November 1988

A Complete Axiomatization of Real-Time Processes

Richard Gerber
University of Pennsylvania

Insup Lee
University of Pennsylvania, lee@cis.upenn.edu

Amy E. Zwarico
Johns Hopkins University

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-88-88.

A Complete Axiomatization of Real-Time Processes

Abstract

Once strictly the province of assembly-language programmers, real-time computing has developed into an area of important theoretical interest. Real-time computing incorporates all of the theoretical problems encountered in concurrent processing and introduces the additional complexity of accounting for the temporal behavior of processes. In this paper we investigate two problems in the theory of real-time processes: defining realistic semantic models and developing proof systems for real-time processes. We present here a semantic domain for real-time processes that captures the temporal constraints of concurrent programs. A partial ordering based on process containment is defined and shown to be a complete partial order on the domain. The domain is used to define the denotational semantics of a CSP-like language that incorporates pure time delay. An axiomatization of process containment is presented and shown to be complete for finite terms in this language. The axiomatization is useful for proving properties of real-time processes and deriving their temporal behavior.

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-88-88.

A Complete Axiomatization Of Real-Time Processes

MS-CIS-88-88
GRASP LAB 162

A. Zwarico
R. Gerber
I. Lee

Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104

November 1988

ACKNOWLEDGEMENTS:
This research was supported in part by NSF grants IRI86-10617, DCR 8501482, DMC 8512838, MCS 82119196-CER, U.S. Army grants DAA29-84-K-0061, DAA29-84-9-0027 and a grant from AT&T's Telecommunications Program at the University of Pennsylvania.

A Complete Axiomatization of Real-Time Processes *

R. Gerber and I. Lee
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104

A. Zwarico
Computer Science Department
The Johns Hopkins University
Baltimore, Md 21218

November 9, 1988

Abstract

Once strictly the province of assembly-language programmers, real-time computing has developed into an area of important theoretical interest. Real-time computing incorporates all of the theoretical problems encountered in concurrent processing and introduces the additional complexity of accounting for the temporal behavior of processes. In this paper we investigate two problems in the theory of real-time processes: defining realistic semantic models and developing proof systems for real-time processes. We present here a semantic domain for real-time processes that captures the temporal constraints of concurrent programs. A partial ordering based on process containment is defined and shown to be a complete partial order on the domain. The domain is used to define the denotational semantics of a CSP-like language that incorporates pure time delay. An axiomatization of process containment is presented and shown to be complete for finite terms in this language. The axiomatization is useful for proving properties of real-time processes and deriving their temporal behavior.

1 Introduction

In recent years there have been several significant attempts toward defining formal semantic models for real-time computing. Once strictly the province of assembly-language programmers, real-time computing has developed into an important theoretical interest. Furthermore, it is becoming generally recognized that the semantic models used for untimed computing cannot be trivially modified to incorporate the notion of time. This is especially true when one attempts to describe the semantics of time-dependent concurrency.

In this paper we present the *Timed Acceptances Model*, which captures the temporal constraints of concurrent programs. The model consists of 1) an abstract, CSP-based language, 2) a partially ordered semantic domain, and 3) a complete axiom system, enabling us to prove critical correctness properties of real-time programs.

*This research was supported in part by NSF DCR 8501482, NSF DMC 8512838, NSF MCS 8219196-CER, ARO DAA6-29-84-k-0061, and a grant from AT&T's Telecommunications Program at the University of Pennsylvania.

Other contributions in this area range from the abstract requirements languages such as RTL [9, 8], to the variable-state based models such as CSP-R [10] and DNP-R [7], to the purely operational models of SCCS [12] and CIRCAL [11]. Our model occupies the middle ground, in that it retains the *structure* of concurrent processes, but solely captures the timing information retained within each process. To this extent, its expressibility is most similar to that found in TCSP [13] and ECP [2]. However, while all three models share a CSP-like syntax, our semantic domain is not based on the notion of *refusals*. Instead, we represent nondeterminism using a temporal extension of Hennessy's Acceptance Tree Model [4]. Most importantly, our model incorporates a complete axiom system, which should provide the foundation for automated analysis of real-time systems.

2 The Language of Real-Time Processes

The syntax of our language is similar to that of untimed CSP [6], with one major exception: The traditional prefix operator, $a \rightarrow P$, has been replaced by the *timed action* operator $A \overset{i}{\rightsquigarrow} P$. In our model *timed action* incorporates the notion of true concurrency, and it allows the specification of pure time delay. As in the untimed CSP, processes communicate by synchronously engaging in *events*, which are considered instantaneous. At any time during its execution, a process may simultaneously engage in a set of such events, after which it delays for some nonzero period of time. The terms of the language are defined as follows:

$$P ::= \text{CHAOS} \mid \text{STOP} \mid \text{SKIP} \mid A \overset{i}{\rightsquigarrow} P \mid P \,\Box\, P \mid P \sqcap P \mid P \|_A P \mid P \backslash A \mid \mu P.F(P)$$

Here, events are members of a finite alphabet Σ, and are denoted a, b, and c. Similarly, the letters A, B and C range over subsets of Σ, while i, j and k range over the natural numbers. The letters P, Q and R range over terms in our language, while F also ranges over terms, but possibly contains a free term P.

While the syntax of the language is similar to its untimed counterpart, the semantics of each term incorporates an additional factor, the passage of time. CHAOS represents the most nondeterministic process. At any moment, CHAOS may execute an arbitrary subset of events from Σ. STOP represents abnormal termination (deadlock) or divergence. Since these two conditions cannot be distinguished experimentally, we let a single process describe them both. Once executed STOP cannot share any event with its environment. On the other hand, SKIP is the process that terminates correctly.

The timed action operator, $A \overset{i}{\rightsquigarrow} P$, is defined for an event set A and a natural number $i > 0$. At time 0 the events in A are simultaneously executed, and after a delay of exactly i time units the process P is executed. If $A = \emptyset$, the execution of P is delayed by exactly i time units. Choice, $P \,\Box\, Q$, differs from its untimed counterpart in that it also represents an external decision made on the occurrence time of an event. For example, let

$$P = (\emptyset \overset{1}{\rightsquigarrow} \{a\} \overset{1}{\rightsquigarrow} \text{STOP}) \,\Box\, (\emptyset \overset{2}{\rightsquigarrow} \{b, c\} \overset{1}{\rightsquigarrow} \text{STOP}).$$

P *must* accept either a at time 1 or both b and c at time 2. If a is offered at time 2, P will not accept it. On the other hand, the nondeterministic process $P \sqcap Q$, internally "decides" at time 0 to behave like P or Q. Consider the process P above, with the "$\Box$" operator replaced by "$\sqcap$". Here, the a-event *may* be accepted at time 1, or b and c *may* be accepted at time 2. However, the decision on which choices to offer is made internally by the process.

The parallel operator, or $P \parallel_A Q$, captures the semantics of concurrency by combining both the delay times and event executions of the two processes. The set A denotes the events on which both P and Q must synchronize, while the processes may execute events in $\Sigma - A$ independently. Consider the following processes:

$$P = (\emptyset \overset{1}{\rightsquigarrow} \{a\} \overset{1}{\rightsquigarrow} \{c\} \overset{1}{\rightsquigarrow} \text{SKIP}) \,\Box\, (\emptyset \overset{4}{\rightsquigarrow} \{b\} \overset{5}{\rightsquigarrow} \{d\} \overset{1}{\rightsquigarrow} \text{SKIP}), \quad Q = \emptyset \overset{1}{\rightsquigarrow} \{a\} \overset{8}{\rightsquigarrow} \text{SKIP}.$$

The process $P \parallel_{\{a,d\}} Q$ is equivalent to $(\emptyset \overset{1}{\rightsquigarrow} \{a\} \overset{1}{\rightsquigarrow} \{c\} \overset{7}{\rightsquigarrow} \text{SKIP}) \,\Box\, (\emptyset \overset{4}{\rightsquigarrow} \{b\} \overset{5}{\rightsquigarrow} \text{STOP})$.

Examining the left-hand alternative, we see that both processes can synchronize on a at time 1. If this choice is taken, then P will execute its local event b one time unit later. Since in this case both processes successfully terminate, the combined termination time is the maximum of the two processes' individual termination times. The right-hand alternative shows that P may execute b at time 4; here synchronization with Q is not necessary. However, since Q *must* synchronize on a at time 1, this step leaves Q deadlocked. The deadlock finally infects P at time 9, when synchronization on the event d becomes necessary. Unlike the parallel composition described in several other models [12, 11, 10, 7], this parallel operator permits n-way synchronization between processes.

Concealment, or $P\backslash A$, hides the process' execution of events in A from the environment. However, concealment does not affect the possible times that the events in $\Sigma - A$ may occur. Thus, the operator may introduce nondeterminism into a process where none previously existed. For example,

$$\begin{aligned}&((\emptyset \overset{1}{\rightsquigarrow} \{a\} \overset{2}{\rightsquigarrow} \{b\} \overset{1}{\rightsquigarrow} \text{STOP}) \,\Box\, (\emptyset \overset{2}{\rightsquigarrow} \{a\} \overset{2}{\rightsquigarrow} \{c\} \overset{1}{\rightsquigarrow} \text{STOP}))\backslash\{a\} \\ &\quad = ((\emptyset \overset{3}{\rightsquigarrow} \{b\} \overset{1}{\rightsquigarrow} \text{STOP}) \sqcap (\emptyset \overset{4}{\rightsquigarrow} \{c\} \overset{1}{\rightsquigarrow} \text{STOP})).\end{aligned}$$

Finally, the recursion operator, or $\mu P.F(P)$, is used to specify infinite processes.

3 The Semantic Model

The semantic domain is based on two well-known untimed paradigms: Hoare's Trace Algebra [6] and Hennessy's Acceptance Tree model [4]. In our model, the trace algebra is temporally extended to depict the observable execution sequences of real-time processes. However, traces alone are unable to capture the meaning of nondeterminism; thus we include a set of states to accompany each trace. This state set denotes the array of choices – both internal and external – that a process may make after executing the trace. The

representation for the state set is similar to the acceptance tree model of nondeterminism, although it explicitly incorporates the notion of time. Here we depart from other CSP-based models [2, 13], which capture nondeterminism though the use of *refusals*. We have found it considerably more natural to specify those events a process *can* execute after a trace, instead of those that it *cannot* execute.

In our model a process is defined by all of its possible traces, and the state sets corresponding to those traces. Our semantic domain forms a complete partial order, where $P \sqsubseteq Q$ if Q is a deterministic refinement of P. All of our operators are monotonic with respect to the partial order; except for hiding, they are also continuous.

3.1 Primitives

Time Domain. Our time domain is discrete. It is represeted by the nonnegative integers $\boldsymbol{N}$, augmented with "∞" to denote infinite time intervals. The domain is designated as $\boldsymbol{N}^{\infty}$.

Events. As noted above, events are instantaneous visible actions that are members of the finite alphabet Σ. Additionally, there is a distinguished event "$\surd$" that designates a process' termination. As in the untimed CSP model "$\surd$" is a "silent" event, in that it is not observable.

Timed Traces. A visible sequence of events executed during a finite interval is represented by a *timed trace*. Timed traces are denoted as

$$\langle (A_1, n_1), (A_2, n_2), \ldots, (A_m, n_m) \rangle \in ((\mathcal{P}(\Sigma - \{\surd\}) - \emptyset) \times \boldsymbol{N})^*.$$

Each A_i denotes a set of events that are executed simultaneously; each n_i measures the delay between the executions of A_{i-1} and A_i (with n_0 denoting the absolute execution time of A_0). We let the letters s, t and u range over timed traces.

The *concatenation* of two traces, denoted $s \hat{} t$, is defined by normal string concatenation. The identity element of this operation is the empty trace, denoted "$\langle\rangle$", with $s \hat{} \langle\rangle = \langle\rangle \hat{} s = s$. Furthermore, if s is a trace such that $s = t_1 \hat{} \langle (A, i) \rangle \hat{} \langle (B, 0) \rangle \hat{} t_2$, then $s = t_1 \hat{} \langle (A \cup B, i) \rangle \hat{} t_2$.

Restriction, or $s \upharpoonright A$, removes from s all events in $\Sigma - A$ without altering the absolute occurrence times of any remaining event. For example, $\langle (\{a, b\}, 1), (\{c\}, 2), (\{d\}, 1) \rangle \upharpoonright \{b, d\} = \langle (\{b\}, 1), (\{d\}, 3) \rangle$.

The *duration* of a trace $\delta(s)$ is the absolute occurrence time of the last set of events in s, with $\delta(\langle\rangle) = 0$. Finally, $s \,|||\, t$ merges the traces s and t, while preserving each event's absolute execution time.

States. Accompanying each trace s is a state set, which represents possible behaviors of the process after executing s. Each member of the state set is called a *state*, containing events that are deterministically offered to the environment. The state domain is defined as

$$\text{STATES} = \{x \cup \{(\{\surd\}, \infty)\} \mid x \in \mathcal{P}((\mathcal{P}(\Sigma) - \emptyset) \times \boldsymbol{N})\},$$

with the letter σ representing a single state. Each pair $(A, i) \in \sigma$ represents events that are offered after a trace s, and the time after $\delta(s)$ at which they may occur.

Note that the pair $(\{\surd\}, \infty)$ is in every state. This denotes that if the environment fails to select any of the other events in the state, the process will diverge. Furthermore, note that no *other* event set may have an occurrence time of ∞. Obviously if there is an infinite delay, the events in the set can never be executed. Thus we stipulate that $\forall A \in \mathcal{P}(\Sigma)\ \lim_{i \to \infty}(A, i) = (\{\surd\}, \infty)$. This fact becomes essential when we prove that the semantic domain forms a complete partial order.

State Sets. After executing a trace, a process may nondeterministically choose a state to offer the environment. This array of choices is represented by the *state set*, which we denote by $\bar{\sigma}$, and which has the following definition.

Definition 1 *A set* $\bar{\sigma} \in (\mathcal{P}(\text{STATES}) - \{\emptyset\})$ *is a valid state set if it is saturated, defined by the following four properties:*

1. $\bar{\sigma}$ *is closed under union, or* $\bar{\sigma}' \subseteq \bar{\sigma} \Longrightarrow (\bigcup_{\sigma \in \bar{\sigma}'} \sigma)\ \in \bar{\sigma}$.

2. $\bar{\sigma}$ *is convex closed with respect to set containment:*

$$\forall \sigma \in \mathit{STATES}\ ((\exists \sigma_1, \sigma_2 \in \bar{\sigma}(\sigma_1 \subseteq \sigma \wedge \sigma \subseteq \sigma_2)) \Longrightarrow \sigma \in \bar{\sigma}).$$

3. *In the partial order formed by set inclusion, if a directed chain is in* $\bar{\sigma}$, *the greatest lower bound of the chain is also in* $\bar{\sigma}$: **if** $\sigma_1, \sigma_2, \ldots \in \bar{\sigma}$ **and** $\sigma_1 \supseteq \sigma_2 \supseteq \ldots$ **then** $(\bigcap \sigma_i) \in \bar{\sigma}$.

4. $\bar{\sigma}$ *is the smallest such set preserving properties 1, 2 and 3.* □

If a set $\bar{\sigma}$ is saturated, then we write that $\bar{\sigma} = sat(\bar{\sigma})$. Note that our requirement for saturated state sets is the analogue of the "failure closure" rule found in [2] and other refusal-based models.

State set addition, $\bar{\sigma} + i$, adds i to every event time in $\bar{\sigma}$, while state set restriction, $\bar{\sigma} \uparrow A$, eliminates the events in $\Sigma - A$ from $\bar{\sigma}$:

$$\begin{aligned} \bar{\sigma} + i &= \{\sigma \mid \exists \sigma' \in \bar{\sigma}\,(\forall (A, j) \in \sigma'\,(i + j \geq 0 \Longleftrightarrow (A, i + j) \in \sigma))\} \\ \bar{\sigma} \uparrow A &= \{\sigma \mid \exists \sigma' \in \bar{\sigma}\,.\,\sigma = \{(B \cap A, i) \mid (B, i) \in \sigma' \wedge B \cap A \neq \emptyset\}\} \end{aligned}$$

Acceptances. An *acceptance* $(s, \bar{\sigma})$ represents a possible execution of a process, where $\bar{\sigma}$ is the set of states reachable after engaging in the trace s. At that time, a single state $\sigma \in \bar{\sigma}$ is nondeterministically offered to the environment. Within our semantic domain, a process is totally defined by its entire set of potential acceptances. Note that since Σ is finite and $\boldsymbol{N}$ is discrete, a process may only execute a finite number of events during a finite time interval.

Now we define several operators on acceptances, letting the symbols $\mathcal{A}$, $\mathcal{A}_1$ and $\mathcal{A}_2$ denote arbitrary acceptance sets.

$$\begin{array}{ll}
\text{Acceptance Set Intersection:} & \mathcal{A}_1 \cap_{\mathcal{A}} \mathcal{A}_2 = \{(s, \bar{\sigma}_1 \cap \bar{\sigma}_2) \mid (s, \bar{\sigma}_1) \in \mathcal{A}_1 \wedge (s, \bar{\sigma}_2) \in \mathcal{A}_2\} \\
\text{Acceptance Set Containment:} & \mathcal{A}_1 \subseteq_{\mathcal{A}} \mathcal{A}_2 \Longleftrightarrow \forall (s, \bar{\sigma}_1) \in \mathcal{A}_1 \; \exists (s, \bar{\sigma}_2) \in \mathcal{A}_2 \; (\bar{\sigma}_1 \subseteq \bar{\sigma}_2) \\
\text{Acceptance Set Addition:} & \mathcal{A} + i = \{(\langle\rangle, \bar{\sigma} + i) \mid (\langle\rangle, \bar{\sigma}) \in \mathcal{A}\} \\
& \quad \cup \{(\langle (A, i+j) \rangle^\frown t, \bar{\sigma}) \mid (\langle (A, j) \rangle^\frown t, \bar{\sigma}) \in \mathcal{A}\}
\end{array}$$

Note that the addition operator delays the starting time of each acceptance by i time units.

3.2 The Domain and Process Containment

In the timed acceptances model, a *real-time process* P is characterized by a set of acceptances representing its potential execution behaviors. We define the trace set of P as $TRS(P) = \{s | (s, \bar{\sigma}) \in P\}$ and the state set associated with a trace s of P as $STS(s, P) = \{\sigma \in \bar{\sigma} | (s, \bar{\sigma}) \in P\}$. For consistency, we define $STS(s, P) = \emptyset$ if $s \notin TRS(P)$.

Definition 2 *A real-time process P is a set of acceptances satisfying the following constraints:*

1. $\langle\rangle \in TRS(P)$
2. $TRS(P) = cl_{\leq}(TRS(P))$, *where $cl_{\leq}$ is the prefix closure of a set of traces.*
3. $P \subseteq (\Sigma \times \mathbb{N})^* \times (\mathcal{P}(STATES) - \{\emptyset\})$
4. $s^\frown\langle(A, i)\rangle \in TRS(P) \Longrightarrow \exists \sigma \in STS(s, P) \,.\, (A, i) \in \sigma$
5. $\forall \sigma \in STS(s, P) \forall (A - \{\surd\} \neq \emptyset, i) \in \sigma \,.\, s^\frown\langle(A - \{\surd\}, i)\rangle \in TRS(P)$
6. $s \in TRS(P) \Longrightarrow STS(s, P) = sat(STS(s, P))$ □

Properties 1 through 3 are consistent with the definition of a process found in [1], while properties 5 and 6 retain the flavor of the Acceptance Tree definitions in [4]. The acceptance sets satisfying all of the above rules form the domain of real-time processes, which we denote $\mathcal{RT}$. The domain is partially ordered by *process containment*. That is, P *contains* Q, or $P \sqsubseteq Q$, if P can execute all of Q's traces and make at least as many nondeterministic decisions after executing each trace.

Definition 3 (Partial Order) $P \sqsubseteq Q$ *if and only if* $Q \subseteq_{\mathcal{A}} P$ □

We proceed to show that process containment forms a complete partial order (Theorem 1). To do this we require the following four lemmas, the first two of which we state without proof. Lemma 3 is necessary to prove that a directed chain of processes has a least upper bound.

Lemma 1 *Process containment forms a partial order.*

Lemma 2 *The domain $\mathcal{RT}$ contains a least element,* CHAOS, *such that* $\forall P \in \mathcal{RT}$ CHAOS $\sqsubseteq P$.

Lemma 3 *Given a chain of processes $P_0 \sqsubseteq P_1 \sqsubseteq \ldots$, if there exists a trace s such that for all i, $s \in TRS(P_i)$, then $\bigcap_{i\geq 0} STS(s, P_i) \neq \emptyset$.*

Proof Let s be a trace such that $\forall i\, s \in TRS(P_i)$, and for notational convenience, denote $STS(s, P_i)$ as $\bar{\sigma}_i$. By the definition of the partial order, for all i, $\bar{\sigma}_i \supseteq \bar{\sigma}_{i+1}$. Now for each i, choose some state σ_i in $\bar{\sigma}_i$. We proceed to show that there exists a state σ such that for all i, $\sigma \in \bar{\sigma}_i$. For each i, define the state σ_i' as:

$$\sigma_i' = \bigcup_{j\geq i} \sigma_j$$

By this construction, we see that $\forall i\ \sigma_i' \supseteq \sigma_{i+1}'$. Furthermore, as state sets are closed under union, we also have $\forall i\ \forall j \geq i,\ \sigma_j' \in \bar{\sigma}_i$. And since state sets are closed under glb's, we derive that

$$\forall i \bigcap_{j\geq i} \sigma_j' \in \bar{\sigma}_i.$$

Now define $\sigma = \bigcap_{i\geq 0} \sigma_i'$. Sinces the σ_i''s form a nonincreasing sequence of states, we deduce that $\forall i,\ \sigma \in \bar{\sigma}_i$.

It remains to be shown that σ is in the domain STATES. However, this is trivial, as every state contains the pair $(\{\surd\}, \infty)$, and thus, any intersection of states does as well. Therefore, $\bigcap_{i\geq 0} STS(s, P_i) \neq \emptyset$. □

Lemma 4 (Least Upper Bound) *Given a chain of processes $\mathbf{P} = \{P_i | i \geq 0, P_i \sqsubseteq P_{i+1}\}$ in $\mathcal{RT}$, $\bigsqcup \mathbf{P} = \{\cap_{\mathcal{A}} P_i \mid P_i \in \mathbf{P}\}$ defines a process and is the least upper bound of the chain.*

Proof To show that $\bigsqcup \mathbf{P}$ is in fact an upper bound, we must first show that it satisfies the properties that define a process. For convenience, denote $\bigsqcup \mathbf{P}$ as P. That P satisfies properties 1, 2 and 3 follows directly from the definition of a process and Lemma 3. Properties 5 and 6 follow from the definitions of saturation and state set intersection. However, property 4 is not so obvious, and we prove it here.

Since every $P_i \in \mathbf{P}$ is a process, every P_i observes property 4. So,

$$s\hat{}\langle(A, j)\rangle \in TRS(P_i) \Longrightarrow \exists \sigma_i \in STS(s, P_i)\,.\,(A, j) \in \sigma_i.$$

Now, define $\sigma_i' = \bigcup_{k\geq i} \sigma_k$. Let $\sigma = \bigcap_{i\geq 0} \sigma_i'$. By the same argument made in Lemma 3, σ is in every $STS(s, P_i)$, and so, σ is in $STS(s, P)$. Also, since (A, j) is in σ, P obeys property 4.

Hence P is an upper bound of the chain $\mathbf{P}$. The proof that P is the least upper bound is straightforward and thus omitted. □

Theorem 1 *The domain $\mathcal{RT}$ forms a complete partial order.*

3.3 Defining the Semantics of the Operators

We now define a meaning function, $\mathcal{M}$ that maps each term in the language to the real-time process that it denotes. The formal semantics of each operator corresponds to the intuitive semantics presented above.

$$\mathcal{M}[\![\text{CHAOS}]\!] = (\mathcal{P}(\Sigma - \{\surd\}) \times \mathbb{N})^* \times (\mathcal{P}(\text{STATES}) - \emptyset)$$

$$\mathcal{M}[\![\text{STOP}]\!] = \{(\langle\rangle, \{\{(\{\surd\}, \infty)\}\})\}$$

$$\mathcal{M}[\![\text{SKIP}]\!] = \{(\langle\rangle, \{\{(\{\surd\}, i) | i \geq 0\}\})\}$$

$$\mathcal{M}[\![A \overset{i}{\rightsquigarrow} P]\!] = \begin{cases} \mathcal{M}[\![P]\!] + i & \text{if } A = \emptyset \\ \{(\langle\rangle, \{\{(A,0), (\{\surd\}, \infty)\}\})\} \cup \{(\langle(A,0)\rangle^\frown s, \bar{\sigma}) | (s, \bar{\sigma}) \in \mathcal{M}[\![P]\!] + i\} & \text{otherwise} \end{cases}$$

$$\mathcal{M}[\![P \,\Box\, Q]\!] = \{(\langle\rangle, \{\sigma_P \cup \sigma_Q | \sigma_P \in STS(\langle\rangle, P) \wedge \sigma_Q \in STS(\langle\rangle, Q)\})\} \cup \{(s \neq \langle\rangle, sat(STS(s,P) \cup STS(s,Q))) | s \in TRS(P) \cup TRS(Q)\}$$

$$\mathcal{M}[\![P \sqcap Q]\!] = \{(s, sat(STS(s,P) \cup STS(s,Q))) | s \in TRS(P) \cup TRS(Q)\}$$

$$\mathcal{M}[\![P \|_A Q]\!] = \{(s, sat(\bar{\sigma}_P \cup^{\|}_{\vartheta} \bar{\sigma}_Q)) \mid \exists (s_P, \bar{\sigma}_P) \in \mathcal{M}[\![P]\!] \;\; \exists (s_Q, \bar{\sigma}_Q) \in \mathcal{M}[\![Q]\!] \; . \; s = s_P \,|||\, s_Q \;\wedge\; s_P \uparrow A = s_Q \uparrow A\},$$

where:

$$\bar{\sigma}_P \cup^{\|}_{\vartheta} \bar{\sigma}_Q = \{\sigma \mid \exists \sigma_P \in \bar{\sigma}_P + \delta(s_P) - \delta(s) \;\; \exists \sigma_Q \in \bar{\sigma}_Q + \delta(s_Q) - \delta(s). \; \sigma = \{(B,i) | \exists (B,i) \in (\sigma_P \cup \sigma_Q) \; . \; B \cap (A \cup \{\surd\}) = \emptyset\} \cup \{(B \cup C, i) | \exists (B,i) \in \sigma_P \; \exists (C,i) \in \sigma_Q . \; ((B \cup C) \cap (A \cup \{\surd\})) \subseteq (B \cap C)\}\}$$

$$\mathcal{M}[\![P \backslash A]\!] = \{(t, STS_{\text{HIDE}}(sat(\bar{\sigma}), A)) \mid \exists s \in TRS(P) \wedge t = s \uparrow (\Sigma - A) \wedge \bar{\sigma} = \{\sigma \mid \exists (s', \bar{\sigma}') \in \mathcal{M}[\![P]\!] \, . \, s' \uparrow (\Sigma - A) = t \wedge \sigma \in \bar{\sigma}' + (\delta(s') - \delta(t))\}\},$$

where:

$$STS_{\text{HIDE}}(\bar{\sigma}, A) = ((\bar{\sigma} \uparrow (\Sigma - (A - \{\surd\})) - \{\{(\{\surd\}, \infty)\}\}) \cup (\bar{\sigma} \cap \{\{(\{\surd\}, \infty)\}\})$$

$$\mathcal{M}[\![\mu P.F(P)]\!] = \bigsqcup_{i \geq 0} (F^i(\text{CHAOS}))$$

Note that the semantics of the parallel operator induce synchronization on the event "$\surd$", whether or not it is included in the synchronizing event set. Thus, two concurrent processes terminate in finite time if and only if both constuent process do. On the other hand, concealment implicitly excludes "$\surd$" from the hidden event set, maintaining the well-definedness of the operator. Furthermore, concealment intruduces no "artificial" deadlocks. STS_{HIDE} preserves only the stopping states induced by true deadlock, or by divergence due to hiding.

4 Axiomatizing Real-time Processes

We now describe a sound and complete axiomatization of process containment for finite processes; that is, those processes that can be represented by terms containing no occurrences of CHAOS or recursion. The semantic function, denoted $\mathcal{M}$, maps these terms to acceptance sets as defined in Section 3. The logical assertions are of the form $P \sqsubseteq Q$ or $P = Q$, where the latter means $P \sqsubseteq Q$ and $Q \sqsubseteq P$. We write $\vdash P \sqsubseteq Q$

if there is a proof of $P \sqsubseteq Q$ in the axiom system. We then introduce axioms that allow us to reason about infinite terms (terms with CHAOS and recursion) and explain a possible extension to the model that allows us to obtain a complete axiomatization of infinite processes. The infinitary axiom used for reasoning about recursively defined terms is based on the well-known idea of *syntactic approximation* of terms [3]. Our approach is similar to that used by Brookes [1] for untimed CSP and Hennessy [5, 4] for CCS.

4.1 Finite Processes

We begin by considering terms formed without parallel composition or hiding. Table 1 lists the axioms and inference rules for these terms. This axiomatization is consistent with that of untimed CSP [1]. In addition, (TA1), (TA2), (CH5), and (D2) allow for reasoning about time. (TA1) and (D2) represent the division of time intervals and the distributivity of delay. (TA2) and (CH5) represent the persistence of deadlock and termination.

The soundness of the system follows from the semantics of the terms in the language as defined in Section 3. We state the soundness of the axiom system without proof.

Theorem 2 *For all terms P and Q, $\vdash P \sqsubseteq Q \Longrightarrow \mathcal{M}[\![P]\!] \supseteq_{\mathcal{A}} \mathcal{M}[\![Q]\!]$.* □

To show that the proof system is complete, we introduce a normal form for finite terms and show that every term can be provably converted to a unique normal form. We then show the axiom system complete for terms in normal form. The completeness for arbitrary terms follows from these two results.

A term in normal form is a nondeterministic composition of deterministically guarded terms. To ensure the uniqueness of normal forms, the nondeterministic composition is indexed by a saturated state set and every subterm reachable by the same execution sequence is unique. In addition, each subterm of the nondeterministic composition is indexed by either $\{(\{\surd\}, \infty)\}$ or a state from which all stopping times except the earliest finite one has been removed. Given a state σ, this set is defined by

$$min_t(\sigma) = \begin{cases} \{(A \neq \{\surd\}, i) \in \sigma\} \cup \{(\{\surd\}, j \neq \infty) | j = min\{i | (\{\surd\}, i) \in \sigma\}\} & \text{if } \sigma \neq \{(\{\surd\}, \infty)\} \\ \{(\{\surd\}, \infty)\} & \text{otherwise} \end{cases}$$

For a given term P, we define a subterm $P_{(i,A,1)}$ to be the process P at time $i+1$ after executing the events in A at time i.

Definition 4 *A term P is in normal form if and only if it has the structure*

$$P = \begin{cases} \text{STOP} \\ \text{SKIP} \\ \sqcap_{\sigma \in \bar{\sigma}} (\Box_{(A,i) \in min_t(\sigma)} (\emptyset \overset{i}{\rightsquigarrow} A \overset{1}{\rightsquigarrow} P_{(i,A,1)})) \end{cases}$$

where $\bar{\sigma}$ is a saturated state set, and each $P_{(i,A,1)}$ is unique and also in normal form. We write $\emptyset \overset{i}{\rightsquigarrow}$ SKIP for $(A, i) = (\{\surd\}, i \neq \infty)$ and STOP *for $(A, i) = (\{\surd\}, \infty)$.* □

Timed Action	
(TA1)	$A \overset{i}{\rightsquigarrow} \emptyset \overset{j}{\rightsquigarrow} P = A \overset{i+j}{\rightsquigarrow} P$
(TA2)	$\emptyset \overset{1}{\rightsquigarrow} \mathrm{STOP} = \mathrm{STOP}$
Nondeterminism	
(ND1)	$P \sqcap P = P$
(ND2)	$P \sqcap Q = Q \sqcap P$
(ND3)	$P \sqcap (Q \sqcap R) = (P \sqcap Q) \sqcap R$
Choice	
(CH1)	$P \,\Box\, P = P$
(CH2)	$P \,\Box\, Q = Q \,\Box\, P$
(CH3)	$P \,\Box\, (Q \,\Box\, R) = (P \,\Box\, Q) \,\Box\, R$
(CH4)	$P \,\Box\, \mathrm{STOP} = P$
(CH5)	$(\emptyset \overset{i}{\rightsquigarrow} \mathrm{SKIP}) \,\Box\, \mathrm{SKIP} = \mathrm{SKIP}$
(CH6)	$(A \overset{i}{\rightsquigarrow} P) \,\Box\, (A \overset{j}{\rightsquigarrow} Q) = (A \overset{i}{\rightsquigarrow} P) \sqcap (A \overset{j}{\rightsquigarrow} Q)$ if $A \neq \emptyset$
Distributivity	
(D1)	$A \overset{i}{\rightsquigarrow} (P \sqcap Q) = (A \overset{i}{\rightsquigarrow} P) \sqcap (A \overset{i}{\rightsquigarrow} Q)$
(D2)	$\emptyset \overset{i}{\rightsquigarrow} (P \,\Box\, Q) = (\emptyset \overset{i}{\rightsquigarrow} P) \,\Box\, (\emptyset \overset{i}{\rightsquigarrow} Q)$
(D3)	$P \sqcap (Q \,\Box\, R) = (P \sqcap Q) \,\Box\, (P \sqcap R)$
(D4)	$P \,\Box\, (Q \sqcap R) = (P \,\Box\, Q) \sqcap (P \,\Box\, R)$
Partial Ordering	
(PO1)	$P \sqcap Q \sqsubseteq P$
(PO2)	$P \sqsubseteq Q \wedge Q \sqsubseteq P \Longrightarrow P = Q$
(PO3)	$P = Q \Longrightarrow P \sqsubseteq Q \wedge Q \sqsubseteq P$
(PO4)	$P \sqsubseteq Q \wedge Q \sqsubseteq R \Longrightarrow P \sqsubseteq R$
Monotonicity	
(M1)	$P \sqsubseteq Q \Longrightarrow (A \overset{i}{\rightsquigarrow} P) \sqsubseteq (A \overset{i}{\rightsquigarrow} Q)$
(M2)	$P_1 \sqsubseteq Q_1 \ \wedge \ P_2 \sqsubseteq Q_2 \Longrightarrow P_1 \sqcap P_2 \sqsubseteq Q_1 \sqcap Q_2$
(M3)	$P_1 \sqsubseteq Q_1 \ \wedge \ P_2 \sqsubseteq Q_2 \Longrightarrow P_1 \,\Box\, P_2 \sqsubseteq Q_1 \,\Box\, Q_2$

Table 1: A Proof System for Finite Processes Without $||_A$ and $\setminus$

Note that to simplify the presentation, we write $\emptyset \overset{0}{\rightsquigarrow} A \overset{1}{\rightsquigarrow} P_{(0,A,1)}$, $\emptyset \overset{0}{\rightsquigarrow}$ SKIP and $\emptyset \overset{0}{\rightsquigarrow}$ STOP instead of $A \overset{1}{\rightsquigarrow} P_{(0,A,1)}$, SKIP and STOP, respectively. Also note that for a term in normal form, the indexing set cannot be empty. We frequently abbreviate the normal form of a term P as $\sqcap_{\sigma_P \in \bar{\sigma}_P} P_{min_t(\sigma_P)}$.

Since the proofs of the existence of normal forms and the completeness of the axiom system use structural induction on terms, we define the length of term P, $|P|$, as follows: $|\text{SKIP}| = |\text{STOP}| = 0$, $|A \overset{i}{\rightsquigarrow} P| = 1+|P|$ for $A \neq \emptyset$, $|\emptyset \overset{i}{\rightsquigarrow} P| = |P|$, $|P \Box Q| = |P \sqcap Q| = 1+max(|P|,|Q|)$, $|P\|_A Q| = 1+|P|+|Q|$, and $|P \backslash A| = 1+|P|$.

The following four axioms can be derived from the axioms in Table 1. They are used in the derivation of normal forms.

(S1) $P \sqcap Q = P \sqcap Q \sqcap (P \Box Q)$

(S2) $P \sqcap (P \Box Q \Box R) = P \sqcap (P \Box Q) \sqcap (P \Box Q \Box R)$

(S3) $((A \overset{i}{\rightsquigarrow} P1) \Box R1)) \sqcap ((A \overset{i}{\rightsquigarrow} P2) \Box R2) =$
$((A \overset{i}{\rightsquigarrow} P1) \Box (A \overset{i}{\rightsquigarrow} P2) \Box R1) \sqcap ((A \overset{i}{\rightsquigarrow} P1) \Box (A \overset{i}{\rightsquigarrow} P2) \Box R2)$

Note that we do not need an axiom to ensure the inclusion of *glb*'s since we only consider finite processes.

Lemma 5 *Every term P that does not contain $\|_A$ and $\backslash$ can be transformed into a normal form using the proof system defined in Table 1.*

Proof. By induction on the length and structure of the term.

If $|P| = 0$, then P is SKIP or STOP and the proof is obvious. Now assume that every term P, with $|P| < n$, can be transformed into an equivalent normal form using the proof system. We show that given P and Q in normal form with $|P| < n$ and $|Q| < n$, $P \sqcap Q$, $P \Box Q$, and $A \overset{k}{\rightsquigarrow} P$ are reducible to normal form.

Using (ND1)-(ND3), rewrite $P \sqcap Q$ as $\sqcap_{\sigma_P \in \bar{\sigma}_P} \sqcap_{\sigma_Q \in \bar{\sigma}_Q} (P_{min_t(\sigma_P)} \sqcap Q_{min_t(\sigma_Q)})$. To reduce this new term to an equivalent normal form we need to show that (1) for each $(A \neq \{\surd\}, i) \in \bigcup_{\sigma \in \bar{\sigma}_P \cup \bar{\sigma}_Q} \sigma$, we can derive a unique subterm after each (A, i) and (2) $\sqcap_{\bar{\sigma}_P} \sqcap_{\bar{\sigma}_Q}$ can be replaced by a single occurrence of $\sqcap$ that is indexed by an equivalent saturated state set.

We construct a unique $R_{(i,A,1)}$ for each $(A \neq \{\surd\}, i) \in \bigcup_{\sigma \in \bar{\sigma}_P \cup \bar{\sigma}_Q} \sigma$ as follows. Let $\bar{\sigma} = \bar{\sigma}_P \cup \bar{\sigma}_Q$ and define the terms $R_{(i,A,1)}$ for $(A \neq \{\surd\}, i) \in \bigcup_{\sigma \in \bar{\sigma}} \sigma$ by

$$R_{(i,A,1)} = \begin{cases} P_{(i,A,1)} & \text{if } (A,i) \in \bigcup_{\sigma \in \bar{\sigma}_P} \sigma \wedge (A,i) \notin \bigcup_{\sigma \in \bar{\sigma}_Q} \sigma \\ Q_{(i,A,1)} & \text{if } (A,i) \notin \bigcup_{\sigma \in \bar{\sigma}_P} \sigma \wedge (A,i) \in \bigcup_{\sigma \in \bar{\sigma}_Q} \sigma \\ P_{(i,A,1)} \sqcap Q_{(i,A,1)} & \text{if } (A,i) \in \bigcup_{\sigma \in \bar{\sigma}_P} \sigma \wedge (A,i) \in \bigcup_{\sigma \in \bar{\sigma}_Q} \sigma \end{cases}$$

Each $R_{(i,A,1)}$ is unique since both $P_{(i,A,1)}$ and $Q_{(i,A,1)}$ are unique. Using (S3) and the distributivity laws we derive

$$P \sqcap Q = \sqcap_{\sigma \in \bar{\sigma}} \Box_{(A,i) \in min_t(\sigma)} (\emptyset \overset{i}{\rightsquigarrow} A \overset{1}{\rightsquigarrow} R_{(i,A,1)})$$

To find the saturated indexing set, we apply (S1) and (S2) to $\sqcap_{\sigma \in \bar{\sigma}_P \cup \bar{\sigma}_Q} R_{min_t(\sigma)}$ obtaining

$$\vdash P \sqcap Q = \sqcap_{\sigma \in sat(\bar{\sigma}_P \cup \bar{\sigma}_Q)} R_{min_t(\sigma)}$$

The proof that $P \,\Box\, Q$ can be reduced to a normal form is similar to the proof for $P \sqcap Q$ and is thus omitted.

To prove $(C \overset{k}{\rightsquigarrow} P)$, we consider three cases: (1) $C \neq \emptyset$ and $k = 1$, (2) $C = \emptyset$ and (3) $C \neq \emptyset$ and $k > 1$. (1) is true because $C \overset{1}{\rightsquigarrow} P$ is in normal form. (2) is proven using (D1) and (D2) to distribute the delay over choice and nondeterminism and (TA1) to eliminate unnecessary occurrences of $\emptyset$. To prove (3), apply (TA1), rewriting $C \overset{k}{\rightsquigarrow} P$ as $C \overset{1}{\rightsquigarrow} \emptyset \overset{k-1}{\rightsquigarrow} P$. Then, using the same technique as in (2), reduce $\emptyset \overset{k-1}{\rightsquigarrow} P$ to its normal form. □

Lemma 6 (Normal Form Completeness) *Given two normal forms* $P = \sqcap_{\sigma_P \in \bar{\sigma}_P} P_{min_t(\sigma_P)}$ *and* $Q = \sqcap_{\sigma_Q \in \bar{\sigma}_Q} Q_{min_t(\sigma_Q)}$, *if* $\mathcal{M}[\![P]\!] \supseteq_{\mathcal{A}} \mathcal{M}[\![Q]\!]$, *then* $\vdash P \sqsubseteq Q$.

Proof. By induction on the length of the normal forms.

The base case, when both terms have zero length, is obvious since they are SKIP or STOP. We assume that the theorem holds for all terms P and Q such that $|P| < n$ and $|Q| < n$.

Suppose that $\mathcal{M}[\![P]\!] \supseteq_{\mathcal{A}} \mathcal{M}[\![Q]\!]$ and $|P|, |Q| < n + 1$. Since P and Q are in normal form, $\bar{\sigma}_P$ and $\bar{\sigma}_Q$ are saturated. Furthermore, each $P_{(i,A,1)}$ and $Q_{(j,B,1)}$ is unique and in normal form with $|P_{(i,A,1)}|, |Q_{(j,B,1)}| < n$. In order to use the structure of the normal form to prove completeness, we need to show that (1) $\bar{\sigma}_P \supseteq \bar{\sigma}_Q$, and (2) for all $\sigma_Q \in \bar{\sigma}_Q$ and $(B, i) \in \sigma_Q$, $\mathcal{M}[\![P_{(i,B,1)}]\!] \supseteq_{\mathcal{A}} \mathcal{M}[\![Q_{(i,B,1)}]\!]$.

Since $\bar{\sigma}_P = STS(\langle\rangle, P)$ and $\bar{\sigma}_Q = STS(\langle\rangle, Q)$, (1) follows immediately from $\mathcal{M}[\![P]\!] \supseteq_{\mathcal{A}} \mathcal{M}[\![Q]\!]$. To prove (2), consider a state $\sigma_Q \in \bar{\sigma}_Q$ and a pair $(B \neq \{\surd\}, i) \in \sigma_Q$. Since each $P_{(i,B,1)}$ and $Q_{(i,B,1)}$ is unique, we have

$$TRS(P_{(i,B,1)}) = \{s | \langle (B,i) \rangle \hat{} s \in TRS(P)\} \quad \text{and} \quad TRS(Q_{(i,B,1)}) = \{s | \langle (B,i) \rangle \hat{} s \in TRS(Q)\}$$

Since $TRS(P) \supseteq TRS(Q)$, it is easy to show that $TRS(P_{(i,B,1)}) \supseteq TRS(Q_{(i,B,1)})$. Furthermore, for all $s \in TRS(Q_{(i,B,1)})$, it follows that $STS(s, Q_{(i,B,1)}) = STS(\langle (B,i) \rangle \hat{} s, Q) \subseteq STS(\langle (B,i) \rangle \hat{} s, P) = STS(s, P_{(i,B,1)})$. Thus, $\mathcal{M}[\![P_{(i,B,1)}]\!] \supseteq_{\mathcal{A}} \mathcal{M}[\![Q_{(i,B,1)}]\!]$.

For all $(B \neq \{\surd\}, i) \in \bigcup_{\sigma \in \bar{\sigma}_Q} \sigma$, we have $|P_{(i,B,1)}|, |Q_{(i,B,1)}| < n$. Thus, it follows that $\vdash P_{(i,B,1)} \sqsubseteq Q_{(i,B,1)}$ from the induction hypothesis. Since $\bar{\sigma}_Q \subseteq \bar{\sigma}_P$, for all $\sigma_Q \in \bar{\sigma}_Q$, we have $\sigma_Q \in \bar{\sigma}_P$. We then derive $P_{min_t(\sigma_Q)} \sqsubseteq Q_{min_t(\sigma_Q)}$ by applying (M3) a finite number of times. Therefore, $\vdash P \sqsubseteq Q$. □

Adding parallel composition and hiding. To make the proof system complete for all terms with no occurrences of CHAOS and recursion, we present axioms that can be used to eliminate parallelism and concealment in Table 2. Since all the operators distribute over nondeterminism, axioms (P1) and (C4) are defined for terms whose outermost operator is $\Box$.

Before describing the new axioms, we define $P_{(i,\emptyset,1)}$ which represents the behavior of P at time $i + 1$ if

Parallelism

For axioms (P1) and (C4), let

$$P = \Box_{(B,j)\in min_t(\sigma_P)}(\emptyset \overset{j}{\rightsquigarrow} B \overset{1}{\rightsquigarrow} P_{(j,B,1)})$$

$$Q = \Box_{(B,j)\in min_t(\sigma_Q)}(\emptyset \overset{j}{\rightsquigarrow} B \overset{1}{\rightsquigarrow} Q_{(j,B,1)})$$

(P1)
$$\begin{aligned} P\|_A Q = {} & \Box_{(B,i)\in\sigma_P\uparrow(\Sigma-A)}(\emptyset \overset{i}{\rightsquigarrow} B \overset{1}{\rightsquigarrow} (P_{(i,B,1)}\|_A Q_{(i,\emptyset,1)})) \\ & \Box_{(B,i)\in\sigma_Q\uparrow(\Sigma-A)}(\emptyset \overset{i}{\rightsquigarrow} B \overset{1}{\rightsquigarrow} (P_{(i,\emptyset,1)}\|_A Q_{(i,B,1)})) \\ & \Box_{(C\neq\{\surd\},i)\in\sigma_P\wedge(D\neq\{\surd\},i)\in\sigma_Q\wedge C\cup D\uparrow A\subseteq C\cap D}(\emptyset \overset{i}{\rightsquigarrow} (C\cup D) \overset{1}{\rightsquigarrow} (P_{(i,C,1)}\|_A Q_{(i,D,1)})) \\ & \Box_{(\{\surd,i\neq\infty)\in min_t(\sigma_P\cap\sigma_Q)}(\emptyset \overset{i}{\rightsquigarrow} \mathrm{SKIP}) \\ & \Box_{(\surd,\infty)\in min_t(\sigma_P\cap\sigma_Q)}\mathrm{STOP} \end{aligned}$$

Concealment

(C1) $(A \overset{i}{\rightsquigarrow} P)\backslash B = (A - B) \overset{i}{\rightsquigarrow} (P\backslash B)$

(C2) $\mathrm{STOP}\backslash A = \mathrm{STOP}$

(C3) $\mathrm{SKIP}\backslash A = \mathrm{SKIP}$

For axiom (C4), let

$$P = \Box_{(B,i)\in min_t(\sigma)}(\emptyset \overset{i}{\rightsquigarrow} B \overset{1}{\rightsquigarrow} P_{(i,B,1)})$$

(C4)
$$\begin{aligned} P\backslash A = {} & (\Box_{(B,i)\in min_t(\sigma)\wedge(B-A)\neq\emptyset}(\emptyset \overset{i}{\rightsquigarrow} (B-A) \overset{1}{\rightsquigarrow} (P_{(i,B,1)}\backslash A))) \\ & \sqcap(\sqcap_{(B,i)\in min_t(\sigma)\wedge(B-A)=\emptyset}(\emptyset \overset{i+1}{\rightsquigarrow} (P_{(i,B,1)}\backslash A))) \end{aligned}$$

Distributivity

(D5) $P\|_A(Q \sqcap R) = (P\|_A Q) \sqcap (P\|_A R)$

(D6) $(P \sqcap Q)\backslash A = (P\backslash A) \sqcap (Q\backslash A)$

Table 2: Additional axioms to the proof system for $\|_A$ and $\backslash$

it has not executed any events up to time i.

$$P_{(i,\emptyset,1)} = \begin{cases} \Box_{(A,j>i)\in min_t(\sigma_P)}(\emptyset \overset{j-(i+1)}{\rightsquigarrow} A \overset{1}{\rightsquigarrow} P_{(j,A,1)}) \\ \Box_{(\{\surd\},j\leq i+1)\in min_t(\sigma_P)}\mathrm{SKIP} \\ \Box_{(\{\surd\},\infty)\in min_t(\sigma_P)}\mathrm{STOP} \end{cases}$$

Axiom (P1) reflects the semantics of parallel composition. That is, interleaving with respect to time. Note that if P and Q can only engage in events in A but not at common finite times, then $P\|_A Q$ reduces to STOP by the last subterm. The axioms for concealment preserve the possible occurrence times of the unconcealed events and capture the possible introduction of nondeterminism by concealment. The soundness of the axioms in Table 2 follows from the semantics of the operators.

These new axioms allow us to derive unique normal forms for all terms denoting finite processes.

Lemma 7 (Normal Form) *Every P can be transformed into a normal form using the proof system defined in Tables 1 and 2.*

Proof. By induction on the length of normal forms.

The case where P does not contain $||_A$ and $\backslash$ is proven in Lemma 5. To prove that $P||_A Q$ is reducible to a unique normal form, we assume that P and Q are already in normal form. Then, we apply axiom (P1) to $P||_A Q$. It is easily shown that the resulting subterms each have length less that $P||_A Q$ and that there are a finite number of such subterms. By the induction hypothesis, each subterm can be converted to a normal form. The deterministic composition of these normal subterms can then be converted to a normal form by Lemma 5.

The proof for the reducibility of $P\backslash A$ is similar. Thus, all finite terms can be converted to a unique normal form. □

Note that in the axiom system we have not included laws for the commutativity and associativity of $||_A$. These laws are derivable since $P||_A Q$ and $Q||_A P$ have an equivalent normal form.

The completeness of the axiom system follows from the Normal Form Lemma and the Normal Form Completeness Lemma.

Theorem 3 (Completeness) *For all terms P and Q, if $\mathcal{M}[\![P]\!] \supseteq_{\mathcal{A}} \mathcal{M}[\![Q]\!]$, then $\vdash P \sqsubseteq Q$.* □

4.2 Infinite Processes

The axiom system can be extended to include infinite processes without hiding (those containing occurrences of CHAOS or recursion) by the following axioms:

(BOT)	$\text{CHAOS} \sqsubseteq P$
(R)	$P[(\mu x.P)\backslash x] \sqsubseteq \mu x.P$
(INF)	$\forall Q \in \text{FIN}(P).Q \sqsubseteq R \Longrightarrow P \sqsubseteq R$

(BOT) states that CHAOS is the least element with respect to $\sqsubseteq$ of the domain. (R) represents the standard ordering used in reasoning about recursive processes, where $P\backslash x$ denotes the substitution of P for each occurrence of x. (INF) is an infinitary rule. It states that any property of a term P is deducible from the properties of its finite approximations $\text{FIN}(P)$. The soundness of these rules follows from the definitions of the operators and the definition of finite approximation.

The introduction of these axioms allows us to reason about infinite processes, but it does not yield a complete axiomatization of real-time processes. That is, although $P \sqsubseteq Q$ may hold for terms P and Q, a syntactic proof of this may not exist. The problem arises because many of the operators are not strict with respect to CHAOS. Thus, instead of a single infinite process corresponding to CHAOS, we have infinitely many processes that are essentially chaotic in nature. In order to obtain a complete axiomatization for infinite processes, the model can be augmented with the set of traces that lead to chaotic behavior. This

technique was used for CSP [1]. In doing so, we are able to make all operators strict with respect to CHAOS. This enables us to define unique normal forms for all processes and to obtain the desired completeness results.

5 Conclusion

We have developed a partially ordered domain for expressing the semantics of real-time processes and an axiomatization of processes in this domain. We argue that the semantic model is realistic because it allows us to represent simultaneously occurring events, to derive the absolute occurrence time of events and to model nondeterministic behavior. Furthermore, it treats the anomalous conditions of deadlock and divergence as equivalent. This is consistent with their operational behavior. The proof system, proven complete for finite terms, allows us to reason about the temporal properties of processes and is consistent with related axiomatizations of untimed models of concurrency.

In real-time computing, the temporal behavior of processes depends not only on synchronization between processes, but also on the scheduling of resources. Our model and all other existing models treat time uniformly and thus allow more possible behaviors that are possible. As a first step toward developing a model that incorporates resource availability and scheduling in real-time computation, we are studying the extensions necessary to distinguish between execution time and wait time, and investigating the incorporation of scheduling into the model.

6 Acknowledgements

The authors would like to thank Richard Beigel, Rob Gerth, Caase Huizing, Mike Reed, Bill Roscoe and Scott Smith for helpful discussions and suggestions.

References

[1] S. Brookes. *A Model for Communicating Sequential Processes.* Technical Report CMU-CS-83-149, Department of Computer Science, Carnegie–Mellon University, 1983.

[2] R. Gerth and A. Boucher. A Timed Failure Semantics for Extended Communicating Processes. In *Proceedings of ICALP '87, LNCS 267*, 1987.

[3] I. Guessarian. *Algebraic Semantics.* Volume 99 of *Lecture Notes in Computer Science*, Springer-Verlag, 1981.

[4] M. Hennessy. Acceptance Trees. *Journal of the ACM*, 32(4):896–928, October 1985.

[5] M. Hennessy. Synchronous and Asynchronous Experiments on Processes. *Journal of Information and Control*, 59(1-3):36–83, 1983.

[6] C. Hoare. *Communicating Sequential Processes.* Prentice-Hall, 1985.

[7] C. Huizing, R. Gerth, and W. de Roever. Full Abstraction of a Denotational Semantics for Real-time Concurrency. In *Proc.* 14^{th} *ACM Symposium on Principles of Programming Languages*, pages 223–237, 1987.

[8] F. Jahanian and A. Mok. A Graph-Theoretic Approach for Timing Analysis and its Implementation. *IEEE Transactions on Computers*, C-36(8):961–975, August 1987.

[9] F. Jahanian and A. Mok. Safety analysis of timing properties in real-time systems. *IEEE Transactions on Software Engineering*, SE-12(9):890–904, September 1986.

[10] R. Koymans, R. Shyamasundar, W. de Roever, R. Gerth, and S. Arun-Kumar. Compositional Semantics for Real-Time Distributed Computing. In *Logic of Programs Workshop '85, LNCS 193*, 1985.

[11] G. Milne. CIRCAL and the Representation of Communication, Concurrency, and Time. *ACM Transactions on Programming Languages and Systems*, 7(2):270–298, April 1985.

[12] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25:267–310, 1983.

[13] G. Reed and A. Roscoe. Metric Spaces as Models for Real-Time Concurrency. In *Proceedings of Math. Found. of Computer Science, LNCS 298*, 1987.