A Testing Theory for Real-Time Systems

Stefan D. Bruda and Chun Dai

***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 (namely, TPTL), so that a process must pass the generated test if and only if the process satisfies the given temporal logic formula. Beside the obvious use of such an algorithm (to generate tests), our result also establishes a correspondence between timed must testing and timed temporal logic.**

***Index Terms*—Formal methods, Real-time systems, Model-based testing, May testing, Must testing, Testing preorders, Test generation, Timed temporal logic, TPTL**

I. Introduction

How to guarantee validity and reliability of software and hardware is one of the most pressing problems. *Conformance testing* [1] is known in this area for its succinctness and high automatization. Its aim is to check whether an implementation conforms to a given specification.

In the context of conformance testing system specifications can be mainly classified into two kinds: algebraic and logic. The first favours refinement, where a single algebraic formalism is equipped with a refinement relation to represent and relate both specifications and implementations [2]. An implementation is validated if it refines its specification. Process algebrae, labelled transition systems, and finite automata are commonly used in this classification, with traditional refinement relations being either behavioural equivalences or preorders [3], [4]. A typical example is model-based testing [3]. The second approach prefers assertive constructs. Different formalisms describe the properties of specifications and implementations; specifications are defined logically 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 [5].

The domain of conformance testing consists in reactive systems, which interact with their environment. 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. Real-time specifications are then used as the basis of conformance testing for such systems.

Stefan D. Bruda and Chun Dai are with the Department of Computer Science, Bishop's University, Sherbrooke, QC J1M 1Z7, Canada; e-mail: stefan@bruda.ca, cdai@cs.ubishops.ca.

The aim of this paper is to develop a semantic theory for real-time system specification based on timed transition systems modeling the behaviour of real-time processes. We first develop a semantic theory for real-time system specification. Using a theory of timed ω-final states (inspired from the theory of timed automata [6]) as well as a timed testing framework based on De Nicola and Hennessy's testing [4], we develop timed may and must preorders that relate timed processes on the basis of their responses to timed tests. 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 abound, they mostly restrict the real-time domain to make it tractable. 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, so our theory is general. Still, our general approach is more practical than one might expect. Indeed, the characterization of the timed preorders uses a surprisingly concise set of timed tests.

To further address the practicality issue we then tackle now automatic test generation. We show how to algorithmically build tests starting from timed propositional temporal logic (or TPTL) [7] formulae. This algorithm is also a first yet significant step toward an integration of operational and assertive specification styles in the area of real-time systems, to obtain heterogeneous (algebraic and logic) 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 III, where we also introduce our timed preorders. Section IV characterizes the timed preorders and Section V presents our conversion of TPTL formulae into equivalent timed tests. We conclude in Section VI.

II. 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.

Our constructions are based on some 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 clocks C is a set of variables over L. We use A, L, and C in sans-serif face exclusively for this purpose, so that their purpose is often consider understood throughout the paper.

A clock interpretation for a set C of clocks is a mapping $\mathsf{C} \to \mathsf{L}$. 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 also be reset to zero.

If x is a clock and r is a real then $x \sim r$ is a time constraint, $\sim \in \{\leq, <, =, \neq, >, \geq\}$. Constraints can be joined in conjunctions or disjunctions, together with the special constraint $\top$ (which is true in any interpretation). $\mathbb{T}(\mathsf{C})$ denotes the set of all time constraints over a set C of clocks.

A. Timed Transition Systems

Labelled transition systems [3] 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 [6], [8] 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 in this section; 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.

For a set A of observable actions ($\tau \notin \mathsf{A}$), a set L of times values, and a set C of clocks, 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 (we use $p \xrightarrow[\mathbb{t}, C]{(a,\delta)} p'$ instead of $(p, (a, \delta), p', \mathbb{t}, C) \in \rightarrow$, omitting C whenever $C = \emptyset$ and also $\mathbb{t}$ whenever $\mathbb{t} = \top$); p_0 is the initial state.

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 do not affect clock interpretations and cannot be time constrained. 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). A timed trace over A, L, and C is a member of $(\mathsf{A} \times \mathsf{L} \times \mathbb{T}(\mathsf{C}) \times 2^{\mathsf{C}})^* \cup (\mathsf{A} \times \mathsf{L} \times \mathbb{T}(\mathsf{C}) \times 2^{\mathsf{C}})^\omega$.

If both L and $\mathbb{T}(\mathsf{C})$ (or equivalently C) are empty the timed transition system becomes an LTS, and its timed traces are normal traces. One of L or $\mathbb{T}(\mathsf{C})$ could be empty and we still obtain a timed trace; we will use this to differentiate between processes and specifications.

As usual we write $p \xRightarrow[\mathbb{t}, C]{(a,\delta)} p'$ if and only if $p \xrightarrow{\tau} \cdots \xrightarrow{\tau} p_n \xrightarrow[\mathbb{t}, C]{(a,\delta)} p'$ and $p \overset{\varepsilon}{\Rightarrow} p'$ if and only if $p = p_0 \xrightarrow{\tau} \cdots \xrightarrow{\tau} p_n = p'$, for some $n \geq 0$. Furthermore $p \xRightarrow{w} q$ whenever $w = (a_i, \delta_i, \mathbb{t}_i, C_i)_{0<i\leq k}$ and $p \xRightarrow[\mathbb{t}_1, C_1]{(a_1,\delta_1)} p_1 \xRightarrow[\mathbb{t}_2, C_2]{(a_2,\delta_2)} p_2 \cdots \xRightarrow[\mathbb{t}_k, C_k]{(a_k,\delta_k)} p_k = p'$. That a process p cannot evolve any further (via either internal or external actions) is denoted by $p \not\rightarrow$.

A timed path π of a timed transition system $M = ((\mathsf{A} \times \mathsf{L}) \cup \{\tau\}, \mathsf{C}, S, \rightarrow, p_0)$ is a potentially infinite sequence $(p_{i-1}, (a_i, \delta_i, \mathbb{t}_i, C_i), p_i)_{0<i<k}$, where $p_{i-1} \xRightarrow[\mathbb{t}_i, C_i]{(a_i,\delta_i)} p_i$ for all $0 < i \leq k$; the length of π is k and is denoted by $|\pi|$. If $|\pi| = \omega$, π is *infinite*; otherwise (that is, if $|\pi| \in \mathbb{N}$) π is *finite*. If $|\pi| \in \mathbb{N}$ and $p_{|\pi|} \not\rightarrow$ (that is, $p_{|\pi|}$ is a deadlock state), then the timed path π is called maximal. trace(π), the (timed) trace of π is defined as the sequence $(a_i, \delta_i, \mathbb{t}_i, C_i)_{0\leq i\leq|\pi|, a_i\neq\tau} \in (\mathsf{A} \times \mathsf{L} \times \mathbb{T}(\mathsf{C}) \times 2^{\mathsf{C}})^*$. $\Pi_f(p')$, $\Pi_m(p')$, $\Pi_I(p')$ 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.

State p' of transition system p is *timed divergent*, denoted by $p' \Uparrow_p$ (or just $p' \Uparrow$ when there is no ambiguity), if $\exists \pi \in \Pi_I(p')$: 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 $\exists l \in \mathbb{N}, p'' \in S : l \leq k, p' \xRightarrow{w'} p'', p'' \Uparrow_p$, with $w' = (a_i, \delta_i, \mathbb{t}_i, C_i)_{0<i<l}$. State p' is timed convergent or timed w-convergent ($p' \Downarrow_p$ and $p \Downarrow_p w$, respectively, again omitting the subscript p 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 p' is $\text{init}_p(p') = \{(a, \delta, \mathbb{t}, C) \in \mathsf{A} \times \mathsf{L} \times \mathbb{T}(\mathsf{C}) \times 2^{\mathsf{C}} : \exists p'' : p' \xRightarrow[\mathbb{t}, C]{(a,\delta)} p''\}$.

Definition 1: TIMED TRACE LANGUAGES. For a timed transition system (state) p the timed finite-trace language $L_f(p)$, maximal-trace (complete-trace) language $L_m(p)$, infinite-trace language $L_I(p)$, and divergence language $L_D(p)$ of p are

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

We defined timed languages slightly differently from the original [6] to reflect their use for system specification and also to simplify the presentation. However if we omit the clocks and their constraints (which we will do for processes) there is a natural bijection between our definition and the original.

Once more similar to the theory of timed automata [6] we introduce a set of timed ω-final states. Then:

Definition 2: 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 $\pi \in \Pi_\omega(p)$.

We exclude henceforth Zeno behaviours from all the languages that we consider; that is, 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 [6], [9]).

B. Timed Propositional Temporal Logic

Timed Propositional Temporal Logic (TPTL) [7] is one of the most general temporal logics with time constraints [10]. TPTL extends linear-time temporal logic (LTL) [5], [10] by adding time constraints, so that its semantics is given with respect to timed traces[1], that is, timed words in $(\mathsf{A} \times \mathsf{L})^* \cup (\mathsf{A} \times \mathsf{L})^\omega$. We use TPTL without congruence, but we just call it TPTL for short.

For presentation convenience we use a slightly modified form of TPTL without congruence. However, it is immediate that our form is equivalent to the original so we 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.

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

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

Let $\mathcal{F}$ be the set of all TPTL formulae. A timed trace $w = (a_i, \delta_i)_{0<i\leq k} \in (\mathsf{A} \times \mathsf{L})^* \cup (\mathsf{A} \times \mathsf{L})^\omega$ satisfies ϕ if and only if $w \vDash_\gamma \phi$. The relation $\vDash_\gamma \subseteq ((\mathsf{A} \times \mathsf{L})^* \cup (\mathsf{A} \times \mathsf{L})^\omega) \times \mathcal{F}$ 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 $\gamma : \mathsf{C} \to \mathsf{L}$ being some clock interpretation.

- $\theta_1 \leq \theta_2$ if and only if $\gamma(\theta_1) \leq \gamma(\theta_2)$,
- $w \vDash_\gamma \top$ and $w \nvDash_\gamma \bot$ for any w,
- $w \vDash_\gamma a$ if and only if $w \neq \varepsilon$ and $a_1 = a$,
- $w \vDash_\gamma \neg\phi$ if and only if $\neg(w \vDash_\gamma \phi)$,
- $w \vDash_\gamma \phi_1 \wedge \phi_2$ if and only if $w \vDash_\gamma \phi_1$ and $w \vDash_\gamma \phi_2$,
- $w \vDash_\gamma \mathsf{X}\ \phi$ if and only if $w_2 \vDash_{\gamma+\delta_1} \phi$,
- $w \vDash_\gamma \phi_1\ \mathsf{U}\ \phi_2$ if $\exists 0 < i \leq k$: $(\forall i \leq r \leq k$: $w_r \vDash_{\gamma+\sum_{j-1}^{r}\delta_j} \phi_2$, $\forall 0 < s < i$: $w_s \vDash_{\gamma+\sum_{j-1}^{s}\delta_j} \phi_1)$,
- $w \vDash_\gamma x.\phi$ if and only if $w \vDash_{\gamma[0/x]} \phi$.

We denote by $\gamma + c$ a clock interpretation in which $(\gamma + c)(x) = \gamma(x) + c$ for all clocks x. We require 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 $t \in \mathsf{L}$. 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. These restrictions are sufficient to model most phenomenae from other timed temporal logics [7].

As usual one can also introduce the derived operators G ("globally") and F ("eventually") as $\mathsf{G}\ \phi = \bot\ \mathsf{R}\ \phi$ and $\mathsf{F}\ \phi = \top\ \mathsf{U}\ \phi$, respectively. The operator R ("releases") is the dual of the operator U. A timed process[2] p satisfies the TPTL formula ϕ, written $p \vDash_\gamma \phi$, if and only if $\forall w \in L_f(p) \cup L_m(p) \cup L_\omega(p) \cup L_D(p) : w \vDash_\gamma \phi$.

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

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

III. A Testing Theory for Real Time

We are now ready to extend the testing theory of De Nicola and Hennessy [4] 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 [8], [11], [12] 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 [13], though this theory is not real time and to our knowledge has not been pursued any further.

The traditional testing framework defines behavioural preorders that relate labelled transition systems according to their responses to tests [4], [14]. The tests are thus 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.

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[3] (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 3: Timed processes and tests. A timed process $((\mathsf{A} \times \mathsf{L}) \cup \{\tau\}, S, \to, p_0)$ is a timed transition system $((\mathsf{A} \times \mathsf{L}) \cup \{\tau\}, \emptyset, S, \to, p_0)$ with an empty set of clocks (and thus with no time constraints). It follows that all the traces of any timed process are in the set $(\mathsf{A} \times \mathsf{L})^* \cup (\mathsf{A} \times \mathsf{L})^\omega$.

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

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[4]. 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 $\mathbb{T}(\mathsf{C}) = \emptyset$ which means there is no time constraint in the test, we call the test classical. The set of all timed tests is denoted by $\mathcal{T}$.

Definition 4: Partial computation. A partial computation c of a timed process p and a timed test t is a potentially

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

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

infinite sequence $(\langle p_{i-1}, t_{i-1}\rangle \overset{(a_i,\delta_i)}{\underset{\mathbb{t}_i, C_i}{\longmapsto}}_R \langle p_i, t_i\rangle)_{0<i\leq k}$, where $k \in \mathbb{N} \cup \{\omega\}$, such that $p_i \in P$ and $t_i \in T$ for all $0 < i \leq k$; and $\delta_i \in \mathsf{L}$ is taken from p, $\mathbb{t}_i$ 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:

- $\langle p_{i-1}, t_{i-1}\rangle \xmapsto{\tau}_1 \langle p_i, t_i\rangle$ if $a_i = \tau$, $p_{i-1} \xrightarrow{\tau}_p p_i$, $t_{i-1} = t_i$, and $t_{i-1} \notin \Sigma$,
- $\langle p_{i-1}, t_{i-1}\rangle \xmapsto{\tau}_2 \langle p_i, t_i\rangle$ if $a_i = \tau$, $p_{i-1} = p_i$, $t_{i-1} \xrightarrow{\tau}_t t_i$, and $t_{i-1} \notin \Sigma$,
- $\langle p_{i-1}, t_{i-1}\rangle \overset{(a_i,\delta_i)}{\underset{\mathbb{t}_i, C_i}{\longmapsto}}_3 \langle p_i, t_i\rangle$ if $(a_i,\delta_i) \in \mathsf{A} \times \mathsf{L}$, $p_{i-1} \xrightarrow{(a_i,\delta_i)}_p p_i$, $t_{i-1} \overset{(a_i,\delta_i)}{\underset{\mathbb{t}_i, C_i}{\longrightarrow}}_t t_i$, and $t_{i-1} \notin \Sigma$.

The first two expressions in the definition of $\mapsto$ indicate that when the test or the process under test is executing an internal action, the other process 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. 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 $(t_{i-1}, (a_i, \delta_i, \mathbb{t}_i, C_i), t_i)_{i \in I_t^c} \in \Pi(t)$, where $I_t^c = \{0 < i \leq k : R_i \in \{2,3\}\}$.

Definition 5: COMPUTATION. A partial computation c is a computation whenever: If $k \in \mathbb{N}$ then c is maximal, that is, $p_k \overset{\tau}{\not\longrightarrow}_p$, $t_k \overset{\tau}{\not\longrightarrow}_t$, and $\text{init}_p(p_k) \cap \text{init}_t(t_k) = \emptyset$ or the time delay of p_k does not satisfy the time constraint of t_k; If $k = \omega$ then $\text{proj}_p(c) \in \Pi_I(p)$. $C(p,t)$ is the set of all computations of p and 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 6: TIMED MAY AND MUST PREORDERS. *p may pass t* (written p $\text{may}_\mathbb{T}$ t), if and only if there exists at least one successful computation $c \in C(p,t)$; *p must pass t* (written p $\text{must}_\mathbb{T}$ t) if and only if every computation $c \in C(p,t)$ is successful.

$p \sqsubseteq_\mathbb{T}^{may} q$ if and only if $\forall t \in \mathcal{T} : p \text{ may}_\mathbb{T}\ t \implies q \text{ may}_\mathbb{T}\ t$; and $p \sqsubseteq_\mathbb{T}^{must} q$ if and only if $\forall t \in \mathcal{T} : p \text{ must}_\mathbb{T}\ t \implies q \text{ must}_\mathbb{T}\ t$.

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. 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 $\mathcal{T}$ to classical tests).

IV. ALTERNATIVE CHARACTERIZATIONS OF TIMED PREORDERS

We now present alternative characterizations for 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 [14], [15] and relates timed testing directly with the behaviour of processes.

Theorem 1:

1) $p \sqsubseteq_\mathbb{T}^{may} q$ if and only if $L_f(p) \subseteq L_f(q)$ and $L_\omega(p) \subseteq L_\omega(q)$.
2) $p \sqsubseteq_\mathbb{T}^{must} q$ if and only if 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 [2], [15]. 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, $p' \xrightarrow{\tau}_p$ implies $p' \overset{(a,\delta)}{\not\longrightarrow}_p$, and $|\{((a,\delta),p'') : p' \xrightarrow{(a,\delta)}_p p''\}| = 1$. 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 $p' \xrightarrow{(a,\delta)}_p p''$ into two transitions $p' \xrightarrow{\tau}_p p_{\langle p',(a,\delta),p''\rangle}$ and $p_{\langle p',(a,\delta),p''\rangle} \xrightarrow{(a,\delta)}_p p''$, 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 $p \sqsubseteq_\mathbb{T}^{must} q$ if and only if all of the following hold:

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

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 characterization theorems 1 and 2 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 $t_w^{May,*} = (\mathsf{A} \cup \{\tau\}, \mathsf{C}, T, \rightarrow, \emptyset, 0, k)$, where $T = \{0, 1, \ldots, k\}$ and $\rightarrow = \{(i-1, a_i, i, c_i = \sum_{j=0}^{i} \delta_i) : 0 < i \leq k\}$.
- For $w = (a_i, \delta_i)_{0<i\leq k} \in (\mathsf{A} \times \mathsf{L},)^\omega$, let $t_w^{May,\omega} = (\mathsf{A} \cup \{\tau\}, \mathsf{C}, T, \rightarrow, T, 0, \emptyset)$, where $T = \mathbb{N}$, $\rightarrow = \{(i-1, a_i, i, c_i = \sum_{j=0}^{i} \delta_i) : i > 0\}$.
- For $w = (a_i, \delta_i)_{0<i\leq k} \in (\mathsf{A} \times \mathsf{L})^*$, let $t_w^{May,div} = (\mathsf{A} \cup \{\tau\}, \mathsf{C}, T, \rightarrow, \{k\}, 0, \emptyset)$, where $T = \{0, 1, \ldots, k\}$, $\rightarrow = \{(i-1, a_i, i, c_i = \sum_{j=0}^{i} \delta_i) : 0 < i \leq k\} \cup \{(k, \tau, k, \top)\}$.
- For $w = (a_i, \delta_i)_{0<i\leq k} \in (\mathsf{A} \times \mathsf{L})^*$, let $t_w^{\Downarrow,*} = (\mathsf{A} \cup \{\tau\}, \mathsf{C}, T, \rightarrow, \emptyset, 0, \{s\})$, where $T = \{0, 1, \ldots, k\} \cup \{s\}$,

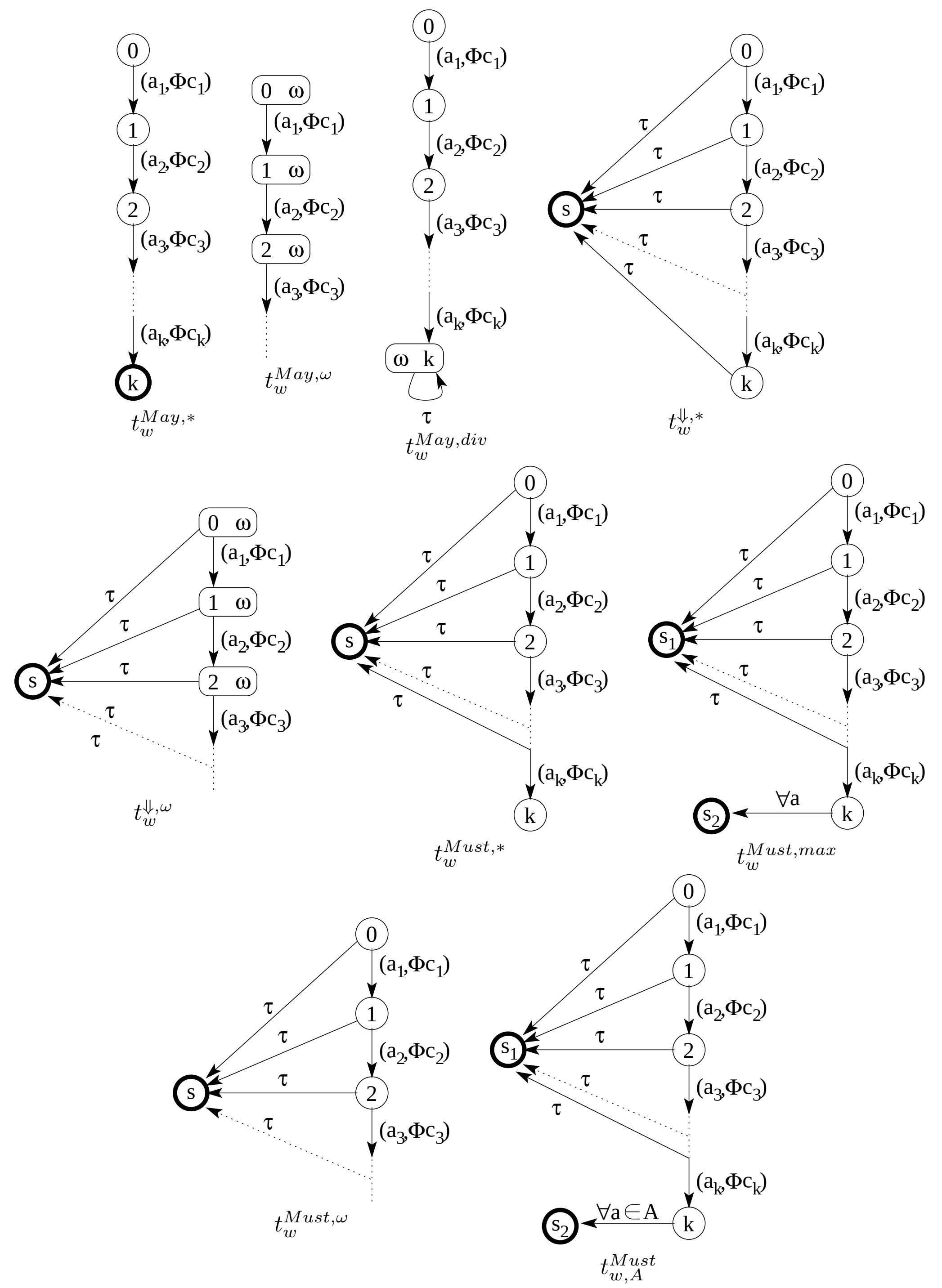

Fig. 1. Timed tests used for the characterization of timed may and must preorders.

$\to = \{(i-1, a_i, i, c_i = \sum_{j=0}^{i} \delta_i) : 0 < i \le k\} \cup \{(i, \tau, s, \top) : 0 < i \le k\}$.

- For $w = (a_i, \delta_i)_{0<i\le k} \in (\mathsf{A} \times \mathsf{L})^\omega$, let $t_w^{\Downarrow,\omega} = (\mathsf{A} \cup \{\tau\}, \mathsf{C}, T, \to, \{s\}, 0, \{s\})$, where $T = \mathbb{N} \cup \{s\}$, $\to = \{(i-1, a_i, i, c_i = \sum_{j=0}^{i} \delta_i) : i > 0\} \cup \{(i, \tau, s, \top) : i > 0\}$.
- For $w = (a_i, \delta_i)_{0<i\le k} \in (\mathsf{A} \times \mathsf{L})^*$, let $t_w^{Must,*} = (\mathsf{A} \cup \{\tau\}, \mathsf{C}, T, \to, \emptyset, 0, \{s\})$, where $T = \{0, 1, \ldots, k\} \cup \{s\}$, $\to = \{(i-1, a_i, i, c_i = \sum_{j=0}^{i} \delta_i) : 0 < i \le k\} \cup \{(i, \tau, s, \top) : 0 \le i < k\}$
- For $w = (a_i, \delta_i)_{0<i\le k} \in (\mathsf{A} \times \mathsf{L})^*$, let $t_w^{Must,max} = (\mathsf{A} \cup \{\tau\}, \mathsf{C}, T, \to, \emptyset, 0, \{s_1, s_2\})$, where $T = \{0, 1, \ldots, k\} \cup \{s_1, s_2\}$, $\to = \{(i-1, a_i, i, c_i = \sum_{j=0}^{i} \delta_i) : 0 < i \le k\} \cup \{(i, \tau, s_1, \top) : 0 \le i < k\} \cup \{(k, a, s_2, \top) : (a, \mathbb{t}) \in \mathsf{A} \times \mathbb{T}()$.
- For $w = (a_i, \delta_i)_{0<i\le k} \in (\mathsf{A} \times \mathsf{L})^\omega$, let $t_w^{Must,\omega} = (\mathsf{A} \cup \{\tau\}, \mathsf{C}, T, \to, \emptyset, 0, \{s\})$, where $T = \mathbb{N} \cup \{s\}$, $\to = \{(i-1, a_i, i, c_i = \sum_{j=0}^{i} \delta_i) : i > 0\} \cup \{(i, \tau, s, \top) : i \in \mathbb{N}\}$.
- For $w = (a_i, \delta_i)_{0<i\le k} \in (\mathsf{A} \times \mathsf{L})^*$ and $A \subseteq \mathsf{A}$, let $t_{w,A}^{Must} = (\mathsf{A} \cup \{\tau\}, \mathsf{C}, T, \to, \emptyset, 0, \{s_1, s_2\})$, where $T = \{0, 1, \ldots, k\} \cup \{s_1, s_2\}$, $\to = \{(i-1, a_i, i, c_i =$

$\sum_{j=0}^{i}\delta_i) : 0 < i \le k\} \cup \{(i,\tau,s_1,\top) : 0 \le i < k\} \cup \{(k,a,s_2,\top) : a \in A\}$.

These tests are depicted graphically in Figure 1. In the figure ω-final states are marked by the symbol ω and success states are distinguished from regular states by thick borders.

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-state 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 in must-testing.

Finally, timed test $t_{w,A}^{Must}$ is capable of comparing the initial action sets of states reached when executing trace w with respect to a subset $A \subseteq \mathsf{A}$.

Note that we use the tightest time constraint possible in our test. We denote $c_i = \sum_{j=0}^{i}\delta_i$ by $\mathbb{t}_i$ in what follows.

Our specific timed tests satisfy the following desired properties:

Lemma 3:

1) Let $w \in (\mathsf{A}\times\mathsf{L})^*$. Then, $w \in L_f(p)$ if and only if p may$_{\mathbb{T}}$ $t_w^{May,*}$.
2) Let $w \in (\mathsf{A}\times\mathsf{L})^\omega$. Then $w \in L_\omega(p)$ if and only if p may$_{\mathbb{T}}$ $t_w^{May,\omega}$.
3) Let $w \in (\mathsf{A}\times\mathsf{L})^*$. Then, $w \in L_\omega(p)$ if and only if p may$_{\mathbb{T}}$ $t_w^{May,div}$.
4) Let $w \in (\mathsf{A}\times\mathsf{L})^*$. Then, $p \Downarrow w$ if and only if p must$_{\mathbb{T}}$ $t_w^{\Downarrow,*}$.
5) Let $w \in (\mathsf{A}\times\mathsf{L})^* \cup (\mathsf{A}\times\mathsf{L})^\omega$. Then, $p \Downarrow w$ if and only if p must$_{\mathbb{T}}$ $t_w^{\Downarrow,\omega}$.
6) Let $w \in (\mathsf{A}\times\mathsf{L})^*$ such that $p \Downarrow w$. Then, $w \notin L_f(p)$ if and only if p must$_{\mathbb{T}}$ $t_w^{Must,*}$.
7) Let $w \in (\mathsf{A}\times\mathsf{L})^*$ such that $p \Downarrow w$. Then, $w \notin L_m(p)$ if and only if p must$_{\mathbb{T}}$ $t_w^{Must,max}$.
8) Let $w \in (\mathsf{A}\times\mathsf{L})^\omega$ such that $p \Downarrow w$. Then, $w \notin L_\omega(p)$ if and only if 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 $w = (a_i,\delta_i)_{0<i\le k}$ for some $k \in \mathbb{N}\cup\{\omega\}$.

- Item 1, $\Rightarrow$: $w \in L_f(p)$, and thus $p_0 \overset{(a_1,\delta_1)}{\Longrightarrow} p_1 \overset{(a_2,\delta_2)}{\Longrightarrow} \cdots \overset{(a_k,\delta_k)}{\Longrightarrow} p_k$ (Definition 1). On the other hand, $t_0^{May,*}\underset{\mathbb{t}_1}{\overset{(a_1,\delta_1)}{\Longrightarrow}}t_1^{May,*}\underset{\mathbb{t}_2}{\overset{(a_2,\delta_2)}{\Longrightarrow}}\cdots\underset{\mathbb{t}_k}{\overset{(a_k,\delta_k)}{\Longrightarrow}}t_k^{May,*}$ (definition of $t_w^{May,*}$ including the form of $\mathbb{t}_i$). Therefore, $(\langle p_{i-1},t_{i-1}^{May,*}\rangle\underset{\mathbb{t}_i}{\overset{(a_i,\delta_i)}{\longmapsto}}_R\langle p_i,t_i^{May,*}\rangle)_{0<i\le k}$, 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, $t_k^{May,*} \overset{\tau}{\not\longrightarrow}$ and $I_p(p_k)\cap I_t(t_k^{May,*}) = \emptyset$), and is further the trace of a successful computation (since $t_k^{May,*} \in Suc$). 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, $(\langle p_{i-1},t_{i-1}^{May,*}\rangle\underset{\mathbb{t}_i}{\overset{(a_i,\delta_i)}{\longmapsto}}_R\langle p_i,t_i^{May,*}\rangle)_{0<i\le k}$, $t_k^{May,*} \overset{\tau}{\not\longrightarrow}$, $I_p(p_k)\cap I_t(t_k^{May,*}) = \emptyset$, and $t_w^{May,*} = t_k^{May,*} \in Suc$. By a reverse argument we conclude then that $w \in L_f(p)$ ($t_0^{May,*}\underset{\mathbb{t}_1}{\overset{(a_1,\delta_1)}{\Longrightarrow}}t_1^{May,*}\underset{\mathbb{t}_2}{\overset{(a_2,\delta_2)}{\Longrightarrow}}\cdots\underset{\mathbb{t}_k}{\overset{(a_k,\delta_k)}{\Longrightarrow}}t_k^{May,*}$, then $p_0 \overset{(a_1,\delta_1)}{\Longrightarrow} p_1 \overset{(a_2,\delta_2)}{\Longrightarrow} \cdots \overset{(a_k,\delta_k)}{\Longrightarrow} p_k$, 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 $0 < l \le k$ there exists one trace $p_0 \overset{(a_1,\delta_1)}{\Longrightarrow} p_1 \overset{(a_2,\delta_2)}{\Longrightarrow} \cdots \overset{(a_l,\delta_l)}{\Longrightarrow} p_l \overset{\tau}{\longrightarrow} p_l \cdots$ 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 $0 < l \le k$ there exists one trace $p_0 \overset{(a_1,\delta_1)}{\Longrightarrow} p_1 \overset{(a_2,\delta_2)}{\Longrightarrow} \cdots \overset{(a_l,\delta_l)}{\Longrightarrow} p_l \overset{\tau}{\longrightarrow} p_l \cdots$ 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 $p_0 \overset{(a_1,\delta_1)}{\Longrightarrow} p_1 \overset{(a_2,\delta_2)}{\Longrightarrow} \cdots \overset{(a_i,\delta_i)}{\Longrightarrow} p_i \overset{\tau}{\longrightarrow} p_i \cdots$, or $p_0 \overset{(a_1,\delta_1)}{\Longrightarrow} p_1 \overset{(a_2,\delta_2)}{\Longrightarrow} \cdots \overset{(a_k,\delta_k)}{\Longrightarrow} p_k$. 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.

■

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 Suc)—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 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 process such that p may$_{\mathbb{T}}$ t, that is, there exists a successful computation $c \in C(p,t)$ with $w = \text{trace}(\text{proj}_p(c)) =$

trace(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 $w =$ trace(proj$_q(c')$) = trace(proj$_t(c')$) and proj$_t(c') =$ proj$_t(c)$. It follows that q may$_\mathbb{T}$ t and therefore $p \sqsubseteq^{may}_\mathbb{T} q$. If $|w| < \omega$, we can split the proof into two cases: either $w \in L_f(p)$ or $w \in L_\omega(p)$. We can then establish that q may$_\mathbb{T}$ t as above.

On to Item 2 now. For the $\Rightarrow$ direction we have that $p \sqsubseteq^{must}_\mathbb{T} q$, $w \in (\mathsf{A}\times\mathsf{L})^* \cup (\mathsf{A}\times\mathsf{L})^\omega$ such that $p \Downarrow w$. Then p must$_\mathbb{T}$ $t^{\Downarrow,*}_w$ or p must$_\mathbb{T}$ $t^{\Downarrow,\omega}_w$ (Lemma 3), then q must$_\mathbb{T}$ $t^{\Downarrow,*}_w$ or q must$_\mathbb{T}$ $t^{\Downarrow,\omega}_w$ (Definition 6), 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 $I_p(p') \subseteq I_q(q')$. Suppose that $p \overset{w}{\not\Longrightarrow}$, that is, $w \notin L_f(p)$. Then p must$_\mathbb{T}$ $t^{Must,*}_w$ (Lemma 3), so q must$_\mathbb{T}$ $t^{Must,*}_w$ (Definition 6). However, q must$_\mathbb{T}$ $t^{Must,*}_w$ does not hold (contrapositive of Lemma 3), a contradiction. Suppose now that $p \overset{w}{\Longrightarrow}$. Let then $X = \{(a,\delta) \in I_p(p') : p \overset{w}{\Longrightarrow} p'\} \neq \emptyset$. Since $I_p(p') \not\subseteq I_q(q')$ (assumption), for every $A \in X$ there exists an $(a,\delta) \in A \setminus I_q(q')$. Let B be the set of all such actions a (ignoring the time actions). It is then immediate then that p must$_\mathbb{T}$ $t^{Must}_{w,B}$ (by the construction of $t^{Must}_{w,B}$); however, it is not the case that q must$_\mathbb{T}$ $t^{Must}_{w,B}$ (since $q' \overset{(a,\delta)}{\not\Longrightarrow}$ for any $(a,\delta) \in (B,\mathsf{L})$). This contradicts the assumption that $p \sqsubseteq^{must}_\mathbb{T} q$.

If on the other hand $|w| = \omega$, assume that $w \notin L_\omega(p)$. Then p must$_\mathbb{T}$ $t^{Must,\omega}_w$ (Definition 6) 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 process such that q must$_\mathbb{T}$ t does not hold, that is, there exists an unsuccessful computation $c = (\langle q_{i-1}, t_{i-1}\rangle, (a_i, \delta_i, \mathbb{t}_i), \langle q_i, t_i\rangle)_{0<i\le k} \in C(q,t)$ (Definition 6). Let $w =$ trace(proj$_p(c)$) = trace(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 proj$_p(c') \in \Pi_\omega(p)$ and proj$_t(c') \notin \Pi_\omega(t)$ (because $|proj_p(c')| < \omega$).This implies that p must$_\mathbb{T}$ t does not hold (Definition 6) and thus $p \sqsubseteq^{must}_\mathbb{T} q$ (since q must$_\mathbb{T}$ t does not hold, by the contrapositive of Definition 6).

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 $|c| < \omega$, we have:

(a) $w \in L_f(q)$, $q \overset{w}{\Longrightarrow} q'$ for some q' and $t_k \neq Suc$ by definition of $t^{Must,*}_w$,

(b) $q_k \overset{\tau}{\not\longrightarrow}$, $t_k \overset{\tau}{\not\longrightarrow}$, $I^c_q(q_k) \cap I^c_t(t_k) = \emptyset$ by definition of $t^{Must,max}_w$; and

(c) $\exists p' : p \overset{w}{\Longrightarrow} p', I^c_p(p') \subseteq I^c_q(q')$ 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 proj$_t(c') =$ proj$_t(c)$ and $\langle p_l, t_l\rangle = \langle p'', t_k\rangle$, where $p' \overset{\varepsilon}{\Longrightarrow} p''$ for some $p'' \overset{\tau}{\not\longrightarrow}_p$. Note that such a p'' must exist since $p \Downarrow w$. Then $I^c_p(p'') \subseteq I^c_p(p')$, definition of c' and p'', and observations (a) and (b) above imply that $I^c_p(p'') \cap I^c_t(t_k) \subseteq I^c_q(q') \cap I^c_t(t'_l) \subseteq I^c_q(q_k) \cap I^c_t(t_l)$; thus c' cannot be extended. Since $t'_l = t_k \notin Suc$, c' is unsuccessful, so p must$_\mathbb{T}$ t does not hold.

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

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

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

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

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

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

On to the $\Leftarrow$ direction now. We assume that Relations (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 $p \sqsubseteq^{must}_\mathbb{T} q$ 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 $w =$ trace(proj$_q(c)$) $= trace($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 $t_k \notin Suc$ we distinguish two cases:

1) Let $w \in L_f(q) \setminus L_m(q)$. Then there exists some $(a,\delta) \in \mathsf{A}\times\mathsf{L}$ such that $q_k \overset{(a,\delta)}{\longrightarrow}_q$ but $t_k \overset{(a,\delta)}{\not\longrightarrow}_t$. That is, $w\cdot(a,\delta) \in L_m(q)$ and so (by Relation (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 proj$_t(c) =$ proj$_t(c')$, $t'_l = t_k$, and $p_l \overset{(a,\delta)}{\longrightarrow}_p$. The computation is maximal (since $t_k = t'_j \overset{(a,\delta)}{\not\longrightarrow}_{t'}$) and unsuccessful (since $|c'| < \omega$ and $t'_l \notin Suc$). Therefore, p must$_\mathbb{T}$ t does not hold.
2) 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 proj$_t(c) \notin \Pi_\omega(t)$, 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 proj$_t(c) =$ 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 $p \sqsubseteq^{must}_\mathbb{T} q$, as desired. ■

V. 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 [6] approaches to LTL model checking [4], [5], [16]–[18].

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 $\text{must}_{\mathbb{T}}$ T_ϕ for any timed process p. T_ϕ can be algorithmically constructed starting from ϕ.

Proof: Let p be an arbitrary timed process. T_ϕ is first constructed to consider only infinite computations, and will then be modified to consider maximal traces. A sub-formula of ϕ is defined inductively:

1) ϕ is a sub-formula of ϕ,
2) any formula t formed by 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 (a time formula is indivisibly a sub-formula),
3) if $\neg\xi$ is a sub-formula of ϕ, then so is ξ,
4) if $O\ \xi$ is a sub-formula 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, $O \in \{\wedge, \vee, \mathsf{U}\}$.

Let C_ϕ be the set of exactly all the clocks that occur in a TPTL formula ϕ, and let $\mathcal{C}(\phi)$ be the closure of ϕ, that is, the set of exactly all the sub-formulae of ϕ. Furthermore, let $\Theta(\phi) \subseteq \mathcal{C}(\phi)$ contain exactly all the sub-formulae of ϕ that are time formulae.

The construction of T_ϕ is then based on the untimed construction developed by Vardi and Wolper [17]. We first consider the "local" automaton $L_\phi = (2^{\mathcal{C}(\phi)}, \mathsf{C}_\phi, N_L, \rightarrow_L, \emptyset, N_L, s_0)$. The set of states contain all the subsets of $\mathcal{C}(\phi)$ that have no internal inconsistency, plus one designated initial state. The local automaton does not impose any acceptance condition. A state s has no internal inconsistency if and only if:

1) $\psi \in s$ if and only if $\neg\psi \notin s$ for all $\psi \in \mathcal{C}(\phi)$,
2) $\xi \wedge \psi \in s$ if and only if $\xi \in s$, $\psi \in s$ for all $\xi \wedge \psi \in \mathcal{C}(\phi)$,
3) $x.\psi \in s$ implies $\psi \in s$.

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

1) $s = s_0$ and $\phi \in a$, or
2) $s \neq s_0$, for all $\psi \in \mathcal{C}(\phi)$, $\mathsf{X}\ \psi \in s$ if and only if $\psi \in t$, and for all $\xi\ \mathsf{U}\ \phi \in \mathcal{C}(\phi)$ either $\psi \in s$, or $\xi \in s$ and $\xi\ \mathsf{U}\ \psi \in t$.

L_ϕ does not impose any acceptance conditions as mentioned, but enforces all the time constraints present in the original formula ϕ. Indeed, at every moment frozen in time by a construction $x.\psi$ we reset the respective clock in the local automaton (for 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. Checking the time formula to determine that the transition is enabled has the intended effect: 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 time.

Acceptance is handled by the "eventuality" automaton $E_\phi = (2^{\mathcal{C}(\phi)}, \emptyset, 2^{\mathcal{E}(\phi)}, \rightarrow_E, \emptyset, \{\emptyset\}, \emptyset)$, with $\mathcal{E}(\phi) = \{\xi\ \mathsf{U}\ \psi \in \mathcal{C}(\phi)\}$ and $s \xrightarrow[\top,\emptyset]{a}_E t$ if and only if

1) $s = \emptyset$ and $\xi\ \mathsf{U}\ \psi \in t$ if and only if $\psi \notin a$ for all $\xi\ \mathsf{U}\ \psi \in a$,
2) $s \neq \emptyset$ and $\xi\ \mathsf{U}\ \psi \in t$ if and only if $\psi \notin a$ for all $\xi\ \mathsf{U}\ \psi \in s$.

The eventuality automaton is identical to the one developed elsewhere [17]. It tries to satisfy the eventualities of the formula (with no regard for time constraints). 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 [17], for 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 $2^{\mathcal{C}(\phi)} \times \mathsf{L}$; in order to switch to $\mathsf{A} \times \mathsf{L}$ we project over A the action labels of all the transitions, as done previously [17].

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 $\text{trace}(\text{proj}_p(c)) \vDash \phi$ for exactly all the successful infinite computations $c \in C(p, T_\phi)$ stripped of time information. The timing information is added (via L_ϕ), as detailed above. In all,

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

We now enhance T_ϕ so that it also accepts finite maximal traces: 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 fixed trace) 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 new state. We use Δ to distinguish from other states also having no outgoing transitions; these states represent deadlock due to inconsistent sub-formulae of ϕ. The final states of T_ϕ will then be the set containing only the state Δ thus introduced. We have:

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

Indeed, $\chi(p) \vDash_\gamma \phi$ (with $\chi(x) = \text{trace}(\text{proj}_x(c))$) implies $s_0 \overset{\chi(T_\phi)}{\Longrightarrow} s \xrightarrow{\tau} \Delta$ (where s_0 is the initial state of T_ϕ). That is $s_0 \overset{\chi(T_\phi)}{\Longrightarrow} s \not\rightarrow$. Thus, c is maximal. Conversely, if c is successful and maximal, then there exits $s_0 \overset{\chi(T_\phi)}{\Longrightarrow} s \not\rightarrow$ in T_ϕ. According to the algorithm there exits then $s_0 \overset{\chi(T_\phi)}{\Longrightarrow} s \xrightarrow{\tau} \Delta$. Thus, c is maximal.

That $p \vDash_\gamma \phi$ if and only if p $\text{must}_{\mathbb{T}}$ T_ϕ follows from Properties (5) and (6), as desired. ■

VI. Conclusions

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. 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 [11], [12], 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. 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 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 [7], but we are still significantly above the expressiveness of most real-time temporal logics [7], [10].

We avoid the discussion of discrete versus continuous time. All the results and definitions 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 established an algorithm for constructing timed tests from TPTL formulae, but 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.

Acknowledgment

This research was supported by the Natural Sciences and Engineering Research Council of Canada. Part of this research was additionally supported by Bishop's University.

References

[1] J. Tretmans, "A formal approach to conformance testing," in *Protocol Test System, VI.* Elsevier, 1994, pp. 257–276.
[2] ——, "Conformance testing with labelled transition systems: Implementation relations and test generation," *Computer Networks and ISDN Systems*, vol. 29, pp. 49–79, 1996.
[3] M. Broy, B. Jonsson, J.-P. Katoen, M. Leucker, and A. Pretschner, Eds., *Model-Based Testing of Reactive Systems: Advanced Lectures*, ser. Lecture Notes in Computer Science. Springer, 2005, vol. 3472.
[4] R. De Nicola and M. C. B. Hennessy, "Testing equivalences for processes," *Theoretical Computer Science*, vol. 34, pp. 83–133, 1984.
[5] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model Checking.* MIT Press, 1999.
[6] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, pp. 183–235, 1994.
[7] R. Alur and T. A. Henzinger, "Real-time logics: Complexity and expressiveness," *Information and Computation*, vol. 104, pp. 35–77, 1993.
[8] T. A. Henzinger, Z. Manna, and A. Pnueli, "Temporal proof methodologies for real-time systems," *Information and Computation*, vol. 112, pp. 273–337, 1994.
[9] S. D. Bruda and S. G. Akl, "Real-time computation: A formal definition and its applications," *International Journal of Computers and Applications*, vol. 25, no. 4, pp. 247–257, 2003.
[10] P. Bellini, R. Mattolini, and P. Nesi, "Temporal logics for real-time system specification," *ACM Computing Surveys*, vol. 32, pp. 12–42, 2000.
[11] 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.
[12] J. Springintveld, F. Vaandrager, and P. R. D'Argenio, "Testing timed automata," *Theoretical Computer Science*, vol. 254, pp. 225–257, 2001.
[13] R. Cleaveland and G. Lüttgen, "Model checking is refinment—Relating Büchi testing and linear-time temporal logic," ICASE, Langley Research Center, Hampton, VA, Tech. Rep. 2000-14, Mar. 2000.
[14] S. Abramsky, "Observation equivalence as a testing equivalence," *Theoretical Computer Science*, vol. 53, pp. 225–241, 1987.
[15] S. D. Bruda, "Preorder relations," in *Model-Based Testing of Reactive Systems: Advanced Lectures*, ser. Lecture Notes in Computer Science, M. Broy, B. Jonsson, J.-P. Katoen, M. Leucker, and A. Pretschner, Eds., vol. 3472. Springer, 2005, pp. 117–149.
[16] 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.

[17] 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.

[18] M. Y. Vardi and P. Wolper, *Reasoning about Infinite Computations*. Academic Press, 1994.