A Timed Calculus of Communicating Systems

by

Liang Chen
Stuart Anderson
Faron Moller

A Timed Calculus of Communicating Systems

LFCS Report Series

ECS-LFCS-90-127

LFCS
Department of Computer Science
University of Edinburgh
The King's Buildings
Edinburgh EH9 3JZ

December 1990

A Timed Calculus of Communicating Systems

Liang Chen[1] Stuart Anderson Faron Moller

Laboratory for the Foundations of Computer Science
Department of Computer Science
University of Edinburgh

Abstract: We introduce a timed calculus of communicating systems, which is an extension of CCS and allows for the expression and analysis of real-time systems using a dense time. A semantic theory based on bisimulation is developed for timed CCS. We present the expansion laws and unique solution result for guarded equations. Also we discuss a weak bisimulation which is essential for process verification.

1 Introduction

The analysis of real time systems in which interaction with the environment must satisfy some time constraints emphasizes the need to develop formal models for real-time concurrency. Examples are timeout in fault-tolerant systems and duration control in safety critical systems. Process algebras, such as CCS [6, 8], CSP [5] and ACP [1, 2], are good formal models of concurrent systems having a variety of well developed semantic theories and verification methods, but none consider the real time aspects of concurrent systems. Instead they usually deal with the quantitative aspects of time of a concurrent system in a qualitative way and ignore explicit time information. This greatly reduces the complexity of the theory of process algebras and specifying and verifying systems. However, it is not always appropriate to deal with quantitative aspects of time in a qualitative way. There are many systems and applications, specifically in the area of reactive systems, for which purely qualitative specification and analysis are inadequate. As an example, consider the case: *it is always the case that after action a happens, action b must happen within no more than 2 seconds*, which mentions time explicitly. It is not enough to deal with this question in a qualitative way.

Recently there have been several attempts to introduce a concept of time in process algebras so that temporal properties of concurrent systems can be explicitly formalised. However, many of them enforce an assumption that agents are synchronous, such as SCCS which is a timed calculus for synchronous agents in which time is modelled by internal action based on the assumption that every action takes one unit time. Others assume a global clock and require actions to happen at precise moments which are measured by that clock. In [3], times associated to actions are global and absolute and the resulting theory is far from trivial. To give an operational semantics for ACP_ρ, one must suppose the existence of a global clock. Yet others assume a discrete time domain and some regard the proceeding of a unit time as a distinguished action. However, the time

[1]The author is supported by TC Scholarship. e-mail: cl@lfcs.ed.ac.uk

action is not the same thing as the normal actions, since the normal actions may be prevented from happening by the environment and time can never be blocked. Also parallel composition is synchronous with respect to time events, however it is asynchronous for the normal actions except communications between its different components.

For the synchronous system, one may use a discrete time domain (e.g. the integers) since all the processes refer to the same global clock and events happen at certain points of time. However, for the asynchronous case, any two processes may perform actions at times which are not equal but arbitarily close to each other. Hence, we need a dense time domain for asychronous systems.

The aim of this paper is to explore the possibilities of incorporating real time (dense time) in a well developed process algebra and develop a formal timed model for asynchronous behaviour and synchronous communication. We provide an extension to Milner's CCS and express explicitly the relative time interval in which an action can happen and allow the time domain to be any arbitrary linearly ordered set. As an example,

$$a(t)\,|_1^6\,.b(s)\,|_0^{10}\,.nil(0)$$

represents the process which can at first perform the action a at any time t, $1 \leq t \leq 6$, and after that, i.e. a happens, the process can perform action b immediately, or delay some time between 0 and 10 before b happens. After b happens, the process cannot perform any action, nor let time proceed.

We develop our calculus under the following assumptions:

1. The actions take no times.

2. The maximal progress principle, i.e. communications must occur whenever they are possible. In other words, a process cannot idle if it can perform a communication.

The assumption that actions have no duration just simplifies the calculus. However, we could model actions taking time by thinking that an action must delay at least the time it will take. The assumption of maximal progress principle is for the study of the theory of abstraction in the real time context, which is essential for a process algebra serving for specification and verification of real time system.

In section 2, we first present the syntax of timed CCS, then we give a timed implementation of the Alternating Bit Protocol to demonstrate the utility of the calculus. In section 3, we present its formal semantics; the operational semantics of our language is given by a labelled transition system. In section 4, we define strong bisimulation between agent expressions and present some properties for strong bisimulation, including the unique solution result for certain recursive equations. In section 5, we define weak bisimulation between agents and present some properties for weak bisimulation. We develop the theory of weak bisimulation under the assumption of maximal progress principle.

2 The language of timed CCS

To define our timed CCS, we first introduce some notions and notations. Let Λ represent the set of atomic actions not containing action τ. We presuppose a structure on Λ that it can be divided into two parts which are isomorphic under a bijection *complementation*:$^-$, with an extension that $a = \bar{\bar{a}}$. We call $\bar{a}$ the co-action or complementary action of a. The simultaneous occurences of a pair of complementary actions represent a communication which is labelled by the internal action τ. Let $Act = \Lambda \bigcup \{\tau\}$, which is ranged over by a, b, u . $\mathcal{T}$ represents the time domain which may be any linearly ordered set. For convenience, we assume $\mathcal{T}$ represents the set of non-negative real numbers, ranged over by d, possibly indexed. We presuppose an infinite set Var_t of time variables, ranged over by r, s, t, possibly indexed. Time expressions, ranged over by e, f, are just usual arithmetic expressions on $\mathcal{T}$.

Remark For convenience, we let $\infty \in \mathcal{T}$ to represent the infinite time and have the following properties:

1. $\forall d \in \mathcal{T},\ d \leq \infty$
2. $\forall d \in \mathcal{T}$, if $d \neq \infty$, then $\infty - d \ = \ \infty \dot{-} d \ = \ \infty$.
3. $\forall d \in \mathcal{T}, \infty + d \ = \ \infty$.

Notation In the sequel, *free(e)* represents the set of all time variables of e.

We also presuppose an infinite set $Var = \{x, y, x_1, ..., y_1, ...\}$ of agent variables. The agent expressions of our timed CCS, ranged over by P, Q, possibly indexed, is defined by the following BNF expression, where $a \in Act$, an arbitrary action, $x \in Var$, an agent variable, e, e' time expressions, t a time variable and $S : \Lambda \rightarrow \Lambda$ the relabelling function, which is a bijection and satisfies $\overline{S(a)} = S(\bar{a})$, $S(\tau) = \tau$.

$$P ::= nil \mid x \mid a(t) \mid_e^{e'} .P \mid P + Q \mid P \| Q \mid P \backslash a \mid P[S] \mid \mu x.P$$

The informal interpretation of the language is

1. *nil* represents the process which can do no action, but let any time proceed.
2. x represents the process bound to the variable x.
3. $a(t) \mid_e^{e'} .P$ represents the process which can perform action a at some time between e and e', in doing so it will evolve to the process $P[d/t]$, where d is the time at which action a happens. As usual, we use $P[d/t]$ to represent the process resulting from the substitution of all free occurrences of time variable t in P by d. Time expressions e and e' in the prefix represent relative times and are called the lower bound and upper bound of action a, respectively. The occurrence of time variable t in the prefix may be regarded as time variable binding occurrence and all occurrences of time variable t in P are said to be bound by the prefix.
4. $P + Q$ represents the process which may behave like P or Q. The choice may be made at the time of the first action of P or Q, or at the moment when one summand can let time pass while another cannot. In the later case, the *stopped* summand will be dropped from the further computation.

5. $P \parallel Q$ represents the parallel composition of the two processes P and Q. Each of them may perform any action independently, or synchronise on complementary actions which represents a communication. Parallel composition is synchronous with respect to time events, i.e. any time allowed by parallel composition must be allowed by all its components

6. $P \setminus a$ represents the process which will behave as P, but will not allow the action a and its complementary action $\bar{a}$ to appear.

7. $P[S]$ represents the process resulting from P by relabling its actions using relabling function S.

8. $\mu x.P$ represents the process defined by the recursive equation $x = P$.

We may formalised the notion of time expressions of a agent expression by the following definition.

Definition 2.1 *We define function Exp(P) inductively on P as follows*:

1. $Exp(nil) = \phi$
2. $Exp(x) = \phi$
3. $Exp(a(t) \mid_e^{e'} .P) = \{e,\ e'\} \bigcup Exp(P)$
4. $Exp(P + Q) = Exp(P) \bigcup Exp(Q)$
5. $Exp(P \| Q) = Exp(P) \bigcup Exp(Q)$
6. $Exp(P \backslash a) = Exp(P)$
7. $Exp(P[S]) = Exp(P)$
8. $Exp(\mu x.P) = Exp(P)$

If $e \in Exp(P)$, we say e is a time expression of P.

Definition 2.2 *We say t is a time variable of P only if there is a time expression $e \in Exp(P)$ such that $t \in free(e)$. Also if t is a time variable of P and there is at least one occurence of t in P which is not bound by prefix, then t is a free time variable of P. If t is bound by some prefix, then it is a bound time variable of P.*

There are two kinds of variables in the definition of our agent expressions, i.e. time variables and agent variables. We say an agent expression P is closed with respect to time variables (agent variables) if there is no free time variable (agent variable) in P. Our agents are just those agent expressions which are closed with respect to time and agent variables. We use $\mathcal{P}$ to denote the set of all agents in our language and $\mathcal{E}$ to denote the set of all general agent expressions. Agent expressions with free time variables or free agent variables may be thought of as agent schemas. In the sequel, we first define an equivalence relation on agents and then extend it to the general agent expressions.

The reason for us to introduce time expressions, not just time constants, as lower and upper bounds of actions is to use an interleaving model for real time concurrent systems over dense time domain. Since our time interval for every action is relative to the time of previous event, only the initial prefixes of all summands of a sequential process will be changed as the result of time events. However, parallel composition is synchronous with respect to time events and all initial prefixes of two components will change as time proceeds. Defining parallel composition in terms of prefix and nondeterministic choice requires the introduction of time variables which refer to the times of action occurences.

For convenience, in the sequel, if P is a closed agent expression with respect to time variables, we usually write $a \mid_{e}^{e'} .P$ instead of $a(t) \mid_{e}^{e'} .P$.

Example 2.3 We can define in our language the specification of a protocol with transmission time $d_{t'}$ as:

$$Protocol_{spec} \overset{\text{def}}{=} accept \mid_{0}^{\infty} .\overline{deliver} \mid_{d_{t'}}^{\infty} .Protocol_{spec}$$

Example 2.4. In this example, we consider a timed implementation of the Alternating Bit Protocol in [8].

There are four components to our protocol, *Sender*, *Receiver* and two unreliable communication lines: *Ack* and *Trans*. We assume that *Ack* and *Trans* lines may lose (but not corrupt) messages. Messages and acknowledgments are sent tagged with bits 0 and 1 alternately.

Notation In the sequel, $\hat{b} = 1 - b$.

Sender, after accepting a message from the environment, sends it with an attached bit b (=0 or 1) along the line *Trans*. Then it awaits an appropriate acknowledgement within some time d_s, which is the time that Sender will wait after sending a message before assuming the message had been lost and retransmitting it. If it gets a correct acknowledgement within time d_s, *Sender* is prepared to accept the next message from the environment and transmits it with bit $\hat{b}$. If it gets a wrong acknowledgement or doesn't get any acknowledgement within time d_s, *Sender* retransmits the message. *Sender* is defined as follows:

$$Sender_b \overset{\text{def}}{=} accept \mid_{0}^{\infty} .\overline{send_b} \mid_{0}^{\infty} .Sending_b$$
$$Sending_b \overset{\text{def}}{=} ack_b \mid_{0}^{d_s} .Sender_{\hat{b}} + ack_{\hat{b}} \mid_{0}^{d_s} .\overline{send_b} \mid_{0}^{\infty} .Sending_b + \overline{send_b} \mid_{d_s}^{\infty} .Sending_b$$

Receiver, which we assume starts in a state awaiting a message tagged with 0, is defined as follows:

$$Receiver \overset{\text{def}}{=} trans_0 \mid_{0}^{\infty} .Deliver_0$$
$$Deliver_b \overset{\text{def}}{=} \overline{deliver} \mid_{0}^{\infty} .\overline{ack_b} \mid_{0}^{\infty} .Reply_b$$
$$Reply_b \overset{\text{def}}{=} trans_b \mid_{0}^{d_s} .\overline{ack_b} \mid_{0}^{\infty} .Reply_b + trans_{\hat{b}} \mid_{0}^{d_s} .Deliver_{\hat{b}} + \overline{ack_b} \mid_{d_s}^{\infty} .Reply_b$$

Two unreliable communication lines *Trans* and *Ack* behave as one cell buffers, except that the message on them may be lost. We assume that the transmission time is d_t

and define the unreliable communication lines as follows:

$$\begin{aligned} Trans \stackrel{\text{def}}{=} & \; send_0 \,|_0^\infty .Trans + send_0 \,|_0^\infty .\overline{trans_0} \,|_{d_t}^\infty .Trans + send_1 \,|_0^\infty .Trans \\ & + send_1 \,|_0^\infty .\overline{trans_1} \,|_{d_t}^\infty .Trans \end{aligned}$$

$$\begin{aligned} Ack \stackrel{\text{def}}{=} & \; reply_0 \,|_0^\infty .Ack + reply_0 \,|_0^\infty .\overline{ack_0} \,|_{d_t}^\infty .Ack + reply_1 \,|_0^\infty .Ack \\ & + reply_1 \,|_0^\infty .\overline{ack_1} \,|_{d_t}^\infty .Ack \end{aligned}$$

Now we can define our protocol as follows:

$$\begin{aligned} Protocol_{impl} \stackrel{\text{def}}{=} & \; (Sender_0 \parallel Trans \parallel Ack \parallel Receiver)\backslash \\ & \{send_0, send_1, trans_0, trans_1, ack_0, ack_1, reply_0, reply_1\} \end{aligned}$$

Since the transmission time is d_t, the procedure of transmitting a message consists of two transmmisions, one is for message and one is for acknowledgement. So we must assume timeout $d_s \geq 2d_t$.

3 The operational semantics of timed CCS

In order to define the meaning for our language, we use the general notion of the labelled transition system

$$(S, \{\xrightarrow{l} : l \in L\})$$

where S is the set of states, L the set of transition labels, and a transition relation $\xrightarrow{l} \subseteq S \times S$ for each $l \in L$.

In the transition system for timed CCS, let S be the set of all agents, $S \subseteq \mathcal{P}$, and $L = Act \bigcup D$, where $D = \{(d) \mid d \in \mathcal{T}\}$ represents the set of time behaviours. Our operational semantics for agents consists of the definition of each transition relation $\xrightarrow{l}$ over S. The definition for semantics is in Plotkin's structured operational semantics style [13].

To define the operational semantics for our language, we first need to define a syntactic predicate which will tell us the maximum time a process can delay before performing any action. We define a function $|\ |_T$: $\mathcal{P} \longrightarrow \mathcal{T}$ inductively as follows:

Definition 3.1

1. $|\, nil \,|_T = \infty$
2. $|\, a(t) \,|_{d_1}^{d_2} .P \,|_T = d_2 \quad (a \neq \tau)$
3. $|\, \tau(t) \,|_{d_1}^{d_2} .P \,|_T = min(d_1,\ d_2)$
4. $|\, P + Q \,|_T = max(|\, P \,|_T,\ |\, Q \,|_T)$
5. $|\, P \parallel Q \,|_T = min(|\, P \,|_T,\ |\, Q \,|_T)$
6. $|\, P\backslash a \,|_T = |\, P \,|_T$

7. $\mid P[S] \mid_T = \mid P \mid_T$

8. $\mid \mu x.P \mid_T = \mid P[\mu x.P/x] \mid_T$

Note that this definition is well defined for recursive case when all recursive variables are guarded by some prefix.

Intuitively, if $\mid P \mid_T = d$, then $\neg \exists d' \exists P'.\ (d' > d \bigwedge P \xrightarrow{(d')} P')$, where $P \xrightarrow{(d')} P'$ will be defined later.

Notation $[d_1,\ d_2] \overset{\text{def}}{=} \{d \mid d_1 \leq d \leq d_2\}$

Definition 3.2 *We define the operators $/d$ on $2^{\mathcal{T}\times\mathcal{T}}$, for each $d \in \mathcal{T}$, as follows:*

$$\mathcal{S}/d = \{[d_1,\ d_2] \mid [d_1,\ d_2] \in \mathcal{S} \bigwedge d_2 \leq d\} \bigcup \{[d_1,\ d] \mid [d_1,\ d_2] \in \mathcal{S} \bigwedge d_1 \leq d < d_2\}$$

To guarantee that communications must occur whenever they are possible. We need to define the function $\mathcal{C} :\ \mathcal{P} \times Act \longrightarrow 2^{\mathcal{T}\times\mathcal{T}}$ inductively as follows

Definition 3.3

1. $\mathcal{C}(nil,\ a) = \phi$
2. $\mathcal{C}(a(t) \mid_{d_1}^{d_2} .P,\ a) = \{[d_1,\ d_2]\} \quad (a \neq \tau\ \bigwedge\ d_1 \leq d_2)$
3. $\mathcal{C}(\tau(t) \mid_{d_1}^{d_2} .P,\ \tau) = \{[d_1,\ d_1]\} \quad (d_1 \leq d_2)$
4. $\mathcal{C}(b(t) \mid_{d_1}^{d_2} .P,\ a) = \phi \quad (a \neq b \bigvee d_1 > d_2)$
5. $\mathcal{C}(P + Q,\ a) = \mathcal{C}(P,\ a) \bigcup \mathcal{C}(Q,\ a)$
6. $\mathcal{C}(P \parallel Q,\ a) = \mathcal{C}(P,\ a)/min(|P|_T,\ |Q|_T) \bigcup \mathcal{C}(Q,\ a)/min(|P|_T,\ |Q|_T) \quad (a \neq \tau)$
7. $\mathcal{C}(P \parallel Q,\ \tau) = \mathcal{C}(P,\ \tau)/min(|P|_T,\ |Q|_T) \bigcup \mathcal{C}(Q,\ \tau)/min(|P|_T,\ |Q|_T)$
   $\bigcup \ \{[d,\ d] \mid \exists a \in Act \exists [d_1,\ d_2] \in \mathcal{C}(P,\ a) \exists [d_1',\ d_2'] \in \mathcal{C}(Q,\ \bar{a}).$
   $(([d_1,\ d_2] \bigcap [d_1',\ d_2']) \neq \phi\ \bigwedge\ d = max(d_1,\ d_1'))\}$
8. $\mathcal{C}(P \backslash b,\ a) = \phi \quad (b = a \bigvee b = \bar{a})$
9. $\mathcal{C}(P \backslash b,\ a) = \mathcal{C}(P,\ a) \quad (b \neq a \bigwedge b \neq \bar{a})$
10. $\mathcal{C}(P[S],\ a) = \{[d_1,\ d_2] | \exists b \in Act.S(b) = a\ \bigwedge\ [d_1,\ d_2] \in \mathcal{C}(P,\ b)\}$
11. $\mathcal{C}(\mu x.P,\ a) = \mathcal{C}(P[\mu x.P/x],\ a)$

Intuitively, $\mathcal{C}(P,\ a)$ represents the set of time intervals in which agent P may perform action a as its first action.

The semantics of our language is the least transition relation defined by the following operational rules.

1. $nil \xrightarrow{(d)} nil \quad (d > 0)$ 2. $a(t) \mid_0^d .P \xrightarrow{a} P[0/t] \quad (d \geq 0)$

$$3.\ a(t)\,|_{d_1}^{d_2}\,.P \xrightarrow{(d)} a(t)\,|_{d_1-d}^{d_2-d}\,.P[t+d/t] \quad (0 < d \le d_2 \bigwedge a \neq \tau)$$

$$4.\ \tau(t)\,|_{d_1}^{d_2}\,.P \xrightarrow{(d)} \tau(t)\,|_{d_1-d}^{d_2-d}\,.P[t+d/t] \quad (0 < d \le min(d_1,\ d_2))$$

$$5.\ \frac{P \xrightarrow{u} P'}{P+Q \xrightarrow{u} P'} \qquad 6.\ \frac{Q \xrightarrow{u} Q'}{P+Q \xrightarrow{u} Q'} \qquad 7.\ \frac{P \xrightarrow{(d)} P' \quad Q \xrightarrow{(d)} Q'}{P+Q \xrightarrow{(d)} P'+Q'}$$

$$8.\ \frac{P \xrightarrow{(d)} P'}{P+Q \xrightarrow{(d)} P'} \quad ((|\,Q\,|_T < d) \bigwedge \neg\exists[d',\ d'] \in \mathcal{C}(Q,\ \tau).d' < d)$$

$$9.\ \frac{Q \xrightarrow{(d)} Q'}{P+Q \xrightarrow{(d)} Q'} \quad ((|\,P\,|_T < d) \bigwedge \neg\exists[d',\ d'] \in \mathcal{C}(P,\ \tau).d' < d)$$

$$10.\ \frac{P \xrightarrow{(d)} P' \quad Q \xrightarrow{(d)} Q'}{P\|Q \xrightarrow{(d)} P'\|Q'} \quad \begin{array}{c}(\neg\exists a \in Act\exists[d_1,\ d_2] \in \mathcal{C}(P,\ a)\exists[d_1',\ d_2'] \in \mathcal{C}(Q,\ \bar{a}).\\ ([d_1,\ d_2] \bigcap [d_1',\ d_2']) \neq \phi\ \wedge\ max(d_1,\ d_1') < d)\end{array}$$

$$11.\ \frac{P \xrightarrow{u} P'}{P\|Q \xrightarrow{u} P'\|Q} \qquad 12.\ \frac{Q \xrightarrow{u} Q'}{P\|Q \xrightarrow{u} P\|Q'} \qquad 13.\ \frac{P \xrightarrow{a} P' \quad Q \xrightarrow{\bar{a}} Q'}{P\|Q \xrightarrow{\tau} P'\|Q'}$$

$$14.\ \frac{P \xrightarrow{u} P'}{P\backslash a \xrightarrow{u} P'\backslash a} \quad (u \neq a \bigwedge u \neq \bar{a}) \qquad 15.\ \frac{P \xrightarrow{(d)} P'}{P\backslash a \xrightarrow{(d)} P'\backslash a}$$

$$16.\ \frac{P \xrightarrow{(d)} P'}{P[S] \xrightarrow{(d)} P'[S]} \qquad 17.\ \frac{P \xrightarrow{u} P'}{P[S] \xrightarrow{S(u)} P'[S]}$$

$$18.\ \frac{P[\mu x.P/x] \xrightarrow{(d)} P'}{\mu x.P \xrightarrow{(d)} P'} \qquad 19.\ \frac{P[\mu x.P/x] \xrightarrow{u} P'}{\mu x.P \xrightarrow{u} P'}$$

The Operational Semantics of Timed CCS

In the above, we assume that communications should occur if they are possible, that is to say a process cannot let time proceed if it can perform a communication. So τ action should happen whenever it is ready to be performed. In this sense, we have $\tau(t)\,|_3^{10}\,.nil(0) = \tau(t)\,|_3^3\,.nil(0)$ and agent $a(t)\,|_0^5\,.P + b(t')\,|_4^{10}\,.P' \parallel \bar{a}(s)\,|_0^8\,.Q$ cannot idle, but perform a communication via the simultaneous occurences of the complementary actions a and $\bar{a}$.

Notation $nil(e) \overset{\text{def}}{=} (b\,|_0^e\,.P)\backslash b$. Clearly $nil(e)$ can do no action, but let time e proceed.

Proposition 3.4 *For any agent P, if $P \xrightarrow{(d)} P'$ and $P' \xrightarrow{(d')} P''$ for some $d,\ d' \in \mathcal{T}$ and agents P' and P'', then $P \xrightarrow{(d+d')} P''$.*

Proof It directly follows by the definition of semantics.

4 Strong bisimulation and its properties

We do not wish to distinguish agents which, in some sense, have the same behaviour. The notion of *bisimulation* between agents captures the idea of having the same behaviour.

Definition 4.1 *A binary relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a strong bisimulation if it satisfies that for any agents P, Q. $P\mathcal{R}Q$, if and only if*
for all $u \in Act$ and for all $d \in T$

$$\begin{array}{ll}
(i) & \text{if } P \xrightarrow{u} P', \text{ then } \exists Q' \text{ such that } Q \xrightarrow{u} Q' \text{ and } P'\mathcal{R}Q'. \\
(ii) & \text{if } Q \xrightarrow{u} Q', \text{ then } \exists P' \text{ such that } P \xrightarrow{u} P' \text{ and } P'\mathcal{R}Q'. \\
(iii) & \text{if } P \xrightarrow{(d)} P', \text{ then } \exists Q' \text{ such that } Q \xrightarrow{(d)} Q' \text{ and } P'\mathcal{R}Q'. \\
(iv) & \text{if } Q \xrightarrow{(d)} Q', \text{ then } \exists P' \text{ such that } P \xrightarrow{(d)} P' \text{ and } P'\mathcal{R}Q'.
\end{array}$$

For any $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$, we can define $\mathcal{F}(\mathcal{R})$ ($\mathcal{F} : \mathcal{P} \times \mathcal{P} \to \mathcal{P} \times \mathcal{P}$) to be the set of pairs $(P,\ Q)$, where $(P,\ Q) \in \mathcal{R}$, which satisfies the above clauses (i) *to* (iv). Clearly, for any binary relation $\mathcal{R}$ over $\mathcal{P}$, $\mathcal{R}$ is a strong bisimulation if and only if $\mathcal{R} \subseteq \mathcal{F}(\mathcal{R})$.

Proposition 4.2 *$\mathcal{F}$ is monotonic over the lattice of binary relations under inclusion.*

We say two agents P and Q are strong bisimulation, denoted by $P \sim Q$, if there is a strong bisimulation $\mathcal{R}$ such that $P\ \mathcal{R}\ Q$.

Definition 4.3 $\sim\, = \bigcup \{\mathcal{R}\ |\mathcal{R} \subseteq \mathcal{F}(\mathcal{R})\}$

It is clear that $\sim$ is a strong bisimulation. In fact $\sim$ is the maximum fixed-point of $\mathcal{F}$. Furthermore, $\sim$ is also an equivalent relation.

Proposition 4.4

1. *$\sim$ is the largest strong bisimulation.*
2. *$\sim$ is an equivalent relation.*

Proof (1) follows directly from the definition of $\sim$ and the fact that the union of strong bisimulations is still a strong bisimulation . For (2), note that the identity $Id_{\mathcal{P}}$ is a bisimulation and the composition and converse of bisimulations are still bisimulations.

Proposition 4.5

1. $P + Q \sim Q + P$
2. $P + (Q + R) \sim (P + Q) + R$
3. $P + nil(d) \sim P \quad (|\ P\ |_T \geq d)$
4. $nil \sim nil(\infty)$
5. $P + P \sim P$

6. $a(t)\,|_{d}^{d'}\,.P \sim a(s)\,|_{d}^{d'}\,.P[s/t] \quad (s\ is\ free\ for\ t\ in\ P)$

7. $a(t)\,|_{d}^{d_1}\,.P + a(t)\,|_{d_1}^{d'}\,.P \sim a(t)\,|_{d}^{d'}\,.P \quad (d \le d_1 \le d')$

8. $a(t)\,|_{d}^{d'}\,.P \sim nil(d') \quad (d > d')$

9. $\tau(t)\,|_{d}^{d+d'}\,.P \sim \tau(t)\,|_{d}^{d}\,.P \quad (d' \ge 0)$

10. $\tau(t)\,|_{d}^{d}\,.P + a(s)\,|_{d+d''}^{d'}\,.Q \sim \tau(t)\,|_{d}^{d}\,.P \quad (d'' > 0)$

Proof All these laws may be proved by exhibiting appropriate strong bisimulations. As an example, in the case (7), we only need to show that S is a strong bisimulation, where

$$S = \{(a(t)\,|_{d \dot{-} d''}^{d_1 \dot{-} d''}\,.P + a(t)\,|_{d_1 \dot{-} d''}^{d' \dot{-} d''}\,.P,\ a(t)\,|_{d \dot{-} d''}^{d' \dot{-} d''}\,.P) \mid 0 \le d'' \le d_1 \bigwedge d \le d_1 \le d'\} \bigcup Id_P$$

is a strong bisimulation. $\square$.

From (1)-(3), it is obvious that $(\mathcal{P},\ +,\ nil(0))$ is a monoid. (6) shows that α-conversion preserves strong bisimulation $\sim$. (7) shows that if an action a may happen within the internal $[d,\ d']$, then this interval may be divided into two continues intervals $[d,\ d_1]$ and $[d_1,\ d']$, $d \le d_1 \le d'$, and action a may happen within either of these two continues intervals. (9) and (10) reflect the applications of maximum progress principle, which says that whenever the process may perform a internal τ action, it cannot delay further.

Proposition 4.6 *If for any $d \in \mathcal{T}$, $d_1 \le d \le d_2$, $P[d/t] \sim Q[d/s]$, then*

$$a(t)\,|_{d_1}^{d_2}\,.P \sim a(s)\,|_{d_1}^{d_2}\,.Q$$

Proof It only needs to show that

$$S = \{(a(t)\,|_{d_1 \dot{-} d}^{d_2 \dot{-} d}\,.P, a(s)\,|_{d_1 \dot{-} d}^{d_2 \dot{-} d}\,.Q) \mid 0 \le d \le d_2 \bigwedge P[d'/t] \sim Q[d'/s]$$

$$for\ all\ d_1 \dot{-} d \le d' \le d_2 - d\} \bigcup \sim$$

is a bisimulation. $\square$

Recall that the occurrences of time variables in the prefix are regarded as time variable binding occurrences. Proposition 4.6 shows that prefix preserves the strong bisimulation provided that whenever the prefix action happens at any moment within its interval, the evolved agents must be bisimilar.

Proposition 4.7

1. $P \parallel Q \sim Q \parallel P$

2. $P \parallel (Q \parallel R) \sim (P \parallel Q) \parallel R$

3. $P \parallel nil \sim P$

4. $(P+Q)\backslash a \;\sim\; P\backslash a + Q\backslash a$
5. $(u(t)\,|_{d}^{d'}\,.P)\backslash a \;\sim\; u(t)\,|_{d}^{d'}\,.(P\backslash a) \qquad (u \neq a \bigwedge u \neq \bar{a})$
6. $(u(t)\,|_{d}^{d'}\,.P)\backslash a \sim nil(d') \qquad (u = a \bigvee u = \bar{a})$
7. $nil(e)\backslash a \sim nil(e)$
8. $(P+Q)[S] \sim P[S] + Q[S]$
9. $(u(t)\,|_{d}^{d'}\,.P)[S] \sim S(u)(t)\,|_{d}^{d'}\,.(P[S])$
10. $nil(e)[S] \sim nil(e)$
11. $A \sim P \qquad (A \overset{\mathrm{def}}{=} P)$
12. $\mu x.P \sim P[\mu x.P/x]$

It is clear that $(\mathcal{P},\ \|,\ nil)$ is a monoid. (4)-(6) show that restrict operators are distributed over prefix and nondeterministic choice. (8) and (9) show that relabelling operators are also distributed over prefix and nondeterministic choice.

Proof All these laws can be proved by exhibiting appropriate strong bisimulations. $\square$

Up to now, we have only defined the strong bisimulation over agents, i.e. expressions with no free time variables or free agent variables. However we may first extend naturally the definition of strong bisimulation to agent expressions which have no free agent variable, but may have free time variables.

Definition 4.8 *Let P and Q be expressions with no agent variable. P and Q contain time variables $\tilde{t}$ at most. We say that $P \sim Q$, if for all time instances $\tilde{d}$, $P[\tilde{d}/\tilde{t}] \sim Q[\tilde{d}/\tilde{t}]$.*

The strong bisimulation is substitutive under all combinators.

Proposition 4.9 *If $P \sim Q$, then for any agent expression R, which is closed with respect to agent variables,*

1. $a(s)\,|_{e}^{e'}\,.P \;\sim\; a(s)\,|_{e}^{e'}\,.Q$
2. $P + R \sim Q + R$
3. $P \| R \sim Q \| R$
4. $P\backslash a \sim Q\backslash a.$
5. $P[S] \sim Q[S].$

Proof All these properties can be showed by exhibiting appropriate strong bisimulation. For (1), suppose e and e' contain time variables $\tilde{t}$ at most, P and Q contain at most time variables $\tilde{t} \bigcup \{s\}$, we can show that

$$\{(a(s)\,|_{e[\tilde d/\tilde t]\dot- d'}^{e'[\tilde d/\tilde t]-d'}\,.P[\tilde d/\tilde t],\ a(s)\,|_{e[\tilde d/\tilde t]\dot- d'}^{e'[\tilde d/\tilde t]-d}\,.Q[\tilde d/\tilde t])\mid P\sim Q\&\ \tilde d\ \mathit{are\ any\ set\ of\ time\ instances}\ \&\ 0\le d'\le e'[\tilde d/\tilde t]\}\quad\bigcup\ \sim$$

is a strong bisimulation. For (3), suppose P, Q and R contain the time variables $\tilde t$ at most, we can show that

$$\{((P\|R)[\tilde d/\tilde t],(Q\|R)[\tilde d/\tilde t])|P\sim Q\ \mathit{and}\ \tilde d\ \mathit{are\ any\ set\ of\ time\ instances}\}\ \bigcup\ \sim$$

is a strong bisimulation. The other cases are similar. □

Propositon 4.10 (Expansion Theorem)

$$\begin{array}{rl} & \sum_{i\in I} a_i(t_i)\,|_{e_i}^{e'_i}\,.P_i\ \|\ \sum_{j\in J} b_j(s_j)\,|_{f_j}^{f'_j}\,.Q_j \\ \sim & \sum_{i\in I\&j\in J\&a_i=\bar b_j}\tau(r_{ij})\,|_{max(e_i,f_j)}^{min(e'_i,f'_j)}\,.(P_i[r_{ij}/t_i]\ \|\ Q_j[r_{ij}/s_j]) \\ + & \sum_{i\in I} a_i(t'_i)\,|_{e_i}^{min(e'_i,\bar f)}\,(P_i[t'_i/t_i]\ \|\ \sum_{j\in J} b_j(s_j)\,|_{f_j\dot- t'_i}^{f'_j-t'_i}\,.Q_j[s_j+t'_i/s_j]) \\ + & \sum_{j\in J} b_j(s'_j)\,|_{f_j}^{min(\bar e,f'_j)}\,(\sum_{i\in I} a_i(t_i)\,|_{e_i\dot- s'_j}^{e'_i-s'_i}\,.P_i[t_i+s'_j/t_i]\ \|\ Q_j[s'_j/s_j]) \end{array}$$

where $\bar e = max\{e'_i \mid i\in I\}, \bar f = max\{f'_j \mid j\in J\}$. r_{ij}, t'_i, s'_j *are fresh variables for* P_i *and* Q_j*, for any* $i\in I$, $j\in J$.

The proof of expansion theorem is in appendix.

We can now extend the definition of $\sim$ further to the agent expressions with agent variables.

Definition 4.11 *Let E and F contain agent variables $\tilde x$ at most. We say $E\sim F$, if $E[\tilde P/\tilde x]\sim F[\tilde P/\tilde x]$ for all agent expressions $\tilde P$ with no free agent variable.*

We will also use $\tilde E\sim\tilde F$ to represent component-wise bisimulation between $\tilde E$ and $\tilde F$.

The above proposition 4.9 for agent expression with no agent variables also holds for agent expressions with agent variables. The following proposition shows that $\sim$ is also preserved by recursion.

Proposition 4.12 *Let $\tilde E$ and $\tilde F$ contain agent variables $\tilde x$ at most. If $\tilde E\sim\tilde F$, then $\mu\tilde x.\tilde E\sim\mu\tilde x.\tilde F$.*

Proof We will only deal with the case of a single recursion equation. Also we assume that the agent expressions are closed with respect to time variables. Therefore, $E\sim F$ and E, F contain at most agent variable x and are closed with respect to time variables.

It will be enough to show that (by taking $G\equiv x$)

$$S=\{(G[\mu x.E/x],\ G[\mu x.F/x])\ |G\ \mathit{contains\ at\ most\ the\ variable}\ x\}$$

is a strong bisimulation up to $\sim$.

To show this, it is enough to prove that (by the symmetric arguments)

1. If $G[\mu x.E/x] \overset{(d)}{\to} P'$, then for some Q' and Q'', $G[\mu x.F/x] \overset{(d)}{\to} Q''$ with $Q'' \sim Q'$ and $(P', Q') \in S$.
2. If $G[\mu x.E/x] \overset{a}{\to} P'$, then for some Q' and Q'', $G[\mu x.F/x] \overset{a}{\to} Q''$, with $Q'' \sim Q'$ and $(P', Q') \in S$.

which may be proved by transition induction on the depth of the inference. $\square$

We now consider the solution of equations $\tilde{x} = \tilde{E}$, which under certain conditions, have unique solutions up to bisimulation.

Definition 4.13 *We say that x is guarded in E, if each occurrence of x is within some subexpression $a(t) \mid_{e}^{e'} .F$ of E, where $a \in Act$.*

Definition 4.14 We say equations $\tilde{x} = \tilde{E}$ are guarded, if all $\tilde{x}$ are guarded in $\tilde{E}$.

Clearly, for any expression E, in which x is guarded, the first action (time action or function action) is independent of the agent substituted for x.

Lemma 4.15 *For any expression E, which is closed with respect to time variables, and any agents $\tilde{P}$, if the variables $\tilde{x}$ are guarded in E and $E[\tilde{P}/\tilde{x}] \xrightarrow{a} P'$ (or $E[\tilde{P}/\tilde{x}] \xrightarrow{(d)} P'$), then P' has the form $E'[\tilde{P}/\tilde{x}]$, for some E'. Furthermore, for any agents $\tilde{Q}$, $E[\tilde{Q}/\tilde{x}] \xrightarrow{a} E'[\tilde{Q}/\tilde{x}]$ (or $E[\tilde{Q}/\tilde{x}] \xrightarrow{(d)} E'[\tilde{Q}/\tilde{x}]$)*

We may prove the lamma by using transition induction on the depth of the inference of $E[\tilde{P}/\tilde{x}] \xrightarrow{(d)} P'$, or $E[\tilde{P}/\tilde{x}] \xrightarrow{a} P'$.

The following proposition shows that under suitable conditions, if the equations $\tilde{x} = \tilde{E}$ have solutions which are agents, then they have unique solutions up to strong bisimulation.

Proposition 4.16 *Let the expressions $E_i (i \in I)$, which are closed with respect to time variables, contain at most the agent variables $x_i (i \in I)$ and all $x_j (j \in I)$ are guarded in $E_i (i \in I)$, If $\tilde{P} \sim \tilde{E}[\tilde{P}/\tilde{x}]$, $\tilde{Q} \sim \tilde{E}[\tilde{Q}/\tilde{x}]$ and $\tilde{P}$, $\tilde{Q}$ are agents, then $\tilde{P} \sim \tilde{Q}$*

Proof We need to prove $P_i \sim Q_i (i \in I)$, and this follows by taking $F \equiv x_i$, if we can show that

$$\begin{aligned} S = \{(F[\tilde{P}/\tilde{x}],\ F[\tilde{Q}/\tilde{x}]) \mid & \textit{ F contains at most agent variables } \tilde{x} \\ & \bigwedge\ \tilde{P} \textit{ and } \tilde{Q} \textit{ are agents}\} \ \bigcup\ Id_P \end{aligned}$$

is a bisimulation up to $\sim$, which may be showed by transition induction on the depth of the inference.

The next proposition shows that every guarded equation set $\tilde{x} = \tilde{E}$, where $\tilde{E}$ are closed with respect to time variables and contain at most free agent variables $\tilde{x}$, has at least one solution $\tilde{P}$.

Proposition 4.17 *For equations $\tilde{x} = \tilde{E}$, where $\tilde{E}$ are closed with respect to time variables and contain at most free agent variables $\tilde{x}$, if all $\tilde{x}$ are guarded in $\tilde{E}$, then there are some agents $\tilde{P}$ such that*

$$\tilde{P} \sim \tilde{E}[\tilde{P}/\tilde{x}]$$

Proof By induction on the number m of equations in $\tilde{x} = \tilde{E}$.

Case 1 m=1, then there is a single equation $x = E$, where E contains at most free agent variable x and x is guarded in E. Recalled that E is closed with respect to time variables. Clearly, agent $\mu x.E$ is a solution of $x = E$.

Case 2 For m+1 equations $\tilde{x} = \tilde{E}$, $x_{m+1} = E_{m+1}$. We have

$$\tilde{x} = \tilde{E}[\mu x_{m+1}.E_{m+1}/x_{m+1}]$$

By induction, there are m agents $\tilde{P}$ such that

$$\tilde{P} \sim \tilde{E}[\mu x_{m+1}.E_{m+1}/x_{m+1}][\tilde{P}/\tilde{x}]$$

Also, equation $x_{m+1} \; = \; E_{m+1}[\tilde{P}/\tilde{x}]$ has a solution

$$P_{m+1} = \mu x_{m+1}.E_{m+1}[\tilde{P}/\tilde{x}]$$

Obviousely, agents $\tilde{P}$, P_{m+1} is the solution we are looking for. □

Therefore, the guarded equations $\tilde{x} = \tilde{E}$, where $\tilde{E}$ are closed with respect to time variables and contain at most free agent variables $\tilde{x}$, have unique solutions up to strong bisimulation.

Proposition 4.18 (Unique Solution of Equations) *For guarded equations $\tilde{x} = \tilde{E}$, where $\tilde{E}$ are closed with respect to time variables and contain at most free agent variables $\tilde{x}$, there are unique agents $\tilde{P}$ (up to strong bisimulation) such that*

$$\tilde{P} \sim \tilde{E}[\tilde{P}/\tilde{x}]$$

Proof The proof follows by propositions 4.16 and 4.17 □

5 Weak bisimulation and its properties

The equivalence discussed above does not consider the problem of abstraction, i.e. every action a (including internal action τ) of one agent must be matched by an a action of the other. In the following, we define a weaker equivalence, i.e. weak bisimulation, and only require each τ action be matched by *zero* or *more* τ actions.

Definition 5.1 *We say $E \overset{(d)}{\Longrightarrow} E'$ if*

$$E(\xrightarrow{\tau})^* \overset{(d_1)}{\longrightarrow} (\xrightarrow{\tau})^* ... (\xrightarrow{\tau})^* \overset{(d_n)}{\longrightarrow} (\xrightarrow{\tau})^* E'$$

for some d_1, ..., d_n, such that $d = d_1 + ... + d_n$.

Definition 5.2 *We say $E \overset{a}{\Longrightarrow} E'$, if $E(\xrightarrow{\tau})^* \xrightarrow{a} (\xrightarrow{\tau})^* E'$*

Note that $E \overset{\epsilon}{\Longrightarrow} E'$ means that $E(\xrightarrow{\tau})^n E'$ for some $n \geq 0$.

Notation For any $a \in Act$, if $a \neq \tau$, then $\hat{a} = a$, and $\hat{\tau} = \epsilon$.

Definition 5.3 *A binary relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ over agents is a weak bisimulation if $(P,\ Q) \in \mathcal{R}$, then for any $a \in Act$ and $d \in \mathcal{T}$*

(i). whenever $P \xrightarrow{a} P'$, $Q \overset{\hat{a}}{\Longrightarrow} Q'$ and $(P', Q') \in \mathcal{R}$ for some Q'.
(ii) whenever $P \xrightarrow{(d)} P'$, $Q \overset{(d)}{\Longrightarrow} Q'$ and $(P', Q') \in \mathcal{R}$ for some Q'.
(iii) whenever $Q \xrightarrow{a} Q'$, $P \overset{\hat{a}}{\Longrightarrow} P'$ and $(P', Q') \in \mathcal{R}$ for some P'.
(iv) whenever $Q \xrightarrow{(d)} Q'$, $P \overset{(d)}{\Longrightarrow} P'$ and $(P', Q') \in \mathcal{R}$ for some P'.

Definition 5.4 *For any $P,\ Q \in \mathcal{P}$, we say P and Q are weakly bisimilar, denoted by $P \approx Q$, if there is a weak bisimulation $\mathcal{R}$ such that $(P, Q) \in \mathcal{R}$, i.e.*

$$\approx\ = \bigcup \{\mathcal{R} : \mathcal{R} \textit{ is weak bisimulation } \}$$

Proposition 5.5

(i) $\approx$ is the largest weak bisimulation.
(ii) $\approx$ is an equivalence relation.

Proof Similar to the cases of strong bisimulation.

Proposition 5.6 *$P \sim Q$ implies $P \approx Q$.*

Proof The proof directly follows the definitions of strong and weak bisimulations.

Thus all the equational laws for $\sim$ also hold for weak bisimulation $\approx$. But the converse is not generally true. Note that $\approx$ is not a congruence relation. Clearly, we have $\tau(t)\ |_2^5\ .a(s)\ |_3^4\ .nil(0)\ \approx\ a(s)\ |_5^6\ .nil(0)$, but $\tau(t)\ |_2^5\ .a(s)\ |_3^4\ .nil(0) + b(r)\ |_3^6\ .nil(0)\ \not\approx\ a(s)\ |_5^6\ .nil(0) + b(r)\ |_3^6\ .nil(0)$.

We define a set $\mathcal{P}_0$ of agents, $\mathcal{P}_0 \subset \mathcal{P}$, as follows:

Definition 5.7 $\mathcal{P}_0 \overset{\text{def}}{=} \{P \mid P \in \mathcal{P} \bigwedge \forall e \in Exp(P) \forall t \in Var_t.(t \in free(e) \longrightarrow \exists e'.\ t \notin free(e') \bigwedge (e = e' - t \bigvee e = e' \dot{-} t))\}$

The following two propositions show that for some agents of $\mathcal{P}_0$, it allows τ action to be ingnored, to some extent, in the sense of weak bisimulation. With the assumption of maximum progress principle, the timed CCS may also serve for process verification.

Proposition 5.8 *Given the agent expressions P, where $P \in \mathcal{P}_0$ and $a(t)\ |_f^{f'}\ .P$ contains at most free time variable s, if for any time expression e of P, s is a time variable of e if and only if t is a time variable of e, then*

$$\tau(s)\ |_d^d\ .a(t)\ |_f^{f'}\ .P \ \approx\ a(t)\ |_{f[d/s]+d}^{f'[d/s]+d}\ .P[0/s] \qquad (f'[d/s] \geq 0)$$

Proof It only needs to show that

$$S = \{(\tau(s)\ |_{d\dot{-}d'}^{d-d'}\ .a(t)\ |_f^{f'}\ .P,\ a(t)\ |_{f[d\dot{-}d'/s]+d-d'}^{f'[d-d'/s]+d-d'}\ .P[0/s])|0 \leq d' \leq d\} \quad \bigcup \quad Id_P$$

is a weak bisimulation.

With the help of expansion law and distributed laws for restrict operator and relabelling operator, any finite agent can be bisimilar to one which contains no parallel composition, restriction or relabelling. The following proposition generalized the above τ law to the agents which are in summation forms.

Proposition 5.9 *Given agent expressions P_i, where $P_i \in \mathcal{P}_0$ and $a_i(t_i)\ |_{f_i}^{f_i'}\ .P_i$ contains at most free time variable t for all $i \in I$, if for any P_i and its any time expression e, t is a time variable of e if and only if t_i is a time variable of e, then*

$$\tau(t)\ |_d^d\ (\sum_{i\in I} a_i(t_i)\ |_{f_i}^{f_i'}\ .P_i) \ \approx\ \sum_{i\in I} a_i\ |_{f_i[d/t]+d}^{f_i'[d/t]+d}\ .P[0/t] \qquad (f_i'[d/t] \geq 0\ for\ all\ i \in I)$$

Proof It only need to show that

$$S = \{(\tau(t)\ |_{d\dot{-}d'}^{d\dot{-}d'}\ .(\sum_{i\in I} a_i(t_i)\ |_{f_i}^{f_i'}\ .P_i,\ \ \sum_{i\in I} a_i(t_i)\ |_{f_i[d\dot{-}d'/t]+d\dot{-}d'}^{f_i'[d\dot{-}d'/t]+d\dot{-}d'}\ .P_i[0/t])\ |\ 0 \leq d' \leq d\}$$

$$\bigcup \quad Id_P$$

is a weak bisimulation. □

Like strong bisimulation $\sim$, weak bisimulation $\approx$ may also be extended to agent expressions.

Proposition 5.10 *If $P \xrightarrow{(d)} P'$ and $0 < d' < d$, then*

$$\exists Q.\ P \xrightarrow{(d')} Q \xrightarrow{(d-d')} P'$$

The proof follows directly from the definition of operational semantics.

Proposition 5.11

1. If $Q \overset{a}{\Longrightarrow} Q'$, then for any $P \in \mathcal{P}$, $P||Q \overset{a}{\Longrightarrow} P||Q'$.

2. If $Q \overset{(d)}{\Longrightarrow} Q'$, $P \xrightarrow{(d)} P'$, and $\neg\exists a \in Act \exists [d_1', d_2'] \in \mathcal{C}(P, a) \exists [d_1'', d_2''] \in \mathcal{C}(Q, \bar{a}).\,([d_1', d_2'] \bigcap [d_1'', d_2'']) \neq \phi \wedge max(d_1', d_1'') < d$, then $P \parallel Q \overset{(d)}{\Longrightarrow} P' \parallel Q'$.

The proofs directly follow from the definitions of $\overset{a}{\Longrightarrow}$, $\overset{(d)}{\Longrightarrow}$ and operational semantics, and also proposition 5.10.

Weak bisimulation is preserved by every combinator, except summation.

Proposition 5.12 If $P \approx Q$, then

1. $a(t) \mid_e^{e'} .P \approx a(t) \mid_e^{e'} .Q$
2. for all $R \in \mathcal{E}.\ P \parallel R \ \approx\ Q \parallel R$
3. $P \backslash a \ \approx\ Q \backslash a$
4. $P[S] \ \approx\ Q[S]$

Note that τ action can neither be restricted nor relabelled.

The proposition may be proved by exhibiting appropriate weak bisimulation.

6 Related work

In [4], [7] and [11], time event, or the progress of time, is modelled by a distinguished action and time is discret. However, the idea that proceeding of time is modeled by an action is not natural in an operational sense. The time event is not the same thing as the normal actions, since the normal actions may be prevented from happening by the environment and time cannot be blocked. Also parallel composition is synchronous with respect to time events, but it is asynchronous for the normal actions, except the required communication between its different components.

Baeten and Bergstra define a real time process algebra ACP_ρ [3] which is a extension of ACP. In ACP_ρ, time is dense and all actions have no duration. However, times associated to actions are global and absolute. To give an operational semantics for ACP_ρ, one must suppose the existence of a global clock. Also, they don't discuss the theory of abstraction which is essential for a process algebra serving for specification and verification of concurrent systems. In [16], Wang represents a calculus for real-time behaviour of asynchronous agents which is an extension of CCS and used dense time domain. All actions take no time and any process may idle arbitary times before willing to perform its actions. Wang develop his theory under assumption of maximal progress principle.

Acknowledgements: We would like to thank Robin Milner for many helpful comments and suggestions.

Appendix

(Proof of the Expansion Theorem) We can show that

$$
\begin{aligned}
S = \{ & (\sum_{i\in I} a_i(t_i) \,|_{e_i \dot{-} d}^{e'_i - d} .P_i \parallel \sum_{j\in J} b_j(s_j) \,|_{f_j \dot{-} d}^{f'_j - d} .Q_j, \\
& \sum_{a=\bar{b}\&i\in I\&j\in J} \tau(r_{ij}) \,|_{max(e_i,f_j)\dot{-} d}^{min(e'_i,f'_j)-d} .(P_i[r_{ij}/t_i] \parallel Q_j[r_{ij}/s_j]) \\
& + \sum_{i\in I} a_i(t'_i) \,|_{e_i \dot{-} d}^{min(e'_i,\bar{f})-d} .(P_i[t'_i/t_i] \parallel \sum_{j\in J} b_j(s_j) \,|_{f_j \dot{-} (t'_i+d)}^{f'_j-(t'_i+d)} .Q_j[s_j + t'_i/s_j]) \\
& + \sum_{j\in J} b_j(s'_j) \,|_{f_j \dot{-} d}^{min(f'_j,\bar{e})-d} .(\sum_{i\in I} a_i(t_i) \,|_{e_i \dot{-} (s'_j+d)}^{e'_i-(s'_j+d)} .P_i[t_i + s'_j/t_i] \parallel Q_j[s'_j/s_j])) \quad | \\
& \forall i \in I, j \in J. 0 \le d \le min(e'_i, f'_j) \bigwedge \bar{e} = max\{e'_i \mid i \in I\} \bigwedge \bar{f} = max\{f'_j \mid j \in J\}\} \\
& \bigcup \; Id
\end{aligned}
$$

is a strong bisimulation.

Case 1

$$\sum_{i\in I} a_i(t_i) \,|_{e_i \dot{-} d}^{e'_i - d} .P_i \| \sum_{j\in J} b_j(s_j) \,|_{f_j \dot{-} d}^{f'_j - d} .Q_j \xrightarrow{(d')} P'$$

for some d' and P'. Then

$$0 < d' \le min(max\{e'_i - d \mid i \in I'\},\ max\{f'_j - d \mid j \in J'\})$$

and P' has the form

$$\sum_{i\in I'} a_i(t_i) \,|_{e_i \dot{-} (d+d')}^{e'_i-(d+d')} .P_i[t_i + d'/t_i] \| \sum_{j\in J'} b_j(s_j) \,|_{f_j \dot{-} (d+d')}^{f'_j-(d+d')} .Q_j[s_j + d'/s_j]$$

where

$$I' = \{i \mid i \in I \bigwedge d' \le e'_i - d\} \;\; and \;\; J' = \{j \mid j \in J \bigwedge d' \le f'_j - d\}$$

Clearly,

$$
\begin{aligned}
& \textstyle\sum_{a_i=\bar{b_j}\&i\in I\&j\in J} \tau(r_{ij}) \,|_{max(e_i,f_j)\dot{-} d}^{min(e'_i,f'_j)-d} .(P_i[r_{ij}/t_i] \parallel Q_j[r_{ij}/s_j]) \\
& + \textstyle\sum_{i\in I} a_i(t'_i) \,|_{e_i \dot{-} d}^{min(e'_i,\bar{f})-d} .(P_i[t'_i/t_i] \parallel \sum_{j\in J} b_j(s_j) \,|_{f_j \dot{-} (t'_i+d)}^{f'_j-(t'_i+d)} .Q_j[s_j + t'_i/s_j]) \\
& + \textstyle\sum_{j\in J} b_j(s'_j) \,|_{f_j \dot{-} d}^{min(f'_j,\bar{e})-d} .(\sum_{i\in I} a_i(t_i) \,|_{e_i \dot{-} (s'_j+d)}^{e'_i-(s'_j+d)} .P_i[t_i + s'_j/t_i] \parallel Q_j[s'_j/s_j])
\end{aligned}
$$

$\xrightarrow{(d')}$

$$
\begin{aligned}
& \textstyle\sum_{a_i=\bar{b_j}\&i\in I'\&j\in J'} \tau(r_{ij}) \,|_{max(e_i,f_j)\dot{-}(d+d')}^{min(e'_i,f'_j)-(d+d')} .(P_i[r_{ij}/t_i][r_{ij} + d'/r_{ij}] \\
& \qquad \parallel \; Q_j[r_{ij}/s_j][r_{ij} + d'/r_{ij}]) \\
& + \textstyle\sum_{i\in I'} a_i(t'_i) \,|_{e_i \dot{-} (d+d')}^{min(e'_i,\bar{f})-(d+d')} .(P_i[t'_i/t_i][t'_i + d'/t'_i]
\end{aligned}
$$

$$\| \ \textstyle\sum_{j\in J'} b_j(s_j) \mid^{f'_j-(t'_i+d)}_{f_j\dot{-}(t'_i+d)} .Q_j[s_j+t'_i/s_j][s_j+d'/s_j])$$

$$+ \textstyle\sum_{j\in J'} b_j(s'_j) \mid^{min(f'_j,\bar{e})-(d+d')}_{f_j\dot{-}(d+d')} .(\sum_{i\in I'} a_i(t_i) \mid^{e'_i-(s'_j+d)}_{e_i\dot{-}(s'_j+d)} .P_i[t_i+s'_j/t_i][t_i+d'/t_i]$$

$$\| \ Q_j[s'_j/s_j][s'_j+d'/s'_j])$$

Since

$$\begin{array}{rcl} P_i[r_{ij}/t_i][r_{ij}+d'/r_{ij}] &=& P_i[t_i+d'/t_i][r_{ij}/t_i] \\ Q_j[r_{ij}/s_j][s_j+d'/s_j] &=& Q_j[s_j+d'/s_j][r_{ij}/s_j] \\ P_i[t'_i/t_i][t'_i+d'/t'_i] &=& P_i[t_i+d'/t_i][t'_i/t_i] \\ Q_j[s_j+t'_i/s_j][s_j+d'/s_j] &=& Q_j[s_j+d'/s_j][s_j+t'_i/s_j] \\ P_i[t_i+s'_j/t_i][t_i+d'/t_i] &=& P_i[t_i+d'/t_i][t_i+s'_j/t_i]' \\ Q_j[s'_j/s_j][s'_j+d'/s'_j] &=& Q_j[s_j+d'/s_j][s'_j/s_j] \end{array}$$

we have
$(\sum_{i\in I'}, a_i(t_i) \mid^{e'_i-(d+d')}_{e_i\dot{-}(d+d')} .P_i[t_i+d'/t_i] \ \| \ \sum_{j\in J'} b_j(s_j) \mid^{f'_j-(d+d')}_{f_j\dot{-}(d+d')} .Q_j[s_j+d'/s_j],$

$$\textstyle\sum_{a_i=\bar{b}_j\&i\in I'\&j\in J'} \tau(r_{ij}) \mid^{min(e'_i,f'_j)-(d+d')}_{max(e_i,f_j)\dot{-}(d+d')} .(P_i[r_{ij}/t_i][r_{ij}+d'/r_{ij}]$$

$$\| \ Q_j[r_{ij}/s_j][r_{ij}+d'/r_{ij}])$$

$$+ \textstyle\sum_{i\in I'} a_i(t'_i) \mid^{min(e'_i,\bar{f})-(d+d')}_{e_i\dot{-}(d+d')} .(P_i[t'_i/t_i][t'_i+d'/t'_i]$$

$$\| \ \textstyle\sum_{j\in J'} b_j(s_j) \mid^{f'_j-(t'_i+d)}_{f_j\dot{-}(t'_i+d)} .Q_j[s_j+t'_i/s_j][s_j+d'/s_j])$$

$$+ \textstyle\sum_{j\in J'} b_j(s'_j) \mid^{min(f'_j,\bar{e})-(d+d')}_{f_j\dot{-}(d+d')} .(\sum_{i\in I'} a_i(t_i) \mid^{e'_i-(s'_j+d)}_{e_i\dot{-}(s'_j+d)} .P_i[t_i+s'_j/t_i][t_i+d'/t_i]$$

$$\| \ Q_j[s'_j/s_j][s'_j+d'/s'_j])) \quad \in S$$

Case 2 $\sum_{i\in I} a_i(t_i) \mid^{e'_i-d}_{e_i\dot{-}d} .P_i \ \| \ \sum_{j\in J} b_j(s_j) \mid^{f'_j-d}_{f_j\dot{-}d} .Q_j \xrightarrow{u} R$ for some u and R,

Case 2.1

$$\sum_{i\in I} a_i(t_i) \mid^{e'_i-d}_{e_i\dot{-}d} .P_i \xrightarrow{a_i} P', \ \ \sum_{j\in J} b_j(s_j) \mid^{f'_j-d}_{f_j\dot{-}d} .Q_j \xrightarrow{b_j} Q'$$

$$a_i=\bar{b}_j, \ \ u=\tau \ \ and \ \ R=P' \ \| \ Q' = P_i[0/t_i] \ \| \ Q_j[0/s_j]$$

Since $e_i\dot{-}d = f_j\dot{-}d = 0$, we have

$$\textstyle\sum_{a_i=\bar{b}_j\&i\in I\&j\in J} \tau(r_{ij}) \mid^{min(e'_i,f'_j)-d}_{max(e_i,f_j)\dot{-}d} .(P_i[r_{ij}/t_i] \ \| \ Q_j[r_{ij}/s_j])$$

$$+ \ \textstyle\sum_{i\in I} a_i(t'_i) \mid^{min(e'_i,\bar{f})-d}_{e_i\dot{-}d} .(P_i[t'_i/t_i] \ \| \ \sum_{j\in J} b_j(s_j) \mid^{f'_j-(t'_i+d)}_{f_j\dot{-}(t'_i+d)} .Q_j[s_j+t'_i/s_j])$$

$$+ \ \textstyle\sum_{j\in J} b_j(s'_j) \mid^{min(f'_j,\bar{e})-d}_{f_j\dot{-}d} .(\sum_{i\in I} a_i(t_i) \mid^{e'_i-(s'_j+d)}_{e_i\dot{-}(s'_j+d)} .P_i[t_i+s'_j/t_i] \ \| \ Q_j[s'_j/s_j])$$

$$\xrightarrow{\tau}$$

$$P_i[0/t_i] \parallel Q_j[0/s_j]$$

and

$$(P_i[0/t_i] \parallel Q_j[0/s_j], P_i[0/t_i] \parallel Q_j[0/s_j]) \in S$$

Case 2.2

$$\sum_{i\in I} a_i(t_i) \mid_{e_i \dot{-} d}^{e_i' - d} .P_i \xrightarrow{a_i} P',\ u = a_i \textit{ and } R = P_i[0/t_i] \| \sum_{j \in J} b_j(s_j) \mid_{f_j \dot{-} d}^{f_j' - d} .Q_j$$

Since $e_i \dot{-} d = 0$, we have

$$\textstyle\sum_{a_i=\bar{b_j} \& i\in I \& j \in J} \tau(r_{ij}) \mid_{max(e_i,f_j) \dot{-} d}^{min(e_i',f_j')-d} .(P_i[r_{ij}/t_i] \| Q_j[r_{ij}/s_j])$$

$$+\ \textstyle\sum_{i\in I} a_i(t_i') \mid_{e_i \dot{-} d}^{min(e_i',\bar{f})-d} .(P_i[t_i'/t_i] \parallel \sum_{j\in J} b_j(s_j) \mid_{f_j \dot{-} (t_i'+d)}^{f_j' - (t_i'+d)} .Q_j[s_j + t_i'/s_j])$$

$$+\ \textstyle\sum_{j\in J} b_j(s_j') \mid_{f_j \dot{-} d}^{min(f_j',\bar{e})-d} .(\sum_{i\in I} a_i(t_i) \mid_{e_i \dot{-} (s_j'+d)}^{e_i' - (s_j'+d)} .P_i[t_i + s_j'/t_i] \parallel Q_j[s_j'/s_j])$$

$$\xrightarrow{a_i}$$

$$P_i[0/t_i] \parallel \textstyle\sum_{j\in J} b_j(s_j) \mid_{f_j \dot{-} d}^{f_j' - d} .Q_j$$

and

$$(P_i[0/t_i] \parallel \sum_{j\in J} b_j(s_j) \mid_{f_j \dot{-} d}^{f_j' - d} .Q_j,\ P_i[0/t_i] \parallel \sum_{j\in J} b_j(s_j) \mid_{f_j \dot{-} d}^{f_j' - d} .Q_j) \in S$$

Case 2.3 $\sum_{j\in J} b_j(s_j) \mid_{f_j \dot{-} d}^{f_j' - d} .Q_j \xrightarrow{b_j} Q',\ u = b_j \textit{ and } R = \sum_{i\in I} a_i(t_i) \mid_{e_i \dot{-} d}^{e_i' - d} .P_i \parallel Q_j[0/s_j]$.
samilar to case 2.2.

The other conditions are similar and the case for $(P, P) \in S$ is trivial. □

References

[1] J.A. Bergstra and J.W. Klop(1984), Process Algebra for Synchronous Communication, Information and Control 60(1-3), pp. 109-137

[2] J.A. Bergstra and J.W. Klop(1985), Algebra of Communicating Processes with Abstraction, Theoretical Computer Science, 37, pp. 77-121

[3] J.C.M. Baeten and J.A. Bergstra(1989), Real Time Process Algebra, Preliminary Version, draft 10/20/89

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

[5] C.A.R. Hoare(1978), Communicating Sequential Processes, Communications of ACM, vol. 21, no. 8, pp. 666-677

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

[7] R. Milner(1983), Calculi for Synchrony and Asynchrony, Theoretical Computer Science, 25, pp. 267-310

[8] R. Milner(1989), Communication and Concurrency, Prentice-Hall international

[9] R. Milner(1989), A Complete Axiomatisation for Observational Congruence of Finite-State Behaviours, Information and Computation 81, pp. 227-247

[10] F. Moller and C. Tofts(1989), A Temporal Calculus of Communicating System, Technical Report LFCS-89-104, University of Edinburgh

[11] X. Nicollin, J.L. Richier, J. Sifakis, J. Voiron(1989), ATP: an Algebra for Timed Processes, Technical Report, Grenoble

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

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

[14] C. Tofts(1988), Temporal Ordering for Concurrency, Technical Report LFCS-88-49, University of Edinburgh

[15] C. Tofts(1989), Timing Concurrent Processes, Technical Report LFCS-89-103, University of Edinburgh

[16] Y. Wang(1990), Real-time Behaviour of Asynchronous Agents, draft, Dept. of Computer Science, Chalmers University of Technology and the University of Gotebory, Sweden