Decidability and Completeness in Real Time Process Algebra

Liang Chen
Laboratory for Foundations of Computer Science
Department of Computer Science
University of Edinburgh
Edinburgh EH9 3JZ, Scotland
cl@lfcs.dcs.ed.ac.uk

Abstract

Time variables are introduced in real time process algebras to represent a notion of time dependency in real time systems. As values of time variables may be real numbers, even a simple finite process (a process which is not recursively defined) can have infinite derivatives. In this paper, we show that strong equivalence (the largest strong bisimulation) for finite processes in real time process algebras with time variables is still decidable. The decidability is independent of the choice of time domain, aloowing time to be discrete or dense. We also present a simple relativised compositional proof system for real time process algebras. The proof system allows us to derive statements of the form $A \vdash E = F$, which means that if the first order formula A holds for any assignment of free time variables, then processes E and F are strong bisimilar under the same assignment. The proof system is again independent of the choice of time domain. We show its soundness for both discrete and dense time domains. We prove that the proof system is complete for finite processes over dense time domains. We also discuss how to restrict occurrences of time variables in time expressions such that the proof system is complete for the sublanguage over discrete time domains. Finally, we consider a possible extension of the proof system for recursively defined processes (infinite processes).

1 Introduction

Process algebras, such as CCS [16, 17], CSP [13] and ACP [3], are structured description languages for concurrent systems and have a variety of well developed semantics theories and verification methods. However, none of them consider temporal aspects of systems. Instead they deal with the quantitative aspects of time of systems in a qualitative way. There are many systems and applications for which purely qualitative specification and analysis are inadequate. The examples are real time systems in which the interactions with their environments must satisfy some hard time constraints.

Recently there are some attempts of incorporating time in the well developed process algebras [1, 6, 9, 12, 18, 22, 23, 19, 24] such that the temporal properties of systems can be explicitly formalised. In [1, 6, 9, 24], time variables or infinite choice operators have been introduced in order to express a notion of time dependency, which means that the time for some actions depends on the happening time of the previous actions. In general, real time process algebras with time variables or infinite choice operators are more expressive than those without them. In these real time process algebras, as time variables may have the real numbers as their values and infinite choice operators may be indexed by the real numbers, even a very simple finite process (a process which is not recursively defined) can have infinite derivatives. As an example, consider a process $a(x)_{10}^{100}.E$ of [6]. The process can perform action a at any time t in the interval $[10, 100]$ and after doing so it proceeds to act like process $E\{t/x\}$, where $E\{t/x\}$ is the result of substituting all free occurrences of x in E by t. The behaviour of this latter process depends on the particular value of t, the happening time of action a, if x has free occurrences in process E.

We consider, in this paper, a general and simple real time process algebra, Timed CCS [6], which is an extension of Milner's CCS with real time. There is no assumption about the underlying nature of time, allowing time to be discrete or dense. For example, we can have the set of nature numbers or the set of non-negative real numbers as the time domain. There is also no assumption of the Maximal Progress Principle. As the language allows time variables, it is very expressive. In this paper, we show that strong equivalence (the largest strong bisimulation) is still decidable for finite processes. The decidability is independent of the choice of the time domain, allowing time to be discrete or dense. We also present a relativised compositional proof system. The proof system is again independent of the choice of the time domain and is sound for both discrete and dense time domains. It is shown complete for finite processes over dense time domains, but complete only for a sublanguage over discrete time domains.

In Section 2, we present the formal syntax and semantics of Timed CCS. The semantics is given by a labelled transition system in the style of Plotkin's structural operational semantics. In Section 3, we generalize the notion of bisimulation to timed processes and study the properties of strong equivalence. To show the decidability of strong equivalence over discrete time domains as well as dense time domains, we construct in Section 4 a characteristic formula $WC(E, F)$ for E and F, called the weakest condition for E to be bisimilar to F. In Section 5, we present a sound proof system for strong equivalence. We show completeness of the proof system over dense time domains and discuss how to restrict occurrences of time variables in time expressions such that the proof system is complete for the restricted language over discrete time domains. A comparison with related work and conclusion are made in section 6.

2 Timed CCS: its Syntax and Semantics

Timed CCS is an extension of Milner's CCS [16, 17] with time. To define it, we presuppose a countable set of names Λ, ranged over by a, b, of atomic actions not containing τ. Let Act $= \Lambda \cup \{\tau\}$, ranged over by α, β. As in CCS, Λ can be partitioned into Γ, the set of names, and $\overline{\Gamma} = \{\bar{a} \mid a \in \Gamma\}$, the set of co-names, with the provision that $\bar{\bar{a}} = a$. Actions a and $\bar{a}$ are called complementary actions which form the basis of communications in the language, analogous to CCS. We also presuppose an infinite set V_t of time variables, ranged over by x, y, z. Let the time domain be $(\mathcal{T} \cup \{\infty\}, \leq)$, where $\mathcal{T}$ contains a least element 0 to represent the starting time and $\leq$ is a linear order over $\mathcal{T}$. There is no assumption about the underlying nature of time, allowing $\mathcal{T}$, for example, to be $\aleph$ (the set of natural numbers), or $\Re^{\geq 0}$ (the set of non-negative reals). Symbol ∞ is introduced to represent infinite time, where $\infty \notin \mathcal{T}$, and time expressions, ranged over by e, f, g, are defined as follows:

Definition 2.1

1 *For any $u \in \mathcal{T}$ and $x \in V_t$, u and x are time expressions.*

2 *For every $u \in \mathcal{T}$ and time expression e, $u \times e$ is a time expression.*

3 *If e and f are time expressions, then $e + f$, $e \dot{-} f$, $max(e, f)$ and $min(e, f)$ are all time expressions, where $e \dot{-} f$ represents the conditional expression if $e < f$ then 0 else $(e - f)$.*

By convention, for any time expression e, we have $e \leq \infty$, $\infty \circ e = \infty$, $max(e, \infty) = \infty$ and $min(e, \infty) = e$, where $\circ$ is $+$ or $\dot{-}$. We will write $e' - e$ in place of $e' \dot{-} e$ whenever $e \leq e'$.

Remark The decidability result of Section 4 will justify our decisions in the choice of time expressions. $\Box$

Process expressions of Timed CCS, ranged over by E, F, are defined by the following BNF expression, where $S : Act \longrightarrow Act$ is a relabelling function which satisfies $\overline{S(a)} = S(\bar{a})$ and $S(\tau) = \tau$.

$$E ::= nil \mid \alpha(x)_e^{e'}.E \mid E + F \mid E \,|\, F \mid E \backslash a \mid F[S]$$

where e' is a time expression or ∞.

All processes defined by the above BNF expression are finite processes. In Section 5.4, we will discuss recursively defined processes (infinite processes).

Process *nil* cannot do any action, but idles indefinitely. Prefix $\alpha(x)_e^{e'}.E$ represents a process which performs action α at any time t between e and e' (inclusive), where e and e' are called the lower bound and upper bound of action α, respectively. After doing so, it evolves to process $E\{t/x\}$, where $E\{t/x\}$ represents the result of substituting all free occurrence of x in E by t. Note that only the upper bounds are allowed to be the infinite time ∞ and therefore $\alpha(x)_\infty^{100.05}.nil$ is not a process. The prefix operator $\alpha(x)_e^{e'}$ in $\alpha(x)_e^{e'}.E$ binds all free occurrences of time variable x in E. This gives, in the usual sense, the notions of free and bound occurrences of time variables. We use $fv_t(E)$ to represent the set of all free time variables occurring in E. We identify those process expressions which are the same up to changes of bound time variables. Agents, ranged over by P, Q, are processes which have no free time variable.

Summation $E+F$ represents choice between processes E and F. The choice is made at the time of the first action of E or F, or at the time when only one process can idle (in this case the process which cannot delay is dropped from the future computation). Clearly the choice is deterministic with respect to time proceeding. Process $E \,|\, F$ represents the parallel composition of processes E and F. Each of them may perform actions independently, or they may synchronise on complementary actions which represent communications between them. Parallel composition is synchronous with respect to time proceeding, i.e. the parallel composition $E \,|\, F$ can delay time u only when both E and F can.

The operational semantics of Timed CCS is defined by a labelled transition system

$$(\mathcal{P}, \{\xrightarrow{(\alpha,t)} \,:\, (\alpha, t) \in (Act \cup \{\epsilon\}) \times \mathcal{T}\})$$

where $\mathcal{P}$ is the set of agents and $\epsilon \notin Act$. For convenience, we will write $P \xrightarrow{\alpha}_t P'$ in place of $(P, P') \in \xrightarrow{(\alpha,t)}$, where $\alpha \in Act$, and $P \longrightarrow_t P'$ in place of $(P, P') \in \xrightarrow{(\epsilon,t)}$.

Transition $P \xrightarrow{\alpha}_t P'$ is read as that agent P performs action α at time t and after doing so it evolves to P'. Transition $P \longrightarrow_t P'$ means that agent P idles up to time t without any action and then evolves to P'.

To define the transition rules, we first define Moller and Tofts' maximal delay time of processes before any action [18].

Definition 2.2

(1) $\lvert nil \rvert_{\mathcal{T}} = \infty$	(4) $\lvert P \,\vert\, Q \rvert_{\mathcal{T}} = min(\lvert P \rvert_{\mathcal{T}}, \lvert Q \rvert_{\mathcal{T}})$
(2) $\lvert \alpha(x)_e^{e'}.\, P \rvert_{\mathcal{T}} = e'$	(5) $\lvert P \backslash a \rvert_{\mathcal{T}} = \lvert P \rvert_{\mathcal{T}}$
(3) $\lvert P + Q \rvert_{\mathcal{T}} = max(\lvert P \rvert_{\mathcal{T}}, \lvert Q \rvert_{\mathcal{T}})$	(6) $\lvert P[S] \rvert_{\mathcal{T}} = \lvert P \rvert_{\mathcal{T}}$

All transition rules of the language are presented in Table 1. The rules are given in the natural deduction style which are read as follows: if the transition or transitions above the inference line can be inferred, then we can infer the transition below the line.

For convenience, we introduce a notation δ to represent a *dead process* $(a(x)_0^0\, nil)\backslash a$. Intuitively, the dead process δ neither performs any action nor idles. We also define some derived operators which will be used in the sequel. The first derived operator is the time prefix $(e)E$ of [18].

Definition 2.3

- $(e)nil = nil$
- $(e)(a(x)_f^{f'}.\, E) = a(x)_{f+e}^{f'+e}.\, (E\{x - e/x\})$ $\quad$ *where* $x \notin fv(e)$
- $(e)(E + F) = (e)E + (e)F$
- $(e)(E \,|\, F) = (e)E \,|\, (e)F$

$$\frac{}{\alpha(x)_v^{v'}.\,P \xrightarrow{\alpha}_t P\{t/x\}} \qquad (v \le t \le v')$$

$$\frac{}{\alpha(x)_v^{v'}.\,P \longrightarrow_t \alpha(x)_{v \dot{-} t}^{v' - t}.\,(P\{t+x/x\})} \qquad (t \le v')$$

$$\frac{}{nil \longrightarrow_t nil} \qquad \frac{P \longrightarrow_t P' \qquad Q \longrightarrow_t Q'}{P + Q \longrightarrow_t P' + Q'}$$

$$\frac{P \longrightarrow_t P'}{P + Q \longrightarrow_t P'} \quad (|Q|_{\mathcal{T}} < t) \qquad \frac{Q \longrightarrow_t Q'}{P + Q \longrightarrow_t Q'} \quad (|P|_{\mathcal{T}} < t)$$

$$\frac{P \xrightarrow{\alpha}_t P'}{P + Q \xrightarrow{\alpha}_t P'} \qquad \frac{Q \xrightarrow{\alpha}_t Q'}{P + Q \xrightarrow{\alpha}_t Q'}$$

$$\frac{P \longrightarrow_t P' \qquad Q \longrightarrow_t Q'}{P \mid Q \longrightarrow_t P' \mid Q'} \qquad \frac{P \xrightarrow{a}_t P' \qquad Q \xrightarrow{\bar{a}}_t Q'}{P \mid Q \xrightarrow{\tau}_t P' \mid Q'}$$

$$\frac{P \xrightarrow{\alpha}_t P' \qquad Q \longrightarrow_t Q'}{P \mid Q \xrightarrow{\alpha}_t P' \mid Q'} \qquad \frac{P \longrightarrow_t P' \qquad Q \xrightarrow{\alpha}_t Q'}{P \mid Q \xrightarrow{\alpha}_t P' \mid Q'}$$

$$\frac{P \longrightarrow_t P'}{P \backslash a \longrightarrow_t P' \backslash a} \qquad \frac{P \xrightarrow{\alpha}_t P'}{P \backslash a \xrightarrow{\alpha}_t P' \backslash a} \quad (\alpha \neq a \wedge \alpha \neq \bar{a})$$

$$\frac{P \longrightarrow_t P'}{P[S] \longrightarrow_t P'[S]} \qquad \frac{P \xrightarrow{\alpha}_t P'}{P[S] \xrightarrow{S(\alpha)}_t P'[S]}$$

Table 1: Operational Rules for Timed CCS

- $(e)(E\backslash a) = ((e)E)\backslash a$

- $(e)(E[S]) = ((e)E)[S]$

Intuitively, $(e)E$ represents a process which, after a delay of time e, has the same behaviour as process E.

Note that ∞ is not a time expression and therefore $(\infty)\delta$ is not a process. However, for convenience, we will allow ourselves to write $(\infty)\delta$ as syntactically identical to the process *nil* in the rest of the paper.

The second derived operator is time shift which is the relative version of that of [1].

Definition 2.4 - $e \gg nil = nil$

- $e \gg (\alpha(x)_f^{f'}.E) = \alpha(x)_{(f \dot{-} e)+(max(e,f')-f')}^{f' \dot{-} e}.(E\{e+x/x\})$ *where* $x \notin fv(e)$

- $e \gg (E + F) = e \gg E + e \gg F$

- $e \gg (E \,|\, F) = e \gg E \,|\, e \gg F$

- $e \gg (E\backslash a) = (e \gg E)\backslash a$

- $e \gg (E[S]) = (e \gg E)[S]$

Informally, for any process E, if $e \leq |E|_{\mathcal{T}}$, then $e \gg E$ represents a state of E after a delay of time e without any action. If $e > |E|_{\mathcal{T}}$, then $e \gg E$ represents the dead process δ. Note the subtlety in the definition of $e \gg (\alpha(x)_f^{f'}.E$. It ensures the lower bound $(f \dot{-} e) + (max(e, f') - f') > 0$ whenever $f' \leq e$.

3 Strong Bisimulation and its Properties

We do not wish to distinguish agents which, in some sense, have the same behaviours. The notion of bisimulation between agents captures the idea of having the same behaviours. We say two agents are not equivalent if a distinction can be detected by an experimenter who interacts with each of them.

Definition 3.1 *A binary relation $\mathcal{R}$ over agents is a strong $\mathcal{T}$-bisimulation if $(P, Q) \in \mathcal{R}$ implies that for all $\alpha \in \mathrm{Act}$ and $u \in \mathcal{T}$*

(1) *if $P \xrightarrow{\alpha}_u P'$, then there is a Q' such that $Q \xrightarrow{\alpha}_u Q'$ and $(P', Q') \in \mathcal{R}$.*

(2) *if $Q \xrightarrow{\alpha}_u Q'$, then there is a P' such that $P \xrightarrow{\alpha}_u P'$ and $(P', Q') \in \mathcal{R}$.*

(3) $|P|_{\mathcal{T}} = |Q|_{\mathcal{T}}$.

Two agents P and Q are strongly bisimilar, written as $P \sim Q$, if there is a strong $\mathcal{T}$-bisimulation $\mathcal{R}$ such that $(P, Q) \in \mathcal{R}$.

It is easy to see that $\sim$ itself is a strong $\mathcal{T}$-bisimulation, the largest $\mathcal{T}$-bisimulation. Moreover, $\sim$ is an equivalence relation, which is called strong equivalence.

Strong equivalence can be extended to any processes as follows:

Definition 3.2 *Given any processes E and F which contain at most free time variables $\vec{x} = (x_1, \ldots, x_n)$, we say $E \sim F$ if for any $\vec{t} \in \mathcal{T}^n$, $E\{\vec{t}/\vec{x}\} \sim F\{\vec{t}/\vec{x}\}$.*

All following laws can be proved in a standard way by demonstrating appropriate strong $\mathcal{T}$-bisimulations. We omit their proofs, which are mostly straightforward and tedious.

Proposition 3.3 (Laws for Prefix)

1 $a(x)_e^{e'}.E \;\sim\; a(y)_e^{e'}.(E\{y/x\})$ $\qquad$ *y is free for x in E*

2 $a(x)_e^{e'}.E \;\sim\; (e')\delta$ $\qquad$ $e > e'$

3 $a(x)_e^{e'}.E \;\sim\; a(x)_e^{e'}.E + (e')\delta$

4 $a(x)_e^{f}.E \;+ a(x)_f^{e'}.X \;\sim\; a(x)_e^{e'}.E$ $\qquad$ $e \leq f \leq e'$

Property (1) shows that α-conversion preserves strong equivalence. By (2), we know that timing may also block actions. For example, the agent $a(t)_{10}^{5}.nil$ cannot perform action a, but idles time 5 before dies.

Proposition 3.4 (Laws for Summation)

1 $E + \delta \;\sim\; E$

2 $E + E \;\sim\; E$

3 $E + F \;\sim\; F + E$

4 $E + (F + G) \;\sim\; (E + F) + G$

The above laws together just say that $(\mathcal{E}, +, \delta)$ is a monoid, where $\mathcal{E}$ is the set of all processes.

Proposition 3.5 (Laws for Restriction)

1 $nil\backslash a \;\sim\; nil$

2 $(\alpha(x)_e^{e'}.E)\backslash a \;\sim\; \alpha(x)_e^{e'}.(E\backslash a)$ $\qquad$ $\alpha \neq a \bigwedge \alpha \neq \bar{a}$

3 $(\alpha(x)_e^{e'}.E)\backslash a \;\sim\; (max(0, e'))\delta$ $\qquad$ $\alpha = a \bigvee \alpha = \bar{a}$

4 $(E + F)\backslash a \;\sim E\backslash a + F\backslash a$

Proposition 3.6 (Laws for Relabelling)

1 $((e)\delta)[S] \;\sim\; (e)\delta$

2 $(\alpha(x)_e^{e'}.\, E)[S] \;\sim\; (S(\alpha))(x)_e^{e'}.\,(E[S])$

3 $(E+F)[S] \;\sim\; E[S]+F[S]$

To simplify a presentation of expansion law, we introduce a notion of normal forms.

Definition 3.7 *A process $E \equiv \sum_{i\in I} a_i(x_i)_{e_i}^{e'_i}.\, E_i + (e)\delta$ is in normal form if for any $i \in I$, $e'_i \leq e$ and E_i is also in normal form.*

Proposition 3.8 (Expansion Law)

Given

$$E \equiv \sum_{i\in I} \alpha_i(x_i)_{e_i}^{e'_i}.\, E_i + (e)\delta \quad \text{ and } \quad F \equiv \sum_{j\in J} \beta_j(y_j)_{f_j}^{f'_j}.\, F_j + (f)\delta$$

which are both in normal form, we have
$E \,|\, F \;\sim$

$$\sum_{i\in I} \alpha_i(z_i)_{e_i}^{min(e_i,f)}.\,(E_i\{z_i/x_i\} \,|\, z_i \gg F)$$

$$+$$

$$\sum_{j\in J} \beta_j(z_j)_{f_j}^{min(f'_j,e)}.\,(z_j \gg E \,|\, F_j\{z_j/y_j\})$$

$$+$$

$$\sum_{\substack{a_i=\bar{b_j}\\ i\in I\\ j\in J}} \tau(z_{ij})_{max(e_i,f_j)}^{min(e'_i,f'_j)}.\,(E_i\{z_{ij}/x_i\} \,|\, F_j\{z_{ij}/y_j\})$$

$$+$$

$$(min(e,f))\delta$$

where for any $i \in I$, $j \in J$, z_i, z_j and z_{ij} are some fresh time variables.

Strong equivalence is a congruence relation, i.e. it is substitutive under all operators.

Proposition 3.9 *For any processes E and F, $E \sim F$ implies*

(1) $\alpha(x)_e^{e'}.\, E \sim \alpha(x)_e^{e'}.\, F$	(4) $E\backslash a \sim F\backslash a$
(2) $E+G \sim F+G$	(5) $E[S] \sim F[S]$
(3) $E \,|\, G \sim F \,|\, G$	

It is easy to see that for any process E, there is a process E' which is in normal form such that $E \sim E'$.

4 Decidability

A variety of equivalences have been proposed in the field of process theory in order to capture the behavioural aspects of processes. Various criteria exist for comparing the merits and deficiencies of these equivalences. One of the most important criteria is decidability.

For CCS, every finite process has only finite derivatives and therefore the decidability of strong equivalence for finite processes is straightforward. However for real time process algebras, as time may be dense, the situation is completely different. Even a very simple finite process can have infinite derivatives. In this section, we show that strong equivalence for finite processes in Timed CCS is still decidable and the decidability is independent of the choice of time domain.

4.1 Characteristic Formulas

For any processes E and F, we construct a formula $WC(E,F)$, the characteristic formula of E and F, of the first-order theory over some structure such that $E \sim F$ if and only if $WC(E,F)$.

Definition 4.1 *For any processes E and F, which contain at most free time variables $\vec{x} = (x_1, \ldots, x_n)$, and a formula A of the first order language, where A has at most free variables $x_1, \ldots, x_n$, we say A is a condition for E to be bisimilar to F, written as $A \models E \sim F$, if for any $\vec{t} \in \mathcal{T}^n$, $A\{\vec{t}/\vec{x}\}$ implies $E\{\vec{t}/\vec{x}\} \sim F\{\vec{t}/\vec{x}\}$.*

Definition 4.2 *A formula A is said to be a weakest condition for E to be bisimilar to F if $A \models E \sim F$ and for any formula B, $B \models E \sim F$ implies $B \rightarrow A$ holds.*

Proposition 4.3 *If A is a weakest condition for E to be bisimilar to F, then*

$$B \models E \sim F \quad \textit{if and only if} \quad B \rightarrow A$$

Proof: Straightforward. $\square$

Corollary 4.4 *If A and B are weakest conditions for E to be bisimilar to F, then $A \leftrightarrow B$ holds.*

We will identify formulas which are logically equivalent. With the above corollary, we have that if weakest conditions for E to be bisimilar to F exist, then they are unique up to this equality.

Now we show that for any processes E and F, the weakest condition for E to be bisimilar to F exists. We first consider the weakest condition $WC(E,F)$ for E to be bisimilar to F, where E and F are both in normal form.

Definition 4.5 *For any normal forms E and F, where*

$$E \equiv \sum_{i\in I} \alpha_i(x_i)^{e_i'}_{e_i}.E_i + (e)\delta \quad \textit{and} \quad F \equiv \sum_{j\in J} \beta_j(y_j)^{f_j'}_{f_j}.F_j + (f)\delta$$

let

$$
\begin{aligned}
WC(E,F) \quad = & \\
& \wedge \ e = f \\
& \wedge \bigwedge_{i\in I} (\forall x.(e_i \leq x \leq e'_i \rightarrow \bigvee_{\substack{\alpha_i=\beta_j \\ j\in J}} (f_j \leq x \leq f'_j \wedge WC(E_i\{x/x_i\}, F_j\{x/y_j\})))) \\
& \wedge \bigwedge_{j\in J} (\forall y.(f_j \leq y \leq f'_j \rightarrow \bigvee_{\substack{\alpha_i=\beta_j \\ i\in I}} (e_i \leq y \leq e'_i \wedge WC(E_i\{y/x_i\}, F_j\{y/y_j\}))))
\end{aligned}
$$

We will show next that $WC(E,F)$, where both E and F are in normal form, is the weakest condition for E to be bisimilar to F. We first show that $WC(E,F)$ is a condition for E to be bisimilar to F.

Proposition 4.6 *For any processes E and F, where both E and F are in normal form, $WC(E,F) \models E \sim F$.*

Proof: Let E be $\sum_{i\in I} \alpha_i(x_i)^{e'_i}_{e_i}.\, E_i + (e)\delta$ and F be $\sum_{j\in J} \beta_j(y_j)^{f'_j}_{f_j}.\, F_j + (f)\delta$. Suppose E and F contain at most free time variables $\vec{z} = (z_1, \ldots, z_n)$. We proceed by induction on the maximum depth of E and F, where the depth $d(E)$ of E is inductively defined as follows:

$$
\begin{array}{ll}
d((e)\delta) = 0 & d(\alpha(x)^{e'}_{e}.\, E) = 1 + d(E) \\
d(E+F) = max(d(E), d(F)) &
\end{array}
$$

For any $\vec{u} = (u_1, \ldots, u_n) \in \mathcal{T}^n$, we suppose $WC(E,F)\{\vec{u}/\vec{z}\}$ holds.

Firstly, $E\{\vec{u}/\vec{z}\} \xrightarrow{\alpha}_v P$, then there is $i \in I$ such that $\alpha = \alpha_i$, $e_i\{\vec{u}/\vec{z}\} \leq v \leq e'_i\{\vec{u}/\vec{z}\}$ and $P \equiv E_i\{\vec{u}/\vec{z}\}\{v/x_i\}$. By definition of $WC(E,F)$, there is $j \in J$, such that $\beta_j = \alpha_i$, $f_j\{\vec{u}/\vec{z}\} \leq v \leq f'_j\{\vec{u}/\vec{z}\}$ and $WC(E_i\{\vec{u}/\vec{z}\}\{v/x_i\}, F_j\{\vec{u}/\vec{z}\}\{v/y_j\})$ holds. By induction, we have $E_i\{\vec{u}/\vec{z}\}\{v/x_i\} \sim F_j\{\vec{u}/\vec{z}\}\{v/y_j\}$ hold. Clearly $F\{\vec{u}/\vec{z}\} \xrightarrow{\beta_j}_v F_j\{\vec{u}/\vec{z}\}\{v/y_j\}$.

Similarly, $F\{\vec{u}/\vec{z}\} \xrightarrow{\alpha}_v Q$ implies $E\{\vec{u}/\vec{z}\} \xrightarrow{\alpha}_v P$ with $P \sim Q$ for some P.

Finally, $e\{\vec{u}/\vec{z}\} = f\{\vec{u}/\vec{z}\}$. Therefore $E\{\vec{u}/\vec{z}\} \sim F\{\vec{u}/\vec{z}\}$. □

Now we show that $WC(E,F)$, where E and F are both in normal form, is the weakest condition for E to be bisimilar to F. To do so we first prove the following lemma.

Lemma 4.7 *For any processes E and F, where both E and F are in normal form, $E \sim F$ implies $WC(E,F)$.*

Proof: We only need to consider the case where both E and F are agents. Let E be $\sum_{i\in I} \alpha_i(x_i)^{v'_i}_{v_i}.\, E_i + (v)\delta$ and F be $\sum_{j\in J} \beta_j(y_j)^{w'_j}_{w_j}.\, F_j + (w)\delta$. Suppose $E \sim F$. We proceed by induction on the maximum depth m of E and F. It is easy to see that $E \sim F$ implies $d(E) = d(F)$.

Case $m = 0$. Then E must have form $(v)\delta$ and F must have form $(w)\delta$. Clearly $E \sim F$ implies $v = w$.

Case $m = m' + 1$.

(1) For any $i \in I$ and $u \in \mathcal{T}$, if $v_i \leq u \leq v'_i$, then $E \xrightarrow{\alpha_i}_u E_i\{u/x_i\}$. So there is a $j \in J$ such that $F \xrightarrow{\beta_j}_u F_j\{u/y_j\}$ and $F_j\{u/y_j\} \sim E_i\{u/x_i\}$, where $\alpha_i = \beta_j$ and $w_j \leq u \leq w'_j$. By induction, $WC(E_i\{u/x_i\}, F_j\{u/y_j\})$ holds.

(2) Similarly, for any $j \in J$ and $u \in \mathcal{T}$, $w_j \leq u \leq w'_j$ implies that there is a $i \in I$ such that $v_i \leq u \leq v'_i$ and $WC(E_i\{u/x_i\}, F_j\{u/y_j\})$ holds.

(3) $v = w$.

Therefore $WC(E, F)$ holds $\Box$

Proposition 4.8 *For any formula A and processes E, F, where E and F are both in normal form, $A \models E \sim F$ implies $A \to WC(E, F)$ holds.*

Proof: Let E be $\sum_{i\in I} \alpha_i(x_i)_{e_i}^{e'_i} E_i + (e)\delta$ and F be $\sum_{j\in J} \beta_j(y_j)_{f_j}^{f'_j} F_j + (f)\delta$. Suppose E and F contain at most time variables $\vec{z} = (z_1, \ldots, z_n)$ and $A \models E \sim F$. For any $\vec{u} = (u_1, \ldots, u_n) \in \mathcal{T}^n$, if $A\{\vec{u}/\vec{z}\}$ holds, then $E\{\vec{u}/\vec{z}\} \sim F\{\vec{u}/\vec{z}\}$. By Lemma 4.7, $WC(E\{\vec{u}/\vec{z}\}, F\{\vec{u}/\vec{z}\})$ holds. Hence $A \to WC(E, F)$ holds. $\Box$

Corollary 4.9 *For any processes E, E' and F, where E, E' and F are all in normal form, if $E \sim E'$ then $WC(E, F)$ is logically equivalent to $WC(E', F)$.*

Now we can consider characteristic formulas $WC(E, F)$ for any processes E and F. Recall that for any process E, there is a process E' such that $E \sim E'$, where E' is in normal form.

Definition 4.10 *For any processes E and F, let $WC(E, F)$ be $WC(E', F')$, where both E' and F' are in normal form and satisfy $E \sim E'$, $F \sim F'$.*

Clearly $WC(E, F)$ is well defined. We will show next that $WC(E, F)$ is the weakest condition for E to be bisimilar to F.

Proposition 4.11 *For any processes E and F, $WC(E, F)$ is the weakest condition for E to be bisimilar to F.*

Proof: We first show that $WC(E, F)$ is a condition for E to be bisimilar to F, i.e. $WC(E, F) \models E \sim F$ holds. Suppose E and F contain at most free time variables $\vec{x} = (x_1, \cdots, x_n)$. Let E' and F' be normal forms such that $E \sim E'$ and $F \sim F'$. For any $\vec{u} \in \mathcal{T}^n$, suppose $WC(E, F)\{\vec{u}/\vec{x}\}$ holds. Then $E'\{\vec{u}/\vec{x}\} \sim F'\{\vec{u}/\vec{x}\}$. So $E\{\vec{u}/\vec{x}\} \sim F\{\vec{u}/\vec{x}\}$ holds.

Now we suppose $A \models E \sim F$ holds. We further assume $A\{\vec{u}/\vec{x}\}$ holds, where $\vec{u} \in \mathcal{T}^n$. Then $E\{\vec{u}/\vec{x}\} \sim F\{\vec{u}/\vec{x}\}$. So $E'\{\vec{u}/\vec{x}\} \sim F'\{\vec{u}/\vec{x}\}$ and therefore $WC(E, F)\{\vec{u}/\vec{x}\}$. $\Box$

4.2 Decidability over Discrete Time Domains

In this section, we consider discrete time. Let $\mathcal{T}$ be $\aleph$, the set of natural numbers. We can define the following relations in the structure $(\aleph, +)$:

(1) Zero ($\{m \,:\, m = 0\}$): $\forall x(x + m = x)$

(2) Ordering ($\{(m,n) : m \leq n\}$): $\exists x(m + x = n)$

(3) Successor ($\{(m,n) : n = S(m)\}$):
$$\exists x \forall y \neg(x = 0) \wedge (y = 0 \vee x \leq y) \wedge m + x = n$$

(4) Subtraction ($\{(l,m,n) : n = l \dot{-} m\}$): $(l < m \wedge n = 0) \vee (l \geq m \wedge l = m + n)$

(5) Max ($\{(l,m,n) : n = max(l,m)\}$): $(l \geq m \wedge n = l) \vee (l < m \wedge n = m)$

(6) Min ($\{(l,m,n) : n = min(l,m)\}$): $(l \geq m \wedge n = m) \vee (l < m \wedge n = l)$

For any time expression e over time domain $(\aleph \cup \{\infty\}, \leq)$, we have $e < \infty$. We can read the terms of the form $e \leq \infty$ as $0 = 0$ throughout $WC(P,Q)$, for any agents P and Q. As a result, $WC(P,Q)$ is a formula of the first-order theory over structure $(\aleph, +)$.

Theorem 4.12 $\sim$ *is decidable for agents.*

Proof: The proof follows from the decidability of the first-order theory of the natural numbers with $+$ [11]. $\Box$

Lemma 4.13 *For any processes E and F which contain at most free time variables $\vec{x} = (x_1, \ldots, x_n)$, we have*

$$E \sim F \quad \textit{if and only if} \quad \forall x_1 \ldots \forall x_n . WC(E,F)$$

Proof: The proof follows from the definition of strong equivalence of processes. $\Box$

Lemma 4.14 *For any processes E and F, where E and F contain at most free time variables $x_1, \ldots, x_n$, the formula $\forall x_1 \ldots \forall x_n . WC(E,F)$ is a sentence of the first-order theory over structure $(\aleph, +)$.*

Proof: Straightforward. $\Box$

Theorem 4.15 $\sim$ *is decidable.*

Proof: Analogous to Theorem 4.12. $\Box$

Since the first-order theory of the natural numbers with $+$ and $\times$, i.e. $\mathrm{Th}(\aleph, +, \times)$, is undecidable, this justifies our decisions in the choice of time expressions.

4.3 Decidability over Dense Time Domains

In this section, we consider dense time. As an example, let $\mathcal{T}$ be $\Re^{\geq 0}$, the set of non-negative real numbers.

As for the structure $(\aleph, +)$, we can also define relations: Zero, Ordering, Subtraction, Max and Min in the structure $(\Re, +, \times)$.

Lemma 4.16 *For any processes E and F, where E and F contain at most free time variables $x_1, \ldots, x_n$, the formula $\forall x_1 \ldots \forall x_n . WC(E, F)$ is a sentence of the first-order theory over structure $(\Re, +, \times)$.*

Proof: Straightforward. □

Theorem 4.17 $\sim$ *is decidable.*

Proof: The result follows from the decidability of the first-order theory of the reals with $+$ and $\times$ [10].

5 A Proof System

In the last section we have shown that for any processes E and F, we have a characteristic formula $WC(E, F)$ such that $E \sim F$ if and only if $WC(E, F)$ holds. We use the notation $A \models E \sim F$ to indicate that the formula A is a condition for process E to be bisimilar to process F. In this section, we describe a proof system in which we derive statements of the form $A \vdash E = F$. The proof system is sound for both discrete and dense time. It will be shown complete for finite processes over dense time domains, but only complete for a restricted language over discrete time domains.

5.1 The Proof System

It is easy to see that every normal form E (process which is in normal form) can be defined by the following BNF expressions:

$$E ::= \delta \mid (e')E \mid \alpha(x)_e^{e'}.E \mid E + F$$

where e is a time expression and e' is a time expression or ∞. For convenience, we write $(\infty)E$, for any process E, as syntactical equal to the process nil. As for any process E there is a normal form E' such that $E \sim E'$, henceforth we will focus on the sublanguage defined by the above syntax.

Table 2 contains all axioms and Table 3 contains all proof rules. The proof rules are in the form

$$\frac{S_1 \ldots S_n}{S}$$

where $S_1, \ldots, S_n, S$ are statements. The rule can be read as: if all premises S_1, $\ldots$, S_n can be derived, then the conclusion S can be deduced.

We say $A \vdash X = Y$ is derivable if it can be derived from the axioms in Table 2 by using the proof rules in Table 3. For convenience, we write $A \vdash X = Y$ to assert that $A \vdash X = Y$ is derivable.

false $\vdash X = Y$	true $\vdash X = X$
true $\vdash X = X + \delta$	true $\vdash X = X + X$
true $\vdash X + Y = Y + X$	true $\vdash X + (Y + Z) = (X + Y) + Z$
true $\vdash X = (0)X$	true $\vdash (e)(e')\delta = (e + e')\delta$
true $\vdash (e)(X + Y) = (e)X + (e)Y$	true $\vdash \alpha(x)_e^{e'}.X = \alpha(x)_e^{e'}.X + (e')\delta$
$e' < e \vdash \alpha(x)_e^{e'}.X = (e')\delta$	
$e \leq y \leq e' \vdash \alpha(x)_e^{e'}.X = \alpha(x)_e^{y}.X + \alpha(x)_y^{e'}.X$	
true $\vdash \alpha(x)_e^{e'}.X = \alpha(y)_e^{e'}.X\{y/x\}$ $\quad$ (y is free for x in X)	
true $\vdash (e)(\alpha(x)_f^{f'}.X) = \alpha(x)_{f+e}^{f'+e}.X\{x - e/x\}$ $\quad$ ($x \notin fv(e)$)	

Table 2: Axioms

1	$$\frac{A \vdash X = Y \qquad A \vdash Y = Z}{A \vdash X = Z}$$
2	$$\frac{A \vdash X = Y \qquad B \vdash X = Y}{A \vee B \vdash X = Y}$$
3	$$\frac{B \vdash X = Y}{A \vdash X = Y} \qquad (A \to B)$$
4	$$\frac{A \vdash X = Y}{A \wedge e = f \vdash (e)X = (f)Y}$$
5	$$\frac{A \wedge e \leq x \leq e' \vdash X = Y}{A \wedge e = f \wedge e' = f' \vdash \alpha(x)_e^{e'}.X = \alpha(x)_f^{f'}.Y} \qquad (x \notin \mathrm{fv}(A))$$
6	$$\frac{A \vdash X = Y \qquad A \vdash X' = Y'}{A \vdash X + X' = Y + Y'}$$
7	$$\frac{A \vdash X = Y \qquad B \vdash X = Z}{A \vee B \vdash X + Y + Z = Y + Z}$$

Table 3: Proof Rules

Lemma 5.1 *The following are derivable:*

(1) $e \leq f \wedge f \leq f' \wedge f' \leq e' \vdash \alpha(x)_f^{f'}. E + \alpha(x)_e^{e'}. E = \alpha(x)_e^{e'}. E$

(2) $e \leq f \wedge f \leq e' \wedge e' \leq f' \vdash \alpha(x)_f^{f'}. E + \alpha(x)_e^{e'}. E = \alpha(x)_e^{f'}. E$

Proof: We only consider (1), as (2) is similar. For (1), we have

$$e \leq f \leq e' \vdash \alpha(x)_e^{e'}. E = \alpha(x)_e^{f}. E + \alpha(x)_f^{e'}. E$$

$$f \leq f' \leq e' \vdash \alpha(x)_f^{e'}. E = \alpha(x)_f^{f'}. E + \alpha(x)_{f'}^{e'}. E$$

Clearly

$$e \leq f \wedge f \leq f' \wedge f' \leq e' \to e \leq f \leq e'$$

and

$$e \leq f \wedge f \leq f' \wedge f' \leq e' \to f \leq f' \leq e'$$

So

$$e \leq f \wedge f \leq f' \wedge f' \leq e' \vdash \alpha(x)_e^{e'}. E = \alpha(x)_e^{f}. E + \alpha(x)_f^{e'}. E$$

and

$$e \leq f \wedge f \leq f' \wedge f' \leq e' \vdash \alpha(x)_f^{e'}. E = \alpha(x)_f^{f'}. E + \alpha(x)_{f'}^{e'}. E$$

Thus

$$e \leq f \wedge f \leq f' \wedge f' \leq e' \vdash \alpha(x)_f^{e'}. E = \alpha(x)_f^{e'}. E + \alpha(x)_f^{f'}. E$$

and

$$e \leq f \wedge f \leq f' \wedge f' \leq e' \vdash \alpha(x)_e^{e'}. E = \alpha(x)_e^{e'}. E + \alpha(x)_f^{f'}. E \qquad \square$$

Lemma 5.2

(1) *If* $A \vdash X = Y$ *and* $x \notin \mathrm{fv}(A)$*, then* $A \wedge (e \leq x \leq e') \vdash \alpha(x)_e^{e'}. X = \alpha(x)_e^{e'}. Y$.

(2) *If* $A \vdash X = Y$ *and* $x \notin \mathrm{fv}(A)$*, then* $A \wedge e = f \wedge e' = f' \vdash \alpha(x)_e^{e'}. X = \alpha(x)_f^{f'}. Y$.

Proof: We only consider (2), as (1) is similar.

Suppose $A \vdash X = Y$ and $x \notin \mathrm{fv}(A)$. Since $A \wedge e \leq x \leq e' \to A$, we have $A \wedge e \leq x \leq e' \vdash X = Y$. Thus $A \wedge e = f \wedge e' = f' \vdash \alpha(x)_e^{e'}. X = \alpha(x)_f^{f'}. Y$. $\square$

Remark Rules (1) and (2) of Lemma 5.2 are equipotent with the proof rule 5. In fact they were the version which was first proposed. Thanks to Faron Moller for suggesting the present rule. $\square$

For convenience, we will write $\vdash E = F$ in place of *true* $\vdash E = F$. Now we consider a simple example and show how the proof system works.

Example Let $E \equiv b(y)_0^{x-(2 \dot{-} x)}. \delta$, $F \equiv b(y)_0^{2x-2}. \delta$ and $G \equiv b(y)_0^{x}. \delta$. We will show that

$$\vdash a(x)_1^{10}. (E + F + G) = a(x)_1^{10}. (F + G)$$

is derivable.

$$\frac{\vdash \delta = \delta}{\dfrac{x - (2 \dot{-} x) = 2x - 2 \vdash b(y)_0^{x-(2\dot{-}x)}.\,\delta = b(y)_0^{2x-2}.\,\delta}{x \leq 2 \vdash b(y)_0^{x-(2\dot{-}x)}.\,\delta = b(y)_0^{2x-2}.\,\delta} \qquad x \leq 2 \rightarrow (x - (2\dot{-}x) = 2x - 2)}$$

and

$$\frac{\vdash \delta = \delta}{\dfrac{x - (2 \dot{-} x) = x \vdash b(y)_0^{x-(2\dot{-}x)}.\,\delta = b(y)_0^{x}.\,\delta}{x > 2 \vdash b(y)_0^{x-(2\dot{-}x)}.\,\delta = b(y)_0^{x}.\,\delta} \qquad x > 2 \rightarrow (x - (2\dot{-}x) = x)}$$

By rule 7

$$x \leq 2 \vee x > 2 \vdash E + F + G = F + G$$

and

$$\frac{\dfrac{x \leq 2 \vee x > 2 \vdash E + F + G = F + G}{1 \leq x \leq 10 \vdash E + F + G = F + G} \qquad 1 \leq x \leq 10 \rightarrow (x \leq 2 \vee x > 2)}{\vdash a(x)_1^{10}.\,(E + F + G) = a(x)_1^{10}.\,(F + G)}$$

5.2 Soundness

The soundness of the proof system is shown by the following proposition.

Proposition 5.3 (Soundness) *If* $A \vdash E = F$, *then* $A \models E \sim F$.

Proof: We only need to show that every instance of the axioms of Table 2 is valid, i.e. if $A \vdash E = F$ is an instance of an axiom, then $A \models E \sim F$, and every proof rule of Table 3 preserves the validity. The validity of every instance of the axioms follows from the properties of strong equivalence.

The proofs of rules (1), (2), (3) and (4) are straightforward.

For (5), we suppose $A \wedge e \leq x \leq e' \models X \sim Y$, where the formula A and the processes $\alpha(x)_e^{e'}.X$, $\alpha(x)_f^{f'}.Y$ contain at most free time variables $\vec{y} = (y_1, \cdots, y_n)$. For any $\vec{v} \in \mathcal{T}^n$, we also suppose $A\{\vec{v}/\vec{y}\}$ holds, $e\{\vec{v}/\vec{y}\} = f\{\vec{v}/\vec{y}\}$ and $e'\{\vec{v}/\vec{y}\} = f'\{\vec{v}/\vec{y}\}$. If $(\alpha(x)_e^{e'}.X)\{\vec{v}/\vec{y}\} \xrightarrow{\beta}_u P$, then $\beta = \alpha$, and $P \equiv X\{\vec{v}/\vec{y}\}\{u/x\}$ where $e\{\vec{v}/\vec{y}\} \leq u \leq e'\{\vec{v}/\vec{y}\}$. Clearly $(\alpha(x)_f^{f'}.Y)\{\vec{v}/\vec{y}\} \xrightarrow{\alpha}_u Y\{\vec{v}/\vec{y}\}\{u/x\}$ and $X\{\vec{v}/\vec{y}\}\{u/x\} \sim Y\{\vec{v}/\vec{y}\}\{u/x\}$ (note that $x \notin fv(A)$). By symmetry, we finish the proof of rule (5).

For (6), we suppose $A \models X \sim Y$ and $A \models X' \sim Y'$. We further assume the formula A, the processes X, X', Y and Y' contain at most free time variables $\vec{y} = (y_1, \cdots, y_n)$. For any $\vec{v} \in \mathcal{T}^n$, we suppose $A\{\vec{v}/\vec{y}\}$ holds. So we have $X\{\vec{v}/\vec{y}\} \sim Y\{\vec{v}/\vec{y}\}$ and $X'\{\vec{v}/\vec{y}\} \sim Y'\{\vec{v}/\vec{y}\}$. Therefore $(X + X')\{\vec{v}/\vec{y}\} \sim (Y + Y')\{\vec{v}/\vec{y}\}$.

The proof of rule (7) is similar to that of rule (6). $\square$

Note that the side condition of rule 5 is important, as otherwise the rule is invalid. For example, let the formula A be $1 \leq x \leq 5$, the processes X and Y be $b(y)_0^{10-x}.\, nil$ and $b(y)_0^x.\, nil$, respectively. Clearly we have $A \wedge 5 \leq x \leq 10 \models X \sim Y$, but $A \not\models a(x)_5^{10}.X \sim a(x)_5^{10}.Y$.

5.3 Completeness

The completeness result of the proof system depends heavily on the existence of normal forms.

Lemma 5.4 *For any process E, there is a normal form E' such that $\vdash E = E'$.*

Proof: We proceed by induction on the structure of E.

Case 1 $E \equiv \delta$. Clearly $\vdash \delta = (0)\delta$.

Case 2 $E \equiv (f)F$. By induction, there is a normal form $\sum_{i\in I} \alpha_i(x_i)_{e_i}^{e_i'}.\, F_i + (e)\delta$ such that

$$\vdash F = \sum_{i\in I} \alpha_i(x_i)_{e_i}^{e_i'}.\, F_i + (e)\delta$$

So

$$\vdash (f)F = (f)(\sum_{i\in I} \alpha_i(x_i)_{e_i}^{e_i'}.\, F_i + (e)\delta)$$

Therefore

$$\vdash (f)F = \sum_{i\in I} \alpha_i(x_i)_{e_i+f}^{e_i'+f}.\, F_i\{x_i - f/x_i\} + (e+f)\delta$$

and $\sum_{i\in I} \alpha_i(x_i)_{e_i+f}^{e_i'+f}.\, F_i\{x_i - f/x_i\} + (e+f)\delta$ is in normal form.

Case 3 $E \equiv \alpha(x)_e^{e'}.\, F$. By induction, there is a normal form F' such that $\vdash F = F'$. So $\vdash \alpha(x)_e^{e'}.\, F = \alpha(x)_e^{e'}.\, F'$ and therefore $\vdash \alpha(x)_e^{e'}.\, F = a(x)_e^{e'}.\, F' + (e')\delta$ where $a(x)_e^{e'}.\, F' + (e')\delta$ is in normal form.

Case 4 $E \equiv E_1 + E_2$. By induction, there is a normal form $\sum_{i\in I} \alpha_i(x_i)_{e_i}^{e_i'}.\, E_i' + (e)\delta$ such that

$$\vdash E_1 = \sum_{i\in I} \alpha_i(x_i)_{e_i}^{e_i'}.\, E_i' + (e)\delta$$

and normal form $\sum_{j\in J} \beta_j(y_j)_{f_j}^{f_j'}.\, E_j'' + (f)\delta$ such that

$$\vdash E_2 = \sum_{j\in J} \beta_j(y_j)_{f_j}^{f_j'}.\, E_j'' + (f)\delta$$

So

$$\vdash E_1 + E_2 = \sum_{i\in I} \alpha_i(x_i)_{e_i}^{e_i'}.\, E_i' + \sum_{j\in J} \beta_j(y_j)_{f_j}^{f_j'}.\, E_j'' + (max(e,f))\delta$$

where $\sum_{i\in I} \alpha_i(x_i)_{e_i}^{e_i'}.\, E_i' + \sum_{j\in J} \beta_j(y_j)_{f_j}^{f_j'}.\, E_j'' + (max(e,f))\delta$ is in normal form. $\square$

5.3.1 Completeness in Dense Time Domains

In this section, we consider dense time. As an example, we assume the time domain $(\Re^{\geq 0} \cup \{\infty\}, \leq)$. By Lemma 5.4, we can further assume that every process is in normal form.

We first show that for any processes E and F, the characteristic formula $WC(E, F)$ can be written in disjunctive normal form.

Lemma 5.5 *For any processes E and F which contain at most free time variables $(x_1, \ldots, x_n)$, $WC(E, F)$ can be written in disjunctive normal form*

$$\bigvee_{i \in I} e_1^i \leq x_1 \leq f_1^i \wedge \cdots \wedge e_n^i(x_1, \ldots, x_{n-1}) \leq x_n \leq f_n^i(x_1, \ldots, x_{n-1})$$

for some finite set I, time expressions $e_k^i(x_1, \cdots, x_{k-1})$ and $f_k^i(x_1, \cdots, x_{k-1})$, where $i \in I$, $k = 1, \cdots, n$, $e_k^i(x_1, \cdots, x_{k-1})$ and $f_k^i(x_1, \cdots, x_{k-1})$ contain at most variables $x_1, \cdots, x_{k-1}$.

Proof: We proceed by induction on the maximal depth of E and F.

For the base case, we have $WC(E, F) \equiv (e = f)$. If $e = e' + u \times x$, where $u \in \mathcal{T}$ is a non-zero constant, $x \notin \mathrm{fv}(e')$ and $x \notin \mathrm{fv}(f)$, then $e = f$ can be replaced by $(f \dot{-} e') \times u^{-1} \leq x \leq (f - e') \times u^{-1}$ which can be written in the above form. If $e = e' \dot{-} u \times x$, where $u \in \mathcal{T}$ and $u \neq 0$, $x \notin \mathrm{fv}(e')$ and $x \notin \mathrm{fv}(f)$, then $e = f$ can be replaced by $((u^{-1} \times e' \leq x \leq \infty) \wedge 0 = f) \vee (0 \leq x \leq u^{-1} \times e' \wedge u^{-1} \times (e' \dot{-} f) \leq x \leq u^{-1} \times (e' - f))$ which can also be written in the above form. All other cases are similar.

For the inductive case, we have $WC(E, F) \equiv (e = f) \wedge$

$$\bigwedge_{i \in I} (\forall x.(e_i \leq x \leq e_i' \rightarrow \bigvee_{\substack{j \in J \\ a_i = b_j}} (f_j \leq x \leq f_j' \wedge WC(E_i\{x/x_i\}, F_j\{x/y_j\}))))$$

$$\bigwedge_{j \in J} (\forall y.(f_j \leq y \leq f_j' \rightarrow \bigvee_{\substack{i \in I \\ a_i = b_j}} (e_i \leq y \leq e_i' \wedge WC(E_i\{y/x_i\}, F_j\{y/y_j\}))))$$

For any $i \in I$ and $j \in J$, by induction

$$WC(E_i\{x/x_i\}, F_j\{x/y_j\}) = \bigvee_{l \in L} (g_l \leq x \leq g_l') \wedge D_{ij}^l$$

where for every $l \in L$, $x \notin \mathrm{fv}(D_{ij}^l)$ and L is finite, and

$$WC(E_i\{y/x_i\}, F_j\{y/y_j\}) = \bigvee_{k \in K} (h_k \leq y \leq h_k') \wedge C_{ij}^k$$

where for every $k \in K$, $y \notin \mathrm{fv}(C_{ij}^k)$ and K is finite.

So $WC(E, F) = (e = f) \wedge$

$$\bigwedge_{i \in I} \forall x.(e_i \leq x \leq e_i' \rightarrow \bigvee_{\substack{j \in J \\ l \in L \\ a_i = b_j}} (max(f_i, g_l) \leq x \leq min(f_j', g_l') \wedge D_{ij}^l))$$

$$\bigwedge_{j\in J} \forall y.(max(0,f_j) \leq y \leq f'_j \rightarrow \bigvee_{\substack{i\in I\\ k\in K\\ a_i=b_j}} (max(e_i,h_k) \leq y \leq min(e'_i,h'_k) \wedge C'^{k}_{ij}))$$

For any $i \in I$, we may replace

$$\forall x.(e_i \leq x \leq e'_i \rightarrow \bigvee_{\substack{j\in J\\ l\in L\\ a_i=b_j}} (max(f_i,g_l) \leq x \leq min(f'_j,g'_l) \wedge D^l_{ij}))$$

by

$$\forall x.(\bigvee_{\substack{(u_1,v_1)\in Inv\\ \vdots\\ (u_n,v_n)\in Inv}} (e_i \geq u_1 \wedge u_1 \leq v_1 \wedge u_2 \leq v_1 \leq v_2 \wedge \cdots \wedge$$
$$u_n \leq v_{n-1} \leq v_n \wedge v_n \geq e'_i \wedge \bigwedge_{l\in L'} D^l_{ij}))$$

where

$$Inv = \{(max(f_j,g_l), min(f'_j,g'_l)) \ : \ j \in J, l \in L\}$$
$$L' = \{l \ : \ max(f_j,g_l) \in \{u_1,\ldots,u_n\}\}$$

in $WC(E,F)$.

Since $x \notin \mathrm{fv}(\{e_i,e'_i,f_j,f'_j,g_l,g'_l,D^{l'}_{ij} \ : \ j \in J, l \in L, l' \in L'\})$, the result can further be replaced by

$$\bigvee_{\substack{(u_1,v_1)\in Inv\\ \vdots\\ (u_n,v_n)\in Inv}} (e_i \geq u_1 \wedge u_1 \leq v_1 \wedge u_2 \leq v_1 \leq v_2 \wedge \cdots \wedge$$
$$u_n \leq v_{n-1} \leq v_n \wedge v_n \geq e'_i \bigwedge_{l\in L'} D^l_{ij})$$

For the other parts, we can do the same replacements. It is easy to show that the final result can be written in the required disjunctive normal form. $\Box$

Remark For discrete time domains, the lemma in general does not hold. For example, the formula $3t = 5s$ cannot be written in the required form. In the next section, we will show how to restrict the occurrences of time variables in time expressions such that we can still retain the lemma for discrete time domains. $\Box$

Proposition 5.6 *For any processes E and F, $WC(E,F) \vdash E = F$.*

Proof: We proceed by induction on the maximal depth of E and F.

For the base case, we have $E \equiv (e)\delta$ and $F \equiv (f)\delta$. Then $WC(E,F) \equiv (e = f)$ and $e = f \vdash E = F$.

For the inductive case, let

$$E \equiv \sum_{i \in I} \alpha_i(x_i)^{e'_i}_{e_i}. E_i + (e)\delta \quad \text{and} \qquad F \equiv \sum_{j \in J} \beta_j(y_j)^{f'_j}_{f_j}. F_j + (f)\delta$$

Suppose $I \neq \emptyset$. If $J = \emptyset$, then $F \equiv (f)\delta$ and

$$WC(E, F) \leftrightarrow (e = f) \wedge \bigwedge_{i \in I} (e_i > e'_i)$$

Clearly

$$(e = f) \wedge \bigwedge_{i \in I} (e_i > e'_i) \vdash E = F$$

Hence $WC(E, F) \vdash E = F$.

Now suppose $J \neq \emptyset$, for any $i \in I$, $j \in J$, by induction

$$WC(E_i\{x/x_i\}, F_j\{x/y_j\}) \vdash E_i\{x/x_i\} = F_j\{x/y_j\}$$

By the lemma, $WC(E_i\{x/x_i\}, F_j\{x/y_j\})$ can be written as

$$\bigvee_{l \in L} (g_l \leq x \leq g'_l \wedge D^l_{ij})$$

where for any $l \in L$, $x \notin \mathrm{fv}(D^l_{ij})$. So, for any $l \in L$

$$g_l \leq x \leq g'_l \wedge D^l_{ij} \vdash E_i\{x/x_i\} = F_i\{x/y_j\}$$

Thus

$$(max(e_i, f_j, g_l) \leq x \leq min(e'_i, f'_j, g'_l)) \wedge D^l_{ij} \vdash E_i\{x/x_i\} = F_i\{x/y_j\}$$

and

$$D^l_{ij} \vdash \alpha_i(x_i)^{min(e'_i, f'_j, g'_l)}_{max(e_i, f_j, g_l)}. E_i = \alpha_i(y_j)^{min(e'_i, f'_j, g'_l)}_{max(e_i, f_j, g_l)}. F_j$$

Let $Inv = \{(max(f_j, g_l), min(f'_l, g'_j)) : j \in J \& l \in L\}$. For any $(u_1, v_1), \ldots, (u_n, v_n) \in Inv$, let

$$C = e_i \geq u_1 \wedge u_1 \leq u_2 \leq v_1 \wedge \cdots \wedge u_{n-1} \leq u_n \wedge u_n \leq v_{n-1} \leq v_n \wedge v_n \geq e'_i$$

Then

$$C \vdash \alpha_i(x_i)^{e'_i}_{e_i}. E_i = \sum_{k=1}^{n} \alpha_i(x_i)^{min(e'_i, v_k)}_{max(e_i, u_k)}. E_i$$

and

$$(\bigwedge_{l \in L'} D^l_{ij}) \wedge C \vdash \alpha_i(x_i)^{e'_i}_{e_i}. E_i = \sum_{\substack{k=1 \\ \alpha_i = \beta_{\pi(k)}}}^{n} \beta_{\pi(k)}(y_{\pi(k)})^{min(e'_i, v_k)}_{max(e_i, u_k)}. F_{\pi(k)}$$

where $L' = \{l : max(f_j, g_l) \in \{u_1, \ldots, u_n\}\}$ and $\pi(k) = j$ where $u_k = max(f_j, g_l)$.

So

$$C \wedge \bigwedge_{l \in L'} D^l_{ij} \vdash \alpha_i(x_i)^{e'_i}_{e_i}. E_i + F = F$$

and

$$\bigvee_{\substack{(u_1,v_1)\in Inv \\ \vdots \\ (u_n,v_n)\in Inv}} (\bigwedge_{l\in L'} D^l_{ij} \wedge C) \vdash \alpha_i(x_i)^{e'_i}_{e_i}.\, E_i + F = F$$

where

$$C \equiv e_i \geq u_1 \wedge u_1 \leq u_2 \leq v_1 \wedge \cdots \wedge u_{n-1} \leq u_n \wedge u_n \leq v_{n-1} \leq v_n \wedge v_n \geq e'_i$$

Hence

$$\bigwedge_{i\in I}(\bigvee_{\substack{(u_1,v_1)\in Inv \\ \vdots \\ (u_n,v_n)\in Inv}} (\bigwedge_{l\in L'} D^l_{ij} \wedge C)) \wedge (e = f) \vdash E + F = F$$

and

$$WC(E,F) \vdash E + F = F$$

Similarly

$$WC(E,F) \vdash E + F = E$$

So

$$WC(E,F) \vdash E = F \qquad \Box$$

Corollary 5.7 *$A \models E \sim F$ implies $A \vdash E = F$.*

Corollary 5.8 (Completeness) *For any processes E and F, $E \sim F$* implies *$\vdash E = F$.*

5.3.2 Completeness in Discrete Time Domains

In this section, we assume time to be discrete. For example, the time domain is $(\aleph \cup \{\infty\}, \leq)$ where $\aleph$ is the set of natural numbers.

As shown in the last section, Lemma 5.5 in general does not hold for discrete time domains. As a result, the proof system is not complete for processes over discrete time domain $(\aleph \cup \{\infty\}, \leq)$. For example, consider processes $P \equiv a(t)^6_6.\, b(s)^{10}_{10}.\, (3s)\delta$ and $Q \equiv a(t)^6_6.\, b(s)^{10}_{10}.\, (5t)\delta$. Clearly we have $\models P \sim Q$, but $\vdash P = Q$ is not derivable from the proof system. However, if we restrict the occurrences of time variables in time expressions, we can still retain the lemma.

We first formalise a notion of time expressions of a process.

Notation Let e be a time expression and S be a set of time expressions. We write $e + S$ in place of the set $\{e + f : f \in S\}$ of time expressions. $\Box$

Definition 5.9 *For any process P, the set of time expressions $Exp_1(P)$ of P is inductively defined as follows:*

$$\begin{array}{ll} Exp_1(\delta) = \{0\} & Exp_1((e')E) = e' + Exp_1(E) \\ Exp_1(\alpha(x)^{e'}_{e}.\, E) = \{e, e'\} & Exp_1(E + F) = Exp_1(E) \cup Exp_1(F) \end{array}$$

Definition 5.10 *For any process E, the sets $Exp_2(E)$ and $Exp_3(E)$ of time expressions are defined as follows:*

$$
\begin{array}{ll}
Exp_2(\delta) = \emptyset & Exp_3((e')E) = Exp_1((e')E) \cup Exp_2(E) \\
Exp_3(\delta) = \emptyset & Exp_3(\alpha(x)_e^{e'}.\, E) = Exp_1(\alpha(x)_e^{e'}.\, E) \cup Exp_3(E) \\
Exp_2(al(x)_e^{e'}.\, E) = Exp_3(E) & Exp_2(E+F) = Exp_2(E) \cup Exp_2(F) \\
Exp_2((e')E) = Exp_2(E) & Exp_3(E+F) = Exp_3(E) \cup Exp_3(F)
\end{array}
$$

Proposition 5.11 *For any normal form E, where $E \equiv \sum_{i \in I} \alpha_i(x_i)_{e_i}^{e_i'}.\, E_i + (e)\delta$, we have*

$$Exp_3(E) = \bigcup_{i \in I} Exp_3(E_i) \cup \{e+0, e_i, e_i' \ : \ i \in I\}$$

Proof: Straightforward. □

For any set S of time expressions, let $Basic(S)$ be the set of time expressions resulted by eliminating max and min in S by the following procedure:

1 Let $Basic(S)$ be S.

2 If $max(e,f) \in Basic(S)$ or $min(e,f) \in Basic(S)$, then replace $max(e,f)$ or $min(e,f)$ by e and f in $Basic(S)$.

3 If $e \circ max(f,f') \in Basic(S)$, or $max(f,f') \circ e \in Basic(S)$, replace them by $e \circ f$ and $e \circ f'$, where $\circ$ is $+$ or $\dot{-}$.

4 If $e \circ min(f,f') \in Basic(S)$, or $min(f,f') \circ e \in Basic(S)$, replace them by $e \circ f$ and $e \circ f'$, where $\circ$ is $+$ or $\dot{-}$.

5 Repeat steps 2 to 4 until there is no occurrence of min and max in $Basic(S)$.

For any set of time expressions S, we say S contains only time expressions which have at most a single occurrence of the same time variables if for any time expression e of $Basic(S)$ and any time variable x, x occurs in e implies that e satisfies one of the following conditions:

(1) $e = f \circ x$ or $e = x \circ f$ for some time expression f, where $x \notin \mathrm{fv}(f)$, and $\circ$ is $+$ or $\dot{-}$.

(2) $e = e_1 \circ e_2$ for some time expressions e_1 and e_2 such that $x \notin \mathrm{fv}(e_2)$ and e_1 satisfies one of the two conditions, or $x \notin \mathrm{fv}(e_1)$ and e_2 satisfies one of the two conditions, where $\circ$ is $+$ or $\dot{-}$.

Let $\mathcal{E}'$ be the set of all processes such that for any $E \in \mathcal{E}'$, $Basic(Exp_3(E))$ contains only time expressions which have at most a single occurrence of the same time variables. The sublanguage of $\mathcal{E}'$ is still very rich. In fact, we have the following property:

Proposition 5.12 *For any processes E and F, if $E, F \in \mathcal{E}'$, then $E + F$ is still in $\mathcal{E}'$. Also if e, e' are time expressions which have at most a single occurrence of the same time variables and $E \in \mathcal{E}'$, then $\alpha(x)_e^{e'}.\, E \in \mathcal{E}'$.*

Proof: Note that $Basic(Exp_3(E + F)) = Basic(Exp_3(E)) \cup Basic(Exp_3(F))$ and $Basic(Exp_3(\alpha(x)_e^{e'}.E)) = Basic(\{e, e'\}) \cup Basic(Exp_3(E))$. The result is straightforward. □

Remark If we consider the language defined by the syntax of Section 2, we only need to restrict the definition of time expressions such that the new definition of time expressions only allows those time expressions which have a single occurrence of the same time variables, i.e. we only allow time expressions defined by the following:

1. for any $u \in \mathcal{T}$ and $x \in V_t$, u and x are time expressions;
2. for any time expression e and time variable $x \in V_t$, if x does not occur in e, then $e+t$, $t+e$, $e \dot{-} t$, $t \dot{-} e$, $max(e,t)$, $max(t,e)$, $min(e,t)$ and $min(t,e)$ are all time expressions. □

Now we can show that for the processes in $\mathcal{E}'$, Lemma 5.5 of the last section still holds.

Lemma 5.13 *For any processes E and F, where $E \in \mathcal{E}'$, $F \in \mathcal{E}'$ and E and F contain at most free time variable $x_1, \ldots, x_n$, $WC(E,F)$ can be written in a disjunctive normal form*

$$\bigvee_{i \in I} e_1^i \leq x_1 \leq f_1^i \wedge \cdots \wedge e_n^i(x_1, \ldots, x_{n-1}) \leq x_n \leq f_n^i(x_1, \ldots, x_{n-1})$$

for some finite set I, time expressions $e_k^i(x_1, \ldots, x_{k-1})$ and $f_k^i(x_1, \ldots, x_{k-1})$, where $i \in I$, $k = 1, \ldots, n$, $e_k^i(x_1, \ldots, x_{k-1})$ and $f_k^i(x_1, \ldots, x_{k-1})$ contain at most variables $x_1, \ldots, x_{k-1}$.

Proof: The proof is similar to that of Lemma 5.5 (Note the restriction over occurrences of time variables in time expressions). □

Proposition 5.14 *For any processes E and F, where $E \in \mathcal{E}'$ and $F \in \mathcal{E}'$, $WC(E,F) \vdash E = F$.*

Proof: Analogous to Proposition 5.6. □

Corollary 5.15 *For any processes $E \in \mathcal{E}'$ and $F \in \mathcal{E}'$, $A \models E \sim F$ implies $A \vdash E = F$.*

Corollary 5.16 (Completeness) *For any processes E and F, where $E \in \mathcal{E}'$ and $F \in \mathcal{E}'$, $E \sim F$* implies $\vdash E = E$.

5.4 Proofs of Recursively Defined Processes

In [6], we also allow recursively defined processes. As an example, $\mu X.E$ represents an infinite process which is recursively defined by the equation $X = E$, where E may contain process variable X. The operational rules of the recursively defined process $\mu X.E$ are:

$$\frac{P\{\mu X.E/X\} \longrightarrow_t P}{\mu X.E \longrightarrow_t P} \qquad \text{and} \qquad \frac{E\{\mu X.E/X\} \xrightarrow{\alpha}_t P}{\mu X.E \xrightarrow{\alpha}_t P}$$

We say a process E is weakly guarded if every process variable X in E is weakly guarded, where X is weakly guarded in E if every occurrence of X is in some subterm of form $\alpha(x)_e^{e'}.\, F$ of E. For example, the process $a(x)_0^{\infty}.\, X + b(y)_1^{10}.\, nil$ is weakly guarded. However the process $a(x)_0^{\infty}.\, X + X$ is not weakly guarded as the second occurrence of X is not weakly guarded in it.

In this section, we consider the proofs of recursively defined processes. We show by an example how the proof system works for recursively defined processes. To do so, the proof system needs to be augmented with an axiom

$$\mathit{true} \vdash \mu X.E = E\{\mu X.E/X\}$$

and some form of induction. We choose a very simple form of induction, namely *Unique Fixpoint Induction*:

$$\frac{\mathit{true} \vdash F = E\{F/X\}}{\mathit{true} \vdash F = \mu X.E}$$

The soundness of the axiom can be proved by demonstrating an appropriate strong $\mathcal{T}$-bisimulation. For a waekly guarded process E, the soundness of the above inductive rule follows from the property of unique solution of weakly guarded equations up to strong bisimulation [7]. However, the inductive rule in general is not valid for processes which are not weakly guarded. For example, if $E \equiv X$, then for any process F we have $F \sim X\{F/X\}$ and clearly $a(x)_0^{10}.\, nil \not\sim \mu X.X$. Also, for any agent P, we have $P + a(t)_0^{10}.\, \delta \sim (a(t)_0^{10}.\, \delta + X)\{(P + a(t)_0^{10}.\, \delta)/X\}$, but $P + a(t)_0^{10}.\, \delta \not\sim \mu X.a(t)_0^{10}.\, \delta + X$ if we let P be $b(s)_0^{10}.\, \delta$.

Now we consider processes

$$E \equiv \mu X.a(x)_1^{10}.\, (b(y)_0^{x-(2\dot{-}x)}.\, X + b(y)_0^{2x-2}.\, X + b(y)_0^{x}.\, X)$$

and

$$F \equiv \mu X.a(x)_1^{10}.\, (b(y)_0^{2x-2}.\, X + b(y)_0^{x}.\, X)$$

and show how to derive

$$\vdash E = F$$

in the extended proof system.

Since F is recursively defined and weakly guarded, by the above induction rule we only need to show that

$$\vdash E = a(x)_1^{10}.\, (b(y)_0^{2x-2}.\, E + b(x)_0^{x}.\, E)$$

However we have

$$\vdash E = a(x)_1^{10}.\,(b(y)_0^{x-(2\dot{-}x)}.\,E + b(y)_0^{2x-2}.\,P + b(y)_0^{x}.\,E)$$

By rule 1 we only need to show

$$\vdash a(x)_1^{10}.\,(b(y)_0^{x-(2\dot{-}x)}.\,E + b(y)_0^{2x-2}.\,E + b(y)_0^{x}.\,E) = a(x)_1^{10}.\,(b(y)_0^{2x-2}.\,E + b(y)_0^{x}.\,E)$$

Clearly

$$\vdash E = E$$

$$\vdash b(y)_0^{2x-2}.\,E = b(y)_0^{2x-2}.\,E$$

and

$$\vdash b(y)_0^{x}.\,E = b(y)_0^{x}.\,E$$

By rule 5, we have

$$x \geq 2 \vdash b(y)_0^{x-(2\dot{-}x)}.\,E = b(y)_0^{x}.\,E$$

and

$$x \leq 2 \vdash b(y)_0^{x-(2\dot{-}x)}.\,E = b(y)_0^{2x-2}.\,E$$

By rule 7, we have

$$x \leq 2 \vee x \geq 2 \vdash b(y)_0^{x-(2\dot{-}x)}.\,E + b(y)_0^{2x-2}.\,E + b(y)_0^{x}.\,E = b(y)_0^{2x-2}.\,E + b(y)_0^{x}.\,E$$

Since $1 \leq x \leq 10 \rightarrow x \leq 2 \bigvee x \geq 2$, by rule 3 we have

$$1 \leq x \leq 10 \vdash b(y)_0^{x-(2\dot{-}x)}.\,E + b(y)_0^{2x-2}.\,E + b(y)_0^{x}.\,E = b(y)_0^{2x-2}.\,E + b(y)_0^{x}.\,E$$

By rule 5, we have the result

$$\vdash a(x)_1^{10}.\,(b(y)_0^{x-(2\dot{-}x)}.\,E + b(y)_0^{2x-2}.\,E + b(y)_0^{x}.\,E) = a(x)_1^{10}.\,(b(y)_0^{2x-2}.\,E + b(y)_0^{x}.\,E)$$

Remark Even untimed CCS is Turing-powerful [17] and therefore no effective complete proof system can exist. By adding sufficiently powerful inductive methods for handing recursively defined processes, we would have a complete (and therefore ineffective) proof system for reasoning about real-timed processes.

6 Conclusion and Related Work

In this paper, we have shown that strong equivalence for finite processes in real time process algebras with time variables is still decidable. The decidability is independent of the choice of time domain, allowing time to be discrete or dense. We have also presented a relativised compositional proof system for real time process algebras with time variables. It is again independent of choice of the time domain. The proof system is sound for both discrete and dense time. It has been shown complete for finite processes over dense time domains, but only complete for a sublanguage over discrete time domains. We have discussed how to restrict occurrences of time variables in time expressions to get the sublanguage. There is no infinite rule in the proof system and therefore it is realistic and (we hope) useful.

Although our discussions are based on Timed CCS, the approach and results can also be applied to other real time process algebras. For example, the approach and results can be applied to a restricted language of Baeten and Bergstra's $ACP\rho I$ (only allow those prefixed integrations) [15]. As shown in [7], the restricted language of $ACP\rho I$ is closely related to Timed CCS. For real time process algebras with an assumption of the Maximal Progress Principle, such as [24], we need two extra axioms:

$$\vdash \tau(x)_e^{e+f}.X = \tau(x)_e^e.X$$

and

$$\vdash \alpha(x)_e^{e'}.X + \tau(y)_f^f.Y = \alpha(x)_e^{min(e',f)}.X + \tau(y)_f^f.Y$$

There is a simple proof system pointed out by Kim Larsen which consists of a single rule

$$\frac{WC(E,F)}{\vdash E = F}$$

It is easy to see that the proof system is sound and complete for finite processes. The soundness and completeness is independent of choice of the time domain.

There is some other work on decidability for real time process algebras. In [14], it has shown that strong equivalence over regular processes without time variables is decidable. The work of [4] has generalised that of [14] by also including parallel composition, but still not allowing time variables.

In [15], Klusener has presented a proof system for the restricted language of $ACP\rho I$ and the work is for absolute time. As there are some powerful infinite rules, the proof system, although sound and complete, is only of theoretical interest.

Acknowledgements: I would like to thank Stuart Anderson, Faron Moller and John Power for many helpful discussions and constructive suggestions, Kim Larsen for pointing out the simple proof system and providing helpful comments, Mark Jerrum for many interesting discussions on decidability.

References

[1] J.C.M. Baeten and J.A. Bergstra, *Real Time Process Algebra*, Formal Aspects of Computing 3(2), pages 142 - 188, 1991.

[2] M. Ben-Or, D. Kozen and J. Reif, *The Complexity of Elementary Algebra and Geometry*, Journal of Computer and System Sciences 32, pages 251 - 264, 1986.

[3] J. A. Bergstra & J. W. Klop, *Algebra of Communicating Processes with Abstraction*, Theoretical Computer Science 37, pages 77 - 121, 1985.

[4] K. Cerans, *Decidability of Bisimulation Equivalences for Parallel Timer Processes*, in Proc. of CAV'92, Lecture Notes in Computer Science 663, 1992.

[5] L. Chen, *Decidability and Completeness in Real-Time Processes*, Technical Report ECS-LFCS-91-185, University of Edinburgh, 1991.

[6] L. Chen, *An Interleaving Model for Real-Time Systems*, in Proc. of Symp. of Logical Foundations of Computer Science, Lecture Notes in Computer Science 620, pages 81 - 92, 1992.

[7] L. Chen, *Timed Processes: Models, Axioms and Decidability*, Ph.D Thesis, CST-101-93 (also published as ECS-LFCS-93-271), University of Edinburgh, 1993.

[8] L. Chen, *Axiomatising Real-Time Processes*, in Proc. of 9th Conf. on Mathematical Foundations of Programming Semantics, Lecture Notes in Computer Science 802, pages 215 - 229, 1993.

[9] L. Chen, S. Anderson and F. Moller, *A Timed Calculus of Communicating System*, Technical Report ECS-LFCS-90-127, University of Edinburgh, 1990.

[10] G. E. Collins, *Quantifer Elimination for Real Closed Fields by Cylindric Algebraic Decomposition*, in Proc. 2nd GI Conference on Automata Theory and Formal Language, pages 134 - 183, 1975.

[11] H. B. Enderton, **A Mathematical Introduction to Logic**, Academic Press, Inc., 1972.

[12] M. Hennessy and T. Regan, *A Temporal Process Algebra*, Technical Report 2/90 University of Sussex, 1990.

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

[14] U. Holmer, K. Larsen & Y. Wang, *Deciding Properties of Regular Real Time Processes*, in Proc. of CAV'91, Lecture Notes in Computer Science, 1991.

[15] A. S. Klusener, *Completeness in Real Time Process Algebra*, Proceedings of CONCUR'91, Lecture Notes in Computer Science 527, pages 96 - 110, 1991.

[16] R. Milner, *A Calculus of Communicating systems*, Lecture Notes in Computer Science 92, Springer-verlag, 1980.

[17] R. Milner, **Communication and Concurrency**, Prentice-Hall international, 1989.

[18] F. Moller and C. Tofts, *A Temporal Calculus of Communicating System*, Lecture Notes in Computer Science 458, pages 401 - 415, 1990.

[19] X. Nicollin, J. Richier, J. Sifakis and J. Voiron, *ATP : an Algebra for Timed Processes*, Laboratoire de Genie Informatique, IMAG-Campus, 38041 Grenoble Cedex, France, 1990.

[20] D.M.R. Park, *Concurrency and Automata on Infinite Sequences*, Lecture Notes in Computer Science 104, Springer-Verlag, 1981.

[21] G. D. Plotkin, *A Structured Approach to Operational Semantics*, DAIMI FN-19, Computer Science Department, Arhus University, Denmark, 1981.

[22] R. Reed and A. W. Roscoe, *A Timed Model for Communicating Sequential Processes*, Theoretical Computer Science 58, pages 249-261, 1988.

[23] S. Schneider, *An Operational Semantics for Timed CSP*, in Procc. of Chalmers Workshop on Concurrency, to appear in Information and Computation, 1991.

[24] Y. Wang, *Real-time Behaviour of Asynchronous Agents*, Lecture Notes in Computer Science 458, pages 502 - 520, 1990.