-time temporal logic*, Tech. Rep. 2000-14, ICASE, Langley Research Center, Hampton, VA, Mar. 2000.

[13] A. COHN, *The notion of proof in hardware verification*, J. Autom. Reasoning, 5 (1989), pp. 127–139.

[14] R. DE NICOLA AND M. C. B. HENNESSY, *Testing equivalences for processes*, Theoretical Computer Science, 34 (1984), pp. 83–133.

[15] R. GERTH, D. PELED, M. Y. VARDI, AND P. WOLPER, *Simple on-the-fly automatic verification of linear temporal logic*, in Proceedings of the IFIP symposium on Protocol Specification, Testing and Verification (PSTV '95), Warsaw, Poland, 1995, pp. 3–18.

[16] T. A. HENZINGER, Z. MANNA, AND A. PNUELI, *Temporal proof methodologies for real-time systems*, Information and Computation, 112 (1994), pp. 273–337.

[17] L. LAMPORT, *Specifying Systems*, Addison-Wesley, 2002.

[18] S. SCHNEIDER, *Concurrent and Real-time Systems: The CSP Approach*, John Wiley & Sons, 2000.

[19] J. SPRINGINTVELD, F. VAANDRAGER, AND P. R. D'ARGENIO, *Testing timed automata*, Theoretical Computer Science, 254 (2001), pp. 225–257.

[20] J. TRETMANS, *A formal approach to conformance testing*, in Protocol Test System, VI, Elsevier, 1994, pp. 257–276.

[21] J. TRETMANS, *Conformance testing with labelled transition systems: implementation relations and test generation*, Computer Networks and ISDN Systems, 29 (1996), pp. 49–79.

[22] M. VARDI AND P. WOLPER, *An automata-theoretic approach to automatic program verification*, in Proceedings of the First Annual Symposium on Logic in Computer Science (LICS '86), 1986, pp. 332–344.

[23] M. Y. VARDI AND P. WOLPER, *Reasoning about Infinite Computations*, Academic Press, 1994.