A Model for Real-Time Systems*

Padmanabhan Krishnan

Department of Computer Science
Ny Munkegade 540
Aarhus University
DK 8000, Aarhus C, Denmark
E-mail: paddyd@aimi.aau.dk

April 1991

Abstract

In this paper we define an equivalence and a modal logic for real-time systems. The equivalence is based on timed processes and the timing specifications they have to satisfy. While the equivalence we define is not a congruence, it does satisfy many laws.

1 Introduction

While process formalisms such as CCS [4], cannot express time delays between events (actions), there are extensions [6, 9] which do. Most of these extensions are called real-time extension. We feel that while the addition of time is an important step towards characterizing real-time, it is by no means sufficient. As [2] explain a real-time system is one which has to satisfy certain properties given certain resource restrictions. The properties that a system is to satisfy play an important role in building architectures/schedulers for it.

*To appear in the 16th International Symposium on Mathematical Foundations of Computer Science (MFCS 1991)

Various notions of equivalence in a concurrent setting have been well established. Consider for example the notion of bisimilarity [8]. If two concurrent terms are bisimilar it means that any behavior exhibited by one can be exhibited by the other. If the bisimulation is a congruence, one item can be substituted for the other in 'arbitrary' contexts without affecting 'behavior'. Real-time systems have timing requirements and if context includes timing requirements, two terms to be equivalent in all contexts will necessarily have identical timing. For example, let timing constraints impose an upper bound on the time by which an execution must terminate. If two terms are equivalent under all timing constraints, they will have to terminate at the same time, as one can specify the upper bound for all times.

While the definitions which are a straight generalization of those used in concurrency results in a general theory which can be used in arbitrary contexts, it is not useful for specific real-time systems. Systems rarely impose precise times when actions are to be taken. Usually, they permit a range of times in which the actions can be taken. So one could define equivalence under a given set of timing constraints rather than arbitrary timing constraints. The notion of predictability is important in real-time systems. Thus, simulation must be defined in a more deterministic context. As a scheduler is an important implementation feature that introduces determinacy, simulation should consider schedulers. In the definition of simulation for concurrent systems, the definition of equivalence etc. is quite general. For example, ((a | b) | c) is trace equivalent to (a | (b | c)) meaning that any trace exhibited by the first can also be exhibited by the second and vice versa. However in the presence of timing constraints and a scheduler the effect of replacing one by the other could have disastrous effects on the safety of the system. Continuing with the above example, let (a after 10) be the timing specification t1 and (a before 30) be the timing specification t2. Let a scheduler for a uniprocessor environment for the above processes translate them as follows: sched((a | b) | c) = c·a·b, sched(a | (b | c)) = a·b·c.

If each · takes 20 units of time (starting at 0), then p1 is equivalent to p2 given the scheduler and timing constraint t1 but not under timing constraint t2. While this example may be rather contrived, the scheduler may behave in this particular fashion due to the presence of certain other timing constraints that it is trying to satisfy. In this paper we introduce definitions which are relevant in a real-time setting. As we are interested in implementation of real-time processes, we concentrate on simulations with bisimulation (or

replaceability) defined as a symmetric simulation.

The aim of this paper is to propose a calculus for real-time systems which expresses delays between events and also properties that the system has to satisfy. The calculus assumes a time domain wee ordered by $\leq$. We like [6, 9] use CCS as the basic untimed language. We also outline how the effect of a scheduler and architecture on a real-time program can be studied.

In section 2 we describe the syntactic elements of our calculus viz., a language to express behavior and a language to express timing constraints. In section 3 we describe an operations semantics for the behavioral aspect of the calculus. In section 4 we develop a notion of equivalence induced by the timing constraints, while in section 5 we present a logical characterization of the equivalence. In section 6 we outline how these ideas can be used to study system issues such as schedulers.

2 Syntax

We define our language RTCCS (for real-time CCS). We as in [4, 5, 6, 9], assume a set of atomic actions Λ on which a bijection $\bar{\cdot}$ can be defined such that for all $a \in \Lambda \bar{\bar{a}} = a$. The time domain we assume is integers (actually any discrete well ordered set would suffice).

The basic syntax of our language is as follows.

$$P = \text{Nil} \mid a;\text{P} \mid (\text{P} \mid \text{P}) \mid \text{P} + \text{P} \mid \text{P} \setminus \text{H} \mid \text{X} \mid \text{rec}\tilde{X} : \tilde{E}$$

Nil is a process which can exhibit no further action. a;P is a process which performs action a and then evolves to process P. The action a is atomic and takes unit time. (It can also be defined to take time greater than 1.) (t)P defines a process which can behave as P after time t. P | Q denotes parallel composition. P + Q denotes non-deterministic choice, P $\setminus$ H restricts the behavior of P to those actions *not* in H. rec$\tilde{X} : \tilde{E}$ represents guarded recursion and we only consider closed terms.

The language defined above is similar to that in [6]. Unlike [9] we do not consider passing of time as values to processes. Rather we specify timing constraints which processes have to satisfy. Generally speaking, there are two types of constraints in real-time systems 1) absolute and 2) relative [1].

Absolute constraints are expressed in terms of what actions must or must not happen in time intervals. For example, A(a,t1,t2)=T requires an 'a' action to occur in the time interval [t1,t2] while A(a,t1,t2)=F requires that an 'a' action should not occur in the interval. An absolute constraint (represented by A) is an element of $\Lambda \times \text{Time} \times \text{Time} \rightarrow \{\text{T}, \text{F}\}$. Relative constraints are expressed in terms of what actions must or must not happen in time intervals after a particular action has occurred. For example, R(a,b,t1,t2) = T requires a 'b' action to occur after t1 time units and within t2 time units after 'a' has occurred viz., t1 and t2 are to be measured after the occurrence of 'a'. Similarly, R(a,b,t1,t2)=F requires that a 'b' action should not occur in the duration interval [t1,t2] after 'a' has occurred. A relative constraint (represented by R) is an element of $\Lambda \times \Lambda \times \text{Time} \times \text{Time} \rightarrow \{\text{T}, \text{F}\}$.

Example 1 *Consider the following process definitions:* $A = (1)a + (2)a + (3)a + \ldots + (10)a$. $B = (2)b + (4)b + (6)b + (8)b + (10)b$. *Let the constraint on the process* $(A \mid B)$ *be:* $R(a, b, 0, 5) = T$. *If an execution of* $(A \mid B)$ *is to satisfy the temporal constraint given one professing element, it cannot select (2)a and (10)b or (10)a and (1)b. The first selection violates the quantitative requirement, while the second violates ordering. However it can choose, (2)a and (4)b or (4)a and (8)b etc. So the available nondeterminism is restricted by the temporal constraints.*

Example 2 *Periodic tasks which are common in real-time systems can be specified as:* $PT = a;\ B; b;\ PT$, *where a and b indicate the start and finish of the periodic task of periodicity P and B a purely sequential process (sequence of atomic actions). The constraints on it are* $\forall\ i$ *in* $0\ ..\ :$ $A(a, i^*P, (i+1)^*P) = T$ *which requires the task to start in the assigned period and* $\forall\ i$ *in* $0\ ..\ :\ A(b, i^*P, (i+1)^*P) = T$ *which requires the task to finish appropriately.*

Example 3 *While a periodic task might be interrupted by a scheduler, (i.e., interleaved with other processes) there may be certain parts of the task which once started should finish quickly (such as reading a sensor which has continuous input). This can be specied as PT = a; B1; s; B2; f; B3; b; PT with a timing constraint* $R(s, f, 0, \epsilon) = T$, *where* ϵ *is the maximum permissible delay.*

Example 4 *Jitter control is also an important aspect in real-time systems and can be specified as: PT = a; B1; s; B2; b; PT with the constraint*

$R(s, s, (P - \epsilon_1), (P + \epsilon_2)) = T$, *where* ϵ_1 *and* ϵ_2 *represent the permissible jitter.*

3 Semantics

Given the set of basic actions (Λ), define a set of actions $Act = \Lambda \cup \{\tau, \delta\}$, where τ represents synchronization and δ idling (or stepping of time without executing an action). Let $\mathcal{P}$ be the set of all processes

Definition: 1 *Define a transition relation* $\longrightarrow$ *as the smallest sub-set of* $(\mathcal{P} \times Act \times \mathcal{P})$, *which satisfies the rules in figure 1.*

We do not define an operational semantics for the timing constraints; rather we define relations induced by them. The semantics of processes is almost identical to the usual semantics of timed CCS. The only difference is that we take a different view regarding the passing of time than [6]. As a waiting process does not need the processor, one can deduct time from it when executing other processes. Towards that we define a function called *Translate* as follows:

$$Translate(P) = \begin{cases} (t-1)Q & if\ P\ is\ (t)Q \\ Translate(P1) + Translate(P2) & if\ P\ is\ P1 + P2 \\ Translate(P1) \mid Translate(P2) & if\ P is\ P1 \mid P2 \\ Translate(Q) \setminus H & if\ P\ is\ Q \setminus H \\ rec\ \tilde{X} : Translate(\tilde{E}) & if\ P\ is\ rec\tilde{X} : \tilde{E} \\ P & otherwise \end{cases}$$

Example 5 *Notice that Translate forces the elapsing of time across choice. Our intuition behind the semantics can be interpreted as that the execution of an action can wait as long as the resource is not allocated. But delaying does not require the processor. Thus,* $a \mid ((1)b + c)$ *after exhibiting a can exhibit* b *as the next action.*

An informal explanation of the transition rules follows. a;P performs 'a' and then behaves as P. Though not shown in the operational rules, this takes unit time. An execution of a step is treated as one clock tick. Hence if a process has to wait for time t, after a step it has to wait for only $(t-1)$; but the processor was unused during that step. The rule for non-determinism

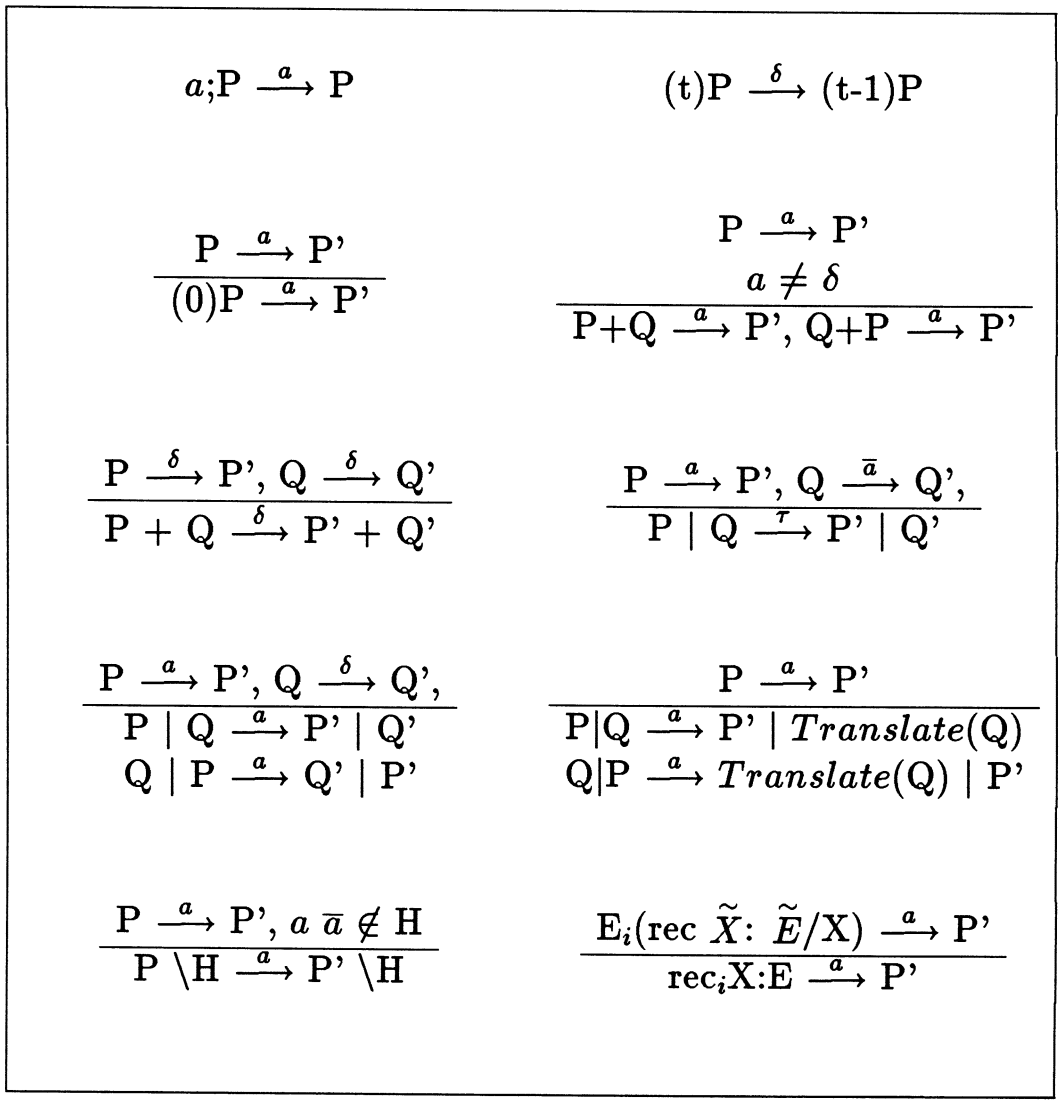

Figure 1: Operational Semantics

exhibiting an action is as usual. However, if both branches of the choice can delay, the choice can be delayed. The rules that determine the behavior under parallel composition are as follows. The first two require the behaviors of processes P and Q to either be synchronizable or at least one of them is idling for the parallel composition to be successful. The third interleaves the execution. The rules for hiding and recursion are as usual.

4 Simulations

In this section we define simulations induced by temporal constraints. [6] show that strong bisimulation of a timed process (defined as: $P \sim Q$ iff $P \xrightarrow{a}$ then $\exists\ Q'$ such that $Q \xrightarrow{a}$ and $P' \sim Q'$ and if $P \overset{t}{\rightsquigarrow} P'$ then $\exists\ Q'$ such that $Q \overset{t}{\rightsquigarrow} Q'$ and $P' \sim Q'$ and vice versa) is a congruence. Thus, the definition requires identical timing of actions for two terms to be bisimilar.

Our concern here is to define a simulation which is more flexible hence which is substitutive in restricted contexts viz., the contexts defined by timing constraints. For example, if a takes 10 units of time in P and a takes 12 units of time in Q, and the constraint is that a occurs no later than 15 units, P can be replaced by Q and vice-versa. However this is not the case if the requirement is that a occurs no later than 11 units of time.

Clearly such a definition cannot be a congruence. What is the use of such a definition? We believe that our work will be relevant in two areas: 1) Reasoning about implementations satisfying specifications and 2) Fault-tolerant real-time systems.

The application of approximations to discuss implementations satisfying specifications is well understood. For example, it is required that any behavior exhibited by the implementation is permitted by the specification, but every behavior permitted by the specification need not be exhibited by the implementation. Real-time systems which have to be fault-tolerant clearly have to satisfy temporal constraints. Also, as they have to be fault tolerant, the occurrence of a fault requires one to 'replace' all the affected processes by 'equivalent' ones. Thus the equivalence need only be defined within the given system and the equivalence need not hold in general.

What are the possible scenarios? Consider two processes P and Q which are to satisfy the timing constraint A(a,0,n)=T. Let P perform an action

different from 'a', and evolve to P', while Q be forced to delay for time t and become Q'. Q' should have the option of performing the same action as P and evolve to Q". How should P' and Q" be related? Firstly they must be observationally related. As P and Q should be related if and only if both satisfy the given constraint the same should hold for P' and Q". However as Q consumed time t units of time while P consumed only 1 unit of time, Q" should be able to produce an 'a' action in time (n−t) while P' should produce an 'a' action within time (n − 1). So P' and Q" are no longer related by a single timing constraint. They are related by two constraints which differ only in the timing aspect of the constraint and not the observation part.

To accommodate the fact that one needs (bi)similarity under a pair of timing constraints, we define (bi)similarity as induced by a pair of timing constraints. In the next few paragraphs we introduce a definition of simulation indexed by a pair of constraints (indicated by $\stackrel{\sqsubseteq}{\sim}_{C1,C2}$). If $\stackrel{\sqsubseteq}{\sim}_{C1,C2}$ is a symmetric relation we write $\sim_{C1,C2}$. For notational convenience we write P $\stackrel{\sqsubseteq}{\sim}_C$ Q, iff P $\stackrel{\sqsubseteq}{\sim}_{C,C}$ Q and P $\sim_C$ Q iff P $\sim_{C,C}$ Q. These definitions use observational simulation which is defined as follows.

Definition: 2 *Define* $P \stackrel{t}{\rightsquigarrow} Q$ *if there is a sequence of transitions rules such that* $P \stackrel{\delta}{\longrightarrow} P_1 \ldots P_{t-1} \stackrel{\delta}{\longrightarrow} Q$. *If* t *is* 0, *then* P *is identical to* Q.

Definition: 3 *Define* $P \stackrel{\sqsubseteq}{\sim} Q$ *(i.e., P observationally simulates Q) iff* $P \stackrel{t_1}{\rightsquigarrow} P' \stackrel{a}{\longrightarrow} P'' \stackrel{t_2}{\rightsquigarrow} P_1$ *then* $\exists\, Q_1$ *such that* $Q \stackrel{t_3}{\rightsquigarrow} Q' \stackrel{a}{\longrightarrow} Q'' \stackrel{t_4}{\rightsquigarrow} Q_1$.

Definitions 4, 5, 6 and 7 define the four types of equivalences induced by the four types of timing constraints.

Definition: 4 $P \stackrel{\sqsubseteq}{\sim}_{A(a,t1,t2)=T,A(a,t3,t4)=T} Q$ *iff*

1. *If* $P \stackrel{tp}{\rightsquigarrow} P' \stackrel{b}{\longrightarrow} P''$ *and* $tp \leq (t1-1)$ $\exists\, Q \stackrel{tq}{\rightsquigarrow} Q' \stackrel{b}{\longrightarrow} Q''$ *and* $tq \leq (t3-1)$ *and* $P'' \stackrel{\sqsubseteq}{\sim}_{A(a,x1,x2)=T,A(a,y1,y2)=T} Q''$ *where* $x1 = t1 - tp - 1, x2 = t2 - tp - 1, y1 = t3 - tq - 1, y2 = t4 - tq - 1$

2. *If* $P \stackrel{tp}{\rightsquigarrow} P' \stackrel{a}{\longrightarrow} P''$ *and* $(t1 - 1) \leq tp \leq (t2 - 1)$ $\exists\, Q \stackrel{tq}{\rightsquigarrow} Q' \stackrel{a}{\longrightarrow} Q''$ *and* $(t3 - 1) \leq tq \leq (t4 - 1)$ *and* $P'' \stackrel{\sqsubseteq}{\sim} Q''$

3. *If* $P \stackrel{tp}{\rightsquigarrow} P' \stackrel{b}{\longrightarrow} P''$ *and* $a \neq b$ *and* $(t1 - 1) \leq tp \leq (t2 - 1)$ $\exists\, Q \stackrel{tq}{\rightsquigarrow} Q' \stackrel{b}{\longrightarrow} Q''$ *and* $(t3 - 1) \leq tq \leq (t4 - 1)$ *and*

$P'' \stackrel{\sqsubseteq}{\sim}_{A(a,x1,x2)=T,A(a,y1,y2)=T} Q''$ *where* $x1 = 0$, $x2 = t2 - tp - 1$, $y1 = 0$, $y2 = t4 - tq - 1$

The first rule relates the behavior before the interval while the second and third relates relevant behavior within the interval. Once the required action is observed then processes are only required to be 'observationally' related. Note that only processes which 'satisfy' the condition are related and thus P $\stackrel{\sqsubseteq}{\not\sim}_{C(0,0),C(0,0)}$ Q as each action takes at least unit time.

Example 6 *Consider the two processes (10)a and (1)a + (3)a. They are bisimilar under the constraint A(a,0,11)=T; as both of them will always satisfy the turning constraint. Thus (10) a* $\sim_{A(a,0,11)=T}$*, (1)a + (3)a. Similarly, (4)a* $\stackrel{\sqsubseteq}{\sim}_{A(a,3,9)=T}$ *(1)a + (6)a.*

Proposition 1 $P \sim_{A(a,t1,t2)=T} Q$ *implies 1)* $P \sim_{A(a,t1-1,t2)=T} Q$ *and 2)* $P \sim_{A(a,t1,t2+1)=T} Q$

Example 7 *The above propositions are intuitive but would not be valid if P and Q were related if both violated the constraint as shown by the following example. (10)a* $\sim_{A(a,4,6)=T}$ *(2)a would be true if processes which violated the requirement were related. However, A(a,3,6) would be satisfied by (2)a but not (10)a and hence no longer bisimilar. Similarly (1)a would be bisimilar to (2)a under the above constraint but not under A(a,3,6). If processes which violate the condition have to be related one has to define similarity between the type and magnitude of error magnitude of error and is not considered here.*

Definition: 5 $P \stackrel{\sqsubseteq}{\sim}_{A(a,t1,t2)=F,A(a,t3,t4)=F} Q$ *iff*

1. $P \stackrel{\sqsubseteq}{\sim} Q$ *implies* $\stackrel{\sqsubseteq}{\sim}_{A(a,0,0)=F,A(a,0,0)=F} Q$
2. *If* $P \stackrel{tp}{\rightsquigarrow} P' \stackrel{b}{\longrightarrow} P''$ $tp \leq (t1 - 1)$ $\exists$ $Q \stackrel{tq}{\rightsquigarrow} Q' \stackrel{b}{\longrightarrow} Q''$ *and* $tq \leq (t3 - 1)$ *and* $P'' \stackrel{\sqsubseteq}{\sim}_{A(a,x1,x2)=F,A(a,y1,y2)=F} Q''$ *where* $x1 = t1 - tp - 1, x2 = t2 - tp - 1, y1 = t3 - tq - 1, y2 = t4 - tq - 1$
3. $P \stackrel{tp}{\rightsquigarrow} P' \stackrel{b}{\longrightarrow} P''$ *and* $tp \geq t2$ *then* $\exists$ $Q \stackrel{tq}{\rightsquigarrow} Q' \stackrel{b}{\longrightarrow} PJ''$ *and* $tq \geq t4$ *and* $P'' \stackrel{\sqsubseteq}{\sim} Q''$

4. $P \overset{tp}{\rightsquigarrow} P' \xrightarrow{b} P''$ *and* $a \neq b$ *and* $(t1-1) \leq tp \leq (t2-1)$ *then* $\exists\ Q \overset{tq}{\rightsquigarrow} Q' \xrightarrow{b} Q''$ *and* $(t3-1) \leq tq \leq (t4-1)$ *and* $P'' \overset{\sqsubset}{\sim}_{A(a,x1,x2)=F,A(a,y2,y2)=F} Q''$ *where* $x1 = 0, x2 = t2 - tp - 1, y1 = 0, y2 = t4 - tq - 1$

The first rule relates processes with are observationally related. As all actions take at least unit time, no action can occur in the interval $[0, 0]$. The second and third ryes relate processes outside the interval, while the final rule requires that the desired action not occur in the interval.

Proposition 2 $P \sim_{A(a,t1,t2)=F} Q$ *implies 1)* $P \sim_{A(a,t1+1,t2)=F} Q$ *and 2)* $P \sim_{A(a,t1,t2-1)=F} Q$.

Example 8 *It is easy to check that (1)a + (2)a + (10)a* $\sim_{A(a,5,9)=F}$ *(3)a + (12)a. Note that (2)a is not bisimilar under the above constraint to (12)a, as the definition requires that behavior before and aver the range be similar.*

This finishes the definitions for the absolute case. Now we turn our attention to the relative case.

Definition: 6 $P \overset{\sqsubset}{\sim}_{R(a,b,t1,t2)=T,R(a,b,t3,t4)=T} Q$ *iff*

1. $P \overset{tp}{\rightsquigarrow} P' \xrightarrow{a} P'' \exists\ Q \overset{tq}{\rightsquigarrow} Q' \xrightarrow{a} Q''$ *and* $P'' \overset{\sqsubset}{\sim}_{NC} Q''$ *and* $P'' \overset{\sqsubset}{\sim}_{R(a,b,t1,t2)=T,R(a,b,t3,t4)=T} Q''$ *where NC is* $A(b, t1, t2) = T, A(b, t3, t4) = T$

2. $P \overset{tp}{\rightsquigarrow} P' \xrightarrow{c} P''$ *and* $a \neq c\ \exists\ Q \overset{tq}{\rightsquigarrow} Q' \xrightarrow{c} Q''$ *and* $P'' \overset{\sqsubset}{\sim}_{R(a,b,t1,t2)=T,R(a,b,t3,t4)=T} Q''$

The first condition starts the clock when an a occurs. The processes after the result of an 'a' action are required to be similar under two conditions. The first condition (NC) is that b occurs within the specified time woe the second requires that the relative constraint be satisfied in future. The second condition in the definition ensures conditional similarity in the future if an 'a' action was not performed.

Proposition 3 $P \sim_{R(a,b,t1,t2)=T} Q$ *implies 1)* $P \sim_{R(a,b,t1-1,t2)=T} Q$ *and 2)* $P \sim_{R(a,b,t1,t2+1)=T} Q$

Definition: 7 $P \overset{\sqsubset}{\sim}_{R(a,b,t1,t2)=F,R(a,b,t3,t4)=F} Q$ *iff*

1. $P \overset{tp}{\rightsquigarrow} P' \xrightarrow{a} P'' \; \exists \; Q \overset{tq}{\rightsquigarrow} Q' \xrightarrow{a} Q''$ *and* $P'' \overset{\sqsubseteq}{\sim}_{NC} Q''$ *and* $P'' \overset{\sqsubseteq}{\sim}_{R(a,b,t1,t2)=F,R(a,b,t3,t4)=F} Q''$ *where* NC *is* $A(b,t1,t2) = F, A(b,t3,t4) = F$

2. $P \overset{tp}{\rightsquigarrow} P' \xrightarrow{c} P''$ *and* $a \neq c$ *then* $\exists \; Q \overset{tq}{\rightsquigarrow} Q' \xrightarrow{c} Q''$ *and* $P'' \overset{\sqsubseteq}{\sim}_{R(a,b,t1,t2)=F,R(a,b,t3,t4)=F} Q''$

As in the previous definition, the first condition starts the clock when an 'a' occurs (but now disallowing 'b'), while the second ensures conditional similarity in the future.

Proposition 4 $P \sim_{R(a,b,t1,t2)=F} Q$ *implies 1)* $P \sim_{R(a,b,t1+1,t2)=F} Q$ *and 2)* $P \sim_{R(a,b,t1,t2-1)=F} Q$

Proposition 5 *Though* $\sim_C$ *is not a congruence, the following hold.*

- *if* $P \overset{\sqsubseteq}{\sim}_{C(t1,t2)} Q$*, where* $C(t1,t2)$ *is* $A(a,t1,t2) = T$ *then*
  1) $(t)P \overset{\sqsubseteq}{\sim}_{C(t1+t,t2+t)} (t)Q$ *2)* $b;P \overset{\sqsubseteq}{\sim}_{C(t1+1,t2+1)} b;Q$
  3) $P + R \overset{\sqsubseteq}{\sim}_{C(t1,t2)} Q + R$ *4)* $P \mid R \overset{\sqsubseteq}{\sim}_{C(t1,t2)} Q \mid R$

- *if* $P \overset{\sqsubseteq}{\sim}_{C(t1,t2)} Q$*, where* $C(t1,t2)$ *is* $A(a,t1,t2) = F$ *then*
  1) $(t)P \overset{\sqsubseteq}{\sim}_{C(t1+t,t2+t)} (t)Q$ *2)* $b;P \overset{\sqsubseteq}{\sim}_{C(t1+1,t2+1)} b;Q$
  3) $P + R \overset{\sqsubseteq}{\sim}_{C(t1,t2)} Q + R$ *4)* $P \mid R \overset{\sqsubseteq}{\sim}_{C(t1,t2)} Q \mid R$

- *if* $P \overset{\sqsubseteq}{\sim}_{C(t1,t2)} Q$*, where* $C(t1,t2)$ *can either be* $R(a,t1,t2) = T$ *or* $R(a,b,t1,t2) = F$ *then*
  1) $(t)P \overset{\sqsubseteq}{\sim}_{C(t1+t,t2)} (t)Q$ *2)* $c;P \overset{\sqsubseteq}{\sim}_{C(t1,t2)} c;Q$ *and* $c \neq a$
  3) $P + R \overset{\sqsubseteq}{\sim}_{C(t1,t2)} Q + R$ *4)* $P \mid R \overset{\sqsubseteq}{\sim}_{C(t1,t2)} Q \mid R$

5 Logic

In this section, we introduce an extension of the Hennessy-Milner logic with recursion [3] to handle time. The formulae in our logic are defined as follows

$$L =< a > L \mid \bigwedge_{i \in I} L_i \mid \neg L \mid [t]L \mid \{t\}L \mid X \mid \nu X : L$$

True is defined to be the conjunction over an empty index set, while Terminated is defined to be the conjunction over the set of actions of ($\neg < a >$ True).

The formulae are interpreted on processes and is defined as follows. Let $\mathcal{P}$ represent the set of all processes.

$[\![< a > \mathrm{L}]\!] = \{\mathrm{P} \in \mathcal{P} \ \exists \ \mathrm{P}',\ \mathrm{P} \xrightarrow{a} \mathrm{P}' \text{ and } \mathrm{P}' \in [\![\mathrm{L}]\!]\}$
$[\![\bigwedge_{i \in I} \mathrm{L}_i]\!] = \bigcap_{i \in I} [\![\mathrm{L}_i]\!]$
$[\![\neg \mathrm{L}]\!] = \mathcal{P} - [\![\mathrm{L}]\!]$
$[\![[\mathrm{t}]\mathrm{L}]\!] = \{\mathrm{P} \in \mathcal{P} \ \exists \ \mathrm{Q} : \mathrm{P} \xrightarrow{a} \mathrm{P}_1 \ldots \mathrm{P}_{t-1} \xrightarrow{a_t} \mathrm{Q} \text{ and } \mathrm{Q} \in [\![\mathrm{L}]\!]\}$
$[\![\{\mathrm{t}\}\mathrm{L}]\!] = \{\mathrm{P} \in \mathcal{P} \ \exists \ \mathrm{Q} : \text{ with } \mathrm{n} \leq \mathrm{t},\ \mathrm{P} \xrightarrow{a_1} \mathrm{P}_1 \ldots \mathrm{P}_{n-1} \xrightarrow{a_n} \mathrm{Q} \text{ and } \mathrm{Q} \in [\![\mathrm{L}]\!]\}$
$[\![\nu \mathrm{X} : \mathrm{L}]\!] = \cup\{\mathrm{Pr} \subseteq \mathcal{P} \text{ such that } [\![L]\!] \supseteq \mathrm{Pr}\}$

The intuition in using [t] and {t} is that the first characterizes the passing of exactly t units of time woe {t} characterizes behavior with in the interval $[0, t]$. The other components have their usual meaning.

The temporal constraints that we have used can be translated into the above logic as follows.

$\mathrm{A}(\mathrm{a}, \mathrm{t1}+1, \mathrm{t2}+1) = \mathrm{T} = [\mathrm{t1}]\{\mathrm{t2}\} < \mathrm{a} > \ \text{True}$

$\mathrm{A}(\mathrm{a}, \mathrm{t1}+1, \mathrm{t2}+1) = \mathrm{F} = \neg([\mathrm{t1}]\{\mathrm{t2}\} < \mathrm{a} > \ \text{True})$

$\mathrm{R}(\mathrm{a}, \mathrm{b}, \mathrm{t1}+1, \mathrm{t2}+1) = \mathrm{T} = \nu \mathrm{C} : (\ (< \mathrm{a} > \ \text{True} \rightarrow [\mathrm{t1}]\{\mathrm{t2}\} < \mathrm{b} > \ \text{True})) \wedge$
(Terminated $\vee$ [1]C)

$\mathrm{R}(\mathrm{a}, \mathrm{b}, \mathrm{t1}+1, \mathrm{t2}+1) = \mathrm{F} = \nu \mathrm{C} : (\neg(< \mathrm{a} > \ \text{True} \wedge [\mathrm{t1}]\{\mathrm{t2}\} < \mathrm{b} > \ \text{True})) \wedge$
(Terminated $\vee$ [1]C)

As expected, $\mathrm{R}(\mathrm{a}, \mathrm{b}, \mathrm{t1}, \mathrm{t2}) = \mathrm{T/F}$ represent safety properties and map to a maximal fixed point formula.

Proposition 6 *$P \sim_C Q$ implies that $P \in [\![L]\!]$ and $Q \in [\![L]\!]$ where L is the translation of C.*

Proposition 7 *$(\forall L, (P \models L)$ iff $(Q \models L)\ )$ implies $P \sim_{st} Q$. where $\sim_{st}$ represents strong timed bisimulation.*

6 System Issues

The above definitions can be used to develop a system for reasoning about timed processes. We outline how the concepts of schedulers can be formalized. Note that we do not present a language in which schedulers can be defined.

Definition: 8 *A scheduler, Sch, is a function which given a process P, yield a process such that $Sch(P) \stackrel{\sqsubseteq}{\sim} P$.*

Schedulers as defined above, can be considered to be implementors of a specification [7]. The general problem of optimal scheduling is usually NP-complete and hence very rarely used. Usually, one either uses a scheduler which is 'satisfactory' or requires more information from the process. Thus, we do *not require* an implementation to satisfy the timing constraints specified for a system. However, the notion of equivalence is considered only for schedulers which can satisfy a given constraint, i.e., as there are many ways of implementing a specification, it is natural to identify similar satisfactory ones.

Definition: 9 *Two schedulers S1 and S2 are defined to be similar with respect to a process P and timing constraint C written as ($S1 \sim_{P,C} S2$), iff $S1(P) \sim_C S2(P)$.*

Definition: 10 *Process P and Q are similar under scheduler Sch and a given constraint C written as ($P \sim_{Sch,C} Q$), iff $Sch(P) \sim_C Sch(Q)$.*

Example 9 *$a \mid b$ is not similar to $b \mid a$ under $A(a, 0, 1) = T$ and a 'FCFS' scheduler defined as follows scheduler($p \mid q$) = schedule(p);schedule(q) and schedule(a) = a.*

Proposition 8 *Given a timing constraint C. If $S1 \sim_{P,C} S2$ and $P \sim_{S1,C} Q$ then $S1 \sim_{Q,C} S2$ iff $P \sim_{S2,C} Q$.*

We extend the above definitions to capture the process of translating a program in a high level language into a more low level language in the context of developing a real-time system. We consider a compilation as converting a program in a some language into a process in RTCCS. We assume that this process of compilation has knowledge of the architecture and hence assigns

times to each action, i.e., prefixes each action by (n) for some n. It could assign a range by using the non-deterministic operator. Thus, a compiler might assign times ranging from 5 to 10 for an action a, while another might assign times ranging from 20 to 30.

Given a program (with timing constraints), and a scheduler one can define 'equivalence' of compilations as follows.

Definition: 11 *Given a program Pgm, a timing constraint C and a scheduler S. Compiler1* $\sim_{C,Pgm,S}$ *Compiler2 iff S(Compiler1 (Pgm))* $\sim_C$ *S(Compiler2 (Pgm))*

Proposition 9 *Given a program Pgm and timing constraint C and schedulers S1 and S2. If Cmp1* $\sim_{C,Pgm,S1}$ *Cmp2 and S1* $\sim_{Cmp1(Pgm),C}$ *S2 then Cmp1* $\sim_{C,Pgm,S2}$ *Cmp2 iff S1* $\sim -Cmp2(P), C$ *S2.*

Acknowledgment

The author is grateful to Uffe Engberg and Peter Mosses for their com-ments and encouragement.

References

[1] F. Jahanian and A. K. Mok. A graph-theoretic approach for timing analysis and its implementation. *IEEE Transactions on Computers*, pages 961–975, August 1987.

[2] M. Joseph and A. Goswami. What's 'Real' about real-time systems? In *IEEE Real-Time Systems Symposium*, pages 78–85, 1988.

[3] K. G. Larsen. Proof systems for Hennessy-Milner logic with recursion. In *13th Colloquim on Trees in Algebra and Programming.* Springer Verlag, 1988.

[4] R. Milner. *A Calculus of Communicating Systems.* Lecture Notes on Computer Science Vol. 92. Springer Verlag, 1980.

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

[6] F. Moller and C. Tofts. A temporal calculus of communicating systems. In J. C. M. Baeten and J. W. Klop, editors, *CONCUR 90, LNCS-458.* Springer Verlag, 1990.

[7] E. R. Olderog and C. A. R. Hoare. Specification-oriented semantics for communicating processes. In *ICALP -83, LNCS 154.* Springer Verlag, 1983.

[8] D. Park. Concurrency and automata on infinite sequences. In *Proceedings of the 5th GI Conference, LNCS-104.* Springer Verlag, 1981.

[9] W. Yi. Real-time behavior of asynchronous agents. In J. C. M. Baeten and J. W. Klop, editors, *CONCUR 90, LNCS-458.* Springer Verlag, 1990.