A Theory of Testing for Real-Time*

Rance Cleaveland
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206
USA

Amy E. Zwarico
Department of Computer Science
Johns Hopkins University
Baltimore, MD 21212
USA

Abstract

This paper develops a framework for generating testing preorders that relate processes on the basis of their timing behavior *as well as their degree of relative nondeterminism. The framework is then applied to two different scenarios. In the first, relations are constructed that relate processes on the basis of* all *timing considerations. In the second, relations are constructed that relate processes on the basis of their relative* speeds. *In both cases, alternative denotational characterizations of the resulting preorders are presented, and examples are given to illustrate the utility of the approach.*

1 Introduction

Over the past decade a number of researchers have investigated the use of *algebraic* techniques for reasoning about *reactive systems* [26]. These systems compute by interacting with their environment; typical examples include communications protocols and process control systems. One result of this line of research has been the development of a number of *behavioral equivalences* and *preorders*, such as observational equivalence [4, 22, 23], the testing/failures relations [6, 12, 17, 15], the divergence preorder [29], and the specification preorder [10], that relate reactive systems on the basis of their externally observable behavior. These relations permit implementations (expressed as networks of reactive systems) to be compared to specifications (also expressed as reactive systems). Moreover, as they usually license the substitution of related processes inside larger systems, they may serve as the basis for *compositional verification* techniques. Behavioral relations have traditionally distinguished reactive systems on the basis of their *extensional behavior* in that they abstract away from any *internal computation* that does not directly affect the external interactions the systems might respond to. Thus, the only internal computation that these relations are sensitive to is that which may result in the *pre-emption* of visible actions.

The goal of this paper is to develop a framework for generating behavioral preorders that are suitable for reasoning about *real-time* characteristics of reactive systems. In a timed setting, internal computation may also cause visible actions to be *delayed*, not just preempted. Our motivation is to develop preorders that are sensitive to this aspect of internal computation. The approach we take derives from the *testing methodology* of DeNicola and Hennessy [11, 12, 15], in that we define a notion of system testing and use it as a basis for developing our behavioral relations. However, in contrast with [11, 12, 15], our testing methodology takes into account the amount of internal computation that processes engage in. As a result, our framework may be used to build preorders relating processes on the basis of their *timing behavior* as well as their nondeterminism.

The remainder of the paper develops as follows. In the next section we review the basic concepts of *transition systems* and testing. Section 3 then introduces *timed testing*, which takes account of the delay exhibited by a process as it attempts to pass a test. The section

*Research supported by the National Science Foundation and DARPA under Grants CCR-9014775 and CCR-8910282.

following investigates the timed testing preorders for two different classes of transition systems and tests, and develops equivalent "denotational" characterizations for them. In Section 5, we present an algebra for the second class of transition systems, while in Section 6 we present examples highlighting the utility and novelty of our approach. The final section contains our conclusions and directions for future research, and a brief discussion of related work.

2 Processes and Testing

We use *labeled transition systems* to model processes. Formally, these may be defined as follows.

Definition 1 *A* labeled transition system *is a quadruple* $\langle Q, A, \rightarrow, q_0\rangle$ *where*

- Q *is a countable set of* states*;*
- A *is a countable set of* actions*, with* $\tau \in A$ *a distinguished* silent*, or* internal*, action;*
- $\rightarrow \subseteq Q \times A \times Q$ *is the* transition relation*; and*
- $q_0 \in Q$ *is the* start state*.*

Intuitively, a labeled transition system represents the operational behavior of a process. Q consists of the set of states that the process may enter, and A contains the set of actions that it may perform. The distinguished state q_0 is the *start state*. The relation $\rightarrow$ describes the actions available in a particular state and the state that may result when the action is executed. In what follows we write $q \xrightarrow{a} q'$ in lieu of $\langle q, a, q'\rangle \in \rightarrow$, and we also use $q \xrightarrow{a}$ to indicate that there is a q' such that $q \xrightarrow{a} q'$. We restrict our attention to *image-finite* transition systems, namely, systems $\langle Q, A, \rightarrow, q_0\rangle$ such that for all $q \in Q$ and $a \in A$, the set $\{\, q' \mid q \xrightarrow{a} q' \,\}$ is finite.

To simplify the presentation we use the following constructions on transition systems.

Definition 2

1. *The transition system* $nil = \langle\{q_0\}, \{\tau\}, \emptyset, q_0\rangle$*.*
2. *Let* $\mathcal{P} = \langle Q, A, \rightarrow, q_0\rangle$ *be a transition system. The* prefixing operation *is defined as follows. Choose* $q' \notin Q$*. Then* $a.\mathcal{P} = \langle Q \cup \{q'\}, A \cup \{a\}, \rightarrow \cup \{q' \xrightarrow{a} q_0\}, q'\rangle$
3. *Let* $\mathcal{P} = \langle Q, A, \rightarrow, q_0\rangle$ *and* $\mathcal{P}' = \langle Q', A', \rightarrow', q_0'\rangle$ *be transition systems. The* nondeterministic composition *of* $\mathcal{P}$ *and* $\mathcal{P}'$ *is defined as follows. Choose* $q' \notin Q \cup Q'$*.*

   $$\begin{array}{l}\mathcal{P} + \mathcal{P}' = \\ \langle Q \cup Q' \cup \{q'\}, A \cup A', \\ \rightarrow \cup \rightarrow' \cup \{\, q' \xrightarrow{a} q'' \mid q_0 \xrightarrow{a} q'' \vee q_0' \xrightarrow{a} q'' \,\}, q'\rangle\end{array}$$

So $a.nil$, $a.nil + b.c.nil$, and $a.(b.nil + c.nil)$ all represent transition systems. In what follows we sometimes abuse notation and leave off trailing *nil*'s; so $b.c$ stands for the process $b.c.nil$. Readers should note the (intended) similarity with Milner's CCS [22, 24].

In general, *testing preorders* and *equivalences* relate processes on the basis of their behavior in response to tests. Particular testing preorders and equivalences may be defined by specifying the *tests* of interest and a means for calculating a process' behavior *in response to* a test. One natural approach is taken by DeNicola and Hennessy [11, 12, 15]: a test may be *any* process in which the set of actions A contains a distinguished action w, the *success* action. Then, a process' behavior in response to a test is given in terms of the *computations* that result when the test is "applied" to the process. These intuitions may be formalized as follows.

Definition 3 *Let* $\mathcal{P} = \langle Q, A, \rightarrow, q_0\rangle$ *be a process and* $\mathcal{T} = \langle T, A' \cup \{w\}, \rightarrow, t_0\rangle$ *a test.*

1. *A* configuration *is a pair* $\langle q, t\rangle \in Q \times T$*. Following [12, 15] we write form* $q||t$ *in lieu of* $\langle q, t\rangle$*.*
2. *Configuration* $q||t$ *is* successful *if* $t \xrightarrow{w}$*.*
3. *The* configuration transition relation*,* $q||t \xrightarrow{a} q'||t'$*, is defined by the following rules.*
   - $a \neq \tau, q \xrightarrow{a} q', t \xrightarrow{a} t' \Rightarrow q||t \xrightarrow{a} q'||t'$*.*
   - $a = \tau, q \xrightarrow{\tau} q' \Rightarrow q||t \xrightarrow{\tau} q'||t$*.*
   - $a = \tau, t \xrightarrow{\tau} t' \Rightarrow q||t \xrightarrow{\tau} q||t'$*.*
4. *A* computation *of* $q||t$ *is a maximal (hence potentially infinite) sequence* $q_1||t_1 \xrightarrow{a_1} q_2||t_2 \xrightarrow{a_2} \cdots$*, where* q_1 *is* q *and* t_1 *is* t*. A computation is* successful *if for some* $i \geq 1$*,* $q_i||t_i$ *is successful.*
5. q **may** t *if* $q||t$ *has a successful computation.* $\mathcal{P}$ **may** $\mathcal{T}$ *if* q_0 **may** t_0*.*
6. q **must** t *if every computation of* $q||t$ *is successful.* $\mathcal{P}$ **must** $\mathcal{T}$ *if* q_0 **must** t_0*.*

Intuitively, the configuration transition relation describes the response of a process in state q to a test in state t. It is designed to abstract away from the internal computation on the part of both the test and the process; either may evolve internally without interacting with the other. Also noteworthy is the fact that since processes are in general nondeterministic, one can distinguish between the *possibility* of success ($\mathcal{P}$ **may** $\mathcal{T}$) and the *inevitability* of success ($\mathcal{P}$ **must** $\mathcal{T}$).

The following testing preorders preorders may now be defined.

Definition 4 *Let $\mathcal{P}$ and $\mathcal{P}'$ be processes.*

1. $\mathcal{P} \sqsubseteq_{may} \mathcal{P}'$ *if for every test $\mathcal{T}$ such that $\mathcal{P}$* **may** *$\mathcal{T}$, $\mathcal{P}'$* **may** *$\mathcal{T}$.*
2. $\mathcal{P} \sqsubseteq_{must} \mathcal{P}'$ *if for every test $\mathcal{T}$ such that $\mathcal{P}$* **must** *$\mathcal{T}$, $\mathcal{P}'$* **must** *$\mathcal{T}$.*
3. $\mathcal{P} \sqsubseteq_{test} \mathcal{P}'$ *if $\mathcal{P} \sqsubseteq_{may} \mathcal{P}'$ and $\mathcal{P} \sqsubseteq_{must} \mathcal{P}'$.*

So $\mathcal{P} \sqsubseteq_{may} \mathcal{P}'$ if $\mathcal{P}'$ may pass all the tests that $\mathcal{P}$ may pass, while $\mathcal{P} \sqsubseteq_{must} \mathcal{P}'$ if $\mathcal{P}'$ must pass all the tests that $\mathcal{P}$ must pass.

A number of alternative characterizations have been developed to ease the task of reasoning using these preorders. Here we present one due to DeNicola [11]. For simplicity, we restrict our attention to *divergence-free* processes, *i.e.*, processes in which no infinite sequence of τ's is possible from any state.

Definition 5 *Let $\mathcal{P} = \langle Q, A, \rightarrow, q_0\rangle$ be a process, with $q, q' \in Q$. Let $s \in (A - \{\tau\})^*$ be a sequence of visible actions.*

1. $q \stackrel{s}{\Longrightarrow} q'$ *is defined inductively as follows:*
   - $q \stackrel{\epsilon}{\Longrightarrow} q'$ *if* $q(\stackrel{\tau}{\rightarrow})^* q'$.
   - $q \stackrel{as'}{\Longrightarrow} q'$ *if* $q \stackrel{\epsilon}{\Longrightarrow}\stackrel{a}{\rightarrow}\stackrel{s'}{\Longrightarrow} q'$.

   We write $q \stackrel{s}{\Longrightarrow}$ if there is a q' such that $q \stackrel{s}{\Longrightarrow} q'$.
2. *The* language, *$L(q)$, of q is defined by: $L(q) = \{\, s \mid q \stackrel{s}{\Longrightarrow} \,\}$. $L(\mathcal{P}) = L(q_0)$.*
3. *The set of states q* **after** *s is defined by: q* **after** *$s = \{\, q' \mid q \stackrel{s}{\Longrightarrow} q' \,\}$.*
4. *Let $L \subseteq A$. Then q* **can** *L holds if for every $q' \in q$* **after** *ϵ, there is an $a \in L$ such that $q' \stackrel{a}{\Longrightarrow}$. For $P \subseteq Q$, define P* **can** *L to hold if for every $q \in P$, q* **can** *L.*[1]

[1] We have changed the notation used in [11] slightly in order to maintain consistency with [12, 15]; the **must** of [11] is our **can**, and our **must** is his **must satisfy**.

Intuitively, $L(q)$ contains the sequences of visible actions that q may engage in, while q **after** s consists of all the states reachable from q via the visible sequence s. The statement q **can** L essentially indicates that, regardless of the internal computation q engages in, an action from L may always be engaged in. Notice that q **after** s **can** L holds of q if for every q' reachable from q via s, q' **can** L. The following theorem is a restatement of one from [11].

Theorem 2.1 *Let $\mathcal{P} = \langle Q, A, \rightarrow, q_0\rangle$ and $\mathcal{P}' = \langle Q', A', \rightarrow, q_0'\rangle$ be processes.*

1. $\mathcal{P} \sqsubseteq_{may} \mathcal{P}'$ *if and only if $L(q_0) \subseteq L(q_0')$.*
2. $\mathcal{P} \sqsubseteq_{must} \mathcal{P}'$ *if and only if for every $s \in (A - \{\tau\})^*$ and $L \subseteq (A - \{\tau\})$ such that q_0* **after** *s* **can** *L, q_0'* **after** *s* **can** *L.*

3 Timed Testing

The traditional testing scenario of [11, 12, 15] ignores the amount of internal computation processes perform in response to tests. In a configuration $q||t$, both q and t may engage in internal computation at any time, regardless of the state of the other component. Accordingly, the resulting preorders are *delay-insensitive*; for any process $\mathcal{P}$ and action a, $a.\tau.\mathcal{P}$ and $a.\mathcal{P}$ are *testing equivalent*. However, from a timing perspective, one may argue that $a.\tau.\mathcal{P}$ and $a.\mathcal{P}$ should *not* be considered equivalent, since after engaging in an a action $a.\tau.\mathcal{P}$ must wait for one "tick" to begin acting like $\mathcal{P}$, while $a.\mathcal{P}$ need not. In this section, we propose a mechanism for capturing this intuition that is consistent with the testing framework outlined in above. We first redefine the configuration transition relation to take account of the (internal computation-caused) delay a process displays in response to tests. We then apply precisely the same treatment described in Section 2 to obtain timed notions of **may** and **must**.

The formal definition of the *timed configuration transition relation*, $\longrightarrow_\chi$, is the following.

Definition 6 *Let $\mathcal{P} = \langle Q, A, \rightarrow, q_0\rangle$ be a process and $\mathcal{T} = \langle T, A' \cup \{w\}, \rightarrow, t_0\rangle$ a test. Let $q \in Q$ and $t \in T$, with $a \in A$. (Note that a may be τ.) Then $\stackrel{a}{\longrightarrow}_\chi$ is defined by the following rules.*

- $q \stackrel{a}{\rightarrow} q', t \stackrel{a}{\rightarrow} t' \Rightarrow q||t \stackrel{a}{\longrightarrow}_\chi q'||t'$.
- $t \stackrel{\tau}{\rightarrow} t', q \stackrel{\tau}{\not\rightarrow} \ \Rightarrow q||t \stackrel{\tau}{\longrightarrow}_\chi q||t'$.

The intuition underlying this definition is the following. We assume that τ actions performed by a process have a duration of one "clock tick", and we wish to interpret the τ actions that a test can perform as "allowable delays". So, when the test is capable of engaging in a τ, it is willing to permit a delay of one "tick", and the process being tested may therefore engage in a τ. If the test does not allow a τ, then the process must accept an action that the test offers in order for the computation to proceed. In other words, the process cannot perform internal actions autonomously. These observations are the motivation for the first part of the above definition. The second part arises from the following property of reactive systems: if a system cannot execute internally, it "idles" (*i.e.*, remains in its current state) until its environment offers an interaction. Accordingly, if a test is willing to delay (by executing internally), and the process is incapable of internal computation, then the test may perform its delay and continue with the rest of the test.

Using this definition of testing, we can define q $\mathbf{may}_\chi$ t, q $\mathbf{must}_\chi$ t, $\mathcal{P}$ $\mathbf{may}_\chi$ $\mathcal{T}$ and $\mathcal{P}$ $\mathbf{must}_\chi$ $\mathcal{T}$ by replacing occurrences of $\xrightarrow{a}$ in Definition 3 with $\xrightarrow{a}_\chi$. We now consider examples of various process constructs and timing properties that can be defined in our framework.

Example 3.1

1. Delay *constructs are possible. For example, to model the process that waits three ticks, then behaves like $\mathcal{P}$, we may use $\tau.\tau.\tau.\mathcal{P}$.*

2. Time-out *operators may be constructed. Suppose we wish to build a process with the following behavior: "Make an a available until the third tick, then time-out, pre-empting the a and making b available." We can represent this as $a + \tau.(a + \tau.(a + \tau.b))$.*

3. Upper bounds *on the amount of permissible delay can be specified by the tests which a process may or must pass. Consider the test $\mathcal{T} = a.\tau.\tau.b.w$. Then $\mathcal{P}$ $\mathbf{may}_\chi$ $\mathcal{T}$ if $\mathcal{P}$ makes an a available immediately, and then two ticks later, can make b available. On the other hand, if $\mathcal{P}$ $\mathbf{must}_\chi$ $\mathcal{T}$, then $\mathcal{P}$ must make a available immediately and, after a two-tick delay, make b available. For example:*
   - $a.(\tau.b + \tau.\tau.\tau.b)$ $\mathbf{may}_\chi$ $a.\tau.\tau.b.w$
   - $a.\tau.\tau.\tau.b$ $\not\mathbf{may}_\chi$ $a.\tau.\tau.b.w$
   - $a.(\tau.b + \tau.\tau.\tau.b)$ $\not\mathbf{must}_\chi$ $a.\tau.\tau.b.w$
   - $a.\tau.b$ $\mathbf{must}_\chi$ $a.\tau.\tau.b.w$

4. Lower bounds *on the amount of permissible delay may also be specified using tests. Consider the test $\mathcal{T} = a + \tau.a.w$. A process $\mathcal{P}$ $\mathbf{must}_\chi$ $\mathcal{T}$ if* no *a is available immediately, but an a is guaranteed to be available after one tick. So we have:*
   - *a $\not\mathbf{must}_\chi$ $a + \tau.a.w$, since the computation $a||a + \tau.a.w \xrightarrow{a}_\chi nil||nil$ is unsuccessful.*
   - *$\tau.a$ $\mathbf{must}_\chi$ $a + \tau.a.w$.*

We now sketch our approach for defining timed testing preorders. There are two parameters in our framework: a class of processes (transition systems), and a class of tests.

Definition 7 *Let $\mathcal{C}$ be a class of transition systems, with $\mathcal{P}, \mathcal{P}' \in \mathcal{C}$. Also let $\mathcal{E}$ be a class of tests. Then the associated* timed testing preorders *are defined by:*

1. *$\mathcal{P} \sqsubseteq^{\mathcal{E}}_{may_\chi} \mathcal{P}'$ if for all $\mathcal{T} \in \mathcal{E}, \mathcal{P}$ $\mathbf{may}_\chi$ $\mathcal{T}$ implies $\mathcal{P}'$ $\mathbf{may}_\chi$ $\mathcal{T}$.*
2. *$\mathcal{P} \sqsubseteq^{\mathcal{E}}_{must_\chi} \mathcal{P}'$ if for all $\mathcal{T} \in \mathcal{E}, \mathcal{P}$ $\mathbf{must}_\chi$ $\mathcal{T}$ implies $\mathcal{P}'$ $\mathbf{must}_\chi$ $\mathcal{T}$.*
3. *$\mathcal{P} \sqsubseteq^{\mathcal{E}}_{test_\chi} \mathcal{P}'$ if $\mathcal{P} \sqsubseteq^{\mathcal{E}}_{may_\chi} \mathcal{P}'$ and $\mathcal{P} \sqsubseteq^{\mathcal{E}}_{must_\chi} \mathcal{P}'$.*

In the next section we illustrate the application of our framework in two different settings. In the first, we consider the classes of all processes and all tests. In the second setting, our goal is to isolate a class of transition systems and tests that yield timed testing preorders conveying a notion of "faster than"; more precisely, our goal is a framework in which for both $\sqsubseteq^{\mathcal{E}}_{may_\chi}$ and $\sqsubseteq^{\mathcal{E}}_{must_\chi}$, it is the case that $\tau.\mathcal{P}$ is generally "strictly less than" $\mathcal{P}$.

4 Applying the Framework

Our goals in this section are two-fold: to continue redeveloping the untimed testing work in a timed setting, and to investigate preorders that capture a notion of "faster than." Toward these ends, we first examine the timed preorders that result from considering the class of all processes and the class of all tests. These relations turn out to have interesting denotational characterizations that are similar in spirit to the alternative characterization of the untimed preorders given in Section 2. However, they turn out to be too fine to capture a notion of "faster than" adequately. Accordingly, in the next subsection, we restrict our attention to a *nonpre-emptive* subclass of transition systems and a class of "linear" tests. The resulting preorders turn out to convey a notion of relative efficiency in the sense that $\tau.\mathcal{P} \sqsubseteq^{\mathcal{E}}_{test_\chi} \mathcal{P}$ while the reverse is not, in general, true.

4.1 General Timed Transition Systems

In the untimed testing preorders of [11, 12, 15], both processes and tests may be arbitrary (image-finite) transition systems. Accordingly, in this subsection we investigate the timed testing preorders that result from applying our framework to this scenario.

Definition 8 *Let $\mathcal{C}$ be the class of all (image-finite) transition systems, and let $\mathcal{E}$ be the class of all tests. Then we write $\sqsubseteq_{may_\chi}$ for $\sqsubseteq^{\mathcal{E}}_{may_\chi}$, $\sqsubseteq_{must_\chi}$ for $\sqsubseteq^{\mathcal{E}}_{must_\chi}$, and $\sqsubseteq_{test_\chi}$ for $\sqsubseteq^{\mathcal{E}}_{test_\chi}$.*

We now consider some examples of processes that are, and are not, related by these preorders.

Example 4.1

1. $\tau.a \sqsubseteq_{may_\chi} a$.
2. $\tau.(a+\tau.b) \not\sqsubseteq_{may_\chi} a+\tau.b$. *To see this, consider the test* $\mathcal{T} = \tau.a.w$. *Now* $\tau.(a+\tau.b)$ $\mathbf{may}_\chi$ $\mathcal{T}$, *but* $a+\tau.b$ $\cancel{\mathbf{may}}_\chi$ $\mathcal{T}$.
3. $b+\tau.b \sqsubseteq_{must_\chi} b$.
4. $\tau.b \not\sqsubseteq_{must_\chi} b$. *Consider the test* $\mathcal{T} = b+\tau.b.w$. *Now* $\tau.b$ $\mathbf{must}_\chi$ $\mathcal{T}$, *but* b $\cancel{\mathbf{must}}_\chi$ $\mathcal{T}$.
5. $a.\tau.b \not\sqsubseteq_{test_\chi} a.b$ *because* $a.\tau.b$ $\mathbf{must}_\chi$ $a.(b+\tau.b.w)$ *but* $a.b$ $\cancel{\mathbf{must}}_\chi$ $a.(b+\tau.b.w)$.

As is demonstrated in part 5 the timed testing preorder and the untimed testing preorder are not equivalent. These examples also illustrate the inadequacy of $\sqsubseteq_{may_\chi}$ and $\sqsubseteq_{must_\chi}$ as *durational* preorders, since in general it is not the case that $\tau.\mathcal{P} \sqsubseteq_{may_\chi} \mathcal{P}$ (part 2) or that $\tau.\mathcal{P} \sqsubseteq_{must_\chi} \mathcal{P}$ (part 4).

The preorders $\sqsubseteq_{may_\chi}$ and $\sqsubseteq_{must_\chi}$ have alternative semantic characterizations that are noteworthy in their similarity to the characterizations of the untimed testing preorders mentioned in Section 2. We need the following definitions.

Definition 9 *Let $\mathcal{P} = \langle Q, A, \longrightarrow, q_0\rangle$ be a transition system, with $q, q' \in Q$, and let $i \geq 0$.*

1. *The set of* timed actions, A_χ, *is defined as* $A_\chi = A \times \mathrm{Int}$.
2. $q \overset{i}{\Rightarrow} q'$ *if either* $q(\overset{\tau}{\longrightarrow})^i q'$, *or there is a* $j < i$ *such that* $q(\overset{\tau}{\longrightarrow})^j q'$ *and* $q' \overset{\tau}{\not\longrightarrow}$.
3. *Let* $\langle a, i\rangle \in A_\chi$. *Then* $q \overset{\langle a,i\rangle}{\Longrightarrow} q'$ *if* $q \overset{i}{\Rightarrow}\overset{a}{\longrightarrow} q'$.
4. *Let* $s \in A^*_\chi$. *Then* $\overset{s}{\Longrightarrow}$ *may be defined inductively by:*
   - $q \overset{\epsilon}{\Longrightarrow} q$.
   - $q \overset{\langle a,i\rangle s'}{\Longrightarrow} q'$ *if* $q \overset{\langle a,i\rangle}{\Longrightarrow}\overset{s'}{\Longrightarrow} q'$.
5. *The* timed language *of q is defined by:* $L_\chi(q) = \{\, s \in A^*_\chi \mid q \overset{s}{\Longrightarrow} \,\}$. *Also,* $L_\chi(\mathcal{P}) = L_\chi(q_0)$.
6. q $\mathbf{after}_\chi$ $\langle s, i\rangle = \{\, q' \mid q \overset{s}{\Longrightarrow}\overset{i}{\Rightarrow} q' \,\}$.
7. *Let* $L \subseteq (A - \{\tau\})$. *Then* q $\mathbf{can}_\chi$ L *if for some* $a \in L, q \overset{a}{\longrightarrow}$. *For* $P \subseteq Q$, P $\mathbf{can}_\chi$ L *holds if for every* $q \in P, q$ $\mathbf{can}_\chi$ L.
8. $\mathrm{SIL}(q) = \{\, \langle s, i, L\rangle \mid q$ $\mathbf{after}_\chi$ $\langle s, i\rangle$ $\mathbf{can}_\chi$ $L \,\}$. *Also,* $\mathrm{SIL}(\mathcal{P}) = \mathrm{SIL}(q_0)$.

This definition may be thought of as the "timed counterpart" of Definition 3. So $q \overset{\langle a,i\rangle}{\Longrightarrow} q'$ holds if, after a delay of i "ticks", q may engage in a and evolve to q'. Then $L_\chi(\mathcal{P})$ can be seen as the set of sequences of "timed actions" that are available to $\mathcal{P}$. We now have the following.

Theorem 4.2 *For any processes $\mathcal{P}$ and $\mathcal{P}'$:*

1. $\mathcal{P} \sqsubseteq_{may_\chi} \mathcal{P}'$ *if and only if* $L_\chi(\mathcal{P}) \subseteq L_\chi(\mathcal{P}')$.
2. $\mathcal{P} \sqsubseteq_{must_\chi} \mathcal{P}'$ *if and only if* $\mathrm{SIL}(\mathcal{P}) \subseteq \mathrm{SIL}(\mathcal{P}')$.

Proof sketch: For the first part, the result follows from the observation that $\sqsubseteq_{may_\chi}$ is completely characterized by tests consisting of a sequence of actions from A (including τ) followed by a w. For the second part, to prove the "only if" direction, it suffices to give a mechanism for constructing a test from a triple $\langle s, i, L\rangle$ so that a process $\mathcal{P}$ $\mathbf{must}_\chi$ the test iff $\langle s, i, L\rangle \in \mathrm{SIL}(\mathcal{P})$. For the "if" direction, it is sufficient to exhibit a technique for building sets of triples of the form $\langle s, i, L\rangle$ from tests so that a process $\mathbf{must}_\chi$ the test iff the set of triples are in SIL of the process. □

4.2 Nonpre-emptive Transition Systems

We now present a class of transition systems and a class of tests for which both the timed may and the timed must preorders capture a notion of "faster than." We then develop alternative characterizations of the preorders as we did in the previous section. In Section 5 we argue that this class of transition systems is reasonable by giving a *process algebra* that generates transition systems in the class.

Intuitively, we are interested in a class of transition systems that are *nonpre-emptive* in the sense that internal computation does not "eliminate" possibilities for external behavior. The formal definition of our notion of nonpre-emptive is the following.

Definition 10 *A labeled transition system* $\langle Q, A, \rightarrow, q_0\rangle$ *is* nonpre-emptive *if the following holds for every* $q \in Q$ *with distinct transitions* $q \xrightarrow{\tau} q_1$ *and* $q \xrightarrow{a} q_2$.

> *Either* $a \neq \tau$ *and* $q_1 \xrightarrow{a} q_2$*; or there is a* q_3 *such that* $q_1 \xrightarrow{a} q_3$ *and* $q_2 \xrightarrow{\tau} q_3$.

Example 4.3

1. $a + \tau.(a + a.b)$ *is nonpre-emptive. Initially, an a-transition to nil is possible, and a* τ*-transition to* $a + a.b$ *is possible; but* $a + a.b$ *has an a-transition to nil.*
2. $b.\tau + \tau.b$ *is nonpre-emptive. The initial b and* τ *transitions lead to states* q_1 *and* q_2 *respectively, and there is a* τ*-transition to nil from* q_1 *and a b-transition from* q_2 *to nil.*
3. $a + \tau.b$ *is* not *nonpre-emptive, since a is no longer possible after the* τ *is executed.*

Let $\mathcal{N}$ be the class of (image-finite) nonpre-emptive transition systems.

The set of tests we use are *linear*; each test has "form" $s.w$, where $s \in A^*$. So $a.w$ and $\tau.b.\tau.\tau.a.w$ are examples of such tests. We let $\mathcal{L}$ denote the set of linear tests. We can now apply our timed preorder framework to generate $\sqsubseteq^{\mathcal{L}}_{may_\chi}$, $\sqsubseteq^{\mathcal{L}}_{must_\chi}$ and $\sqsubseteq^{\mathcal{L}}_{test_\chi}$ over $\mathcal{N}$. The following examples illustrate processes that are, and are not, related by these preorders.

Example 4.4

1. $\tau.a \sqsubseteq^{\mathcal{L}}_{may_\chi} a$.
2. $a \not\sqsubseteq^{\mathcal{L}}_{may_\chi} \tau.a$*, since* a $\mathbf{may}_\chi$ $a.w$*, but* $\tau.a$ $\cancel{\mathbf{may}}_\chi$ $a.w$.
3. $\tau.a \sqsubseteq^{\mathcal{L}}_{must_\chi} a$.
4. $a \not\sqsubseteq^{\mathcal{L}}_{must_\chi} \tau.a$*, since* a $\mathbf{must}_\chi$ $a.w$ *and* $\tau.a$ $\cancel{\mathbf{must}}_\chi$ $a.w$.

In fact, both $\sqsubseteq^{\mathcal{L}}_{may_\chi}$ and $\sqsubseteq^{\mathcal{L}}_{must_\chi}$ embody a notion of "relative efficiency" as formally stated in the following theorem.

Theorem 4.5 *Let* $\mathcal{P} \in \mathcal{N}$ *be a nonpre-emptive transition system. Then* $\tau.\mathcal{P} \sqsubseteq^{\mathcal{L}}_{may_\chi} \mathcal{P}$*, and* $\tau.\mathcal{P} \sqsubseteq^{\mathcal{L}}_{must_\chi} \mathcal{P}$.

It is worth pointing out that, in general, $\mathcal{P} \not\sqsubseteq^{\mathcal{L}}_{may_\chi} \tau.\mathcal{P}$ and $\mathcal{P} \not\sqsubseteq^{\mathcal{L}}_{must_\chi} \tau.\mathcal{P}$. For example, if the start state of $\mathcal{P}$ is *stable* (*i.e.*, incapable of a τ) and not deadlocked (*i.e.*, capable of some action), then both of these statements are true. In addition, it is the case that prefixing preserves both $\sqsubseteq^{\mathcal{L}}_{may_\chi}$ and $\sqsubseteq^{\mathcal{L}}_{must_\chi}$.

As we did for the general timed preorders, we now present an alternative characterization of the nonpre-emptive preorders. It turns out that $\sqsubseteq^{\mathcal{L}}_{may_\chi}$ coincides with $\sqsubseteq_{may_\chi}$ restricted to $\mathcal{N}$. On the other hand, the characterization for $\sqsubseteq^{\mathcal{L}}_{must_\chi}$ is very different. The following definitions are useful.

Definition 11 *Let* $\mathcal{P} = \langle Q, A, \rightarrow, q_0\rangle$ *be a transition system.*

1. *Let* $s \in A^*_\chi$ *and* $q \in Q$*. Then define* $mt(q, s)$ *by induction on* s *as follows:*
   - $mt(q, \epsilon)$ *for all* q.
   - $mt(q, \langle a, 0\rangle s')$ *if* $q \xrightarrow{a}$ *and for all* q' *such that* $q \xrightarrow{a} q'$, $mt(q', s')$.
   - $mt(q, \langle a, i+1\rangle s')$ *if for all* q' *such that* $q \overset{1}{\Rightarrow} q'$, $mt(q', \langle a, i\rangle s')$.

   If $mt(q, s)$ *holds, we say that* s *is a* must trace *of* q*. Also,* $mt(\mathcal{P}, s)$ *if and only if* $mt(q_0, s)$.
2. *The* must language *of* $\mathcal{P}$ *is given by:* $L^M_\chi(\mathcal{P}) = \{\, s \mid mt(\mathcal{P}, s)\,\}$.

Intuitively, s is a must trace of $\mathcal{P}$ if, no matter which states $\mathcal{P}$ enters as it "computes" s, the whole trace s executes to completion. For example $\langle a, 0\rangle$ is a must trace of $a.b + a.c$, while $\langle a, 0\rangle\langle b, 0\rangle$ is not. We now have the following.

Theorem 4.6 *For any nonpre-emptive processes* $\mathcal{P}$ *and* $\mathcal{P}'$*:*

1. $\mathcal{P} \sqsubseteq^{\mathcal{L}}_{may_\chi} \mathcal{P}'$ *if and only if* $L_\chi(\mathcal{P}) \subseteq L_\chi(\mathcal{P}')$.
2. $\mathcal{P} \sqsubseteq^{\mathcal{L}}_{must_\chi} \mathcal{P}'$ *if and only if* $L^M_\chi(\mathcal{P}) \subseteq L^M_\chi(\mathcal{P}')$.

We close this subsection with some observations about $\sqsubseteq^{\mathcal{L}}_{may_\chi}$ and $\sqsubseteq^{\mathcal{L}}_{must_\chi}$. First, one may criticize our definition of nonpre-emptiveness as being *ad hoc*. We address this criticism in the next section by giving a process algebra that generates nonpre-emptive transition systems. Second, one may charge that the set $\mathcal{L}$ is too restrictive. It is true that the resulting must preorder is less sensitive to nondeterminism than the untimed must preorder, and it may be possible to enrich $\mathcal{L}$ to remedy this while retaining the "faster than" characteristics of the may and must preorders. This is a subject of ongoing investigation.

5 A Nonpre-emptive Process Algebra

In this section we introduce a simple process algebra for representing nonpre-emptive transition systems. We give its syntax and operational semantics, and we point out the congruence properties that the nonpre-emptive timed preorders introduced in the previous section enjoy with respect to this language.

Let A be a set of actions (containing τ) ranged over by a, let $\mathcal{V}$ be a set of *process variables* ranged over by X, and let $S \subseteq (A - \{\tau\})$. Also let $f : A \rightarrow A$ be such that $f(a) = \tau$ if and only if $a = \tau$. Then the formal syntax of our language is given as follows.

$$p \quad ::= \quad nil \mid X \mid a.p \mid p [\!] p \mid p||_S p \mid p[f] \mid rec(X.p)$$

Occurrences of X in the term $rec(X.p)$ are bound in the usual sense. We write $p[q/X]$ to represent the simultaneous substitution of q for all free occurrences of X in p.

As is usual in a process algebra, our intention is to define a *global* transition system $\langle P, A, \rightarrow \rangle$, where P is the set of closed terms generated by the above grammar and $\rightarrow$ is given by the *operational semantics* of terms. Then processes are defined by giving the term that corresponds to their start states in the global transition system. The formal semantics of the language appear in Figure 1.

Informally, the operators have the following description. Process $a.p$ may engage in action a and then behave like p. Process $p [\!] q$ may engage in any of the external actions that p or q may, and thereafter behave like the process whose initial external action was chosen; notice that *internal actions* may *not* resolve the choice. Similar choice operators are used in Hennessy [15] and MEIJE [2]. Process $p||_S q$ behaves like the *interleaving* of p and q, with the added stipulation that p and q must synchronize on actions in S (note that syntax restrictions prohibit τ being an element of S); this operator appears in TCSP [6, 17]. The process $p[f]$ behaves like p with actions "relabeled" by f. Finally, the process $rec(X.p)$ can be thought of as a distinguished solution to the equation $X = p$.

The next lemma establishes that the global transition system induced by this language is nonpre-emptive.

Lemma 5.1 *Let p be a closed process term. Then p is nonpre-emptive.*

It also turns out that the timed testing preorders $\sqsubseteq^{\mathcal{L}}_{may_\chi}$ and $\sqsubseteq^{\mathcal{L}}_{must_\chi}$ introduced in the previous section are preserved by the finite (*i.e.*, non-*rec*) operators in this language.

Theorem 5.2 *The relations $\sqsubseteq^{\mathcal{L}}_{may_\chi}$ and $\sqsubseteq^{\mathcal{L}}_{must_\chi}$ are precongruences for the finite language. That is, whenever $p \sqsubseteq^{\mathcal{L}}_{may_\chi} p'$, then*

1. $a.p \sqsubseteq^{\mathcal{L}}_{may_\chi} a.p'$;
2. $p [\!] q \sqsubseteq^{\mathcal{L}}_{may_\chi} p' [\!] q$ *and* $q [\!] p \sqsubseteq^{\mathcal{L}}_{may_\chi} q [\!] p'$;
3. $p||_S q \sqsubseteq^{\mathcal{L}}_{may_\chi} p'||_S q$ *and* $q||_S p \sqsubseteq^{\mathcal{L}}_{may_\chi} q||_S p'$;
4. $p[f] \sqsubseteq^{\mathcal{L}}_{may_\chi} p'[f]$.

The similar substitutivity results hold for $\sqsubseteq^{\mathcal{L}}_{must_\chi}$.

In closing, we remark on a couple of operators that are not in our language. The first is internal choice (often written $\oplus$); we have omitted it in the interest of simplicity, since defining it operationally in a way that does not affect the timing behavior of processes requires additional overhead. In principle, however, it would not be difficult to introduce. A more problematic operator is *hiding*. Given the language that we have defined, an appropriate abstraction operator would have the effect of "relabeling" actions to τ. Traditional hiding operators would destroy the nonpre-emptive character of the processes that may be built in this language; defining a nonpre-emptive hiding operator is possible (and may be done along the lines pursued in [30]), but doing so is beyond the scope of this paper.

6 An Example

In this section we work an example illustrating the application of the $\sqsubseteq^{\mathcal{L}}_{may_\chi}$ and $\sqsubseteq^{\mathcal{L}}_{must_\chi}$ preorders introduced in the previous section. Consider the following, where B_2 is to be interpreted as the specification of a two-place buffer.

$$\begin{aligned} B_2 &= in.B_2' \\ B_2' &= in.B_2'' [\!] out.B_2 \\ B_2'' &= out.B_2' \end{aligned}$$

$$
\begin{array}{rcl}
a.p \xrightarrow{a} p & & \\
p \xrightarrow{a} p', a \neq \tau & \Rightarrow & p \,[\!]\, q \xrightarrow{a} p' \\
q \xrightarrow{a} q', a \neq \tau & \Rightarrow & p \,[\!]\, q \xrightarrow{a} q' \\
p \xrightarrow{\tau} p' & \Rightarrow & p \,[\!]\, q \xrightarrow{\tau} p' \,[\!]\, q \\
q \xrightarrow{\tau} q' & \Rightarrow & p \,[\!]\, q \xrightarrow{\tau} p \,[\!]\, q' \\
p \xrightarrow{a} p', a \notin S & \Rightarrow & p \|_S q \xrightarrow{a} p' \|_S q \\
q \xrightarrow{a} q', a \notin S & \Rightarrow & p \|_S q \xrightarrow{a} p \|_S q' \\
p \xrightarrow{a} p', q \xrightarrow{a} q', a \in S & \Rightarrow & p \|_S q \xrightarrow{a} p' \|_S q' \\
p \xrightarrow{a} p' & \Rightarrow & p[f] \xrightarrow{f(a)} p'[f] \\
p[rec(X.p)/X] \xrightarrow{a} p' & \Rightarrow & rec(X.p) \xrightarrow{a} p'
\end{array}
$$

Figure 1: The operational semantics of the nonpre-emptive language.

The buffer has two communication channels: *in*, from which it reads its inputs, and *out*, onto which it delivers its outputs. Initially, the buffer is empty; after an input, it may either deliver an output or accept another input, at which point it may only output. Our goal is to develop an implementation I such that $B_2 \sqsubseteq^{\mathcal{L}}_{test_\chi} I$, using one-place buffers defined as follows.

$$
\begin{array}{rcl}
B_1 & = & in.B_1' \\
B_1' & = & out.B_1
\end{array}
$$

Traditionally, this problem may be approached in two ways. One solution is to hook two one-place buffers together "in series" by linking the output port of the first buffer to the input port of the second buffer. We can do this as follows.

$$
I_{series} = (B_1[i/out] \,\|_{\{i\}}\, B_1[i/in]) \lfloor \{i\}
$$

Here $\lfloor\{i\}$ is a hiding operator; it essentially "relabels" i actions to τ. In general, $p\lfloor\{i\}$ will not be nonpre-emptive, even if p is; however, in this case, it does turn out that I_{series} is nonpre-emptive. Now, in an untimed setting we may show that B_2 and I_{series} are testing equivalent, and I_{series} is therefore deemed a correct implementation.

In our nonpre-emptive timed setting, however, this is not the case. One may easily show that $B_2 \not\sqsubseteq^{\mathcal{L}}_{test_\chi} I_{series}$; in fact, it is not even the case that $B_2 \sqsubseteq^{\mathcal{L}}_{may_\chi} I_{series}$. To see this, it is sufficient to note that

$$
B_2 \ \mathbf{may}_\chi \ in.out.w
$$

while

$$
I_{series} \ \not{\mathbf{may}}_\chi \ in.out.w.
$$

Intuitively, this is because of the τ that I_{series} must perform in order to "transfer" the input from the first buffer to the second buffer. Thus, B_2 is faster than I_{series}, and I_{series} is therefore not a correct implementation from the timing perspective. Note that it is the case that $I_{series} \sqsubseteq^{\mathcal{L}}_{test_\chi} B_2$.

Now consider the second traditional approach to developing a correct implementation of B_2; this one involves assembling two one-place buffers "in parallel" as follows.

$$
I_{parallel} \quad = \quad (B_1 \|_\emptyset B_1)
$$

Again, in the untimed setting B_2 and $I_{parallel}$ are testing equivalent. The same also holds in the nonpre-emptive timed setting; in particular, $B_2 \sqsubseteq^{\mathcal{L}}_{test_\chi} I_{parallel}$.

7 Conclusions

In this paper we have developed a framework for *testing preorders* [12, 15] that take account of processes' *timing behavior* in addition to their degrees of relative nondeterminism. We have given both operational and "denotational" characterizations of these preorders in two different settings: one in which both tests and processes may be arbitrary transition systems, and one in which processes are *nonpre-emptive* and tests are *linear*. In the latter case, the preorders convey a notion of *relative efficiency* in the sense that when p is "less than" q, p exhibits more delay than q does. We also presented a language for expressing nonpre-emptive processes and established that the nonpre-emptive timed testing preorders are precongruences for the language, and we worked an example illustrating the application of these preorders.

A variety of issues remain to be explored. For instance, we have shown how our framework may be applied to develop preorders representing a notion of "faster than"; the basic idea involved fixing on an appropriate set of tests that express these kind of constraints. One area requiring further investigation is the strengthening of the must preorder in this setting to be more sensitive to nondeterminism. Also, other timing considerations are often of interest: for example, for certain kinds of "hard real-time" systems, lower bounds on delays are often given. Determining sets of tests to capture these notions, and from them timing preorders, would be useful. This paper has focused more on the semantic aspects of our model; investigating various language constructs and inequational axiom systems for reasoning about processes built using them for the general and the "faster than" preorders would be a fruitful avenue to pursue. In another vein, we have seen that in the general class of transition systems, the timed preorders do not license the treatment of internal computation as duration, while nonpre-emptive transition systems do. Some pre-emptive constructs, such as time-outs, are clearly useful in a timed setting; the question arises, then as to the existence of a class of transition systems in which time may treated as duration and yet certain types of pre-emption are permitted. Also, in the context of our nonpre-emptive language a reasonable hiding operator needs to be developed. Finally, a fully abstract denotational model along the lines of Hennessy's acceptance trees [15] for general and nonpre-emptive transition systems would be useful, since this would, among other things, permit the development of algorithms for automatically computing the timed testing relations. Such algorithms could then be implemented automated tools such as the Concurrency Workbench [8, 9].

Related Work

Recent years have witnessed the incorporation of real-time constructs in process algebras [19, 23, 20, 18, 13] [14, 27, 28, 7, 16, 25] [3, 30, 1, 21] and the development of techniques for reasoning about their properties. In contrast with our work, much of this work treats actions as *durationless*. The passage of time is then modeled by assigning times to the occurrence of actions (as in [3, 14]) or by introducing special actions that denote "ticks" or "idles", as in [16, 25]. The equivalences and preorders that result typically do not allow one to reason about differences in relative delays. One exception may be found in [1], which develops a preorder in which CCS processes are related by their speed. The preorder is defined by a bisimulation equivalence in which the processes must reach equivalent states by executing sequences of actions that differ only in their speed (where speed is represented by the number of delay actions separating the visible actions).

References

[1] Arun-Kumar, S., and M. Hennessy. "An Efficiency Preorder for Processes." Technical Report 5/90, Department of Computing Science, University of Sussex, July 1990.

[2] Austry, D. and G. Boudol. "Algèbre de processus et synchronisations." *Theoretical Computer Science*, v. 30, 1984, pp. 91-131.

[3] Baeten, J.C.M., and J.A. Bergstra. "Real Time Process Algebra." Technical Report P8916b, Programming Research Group, University of Amsterdam, March 1990.

[4] Bergstra, J.A. and J.W. Klop. "Algebra of Communicating Processes with Abstraction." *Theoretical Computer Science*, v. 37, 1985, pp. 77-121.

[5] Bloom, B., S. Istrail and A. Meyer. "Bisimulation Can't Be Traced." In *Proceedings of the ACM Symposium on Principles of Programming Languages*, 1988.

[6] Brookes, S.D., C.A.R Hoare and A.W. Roscoe. "A Theory of Communicating Sequential Processes." *Journal of the ACM*, v. 31, n. 3, July 1984, pp. 560-599.

[7] Cardelli, L. "Real Time Agents." In *Proceedings of Ninth Colloquium in Automata, Languages and Programming*, 1982, pp. 94–106.

[8] Cleaveland, R., J. Parrow and B. Steffen. "The Concurrency Workbench." In *Proceedings of the Workshop on Automatic Verification Methods for Finite-State Systems*, 1989, Lecture Notes in Computer Science 407, pp. 24–37. Springer-Verlag, Berlin.

[9] Cleaveland, R., J. Parrow and B. Steffen. "A Semantics based Verification Tool for Finite State Systems", In *Proceedings of the Ninth International Symposium on Protocol Specification, Testing, and Verification*. North Holland, 1989.

[10] Cleaveland, R. and B. Steffen. “A Preorder for Partial Process Specifications.”. In *Proceedings, CONCUR '90*, Lecture Notes in Computer Science 458. Springer-Verlag, Berlin, 1990, pp. 141–151.

[11] DeNicola, R. “Extensional Equivalences for Transition Systems.” *Acta Informatica*, v. 24, 1987, pp. 211–237.

[12] DeNicola, R. and Hennessy, M.C.B. “Testing Equivalences for Processes.” *Theoretical Computer Science*, v. 34, 1983, pp. 83-133.

[13] Gerber, R. and I. Lee. “CCSR: A Calculus for Communicating Shared Resources.” In *Proceedings, CONCUR '90*, Lecture Notes in Computer Science 458. Springer-Verlag, Berlin, 1990.

[14] Gerth, R., and A. Boucher. “A Timed Failure Semantics for Extended Communicating Processes.” In *Proceedings of ICALP '87*, Lecture Notes in Computer Science 267. Springer-Verlag, Berlin, 1987.

[15] Hennessy, M.C.B. *Algebraic Theory of Processes.* MIT Press, Boston, 1988.

[16] Hennessy, M., and T. Regan. “A Temporal Process Algebra.” Technical Report 2/90, Department of Computing Science, University of Sussex, April 1990.

[17] Hoare, C.A.R. *Communicating Sequential Processes.* Prentice-Hall, London, 1985.

[18] Huizing, C., R. Gerth and W.P. de Roever. “Full Abstraction of a Denotational Semantics for Real-time Concurrency.” In *Proceedings of POPL 87*, 1987, pp. 223–237.

[19] G. Jones. *A Timed Model for Communicating Processes.* DPhil Thesis, Oxford University, 1982.

[20] Koymans, R., R.K. Shyamasundar, W.P. de Roever, R. Gerth and S. Arun-Kumar. “Compositional Semantics for Real-Time Distributed Computing.” In *Proceedings of Logic of Programs Workshop '85*, Lecture Notes in Computer Science 193. Springer-Verlag, Berlin, 1985.

[21] Milne, G.J. “CIRCAL and the Representation of Communication, Concurrency, and Time.” TOPLAS, v. 7, n. 2, April 1985, pp. 270–298.

[22] Milner, R. *A Calculus of Communicating Systems*, Lecture Notes in Computer Science 92. Springer-Verlag, Berlin, 1980.

[23] Milner, R. “Calculi for Synchrony and Asynchrony.” *Theoretical Computer Science*, v. 25, 1983, pp. 267–310.

[24] Milner, R. *Communication and Concurrency.* Prentice Hall, 1989.

[25] Nicollin, X., J-L. Richier, J. and J. Voiron. “ATP: An Algebra for Timed Processes.” In *Proceedings of the IFIP TC 2 Working Conference on Programming Concepts and Methods Principles of Programming Languages*, April 1990, pp. 402–429.

[26] Pnueli, A. “Applications of Temporal Logic to the Specification and Verification of Reactive Systems: A Survey of Current Trends.” In *Current Trends in Concurrency*, J. de Bakker, W.-P. de Roever and G. Rozenberg eds., Lecture Notes in Computer Science 224. Springer-Verlag, Berlin, 1986.

[27] Reed, G.M., and A.W. Roscoe. “A Timed Model for Communicating Sequential Processes.” In *Proceedings of ICALP '86*, Lecture Notes in Computer Science 226. Springer-Verlag, Berlin, 1986.

[28] Reed, G.M., and A.W. Roscoe. “Metric Spaces as Models for Real-Time Concurrency.” In *Proceedings of Mathematical Foundations of Computer Science*, Lecture Notes in Computer Science 298. Spinger-Verlag, Berlin, 1987.

[29] Walker, D.J. “Bisimulation Equivalence and Divergence in CCS.” In *Proceedings of the Third Annual Symposium on Logic in Computer Science*, 1988, pp. 186-192.

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