数理解析研究所講究録
996巻 1997年 23-38

Speed-Sensitive Orders for Communicating Processes

Ichiro Satoh*

Department of Information Sciences, Ochanomizu University
2-1-1 Otsuka Bunkyo-ku Tokyo 112 Japan

Abstract

This paper proposes two process calculi for synchronously communicating real-time processes and asynchronously ones. They are extensions of existing process calculi with the ability to express a time-dependent constructor, representing execution time and delayed processing. We also construct a theoretical framework for the performance analysis of communicating processes. It is formulated as two algebraic order relations for synchronously communicating processes and asynchronously ones. The relations can order two behaviorally equivalent processes with respect to their relative execution speeds. It provides a suitable method to verify and optimize real-time communicating systems.

1 Introduction

Over the last few years, many researchers have explored time-extended process calculi for the specification and the verification of real-time systems, for example see [7, 10, 16, 18, 19, 23]. Most of them are extensions of existing untimed process calculi, like CSP [11], CCS [14], and ACP [3], with the ability to quantitatively express temporal features, for example delayed processing, timeout handling, and so on. Some of the researchers also have developed time-sensitive equivalences by extending existing untimed equivalences such as bisimulation [14] and testing equivalence [9] with the notion of time. These equivalences equate two real-time processes when both their functionally behavioral properties and their temporal properties are *completely* matched with each other.

However, the equivalences may often be too strict in the analysis of most time-dependent systems, including non-strict real-time systems. This is because most systems have various temporal uncertainties, for example unpredicted transmission delays in communication, and unexpected interruptions in processors. Therefore, the temporal properties of implementations in the real world are never the same as those of their specification exactly. Also, we can often say that an implementation is able to satisfy its specification, only when the implementation can perform the behavioral properties given in its specification at *earlier* timings than those given in the specification.

*Email: *ichiro@is.ocha.ac.jp*

Therefore, it is convenient to construct a verification framework that can decide whether two processes can perform the same behaviors and whether one of them (e.g. an implementation of a system) can perform the behaviors faster than the other (e.g. the specification of the system). Moreover, in real communicating systems, interprocess communication is often realized by means of *asynchronous* style as well as *synchronous* one. In asynchronous communication settings, the sender of a message cannot know when the message is actually consumed as opposed to synchronous ones. Consequently, the verification of asynchronously communicating processes requires another time-sensitive framework. We address both synchrony and asynchrony in communication.

This paper proposes such a framework. It is formulated through two algebraic order relations on real-time processes with respect to their execution speeds. The relations can order two processes when they have the *same* behavioral properties and one of them can perform the properties *earlier* than the another. To clarify our exposition, we first develop two new time-extended calculi towards synchrony and asynchrony in communication. One of them aims to model synchronous interactions among real-time processes and the other is to model asynchronous interactions among ones. We next equip each calculus with its own speed-sensitive order relation, which corresponds to its communication fashion. We have many occasions to replace a slower subsystem in a system with a faster subsystem. In such a reconstruction, we need to guarantee that the slower subsystem can really be replaced by the faster one without altering the behavioral properties or lowering the execution speed of the whole system. We thus investigate the substitutability of speed-sensitively ordered processes.

The organization of this paper: The organization of this paper is as follows: In the next section we briefly present our basic ideas concerning the process calculi and then define their syntax and semantics. In Section 3 we define two speed-sensitive order relations on synchronously / asynchronously communicating processes and study their basic properties. In Section 4 we compare with related work and in the final section we give some concluding remarks.

2 Calculi

This section constructs two calculi by extending CCS [14] with the notion of quantitative time.

2.1 Basic Framework

Before defining the calculi, we summarize our basic ideas of them.

Time Values

We assume that time is interval between events instead of any absolute time, and is discrete, instead of continuous time. Time values are denoted as positive natural numbers including zero.

The Passage of Time
Time passes in all processes at the same speed. Also, all processes follow the same clock, or different ones but well-synchronized clocks. The advance of time is modeled as a special transition which is labeled by a quantity of time to indicate the amount of the advance, for example $P \xrightarrow{\langle t \rangle} P'$. It means that process P become P' after t time units.

Instantaneous Action
To preserve the pleasant properties of the original calculus, all communication and internal actions are assumed to be instantaneous. Instead, we introduce a special language construction to express temporal costs, including the execution time of processes.

Time-Dependent Behavior
We introduce a new prefix operator whose contents are dependent on the passage of time, called *delay operator*. This operator suspends a process for a specified period, written as $\langle t \rangle$, where t is the amount of the suspension. For instance, $\langle t \rangle.P$ means a process which is idle for t time units and then behaves as P.[1]

Idling
We assume that when an internal or communication action is enabled, processes must perform the action immediately without imposing unnecessary idling.[2] This assumption is the same as the notion of *maximal progress* shown in [10, 19, 23]. It lets us measure the minimum cost in synchronization among parallel processes, and enables the calculus to preserve the observation properties of CCS.

Asynchronous Communication
The word *asynchrony* here means that sender processes can send messages without synchronizing any processes. However, most of existing process calculi, including CCS [14] are formulated based on synchronous communication, they can have the expressive power of asynchronous communication. A way to express asynchronous communication is to restrict the formation of a term, $\overline{a}.P$, in the original calculus to the case where P is a terminate process, written as $\mathbf{0}$. That is, an asynchronous output, $\overline{a}$, followed by a process, P, is the same as the parallel composition $\overline{a}.\mathbf{0}|P$. This approach is essentially similar to several existing ones, for example see [1, 4, 12]. We also assume that the order of message arrival is indeterminate.[3]

[1] We can introduce some operators which restrict the possibility of particular actions due to the passage of time, but leave a calculus with such operators to other paper, e.g. [19]. That result is available in the calculi presented here.

[2] On the contrary, we can assume that when an internal or communication action is enabled, processes may not perform the action soon. We leave further details of this alternative model to another paper [20].

[3] In this paper, we assume that communication delay is negligible. In [20], we present a calculus for communicating systems where processes are located remotely and communication delay cannot be ignored.

3 Definition

This section defines the syntax and the semantics of the calculi.

3.1 Timed Calculus for Synchronous Communication

We present a calculus for synchronous interactions among real-time processes. The calculus is called *TSCS* (*Timed calculus for Synchronously Communicating Systems*) and is an extension of Milner's CCS [14].

Syntax

The syntax of the calculus is coincide with all the construction of CCS except for a new time-dependent operators, called *delay operator*. We first define the notations of time values.

Definition 3.1 Let $\mathcal{T}$ denote the set of the positive natural numbers including 0. We call $\mathcal{T}$ *time domain.* □

Next, we define symbols to present the events of processes.

Definition 3.2

- Let $\mathcal{A}$ be an infinite set of names denoting communication actions. Its elements are denoted as a,b,...
- Let $\overline{\mathcal{A}}$ be an infinite set of co-names. Its elements are denoted as $\overline{a}$,$\overline{b}$,...
- Let $\mathcal{L} \equiv \mathcal{A} \cup \overline{\mathcal{A}}$ be a set of communication action names. Elements of the set are written as $\ell, \ell', \ldots$.
- Let τ denote an internal action.
- Let Γ be the set of actions corresponding the amount of the passage of time. Elements of the set are denoted as $\langle t_1 \rangle, \langle t_2 \rangle, \ldots$, where $t_1, t_2, \ldots \in \mathcal{T}$.
- Let $Act \equiv \mathcal{L} \cup \{\tau\}$ be the set of operational actions. Its elements are denoted as α,β,.... □

$\overline{a}$ is the complementary action of a. τ-action represents all handshake communications and is considered to be unobservable from outside environments.

Definition 3.3 The set $\mathcal{P}_s$ of *TSCS* expressions ranged over by $P, P_1, P_2, \ldots$ is defined recursively by the following abstract syntax:

P	::= $\mathbf{0}$	(*Terminate Process*)	| $\alpha.P$	(*Action Prefix*)
	| $P_1 + P_2$	(*Summation*)	| $P_1 \mid P_2$	(*Composition*)
	| $P \setminus L$	(*Action Restriction*)	| $A \stackrel{\text{def}}{=} P$	(*Recursive Definition*)
	| $\langle t \rangle.P$	(*Delay Prefix*)		

where t is an element of $\mathcal{T}$ and L is a subset of $\mathcal{L}$. A is a process variable in set $\mathcal{K}$. We assume that in $A \stackrel{\text{def}}{=} P$, P is always closed, and each occurrence of A in P is only within some subexpressions $\alpha.A$ where α is not empty, or $\langle t \rangle.A$ where $t > 0$. □

The informal meaning of each process constructor is as follows:

- **0** is a terminate process which can perform no internal nor communication action.
- $\alpha.P$ is a process to perform action α and then behaves like P, where α is an input, output, or internal action.
- $P_1 + P_2$ represents a process which may behave as either P_1 or P_2.
- $P_1 \mid P_2$ represents that process P_1 and P_2 may run in parallel.
- $P \setminus L$ behaves like P but it is prohibited to communicate with external processes at actions in $L \cup \bar{L}$.
- $A \stackrel{\text{def}}{=} P$ means that A is defined as P, where P may include A.
- $\langle t \rangle.P$ represents a process which is suspended for t time units and then behaves like P.

We need the notion of action sort later.

Definition 3.4 The syntactic sort of each process, $\mathcal{L}(P)$, is defined inductively by:

$$\begin{array}{rclrcl} \mathcal{L}(\mathbf{0}) & = & \emptyset & \mathcal{L}(a.P) & = & \{a\} \cup \mathcal{L}(P) \\ \mathcal{L}(\overline{a}.P) & = & \{\overline{a}\} \cup \mathcal{L}(P) & \mathcal{L}(\tau.P) & = & \mathcal{L}(P) \\ \mathcal{L}(P_1 + P_2) & = & \mathcal{L}(P_1) \cup \mathcal{L}(P_2) & \mathcal{L}(P_1|P_2) & = & \mathcal{L}(P_1) \cup \mathcal{L}(P_2) \\ \mathcal{L}(P \setminus L) & = & \mathcal{L}(P) - (L \cup \overline{L}) & \mathcal{L}(\langle d \rangle.P) & = & \mathcal{L}(P) \end{array}$$

when $A \stackrel{\text{def}}{=} P$, we have $\mathcal{L}(P) \subseteq \mathcal{L}(A)$. We often call *sort* simply. □

Semantics

The operational semantics of the calculus can computationally encompass that of CCS and embody the notion of time. The semantics is defined as two tiers of labeled transition rules. One of them defines the semantics of functional behaviors of processes, called *behavioral transition*, written as $\stackrel{\alpha}{\longrightarrow}$ ($\longrightarrow \subseteq \mathcal{P}_s \times Act \times \mathcal{P}_s$) and the another defines the passage of time on processes, called *temporal transition*, written as $\stackrel{\langle t \rangle}{\longrightarrow}$ ($\longrightarrow \subseteq \mathcal{P}_s \times \Gamma \times \mathcal{P}_s$).

Definition 3.5 *TSCS* is a labeled transition system $\langle\ \mathcal{P}_s,\ Act \cup \Gamma,\ \{\ \stackrel{\mu}{\longrightarrow} \subseteq \mathcal{P}_s \times \mathcal{E}_s \mid \mu \in Act \cup \Gamma\ \}\ \rangle$. The transition relation $\longrightarrow$ is defined by two kinds of structural induction rules given in Figure 1 and 2. □

In giving the rules, we adopt the convention that the transition below the horizontal line may be inferred from the transitions above the line.

Example 3.6 We show some basic examples of processes in $\mathcal{P}_s$ as follows:

(1) $\langle 2 \rangle.\overline{a}.P_1$ is a process which performs output action $\overline{a}$ after 2 time units and then behaves like P_1.

$$\langle 2 \rangle.\overline{a}.P_1 \stackrel{\langle 2 \rangle}{\longrightarrow} \overline{a}.P_1 \stackrel{\overline{a}}{\longrightarrow} P_1$$

(2) $\langle 3 \rangle.(a.P_2 + b.P_3)$ is a process which can receive either input action a or b after 3 time units, and then behaves like P_2 or P_3.

(3) After three time units, $\langle 2\rangle.\overline{a}.P_1 \mid \langle 3\rangle.(a.P_2 + b.P_3)$ performs a communicate between $\langle 2\rangle.\overline{a}.P_1$ and $\langle 3\rangle.(a.P_2 + b.P_3)$ at action name a.

$$\langle 2\rangle.\overline{a}.P_1 \mid \langle 3\rangle.(a.P_2 + b.P_3) \xrightarrow{\langle 3\rangle} \overline{a}.P_1 \mid (a.P_2 + b.P_3) \xrightarrow{\tau} P_1 \mid P_2 \qquad \square$$

We present some basic properties of the semantics of the calculus.

Proposition 3.7

(1) *Maximal Progress*
If $P \xrightarrow{\tau} P'$ then, there is not some P'' such that $P \xrightarrow{\langle t\rangle} P''$ where $t > 0$.

(2) *Time Determinacy*
If $P \xrightarrow{\langle t\rangle} P'$ and $P \xrightarrow{\langle t\rangle} P''$ then P' and P' are syntactically identical. $\square$

The first property means that if a process has an executable communication or an internal action, it must perform the action immediately without imposing unnecessary idling. This property allows us to estimate synchronization time in communication exactly. The second means that time is deterministic in the sense that the passage of time does not interfere with any non-determinism.

The transition relation $\longrightarrow$ does not distinguish between observable and unobservable actions. We define two transition relations due to the non-observationability of τ.

Definition 3.8

(i) $P \stackrel{\alpha}{\Longrightarrow} P'$ is defined as $P(\xrightarrow{\tau})^* \xrightarrow{\alpha} (\xrightarrow{\tau})^* P'$

(ii) $P \stackrel{\widehat{\alpha}}{\Longrightarrow} P'$ is defined as $P(\xrightarrow{\tau})^* \xrightarrow{\alpha} (\xrightarrow{\tau})^* P'$ if $\alpha \neq \tau$ and otherwise $P(\xrightarrow{\tau})^* P'$

(iii) $P \stackrel{\langle t\rangle}{\Longrightarrow} P'$ is defined as $P(\xrightarrow{\tau})^* \xrightarrow{\langle t_1\rangle} (\xrightarrow{\tau})^* \cdots (\xrightarrow{\tau})^* \xrightarrow{\langle t_n\rangle} (\xrightarrow{\tau})^* P'$ $(t = t_1 + \cdots + t_n)$. $\square$
where $+$ is a mathematical addition over two numbers.

3.2 Timed Calculus for Asynchronous Communication

We now define a calculus for asynchronous interactions among real-time processes, called *TACS* (*Timed calculus for Asynchronously Communicating Systems*). It is a subset of the previous calculus where there is not any output prefix. We also disallow output guards in any alternative choice. This is because in asynchronous communication settings, it is difficult to realize output guards on choice points without synchronization at the implementation level.

Definition 3.9 The set $\mathcal{P}_a$ of *TACS* expressions, $P, P_1, P_2, \ldots$ is defined by the following expressions:

$$\begin{array}{rcl} P & ::= & \overline{a} \mid P_1 \mid P_2 \mid \langle t\rangle.P \mid P \setminus L \mid A \stackrel{\text{def}}{=} P \mid G \\ G & ::= & \mathbf{0} \mid a.P \mid \tau.P \mid G_1 + G_2 \end{array}$$

where a is a message name in $\mathcal{A}$, t is an element of $\mathcal{T}$. We often denote $\overline{a}$ as $\overline{a}.\mathbf{0}$. $\square$

$$\frac{-}{\alpha.P \xrightarrow{\alpha} P} \qquad \frac{P_1 \xrightarrow{\alpha} P_1'}{P_1 + P_2 \xrightarrow{\alpha} P_1'} \qquad \frac{P_2 \xrightarrow{\alpha} P_2'}{P_1 + P_2 \xrightarrow{\alpha} P_2'}$$

$$\frac{P_1 \xrightarrow{\alpha} P_1'}{P_1 \mid P_2 \xrightarrow{\alpha} P_1' \mid P_2} \qquad \frac{P_2 \xrightarrow{\alpha} P_2'}{P_1 \mid P_2 \xrightarrow{\alpha} P_1 \mid P_2'} \qquad \frac{P_1 \xrightarrow{\ell} P_1',\ P_2 \xrightarrow{\overline{\ell}} P_2'}{P_1 \mid P_2 \xrightarrow{\tau} P_1' \mid P_2'}$$

$$\frac{P \xrightarrow{\alpha} P',\ \alpha \notin L \cup \overline{L}}{P \setminus L \xrightarrow{\alpha} P' \setminus L} \qquad \frac{P \xrightarrow{\alpha} P'}{A \xrightarrow{\alpha} P'} (A \stackrel{\text{def}}{=} P) \qquad \frac{P \xrightarrow{\alpha} P'}{\langle 0 \rangle.P \xrightarrow{\alpha} P'}$$

Figure 1: Inference Rules for Behavioral Transition

$$\frac{-}{\mathbf{0} \xrightarrow{\langle t \rangle} \mathbf{0}} \qquad \frac{-}{\ell.P \xrightarrow{\langle t \rangle} \ell.P} \qquad \frac{P_1 \xrightarrow{\langle t \rangle} P_1',\ P_2 \xrightarrow{\langle t \rangle} P_2'}{P_1 + P_2 \xrightarrow{\langle t \rangle} P_1' + P_2'}$$

$$\frac{P_1 \xrightarrow{\langle t \rangle} P_1',\ P_2 \xrightarrow{\langle t \rangle} P_2'}{P_1 \mid P_2 \xrightarrow{\langle t \rangle} P_1' \mid P_2'} (\neg \exists P' : P_1 | P_2 \xrightarrow{\tau} P')$$

$$\frac{P \xrightarrow{\langle t \rangle} P'}{P \setminus L \xrightarrow{\langle t \rangle} P' \setminus L} \qquad \frac{P \xrightarrow{\langle t \rangle} P'}{A \xrightarrow{\langle t \rangle} P'} (A \stackrel{\text{def}}{=} P)$$

$$\frac{P \xrightarrow{\langle t \rangle} P'}{\langle 0 \rangle.P \xrightarrow{\langle t \rangle} P'} \qquad \frac{-}{\langle t + t' \rangle.E \xrightarrow{\langle t \rangle} \langle t' \rangle.E} (t + t' > 0)$$

Figure 2: Inference Rules for Temporal Transition

We give the intuitive meanings of some important expressions in the language below.

- $(\overline{a}|P)$ represents a process that sends message a without synchronizing any processes and behaves like P. Note that $\overline{a}$ corresponds to the asynchronous message itself. P corresponds to the subsequent computation.
- $a.P$ represents a process that receives message a and then behaves like P.

Semantics

All the expressions in the present calculus are included in those in the previous one. The semantics of the present calculus is defined through that of the previous one.

Definition 3.10 The operational semantics of *TACS* is defined from the transition relation rules given in Definition 3.5. □

Example 3.11 We show some basic examples of processes in $\mathcal{P}_a$ as follows:

(1) $\langle 2 \rangle.(\overline{a}|P_1)$ is a process which sends asynchronous message a after 2 time units and then behaves like P_1.

$$\langle 2 \rangle.(\overline{a}|P_1) \xrightarrow{\langle 2 \rangle} (\overline{a}|P_1) \xrightarrow{\overline{a}} P_1$$

(2) $\langle 3\rangle.(a.P_2 + b.P_3)$ is a process which can receive either message a or b after 3 time units and behaves like P_2 or P_3.

(3) $\langle 2\rangle.(\overline{a}|P_1) \mid \langle 3\rangle.(a.P_2 + b.P_3)$ performs a communication at action name a.

$$\langle 2\rangle.(\overline{a}|P_1) \mid \langle 3\rangle.(a.P_2 + b.P_3) \xrightarrow{\langle 3\rangle} \overline{a} \mid P_1 \mid (a.P_2 + b.P_3) \xrightarrow{\tau} P_1 \mid P_2 \qquad \square$$

In asynchronous communication settings, sender processes cannot observe how the messages which they send will be treated by the other processes. We introduce weak labeled transitions for asynchronous communication.

Definition 3.12 The operational semantics of *TACS* is redefined from the transition relation rules given in Definition 3.5 and the following rules:

$$\frac{-}{\overline{a} \mid P \xrightarrow{\uparrow a} P} \qquad \frac{-}{P \xrightarrow{\langle t\rangle \downarrow a} \langle t\rangle.\overline{a} \mid P} \qquad (t \in \mathcal{T})$$

where we often abbreviate $\langle 0\rangle \downarrow a$ to $\downarrow a$. $\square$

We give intuitive meaning of the above transitions.

- $P \xrightarrow{\uparrow a} P'$ means that a conceptual observer or the environment receives message a from P and P behaves like P'.
- $P \xrightarrow{\langle t\rangle \downarrow a} P'$ means that after t time units, an observer or the environment sends message a and P becomes P'.

Definition 3.13

(1) $P \stackrel{\langle t\rangle \downarrow a}{\Longrightarrow} P'$ is given as $P \, (\xrightarrow{\tau})^* \xrightarrow{\langle t\rangle \downarrow a} (\xrightarrow{\tau})^* \, P'$.

(2) $P \stackrel{\uparrow a}{\Longrightarrow} P'$ is given as $P \, (\xrightarrow{\tau})^* \xrightarrow{\uparrow a} (\xrightarrow{\tau})^* \, P'$.

4 Speed-Sensitive Prebisimulations

Based on time-extended process calculi, several researchers have explored time-sensitive equivalences which are based on trace equivalence, failure equivalence, testing equivalence, and bisimulation equivalence like ours, for example see [10, 16, 18, 19, 23]. These equivalences equate two processes if they cannot be distinguished from each other in their temporal properties as well as their behavioral one. This section develops algebraic order relations on processes with respect to their speeds based on the bisimulation concept.

4.1 An Order for Synchronously Communicating Processes

We define a speed-sensitive relation for synchronously communicating processes based on the bisimulation technique. r

Definition 4.1 A binary relation $\mathcal{R} \subseteq (\mathcal{P}_s \times \mathcal{P}_s) \times \mathcal{T}$ is a *t-synchronous prebisimulation* over synchronously communicating processes if $(P_1, P_2) \in \mathcal{R}_t$ $(t \geq 0)$ implies, for all $\alpha \in Act$;

(i) $\forall d\, \forall P_1'$: $P_1 \stackrel{\langle d \rangle}{\Longrightarrow} \stackrel{\alpha}{\Longrightarrow} P_1'$ *then*
$\exists d'\, \exists P_2'$: $P_2 \stackrel{\langle d \rangle}{\Longrightarrow} \stackrel{\langle d' \rangle}{\Longrightarrow} \stackrel{\widehat{\alpha}}{\Longrightarrow} P_2'$ *and* $(P_1', P_2') \in \mathcal{R}_{t+d'}$

(ii) $\forall d\, \forall P_2'$: $P_2 \stackrel{\langle d \rangle}{\Longrightarrow} \stackrel{\alpha}{\Longrightarrow} P_2'$ *then*
$\exists P_1'$: $P_1 \stackrel{\langle d \rangle}{\Longrightarrow} \stackrel{\langle t \rangle}{\Longrightarrow} \stackrel{\widehat{\alpha}}{\Longrightarrow} P_1'$ *and* $(P_1', P_2') \in \mathcal{R}_0$ □

In the above definition, $\mathcal{R}_t$ is a family of relations indexed by a non-negative time value t. Intuitively, t is the relative difference between the time of P_1 and that of P_2; that is, it means that P_1 precedes P_2 by t time units.[4] The following order relation starts with a prebisimulation indexed by t (i.e., $\mathcal{R}_t^L$) and can change t as the bisimulation proceeds only if $t \geq 0$.

Definition 4.2 We let $P_1 \leq_s^t P_2$ if there exists some t-synchronous prebisimulation such that $(P_1, P_2) \in \mathcal{R}_t$. We call $\leq_s^t$ *speed-sensitive order* on synchronously communicating processes. We shall often abbreviate $\leq_s^0$ as $\leq_s$. □

Hereafter, we usually consider $\leq_s$ only. We show several algebraic properties of the order relation below.

Proposition 4.3 Let $P, P_1, P_2, P_3 \in \mathcal{P}_s$. Then,

(1) $P \leq_s P$
(2) $P_1 \leq_s P_2$ *and* $P_2 \leq_s P_3$ *then* $P_1 \leq_s P_3$ □

From these results, we see that $\leq_s$ is a preorder relation. We also have $P_1 \leq_s^{t_1+t_2} P_3$ if $P_1 \leq_s^{t_1} P_2$ and $P_2 \leq_s^{t_2} P_3$.

Proposition 4.4 Let $P_1, P_2 \in \mathcal{P}_s$, $t_1, t_2 \in \mathcal{T}$ such that $t_1 \leq t_2$. Then,

$$\langle t_1 \rangle.P \leq_s \langle t_2 \rangle.P$$ □

The above proposition shows an important characteristic of $\leq_s$.

Example 4.5 We show some basic examples of $\leq_s$ as follows:

(1) $a.P \leq_s \langle 1 \rangle.a.P$
(2) $\langle 1 \rangle.\overline{a}.\langle 2 \rangle.\overline{b}.P \leq_s \langle 1 \rangle.\overline{a}.\langle 3 \rangle.\overline{b}.P$
(3) $a.P_1 | \langle 1 \rangle.b.P_2 \leq_s \langle 1 \rangle.a.P_1 | \langle 1 \rangle.b.P_2$
(4) $\langle 1 \rangle.(a.P_1 + b.P_2) | \langle 2 \rangle.\overline{a}.P_3 \leq_s \langle 2 \rangle.(a.P_1 + b.P_2) | \langle 2 \rangle.\overline{a}.P_3$ □

Proposition 4.6 Let $P_1, P_2, Q \in \mathcal{P}_s$ such that $P_1 \leq_s P_2$. Then

(1) $\alpha.P_1 \leq_s \alpha.P_2$

[4]This means that the performance of P_1 is at most t time units faster than that of P_2.

(2) $P_1 \setminus L \leq_s P_2 \setminus L$

(3) $\langle t \rangle.P_1 \leq_s \langle t \rangle.P_2$ □

It is convenient to develop a precongruence with respect to speeds in order to guarantee the substitutability between two ordered processes in any parallel context. However, there is an undesirable problem in defining such a pre-congruence with temporal inequality. Suppose three objects: $A_1 \stackrel{\text{def}}{=} \langle 2 \rangle.\overline{a}.\mathbf{0}$, $A_2 \stackrel{\text{def}}{=} \langle 4 \rangle.\overline{a}.\mathbf{0}$, and $B \stackrel{\text{def}}{=} a.P_1 + b.P_2 | \langle 3 \rangle.\overline{b}.\mathbf{0}$. We clearly have $A_1 \leq_s A_2$ but cannot expect that $A_1 | B \leq_s A_2 | B$, because $A_1 | B \stackrel{\langle 2 \rangle}{\rightarrow} \stackrel{\tau}{\rightarrow} P_1 | \langle 1 \rangle.\overline{b}.\mathbf{0}$ and $A_2 | B \stackrel{\langle 3 \rangle}{\rightarrow} \stackrel{\tau}{\rightarrow} \langle 1 \rangle.\overline{a}.\mathbf{0} | P_2$. This anomaly is traced to contexts that restrict the capability to execute a particular computation due to the passage of time, for example *timeout* handling in B. In order to define a rational pre-congruence with respect to speed, we need to impose certain syntactic restrictions on processes that may be composed in parallel.

Definition 4.7

(1) $(\alpha_1 | \cdots | \alpha_n).P$ is defined as $\sum_{1 \leq i \leq n} \alpha_i$, where $(\alpha).P \equiv \alpha.P$. $(\alpha_1 | \cdots | \alpha_{i-1} | \alpha_{i+1} | \cdots | \alpha_n |).P$ is called *confluent summation.*

(2) $P_1 || _L P_2$ is defined as $(P_1 | P_2) \setminus L$. $P_1 || _L P_2$ is *confluent composition* if $\mathcal{L}(P_1) \cap \mathcal{L}(P_1) = \emptyset$ and $\overline{\mathcal{L}(P_1)} \cap \mathcal{L}(P_1) \subseteq L \cup \overline{L}$.

□

Definition 4.8 $P \in \mathcal{P}_s$ is *time-stable* if P is built using only terminate process, action prefix, delay prefix, action restriction, confluent composition, confluent summation, and recursion given in Definition 3.3. □

Example 4.9 We show some basic examples of time-stable processes: $\overline{a}.\mathbf{0}$, $(\langle 3 \rangle.\overline{a}.P_1 | a.P_2) \setminus \{a\}$, $\langle 3 \rangle.(a.b.P + b.a.P)$, and $A \stackrel{\text{def}}{=} \overline{a}.b.A$, where P, P_1, P_2 are time-stable processes. □

This restriction allows processes to perform executable actions in any order. When we focus on processes cooperating only with particular processes, there may indeed be more suitable forms which cover a class of such processes. We here give a lemma before proving that $\leq_s$ is preserved in a parallel composition.

Lemma 4.10 Let $P_1, P_2, Q_1, Q_2 \in \mathcal{P}_s$ be time-stable processes. Then,

$$P_1 \leq_s^t P_2 \quad \textit{and} \quad Q_1 \leq_s^t Q_2 \quad \textit{then} \quad P_1 \,|\, Q_1 \leq_s^t P_2 \,|\, Q_2$$ □

We now come to the main result of this subsection.

Theorem 4.11 Let $P_1, P_2, Q \in \mathcal{P}_s$ be time-stable processes. Then,

$$P_1 \leq_s P_2 \qquad \textit{then} \qquad P_1 \,|\, Q \leq_s P_2 \,|\, Q$$ □

Intuitively, the above result tells that a parallel composition between the faster processes can really perform faster than one between the slower ones. That is, a system when embedding the faster processes can still perform faster than when embedding the slower ones.

4.2 An Order for Asynchronously Communicating Processes

In asynchronous communication settings, sender processes do not have to synchronize their receiver processes and thus can send messages as soon as they can. Therefore, in the settings real-time processes do not need to completely match their own temporal specification, and need only to deliver their required messages to receiver processes at earlier timings than those given in their specification. This means that in the verification of asynchronous communicating real-time processes, a speed-sensitive order relation can be often more suitable than in that of synchronous ones.

Definition 4.12 A binary relation $\mathcal{R} \subseteq (\mathcal{P}_a \times \mathcal{P}_a) \times \mathcal{T}$ is a *t-asynchronous prebisimulation* over asynchronously communicating processes $\mathcal{P}_a$ if $(P_1, P_2) \in \mathcal{R}_t$ implies, for all $a, b \in \mathcal{A}$, and $d \in \mathcal{T}$;

(i) $\forall t_1\ \forall P_1'$: $P_1 \stackrel{\langle d \rangle \natural a}{\Longrightarrow} \stackrel{\langle t_1 \rangle}{\Longrightarrow} \stackrel{\natural b}{\Longrightarrow} P_1'$ *then*
$\exists t_2\ \exists P_2'$: $P_2 \stackrel{\langle d \rangle \natural a}{\Longrightarrow} \stackrel{\langle t_2 \rangle}{\Longrightarrow} \stackrel{\natural b}{\Longrightarrow} P_2'$ *and* $(P_1', P_2') \in \mathcal{R}_{t-t_1+t_2}$

(ii) $\forall t_2\ \forall P_2'$: $P_2 \stackrel{\langle d \rangle \natural a}{\Longrightarrow} \stackrel{\langle t_2 \rangle}{\Longrightarrow} \stackrel{\natural b}{\Longrightarrow} P_2'$ *then*
$\exists t_1\ \exists P_1'$: $P_1 \stackrel{\langle d \rangle \natural a}{\Longrightarrow} \stackrel{\langle t_1 \rangle}{\Longrightarrow} \stackrel{\natural b}{\Longrightarrow} P_1'$ *and* $(P_1', P_2') \in \mathcal{R}_{t-t_1+t_2}$

where a, b may be empty names and $t, t_1, t_2, d \in \mathcal{T}$. □

As t-synchronous prebisimulation, $\mathcal{R}_t$ is a family of relations indexed by a non-negative time value t, where t corresponds to the relative difference between the time of the first process and the second one.

Definition 4.13 We let $P_1 \leq_a^t P_2$ if there exists a t-asynchronous prebisimulation such that $(P_1, P_2) \in \mathcal{R}_t$. We call $\leq_a^t$ *speed-sensitive order* on asynchronously communicating processes. We shall often abbreviate $\leq_a^0$ as $\leq_a$. □

We here state the informal meaning of $P_1 \leq_a^t P_2$. We first assume $(P_1, P_2) \in \mathcal{R}_t$. This assumption means P_1 precedes P_2 by t time units. An observer sends message a to P_1 after d time units (written as $P_1 \stackrel{\langle d \rangle \natural a}{\Longrightarrow}$ in (i)). It also sends the same message to P_2 after d time units (written as $P_2 \stackrel{\langle d \rangle \natural a}{\Longrightarrow}$ in (ii)). And then the observer receives return message b from P_1 after t_1 time units (written as $\stackrel{\langle t_1 \rangle}{\Longrightarrow} \stackrel{\natural b}{\Longrightarrow} P_1'$ in (i)), and from P_2 after t_2 time units (written as $\stackrel{\langle t_2 \rangle}{\Longrightarrow} \stackrel{\natural b}{\Longrightarrow} P_2'$ in (ii)). If the arrival time of the return message from P_1 is earlier than that from P_2,[5] and if P_1' and P_2' can be successfully observed in $(P_1', P_2') \in \mathcal{R}_{t-t_1+t_2}$ in the same way, P_1 and P_2 can perform the same behaviors but P_1 can perform the behaviors *faster* than P_2. We show several algebraic properties of the order relation below.

Proposition 4.14 Let $P, P_1, P_2, P_3 \in \mathcal{P}_a$. Then,

(1) $P \leq_a P$
(2) $P_1 \leq_a P_2$ *and* $P_2 \leq_a P_3$ *then* $P_1 \leq_a P_3$ □

[5] Note that P_2 already precedes P_1 by t time units. Thus, the relative difference between the arrival timing of the message from P_2 and that from P_1 is $t - t_1 + t_2$.

The above proposition says that $\leq_a$ is a preorder relation. We also have $P_1 \leq_a^{t_1+t_2} P_3$ if $P_1 \leq_a^{t_1} P_2$ and $P_2 \leq_a^{t_2} P_3$.

Proposition 4.15 Let $P_1, P_2 \in \mathcal{P}_a$, $t_1, t_2 \in \mathcal{T}$ such that $t_1 \leq t_2$. Then,

$$\langle t_1 \rangle.P \leq_a \langle t_2 \rangle.P$$ □

Example 4.16 We show some basic examples of $\leq_a$ as follows:

(1) $\overline{a} \leq_a \langle 1 \rangle.\overline{a}$
(2) $a.P \leq_a \langle 1 \rangle.a.P$
(3) $a.P_1 | \langle 1 \rangle.b.P_2 \leq_a \langle 1 \rangle.a.P_1 | \langle 1 \rangle.b.P_2$
(4) $\langle 1 \rangle.(\overline{a} | \langle 3 \rangle.(\overline{b} | P)) \leq_a \langle 2 \rangle.(\overline{a} | \langle 2 \rangle.(\overline{b} | P))$ *c.f.* $\langle 1 \rangle.\overline{a}.\langle 3 \rangle.\overline{b}.P \not\leq_s \langle 2 \rangle.\overline{a}.\langle 2 \rangle.\overline{b}.P$ □

In the above example, (4) shows a difference between synchrony and asynchrony in communication.

Proposition 4.17 Let $P_1, P_2, Q \in \mathcal{P}_a$ such that $P_1 \leq_a P_2$. Then,

(1) $a.P_1 \leq_a a.P_2$
(2) $P_1 \setminus L \leq_a P_2 \setminus L$
(3) $\langle t \rangle.P_1 \leq_a \langle t \rangle.P_2$ □

From the above result, we directly know that $\overline{a}|P_1 \leq_a \overline{a}|P_2$ when $P_1 \leq_a P_2$. Next, we show that $\leq_a$ is precongruent with respect to parallel composition. We establish a convention that at most one process in parallel compositions can receive any given message, for the sake of substitutability. As the same of the synchronous communication setting, we need a syntactic restriction on expressions on $\mathcal{P}_a$.

Definition 4.18 $P \in \mathcal{P}_a$ is *time-stable* if P is built using only terminate process, action prefix, delay prefix, action restriction, confluent summation, composition, and recursion and when any subsequence of P occurs in form $P_1|P_2$, $\mathcal{L}(P_1) \cap \mathcal{L}(P_1) \cap \mathcal{A} = \emptyset$. □

Note that $\mathcal{L}(P) \cap \mathcal{A}$ means all the action names included in P. We need a lemma before proving it.

Lemma 4.19 Let $P_1, P_2, Q_1, Q_2 \in \mathcal{P}_a$ be time-stable processes. Then,

$$P_1 \leq_a^t P_2 \quad \textit{and} \quad Q_1 \leq_a^t Q_2 \quad \textit{then} \quad P_1 \,|\, Q_1 \leq_a^t P_2 \,|\, Q_2$$ □

Theorem 4.20 Let $P_1, P_2, Q \in \mathcal{P}_a$ be time-stable processes. Then,

$$P_1 \leq_a P_2 \qquad \textit{then} \qquad P_1 \,|\, Q \leq_a P_2 \,|\, Q$$ □

Intuitively, the result shows that if two processes are behaviorally equivalent and one of them can perform faster than the other, the faster process can be behaviorally substituted for the slower one in a parallel composition. That is, a parallel composition between the faster processes can really perform faster than one between the slower ones. That is, a system embedded with the faster processes can really perform faster than one embedded with the slower ones.

Remarks The definition of each relation can reveal essential differences between synchrony and asynchrony in interactions among processes in time-sensitive contexts. In synchronous communication settings, processes must be blocked until their partner processes are ready to communicate. $\leq_s$ can order two synchronously communicating processes when a conceptual observer cannot distinguish between them in their communications, and when the timings of the communications with one of them are earlier than those with the another. On the other hand, in asynchronous communication settings, the observer cannot exactly know when the messages which it sends are received by processes. It can know only the arrival timings of return messages. In $\leq_a$, an observer sends arbitrary messages to processes and waits for return messages from them. It orders the two processes when the return messages cannot be distinguished from each other, and when the arrival timings of the messages from one of them are earlier than those from the another.

5 Related Work

We here survey some related work. There have indeed been many process calculi for time-dependent systems, for example see [7, 10, 16, 18, 19, 22, 23]. Most of the calculi have been equipped with time-sensitive equivalences as verification methods. However, there have been a few researchers that studied inequalities for time-dependent processes.

Moller and Tofts in [16] proposed a preorder relation over processes with respect to their relative speeds, based on the bisimulation technique. Unlike ours, their calculus assumes to permit an executable communication to be suspended for arbitrary periods of time. As a result, the relation shows only that a process may *possibly* execute faster than the other. Recently, Vogler in [22] and Jenner and Vogler in [13] presented speed-sensitive preorder relations based on the testing equivalence technique [9]. The relation can relate asynchronously communicating processes according to their relative speeds, but its semantics is formulated based on causality between events on the assumption that actions are not instantaneous, unlike ours. Also, some researchers have explored the performance analysis by means of process algebras, for example see [5]. However, most of them are based on non-instantaneous actions. The other assumes that every process proceeds in lockstep and at every instant performs a single action.

Arun-Kumar and Hennessy in [2] propose an approach to relate processes with respect to their relative efficiencies based on the bisimulation technique [14]. Cleaveland and Zwarico in [6] Natarajan and Cleaveland [17] propose a similar approach based on the testing equivalence technique [9]. These approaches relate two processes whether their behavioral properties are equivalent and one of them can perform less internal actions τ than the other. That is to say, a process which needs to take more τ actions to execute the same properties than the other, is a slower process. However, it is very difficult to reflect the execution cost of real systems upon the number of τ-actions exactly in the description of the systems.

On the other hand, there have been several process calculi with the ability to express asynchronous communication (e.g. [1, 4, 8, 12]). Among them, in [4] Baeten and Bergstra proposed a time-extended process calculus with the ability to express asynchronous communication. It represents asynchronous message transmission as the creation of a process corresponding to the message like the methods developed in [12]. However, it provides

just a language to describe asynchronously communicating real-time processes and it does not provide any method to analyze the performance of processes.

6 Concluding Remarks

In this paper, we proposed two time-extended calculi to specify and analyze synchronous interactions among real-time processes and asynchronous ones. They are extensions of CCS with the ability to express quantitatively temporal properties of processes. They allow us to describe the behavioral and temporal properties of interactions among real-time processes.

Based on the calculi, we define algebraic order relations that can distinguish between behaviorally equivalent processes performing at different speeds. The relations can prove that a faster process can behaviorally replace a slower process without altering any behavioral properties nor the performance of the whole system. They can provide a theoretical framework to verify and optimize communicating systems.

This paper gives only a starting point for understanding communications among real-time processes. There are many issues that we leave in this paper, One of the most important issues is to study further refinements of the calculi and the relations. We are also interested in formulating inequalities based on the order relations, and investigating how the relation of each calculus is related with each other.

References

[1] Amadio, R.M., Castellani, I, and Sangiorig, D., *On Bisimulation for the Asynchronous π-calculus*, Proceedings of CONCUR'96, p148-162, Springer-Verlag, August, 1996.

[2] Arun-Kumar, S. and Hennessy, M., *An Efficiency Preorder for Processes*, Acta Informatica, Vol.29, p737-760, 1992.

[3] Baeten, J. C. M., and Bergstra, J. A., *Process Algebra*, Cambridge University Press, 1990.

[4] Baeten, J. C. M., and Bergstra, J. A., *Asynchronous Communication in Real Space Process Algebra*, Proceedings of Formal Techniques in Real-Time and Fault-Tolerant System, LNCS 591, p473-491, Springer-Verlag, May, 1991.

[5] Chen. X. J., Corradini, F., and Gorrieri, R., *A Study on the Specification and Verification of Performace Properties*, Proceedings of Algebraic Methodology and software Technology, LNCS Vol. 1011, Springer-Verlag, 1996.

[6] Cleaveland, R and Zwarico, A.E., *Theory of Testing for Real-Time*, Proceedings, 6th IEEE Symposium on Logic in Computer Science, p110-119, 1990.

[7] Davies, J. W., *Specification and Proof in Real-Time CSP*, Cambridge University Press, 1994.

[8] de Bore, F.S., Klop, J.W., and Palamidessi, *Asynchronous Communication in Process Algebra*, Proceedings of LICS'92, p137-147, June, 1992.

[9] de Nicola, E. and Hennessy, M., *Testing Equivalence for Processes*, Theoretical Computer Science, No.34, 1984.

[10] Hennessy, M., *On Timed Process Algebra: a Tutorial*, Technical Report 2/93, University of Sussex, 1993

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

[12] Honda, K., and Tokoro, M., *An Object Calculus for Asynchronous Communication*, Proceedings of ECOOP'91, LNCS 512, p133-147, June, 1991.

[13] Jenner, L., and Volger, W., *Faster Asynchronous Systems in Dense Time*, Proceedings of ICALP'96, p75-86, LNCS Vol. 1099, Springer-Verlag, 1996.

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

[15] Milner, R., Parrow. J., Walker, D., *A Calculus of Mobile Processes*, Information and Computation, Vol.100, p1-77, 1992.

[16] Moller, F., and Tofts, C., *Relating Processes with Respect to Speed*, Proceedings of CONCUR'91, LNCS 527, Springer-Verlag, August, 1991.

[17] Natarajan, V., and Cleaveland, R., *An Algebraic Theory of Process Efficiency*, Proceedings of LICS'96, p63-72, June, 1996.

[18] Nicollin. X., and Sifakis, J., *An Overview and Synthesis on Timed Process Algebras*, Proceedings of Computer Aided Verification, LNCS 575, p376-398, Springer-Verlag, June, 1991.

[19] Satoh, I., and Tokoro, M., *A Formalism for Real-Time Concurrent Object-Oriented Computing*, Proceedings of ACM Conference on Object Oriented Programming Systems, Langauges, and Applications (OOPSLA'92), p315-326, 1992.

[20] Satoh, I., and Tokoro, M., *A Formalism for Remotely Interacting Processes*, Proceedings of Workshop on Theory and Practice of Parallel Programming (TPPP'94), LNCS 907, p216-228, Springer-Verlag, 1995.

[21] Satoh, I., and Tokoro, M., *Time and Asynchrony in Interactions among Distributed Real-Time Objects*, Proceedings of European Conference on Object-Oriented Programming (ECOOP'95), LNCS 952, p331–350, Springer-Verlag, 1995.

[22] Volger, W., *Faster Asynchronous Systems*, Proceedings of CONCUR'95, p299-312, LNCS 962, Springer-Verlag, 1995.

[23] Wang, Y., *CCS + Time = an Interleaving Model for Real Time Systems*, In proceedings of Automata, Languages and Programming'91, LNCS 510, Springer-Verlag, 1991.

Appendix

It is often useful to give time-sensitive equivalences for the present calculi in studying their basic properties, although such equivalences are often too strict in the verification of real systems. We thus develop time-sensitive bisimulation relations over process expressions by extending CCS's bisimulation with the notion of time. We denote both $\mathcal{P}_s$ and $\mathcal{P}_a$ as $\mathcal{P}$ later.

Definition 6.1 A binary relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a *strong bisimulation* if $(P_1, P_2) \in \mathcal{R}$ implies, for all $\alpha \in Act$ and $t \in \mathcal{T}$,

(i) $\forall P_1'$: $P_1 \xrightarrow{\langle t \rangle} \xrightarrow{\alpha} P_1'$ *then* $\exists P_2'$: $P_2 \xrightarrow{\langle t \rangle} \xrightarrow{\alpha} P_2'$ *and* $(P_1', P_2') \in \mathcal{R}$.
(ii) $\mathcal{R}$ is symmetric.

P_1 and P_2 are *strongly equivalent*, written $P_1 \sim_T P_2$, if there exists a strong bisimulation $\mathcal{R}$ such that $(P_1, P_2) \in \mathcal{R}$. □

Intuitively, if P_1 and P_2 are strongly equivalent, they cannot be distinguished from one another in their behaviors and timings. We can know that $\sim_{\mathcal{T}}$ is symmetric, reflexive, transitive, and congruent. We show a set of laws which are sound with respect to the strong equivalence below.

Proposition 6.2

$$
\begin{array}{rcl}
P_1 + P_2 & \sim_{\mathcal{T}} & P_2 + P_1 \\
P + P & \sim_{\mathcal{T}} & P \\
P_1 \,|\, P_2 & \sim_{\mathcal{T}} & P_2 \,|\, P_1 \\
P \,|\, \mathbf{0} & \sim_{\mathcal{T}} & P \\
P \setminus L & \sim_{\mathcal{T}} & P \\
(P_1 \,|\, P_2) \setminus L & \sim_{\mathcal{T}} & P_1 \setminus L \,|\, P_2 \setminus L \\
(P_1 + P_2) \setminus L & \sim_{\mathcal{T}} & P_1 \setminus L + P_2 \setminus L \\
\langle t \rangle.(P_1 + P_2) & \sim_{\mathcal{T}} & \langle t \rangle.P_1 + \langle t \rangle.P_2 \\
\langle t_1 + t_2 \rangle.P & \sim_{\mathcal{T}} & \langle t_1 \rangle.\langle t_2 \rangle.P
\end{array}
\qquad
\begin{array}{rcl}
P_1 + (P_2 + P_3) & \sim_{\mathcal{T}} & (P_1 + P_2) + P_3 \\
P + \mathbf{0} & \sim_{\mathcal{T}} & P \\
P_1 \,|\, (P_2 \,|\, P_3) & \sim_{\mathcal{T}} & (P_1 \,|\, P_2) \,|\, P_3 \\
\multicolumn{3}{l}{\textit{if } \mathcal{L}(P) \cap (L \cup \overline{L}) = \emptyset} \\
\multicolumn{3}{l}{\textit{if } \mathcal{L}(P_1) \cap \overline{\mathcal{L}(P_2)} \cap (L \cup \overline{L}) = \emptyset} \\
\\
\langle t \rangle.(P_1 | P_2) & \sim_{\mathcal{T}} & \langle t \rangle.P_1 | \langle t \rangle.P_2 \\
\langle 0 \rangle.P & \sim_{\mathcal{T}} & P
\end{array}
$$

□

We define an equivalence with the concept of observation.

Definition 6.3 A binary relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a *weak bisimulation* if $(P_1, P_2) \in \mathcal{R}$ implies, for all $\alpha \in Act$ and $t \in \mathcal{T}$,

(i) $\forall P_1'$: $P_1 \stackrel{\langle t \rangle}{\Longrightarrow} \stackrel{\alpha}{\Longrightarrow} P_1'$ *then* $\exists P_2'$: $P_2 \stackrel{\langle t \rangle}{\Longrightarrow} \stackrel{\widehat{\alpha}}{\Longrightarrow} P_2'$ *and* $(P_1', P_2') \in \mathcal{R}$.
(ii) $\mathcal{R}$ is symmetric.

P_1 and P_2 are *observation-equivalent*, written $P_1 \approx_{\mathcal{T}} P_2$, if there exists a weak bisimulation $\mathcal{R}$ such that $(P_1, P_2) \in \mathcal{R}$. □

Intuitively, if P_1 and P_2 are observation-equivalent, each action of P_1 must be matched by a sequence of actions of P_2 with the same visible contents and timing, and conversely. $\approx_{\mathcal{T}}$ is symmetric, reflexive, transitive. We know that it is congruent except summation, and $P_1 \approx_{\mathcal{T}} P_2$ if $P_1 \sim_{\mathcal{T}} P_2$.

Proposition 6.4

$$
\begin{array}{rcl}
\tau.P & \approx_{\mathcal{T}} & P \\
\tau.P + P & \approx_{\mathcal{T}} & \tau.P \\
\langle t \rangle.\tau.P & \approx_{\mathcal{T}} & \langle t \rangle.P
\end{array}
\qquad
\begin{array}{rcl}
\alpha.\tau.P & \approx_{\mathcal{T}} & \alpha.P \\
\alpha.(\tau.P + Q) & \approx_{\mathcal{T}} & \alpha.(\tau.P + Q) + \alpha.Q
\end{array}
$$

□

On the basis of the two equivalence relations, we can have sound and complete equational laws over finite processes on $\mathcal{P}_s$ and $\mathcal{P}_a$, but we leave them to other papers.