# Time and Asynchrony in Interactions among Distributed Real-Time Objects[⋆]

Ichiro Satoh[⋆⋆] and Mario Tokoro[⋆⋆⋆]

Department of Computer Science, Keio University
3-14-1, Hiyoshi, Kohoku-ku, Yokohama, 223, Japan

**Abstract.** This paper presents a framework of specification and verification for distributed real-time object-oriented systems. An earlier paper [17] introduced a process calculus to describe distributed objects using local clocks. However, it is appropriately based on synchronous communication and thus cannot sufficiently model asynchronous communication in distributed systems. In this paper we propose a new process calculus with the ability to express asynchronous message passing, communication delay, and delayed processing. It can describe temporal and behavioral properties of distributed real-time objects. Based on the new calculus, we develop a verification method by means of algebraic order relations. The relations are speed-sensitive and can decide whether two distributed real-time objects are behaviorally equivalent and whether one of them can perform its behaviors faster than the other. They offer a suitable method to prove the correctness and the reusability of real-time objects in asynchronous communication settings. Some examples are shown to demonstrate utilities of the calculus and the relations.

## 1 Introduction

The notion of concurrent object-oriented computation [21] provides a powerful method to design and develop distributed systems. This is because objects are logically self-contained active entities interacting with one another through message passing; whereas in distributed systems, processors cooperate by sending messages over communication networks. However, these networks have communication delays, a function of geographic distance, communication bandwidth, and communication protocol overhead. The delay is often non-deterministic and seriously affects the arrival timings of messages. Also, for efficiency reasons, communication among remote processors is asynchronous instead of synchronous, although asynchronous communication has more unpleasant non-deterministic aspects. This is because in synchronous communication the sender must be blocked for at least the round-trip time of the message, including communication delay.

[⋆] Part of this work has been done while the first author was visiting Rank Xerox Research Centre, Grenoble Laboratory, France.

[⋆⋆] Email: *satoh@mt.cs.keio.ac.jp*, Partially supported by JSPS Fellowships for Japanese Junior Scientists.

[⋆⋆⋆] Email: *mario@mt.cs.keio.ac.jp*, Also with Sony Computer Science Laboratory Inc. 3-14-13 Higashi-Gotanda, Shinagawa-ku, Tokyo, 141, Japan.

Therefore, delay and asynchrony in communication create serious difficulties in the design and development of distributed systems, even if these systems are based on the object-orientation concept. To construct correct programs for distributed real-time systems, we need to analyze the influences of delay and asynchrony on the behavioral and temporal properties of the systems. In this paper we address this problem and propose a framework of specification and verification for distributed real-time object-oriented systems with these features.

The framework consists of two parts: a description language and a verification method for distributed real-time object-oriented systems. The language is formulated as a process calculus [2, 11, 12], because the powerful expressive capability of process calculi allows many features of concurrent objects to be naturally modeled [10, 15]. For example, objects can be viewed as processes; interactions among objects can be seen as communications; and encapsulation can be modeled by the restriction of visible communications. However, ordinary process calculi are based on synchronous communication and lack the notion of time. Therefore, in this paper we develop a new timed extended calculus, called TACS (*Timed calculus for Asynchronously Communicating Systems*), with the ability to express communication delay, asynchronous message passing, and delayed processing. It allows us to describe temporal and behavioral properties of remotely located real-time objects and their asynchronous interactions with communication delay. In earlier papers [16, 17] the authors introduced timed extended process calculi, called RtCCS and DtCCS, for real-time concurrent and distributed object-oriented computing. Unfortunately, these calculi are based on synchronous communication and thus cannot provide any general framework for asynchronous interactions among distributed objects.

On the other hand, our verification method is formulated on the basis of algebraic relations over objects described in the language. In asynchronous communication settings, a sender object needs only to deliver messages given in its behavioral specification to its receiver objects at *earlier* timings than those specified in its temporal specification. It is very convenient to provide a verification method which decides if two objects can send and receive the same messages and if one of them (e.g., an implementation of an object) can perform faster than the another (e.g., the specification of the object). Also, we have many occasions to replace an older object (a slower object) in a system with a new, faster object. In such a reconstruction, we need to guarantee that the older object can really be replaced by the first object without altering the behavioral properties or lowering the performance characteristics of the whole system. Therefore, we define algebraic order relations which distinguish between behaviorally equivalent objects performing at different speeds. The relations allow us not only to compare two distributed real-time objects with respect to their performances but also to prove that a faster object can behaviorally replace a slower object and that, when embedding the faster object, the new system can really perform faster than the old one.

**Organization of this paper:** In the next section we briefly present our basic ideas concerning the process calculus and then define its syntax and semantics. In Section 3 we define two timed pre-bisimulations that can relate two processes according to their speeds. In Section 4 we briefly survey some related work, and the final section contains some concluding remarks.

## 2 Language

The language called *TACS*, on which our verification framework is based, is formulated as a process calculus [2, 11, 12]. It has the ability to express most features of distributed real-time object-oriented computing: communication delay, asynchronous message passing, and delayed processing. Its syntax and semantics are essentially similar to those of CCS [11], except for the extensions.[4]

### Basic Framework

Before formally defining *TACS*, we summarize its basic idea.

**Non-blocking Message Send:**
In distributed systems, communication between remotely located objects is often realized by means of one-way asynchronous message passing. However, most of process calculi are essentially based on synchronous communication, and thus we need to introduce the ability to express *non-blocking message send*.[5] We adapt an approach to express an asynchronous message by creating a process that does not do anything except for being received by an input port for the message. This is similar to the methods developed in several existing work [3, 10].

**The Passage of Time and Delaying Processing:**
We assume that time is discrete and that the passage of time in every processor elapses at the same rate.[6] The passage of one time unit is represented as a transition that proceeds synchronously in all objects, written as transition $\leadsto$. Also, most distributed real-time systems have behaviors that depend on the passage of time. We introduce an operation that explicitly suspends its execution for a specified period. It is written as $\langle t \rangle$. For example, $\langle t \rangle.S$ is idle for $t$ time units and then behaves as $S$.

[4] In this paper, for focusing temporal aspects in distributed real-time systems, we avoid to introduce the expressive capability for a distributed system where topological connections between objects can change dynamically; but our calculus can easily model such a system by incorporating with the port-passing mechanism developed in [12].

[5] In our framework, we assume that the order of message arrival is indeterminate.

[6] By using the method developed in [17], we can easily introduce the ability to express multiple and inaccurate clocks.

**Location-dependent Communication Delay:**
Communication delay for a message is defined as the difference between the time at which an object is ready to send the message and the time at which the message reaches its destination object. The length of communication delay is often dependent on the distance between the locations of the sender and receiver objects. Therefore, we introduce the concept of object (or process) location as developed in [5] and then represent communication delay as a pair consisting of the sender location and the receiver location.[7] Each object has its location name and can receive only the messages that can arrive at the location. For example, we write object $S$ at location $\ell$ which sends message $a$ to an object at location $\ell'$ as $\ell'\!\uparrow a.S : \ell$.

## Syntax

We first define several notations that we will use hereafter.

**Definition 2.1**

- Let $\mathcal{M}$ be an infinite multi-set of message names, ranged over by $a$,$b$,....
- Let $\mathcal{L}$oc be an infinite set of location names, ranged over by $\ell$,$\ell_1$,$\ell_2$,....
- Let $\mathcal{T}$ be the infinite set of the positive natural numbers including 0.
- Let $\mathcal{D} \subseteq \mathcal{L}\mathrm{oc} \times \mathcal{L}\mathrm{oc} \to 2^{\mathcal{T}}$ be a set of multivalued functions, called communication delay functions, ranged over by *delay*,.... □

In our framework, we describe distributed real-time objects by means of the expressions defined below. In order to clarify our exposition, we divide the expressions into two groups: expressions for describing local objects and expressions for describing located objects.

**Definition 2.2** The set $\mathcal{S}$ of local object expressions, ranged over by $S, S_1, S_2, \ldots$ is the smallest set containing the following expressions:

$$\begin{array}{lll} S ::= & \ell\!\uparrow a.S & (\textit{Asynchronous Message Send}) \\ \mid & \sum_{i\in I} \downarrow a_i.S_i & (\textit{Selective Message Receive}) \\ \mid & \langle t\rangle \,.\, S & (\textit{Delaying Processing}) \\ \mid & S \,|\, S & (\textit{Parallel Execution}) \\ \mid & S \setminus N & (\textit{Message Encapsulation}) \\ \mid & A \overset{\mathrm{def}}{=} S & (\textit{Recursive Definition}) \end{array}$$

The set $\mathcal{P}$ of located object expressions, ranged over by $P, P_1, P_2, \ldots$ is the smallest set containing the following expressions:

$$\begin{array}{lll} P ::= & S\!:\!\ell & (\textit{Located Object}) \\ \mid & P \,|\, P & (\textit{Parallel Execution}) \\ \mid & P \setminus N & (\textit{Message Encapsulation}) \end{array}$$

[7] The concept in this paper is used only for expressing communication dependent on locations; we do not intend to investigate the concept itself.

where $N$ is a subset of $\mathcal{M}$, $I$ is a finite index set $\{1, 2, \ldots, n\}$, $t$ is an element of $\mathcal{T}$, and $A$ is an element of a constant object expression. We assume that in $A \overset{\text{def}}{=} S$ if $S$ contains $A$, $A$ occurs only within subexpressions in $\mathcal{S}$ of the form $\sum_{i \in I} \downarrow a_i.S'_i$ or $\langle t \rangle \,.\, S'$ $(t > 0)$ in $S$. We also assume that $a_i$ is not an empty name in $\sum_{i \in I} \downarrow a_i.S'_i$. We write $\mathbf{0}$ for an empty summation (i.e., $\sum_{i \in \emptyset} \downarrow a_i.S_i$) and abbreviate $\ell {\uparrow} a.\mathbf{0}$ to $\ell {\uparrow} a$, and $\epsilon {\uparrow} a.\mathbf{0}$ to ${\uparrow} a.\mathbf{0}$, where $\epsilon$ is an empty location name. We often use the more readable notation $\downarrow a_1.S_1 + \cdots + \downarrow a_n.S_n$ instead of $\sum_{i \in \{1,\ldots,n\}} \downarrow a_i.S_i$ and write $A{:}\ell \overset{\text{def}}{=} S{:}\ell$ if $A \overset{\text{def}}{=} S$. □

The intuitive meaning of constructors in $\mathcal{S}$ ($\mathcal{P}$) is as follows:

- $\ell' \uparrow a.S$ : sends sends message $a$ to a object at location $\ell'$, and continues to execute $S$ without blocking.
- $\sum_{i \in I} \downarrow a_i.S_i$ behaves as $S_i$ if it receives message $a_i$. $\mathbf{0}$ represents a terminated or deadlocked object.
- $\langle t \rangle.S$ is idle for $t$ time units and then behaves like $S$.
- $S_1 \mid S_2$ allows $S_1$ and $S_2$ to execute in parallel at location $\ell$.
- $S \setminus N$ encapsulates all messages in set $N$.
- $A \overset{\text{def}}{=} S$ says that $A$ is defined as $S$. $S$ is allowed to include $A$.
- $S{:}\ell$ is an object located at $\ell$.
- $P_1 \mid P_2$ allows $S_1$ and $S_2$ to execute in parallel at location $\ell$.

We define a function to extract the name of an object location from $P$ expressions.

**Definition 2.3** The function $|\cdot|_{Loc} : \mathcal{P} \rightarrow 2^{\mathcal{L}\text{oc}}$, which presents the location names of objects, is defined as follows:

$$|S{:}\ell|_{Loc} = \{\ell\}, \quad |P_1|P_2|_{Loc} = |P_2|_{Loc} \cup |P_2|_{Loc}, \quad |P \setminus N|_{Loc} = |P|_{Loc} \quad \square$$

**Semantics**

The operational semantics of *TACS* is given by two kinds of state transition relations: $\rightarrow \subseteq \mathcal{P} \times \mathcal{P}$ (called *behavioral transition*) and $\rightsquigarrow \subseteq \mathcal{P} \times \mathcal{P}$ (called *temporal transition*). Throughout this paper we will use a structural equivalence ($\equiv$) over expressions. This is the method followed by Milner in [12] to deal with the $\pi$-calculus. The use of a structural equivalence provides a basic equivalence between objects and allows us to concentrate on the investigation of inequalities between objects in the following section.

**Definition 2.4** $\equiv$ is the least syntactic equivalence defined by the following equations:

$$P_1 \mid P_2 \equiv P_2 \mid P_1 \qquad P_1 \mid (P_2 \mid P_3) \equiv (P_1 \mid P_2) \mid P_3 \qquad P \mid \mathbf{0}{:}\ell \equiv P$$
$$(S_1 \mid S_1) : \ell \equiv S_1 : \ell \mid S_1 : \ell \qquad \mathbf{0}{:}\ell \setminus N \equiv \mathbf{0}{:}\ell \qquad S \setminus N{:}\ell \equiv S : \ell \setminus N$$

$P \backslash N_1 \backslash N_2 \equiv P \backslash N_2 \backslash N_1 \qquad (P \mid \ell' \uparrow a.\mathbf{0} : \ell) \backslash N \equiv P \backslash N \mid \ell' \uparrow a.\mathbf{0} : \ell$ where $a \notin N$
$\langle 0 \rangle . S : \ell \equiv S : \ell$

We now present the operational semantics of our language.

**Definition 2.5** *Behavioral transition* $\rightarrow \subseteq \mathcal{P} \times \mathcal{P}$ (written $P \rightarrow P'$) is the least binary relation given by the following axioms and rules.

$$\ell' \uparrow a \,.\, S : \ell \;\rightarrow\; S : \ell \mid \langle t \rangle \,.\, \epsilon \uparrow a.\mathbf{0} : \ell' \quad (\, t \in \mathit{delay}(\ell, \ell') \text{ and } \ell' \text{ is not } \epsilon)$$

$$\textstyle\sum_{i \in I} \downarrow a_i.S_i : \ell \mid \epsilon \uparrow a_i.\mathbf{0} : \ell \;\rightarrow\; S_i : \ell \mid \mathbf{0} : \ell$$

$$\frac{P_1 \;\rightarrow\; P_1'}{P_1 \mid P_2 \;\rightarrow\; P_1' \mid P_2} \qquad \frac{P \;\rightarrow\; P'}{P \backslash N \;\rightarrow\; P' \backslash N}$$

$$\frac{S : \ell \;\rightarrow\; S' : \ell}{A : \ell \;\rightarrow\; S' : \ell}\ (A \overset{\text{def}}{=} S) \qquad \frac{P_1 \equiv P_1', \quad P_1 \;\rightarrow\; P_2, \quad P_2 \equiv P_2'}{P_1' \;\rightarrow\; P_2'}$$

where *delay* $\in \mathcal{D}$. We often write $(\equiv)^* \rightarrow (\equiv)^*$ as $\rightarrow$ when not ambiguity. □

**Definition 2.6** *Temporal transition* $\rightsquigarrow \subseteq \mathcal{P} \times \mathcal{P}$ (written $P \rightsquigarrow P$) is the least binary relation defined by the following axioms and rules.

$$\langle t \rangle \,.\, S : \ell \;\rightsquigarrow\; \langle t - 1 \rangle \,.\, S : \ell \quad (t > 0) \qquad\qquad \epsilon \uparrow a \,.\, \mathbf{0} : \ell \;\rightsquigarrow\; \epsilon \uparrow a \,.\, \mathbf{0} : \ell$$

$$\textstyle\sum_{i \in I} \downarrow a_i \,.\, S_i : \ell \;\rightsquigarrow\; \sum_{i \in I} \downarrow a_i \,.\, S_i : \ell$$

$$\frac{P_1 \;\rightsquigarrow\; P_1', \quad P_2 \;\rightsquigarrow\; P_2', \quad P_1 \mid P_2 \not\rightarrow}{P_1 \mid P_2 \;\rightsquigarrow\; P_1' \mid P_2'} \qquad \frac{P \;\rightsquigarrow\; P'}{P \backslash N \;\rightsquigarrow\; P' \backslash N}$$

$$\frac{S : \ell \;\rightsquigarrow\; S' : \ell}{A : \ell \;\rightsquigarrow\; S' : \ell}\ (A \overset{\text{def}}{=} S) \qquad \frac{P_1 \equiv P_1', \quad P_1 \;\rightsquigarrow\; P_2, \quad P_2 \equiv P_2'}{P_1' \;\rightsquigarrow\; P_2'}$$

where we often write $(\equiv)^* \rightsquigarrow (\equiv)^*$ as $\rightsquigarrow$ when not ambiguity. □

Remarks on the above transition rules:

- The first axiom of Definition 2.5 shows that a non-blocking message send, $\ell' \uparrow a.S : \ell$, is translated into two parallel processes: $\langle t \rangle . \epsilon \uparrow a.\mathbf{0} : \ell'$ and $S : \ell$. The former corresponds to the asynchronous message. Before it can be received by an object at location $\ell'$, it is suspended for the amount of communication delay from $\ell$ to $\ell'$, written as $\mathit{delay}(\ell, \ell')$. If communication delay is indeterminate, the suspension time is a time value among all the possible communication delays. The latter corresponds to the subsequent computation. Note that it can continue to execute without any blocking.
- The second axiom of Definition 2.5 defines the semantics of message reception. A message can be consumed only by a receiver having a corresponding input port at the same location as the message's destination location.

- Our semantics forces message sending and receiving to be performed as soon as they are executable, without unnecessary delay. This pre-emptiveness allows us to estimate the necessary execution time of interactions among distributed objects and express several important timed operations, including timeout handling.[8]

In the following section we introduce two order relations with the notion of remote observation, using conceptual observers residing at specified locations and making comparisons between two remote objects (or object groups) by means of sending arbitrary messages to the objects and receiving messages from these. The observers can know only the length of the passage of time and the messages which can arrive at their own locations. They cannot observe any encapsulated computation nor any message passing at the locations except their own locations. The formal representation of the notion is given as labels for behavioral transitions like the weak transitions presented in [11].[9] We define the labeled transitions as follows:

**Definition 2.7** Let $a$ range over $\mathcal{M}$, $t$ range over $\mathcal{T}$, and $\ell, \ell'$ range over $\mathcal{L}$. The labeled translations are denoted as follows:

- $P \overset{\langle t\rangle\uparrow a}{\twoheadrightarrow}_{\ell} P'$ is defined as $P \rightarrow^{*} \langle t\rangle\epsilon\uparrow a.\mathbf{0}\!:\!\ell \mid P'$
- $P \overset{\langle t\rangle\ell'\downarrow a}{\twoheadrightarrow}_{\ell} P'$ is defined as $\langle t\rangle\ell'\uparrow a.\mathbf{0}\!:\!\ell \mid P \rightarrow^{*} P'$
- $P \rightsquigarrow\!\!\twoheadrightarrow^{t} P'$ is defined as $P \underbrace{\rightarrow^{*}\rightsquigarrow\rightarrow^{*} \;\cdots\; \rightarrow^{*}\rightsquigarrow\rightarrow^{*}}_{t \text{ times}} P'$

where $P \rightarrow^{*} P'$ is given as $P \rightarrow \cdots \rightarrow P'$. □

The above labeled transitions have the following intuitive meaning: $P \overset{\langle d\rangle\uparrow a}{\twoheadrightarrow}_{\ell} P'$ means that $P$ behaves as $P'$ and an observer located at $\ell$ receives message $a$ from $P$ after $d$ time units; $P \overset{\langle d\rangle\ell'\downarrow a}{\twoheadrightarrow}_{\ell} P'$ means that an observer located at $\ell$ sends message $a$ to an object at location $\ell'$ after $d$ time units and $P$ behaves as $P'$. Object $P$ may receive the message if it is located at $\ell'$; $P \rightsquigarrow\!\!\twoheadrightarrow^{t} P'$ means that $P$ becomes $P'$ after $t$ time units and arbitrary times of internal computations.

The following example illustrates how to describe distributed real-time objects in *TACS*.

**Example 2.8** We describe a simple communication protocol for an unreliable communication network. The protocol consists of a sender object at location $\ell_S$ and a receiver object at location $\ell_R$. The sender sends a message to the receiver ($\ell_R \uparrow \mathit{send}$) and waits for an acknowledgment ($\downarrow \mathit{ack}$). If it cannot receive the acknowledgment within eight time units, it retransmits the message. The receiver receives the message ($\downarrow \mathit{send}$) and then returns an acknowledgment ($\ell_S \uparrow \mathit{ack}$). These objects are described as follows:

[8] We leave details of a non pre-emptive version of this calculus to another paper [19].

[9] We can formalize the relations without using labels for transitions. However, the use of labeled transitions allows us to present and prove some properties of the relations more simply.

$$\begin{aligned}
\mathit{Sender}\,{:}\,\ell_S &\overset{\text{def}}{=} (\ell_R{\uparrow}\,\mathit{send}\,.\,({\downarrow}\,\mathit{ack}.\mathbf{0} + {\downarrow}\,\mathit{timeout}\,.\,\mathit{Sender})\,|\\
&\qquad\qquad\qquad \langle 8\rangle\,.\,\ell_S{\uparrow}\,\mathit{timeout}\,.\,\mathbf{0}) \setminus \{\mathit{timeout}\}\,{:}\,\ell_S\\
\mathit{Receiver}\,{:}\,\ell_R &\overset{\text{def}}{=} {\downarrow}\,\mathit{send}\,.\,\ell_S{\uparrow}\,\mathit{ack}\,.\,\mathbf{0}\,{:}\,\ell_R
\end{aligned}$$

We assume that communication from $\ell_S$ to $\ell_R$ takes $3\pm1$ time units and may occasionally fail, and one from $\ell_R$ to $\ell_S$ takes $2\pm1$ time units and may occasionally fail.

$$\begin{aligned}
\mathit{delay}(\ell_S,\ell_R) &= \{2,3,4\} \cup \{\infty\}\\
\mathit{delay}(\ell_R,\ell_S) &= \{1,2,3\} \cup \{\infty\}\\
\mathit{delay}(\ell_S,\ell_S) &= \{0\}
\end{aligned}$$

where $\infty$ corresponds to a communication failure. By expanding $(\mathit{Sender} : \ell_S \mid \mathit{Receiver} : \ell_R\ ) \setminus \{\mathit{send}, \mathit{ack}\}$ in the above transition, we can strictly analyze both the behavioral properties and the temporal properties of the entire system, including the influence of non-determinacy in communication delay.

For lack of space, when $\mathit{delay}(\ell_S,\ell_R)$ is three time units and $\mathit{delay}(\ell_R,\ell_S)$ is two time units, we demonstrate interactions between them as follows:

$$\begin{aligned}
&(\mathit{Sender}\,{:}\,\ell_S \mid \mathit{Receiver}\,{:}\,\ell_R) \setminus \{\mathit{send}, \mathit{ack}\}\\
&\ \rightarrow\ (\langle 3\rangle\,.\,\epsilon{\uparrow}\,\mathit{send}\,.\,\mathbf{0}\,{:}\,\ell_R \mid (({\downarrow}\,\mathit{ack}\,.\,\mathbf{0} + {\downarrow}\,\mathit{timeout}\,.\,\mathit{Sender})\,{:}\,\ell_S \mid\\
&\qquad\quad \langle 8\rangle\,.\,\ell_S{\uparrow}\,\mathit{timeout}\,.\,\mathbf{0}\,{:}\,\ell_S) \setminus \{\mathit{timeout}\} \mid {\downarrow}\,\mathit{send}\,.\,\ell_S{\uparrow}\,\mathit{ack}\,.\,\mathbf{0}\,{:}\,\ell_R) \setminus \{\mathit{send}, \mathit{ack}\}\\
&\ \leadsto^3\ (\epsilon{\uparrow}\,\mathit{send}\,.\,\mathbf{0}\,{:}\,\ell_R \mid (({\downarrow}\,\mathit{ack}\,.\,\mathbf{0} + {\downarrow}\,\mathit{timeout}\,.\,\mathit{Sender})\,{:}\,\ell_S \mid\\
&\qquad\quad \langle 5\rangle\,.\,\ell_S{\uparrow}\,\mathit{timeout}\,.\,\mathbf{0}\,{:}\,\ell_S) \setminus \{\mathit{timeout}\} \mid {\downarrow}\,\mathit{send}\,.\,\ell_S{\uparrow}\,\mathit{ack}\,.\,\mathbf{0}\,{:}\,\ell_R) \setminus \{\mathit{send}, \mathit{ack}\}\\
&\ \rightarrow\ ((({\downarrow}\,\mathit{ack}\,.\,\mathbf{0} + {\downarrow}\,\mathit{timeout}\,.\,\mathit{Sender})\,{:}\,\ell_S \mid\\
&\qquad\quad \langle 5\rangle\,.\,\ell_S{\uparrow}\,\mathit{timeout}\,.\,\mathbf{0}\,{:}\,\ell_S) \setminus \{\mathit{timeout}\} \mid \ell_S{\uparrow}\,\mathit{ack}\,.\,\mathbf{0}\,{:}\,\ell_R) \setminus \{\mathit{send}, \mathit{ack}\}\\
&\ \rightarrow\ ((({\downarrow}\,\mathit{ack}\,.\,\mathbf{0} + {\downarrow}\,\mathit{timeout}\,.\,\mathit{Sender})\,{:}\,\ell_S \mid\\
&\qquad\quad \langle 5\rangle\,.\,\ell_S{\uparrow}\,\mathit{timeout}\,.\,\mathbf{0}\,{:}\,\ell_S) \setminus \{\mathit{timeout}\} \mid \langle 2\rangle\,.\,\epsilon{\uparrow}\,\mathit{ack}\,.\,\mathbf{0}\,{:}\,\ell_S \mid \mathbf{0}\,{:}\,\ell_R) \setminus \{\mathit{send}, \mathit{ack}\}\\
&\ \leadsto^2\ ((({\downarrow}\,\mathit{ack}\,.\,\mathbf{0} + {\downarrow}\,\mathit{timeout}\,.\,\mathit{Sender})\,{:}\,\ell_S \mid\\
&\qquad\quad \langle 3\rangle\,.\,\ell_S{\uparrow}\,\mathit{timeout}\,.\,\mathbf{0}\,{:}\,\ell_S) \setminus \{\mathit{timeout}\} \mid \epsilon{\uparrow}\,\mathit{ack}\,.\,\mathbf{0}\,{:}\,\ell_S) \setminus \{\mathit{send}, \mathit{ack}\}\\
&\ \rightarrow\ (\mathbf{0}\,{:}\,\ell_S \mid \mathbf{0}\,{:}\,\ell_R) \qquad\qquad \textit{(successful)}
\end{aligned}$$

## 3 Verification for Distributed Real-Time Objects

This section presents two algebraic order relations on distributed real-time objects with respect to speed. In the last few years, many timed equivalence relations for synchronously communicating processes (or objects) have already been explored in timed extended process calculi (for example see [13, 14, 16]). These relations equate two processes only when their temporal and behavioral properties completely match. The temporal strictness of these relations is appropriate in verifying synchronously communicating real-time objects. This is because every sender object must synchronize its receiver one in synchronous communication settings Therefore, even if a sender object can send messages earlier than its temporal specification, the object may not send them at earlier timings, where its receiver object is as fast as the specification.

On the other hand, in asynchronous communication settings, sender objects do not have to synchronize their receiver objects and thus can send messages as soon as they can. Therefore, in the settings real-time objects do not need to completely match their own temporal specification, and need only to deliver messages to their receiver objects at *earlier* timings than those given in their specification.[10] This reveals that in the verification of asynchronous communicating real-time objects, "speed-sensitive" order relations are more suitable and practical than the above timed equivalences. In this section, we investigate algebraic order relations that decide whether two real-time objects are behaviorally equivalent and whether one of them can perform *faster* than the another.

## Relating Objects with Respect to Speed

Before defining the relations, we first illustrate the basic idea behind them. Suppose two simple objects: $A_1 \overset{\text{def}}{=} \langle 1 \rangle . \ell{\uparrow} a.S{:}\ell_A$ and $A_2 \overset{\text{def}}{=} \langle 3 \rangle . \ell{\uparrow} a.S{:}\ell_A$, where $\ell$ is a conceptual observer's location. $A_1$ can send message $a$ to the observer after one time unit, and $A_2$ can send the same message after three time units. That is, $A_1$ can send the message *sooner* than $A_2$. Therefore, we would consider object $A_1$ to be *faster* than object $A_2$. Now we define such a speed-sensitive order relation based on the notion of the observation bisimulation [11].

**Definition 3.1** A binary relation $\mathcal{R} \subseteq (\mathcal{P} \times \mathcal{P}) \times \mathcal{T} \times 2^{\mathcal{L}\mathrm{oc}}$ is a *$t$-speed prebisimulation* on $L$ if $(P_1, P_2) \in \mathcal{R}_t^L$ ($t \in \mathcal{T}$ and $L \subseteq \mathcal{L}\mathrm{oc}$) implies, for all $a, b \in \mathcal{M} \cup \{\epsilon\}$, $\ell, \ell' \in L$, and $d \in \mathcal{T}$;

(i) $\forall \ell_1, \forall m_1, \forall P_1' :\ P_1 \overset{\langle d \rangle \ell_1 {\downarrow} a}{\twoheadrightarrow}{}_{\ell} \rightsquigarrow\!\!\twoheadrightarrow^{m_1} \overset{\uparrow b}{\twoheadrightarrow}{}_{\ell'} P_1'$ *then*
$\qquad \exists \ell_2, \exists m_2, \exists P_2' :\ P_2 \overset{\langle d \rangle \ell_2 {\downarrow} a}{\twoheadrightarrow}{}_{\ell} \rightsquigarrow\!\!\twoheadrightarrow^{m_2} \overset{\uparrow b}{\twoheadrightarrow}{}_{\ell'} P_2'$ *and* $(P_1', P_2') \in \mathcal{R}^L_{t-m_1+m_2}$

(ii) $\forall \ell_2, \forall m_2, \forall P_2' :\ P_2 \overset{\langle d \rangle \ell_2 {\downarrow} a}{\twoheadrightarrow}{}_{\ell} \rightsquigarrow\!\!\twoheadrightarrow^{m_2} \overset{\uparrow b}{\twoheadrightarrow}{}_{\ell'} P_2'$ *then*
$\qquad \exists \ell_1, \exists m_1, \exists P_1' :\ P_1 \overset{\langle d \rangle \ell_1 {\downarrow} a}{\twoheadrightarrow}{}_{\ell} \rightsquigarrow\!\!\twoheadrightarrow^{m_1} \overset{\uparrow b}{\twoheadrightarrow}{}_{\ell'} P_1'$ *and* $(P_1', P_2') \in \mathcal{R}^L_{t-m_1+m_2}$

where $\ell_1 \in |P_1|_{Loc}$, $\ell_2 \in |P_2|_{Loc}$. We let $P_1 \preceq_t^L P_2$ if there exists a $t$-speed prebisimulation such that $(P_1, P_2) \in \mathcal{R}_t^L$. We call $\preceq_t^L$ *speed order* on $L$. We shall often abbreviate $\preceq_0^L$ as $\preceq^L$. $\square$

In the above definition, $\mathcal{R}_t^L$ 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.[11] $\preceq_t^L$ starts with a prebisimulation indexed by $t$ (i.e., $\mathcal{R}_t^L$) and can change $t$ as the bisimulation proceeds

[10] In our framework the order of the message arrival is assumed to be indeterminate; but if the order is sensitive, messages will need to be delivered without violating the order.

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

if $t \geq 0$. $L$ corresponds to the set of observable locations. The bisimulation makes two objects interact with arbitrary objects at $L$ and relates them according to the temporal and behavioral results of the interactions. $P_1$ (or $P_2$) may be an object or a group of objects at the same or different locations.

We here state the informal meaning of $P_1 \preceq_t^L P_2$. We first assume that conceptual observers are at locations in $L$ and that $P_1$ precedes $P_2$ by $t$ time units. An observer at location $\ell$ ($\ell \in L$) sends a message to $P_1$ after $d$ time units (written as $P_1 \xrightarrow{\langle d\rangle \ell_1 \downarrow a}_{\ell}$ in (i) ). It also sends the same message to $P_2$ after $d$ time units (written as $P_2 \xrightarrow{\langle d\rangle \ell_2 \downarrow a}_{\ell}$ in (ii) ). And then an observer at location $\ell'$ ($\ell' \in L$) receives return messages from both objects, $P_1$ and $P_2$, after $m_1$ time units and $m_2$ time units respectively (written as $\rightsquigarrow^{m_1} \xrightarrow{\uparrow b}_{\ell'} P_1{}'$ and $\rightsquigarrow^{m_2} \xrightarrow{\uparrow b}_{\ell'} P_2{}'$ ). If the arrival time of the return message from $P_1$ is earlier than that from $P_2$,[12] and if $P_1{}'$ and $P_2{}'$ can be successfully observed in $(P_1{}', P_2{}') \in \mathcal{R}^L_{t-m_1+m_2}$ in the same way, the observers judge that $P_1$ and $P_2$ are behaviorally equivalent and that $P_1$ can perform its behaviors *faster* than $P_2$.

We show several basic properties of the order relation below.

**Proposition 3.2** Let $P, P_1, P_2, P_3 \in \mathcal{P}$ $t, t_1, t_2 \in \mathcal{T}$, and $L \subseteq \mathcal{L}\mathrm{oc}$ then,

(1) $P \preceq_t^L P$

(2) *If* $P_1 \preceq_{t_1}^L P_2$ *and* $P_2 \preceq_{t_2}^L P_3$ *then* $P_1 \preceq_{t_1+t_2}^L P_3$

where for all $\ell \in L$ and $\dot{\ell} \in |P|_{Loc}$, $delay(\ell, \dot{\ell})$ and $delay(\dot{\ell}, \ell')$ are constant. □

From these results, we see that $P \preceq^L P$ and that if $P_1 \preceq^L P_2$ and $P_2 \preceq^L P_3$ then $P_1 \preceq^L P_3$. Hence, $\preceq^L$ is a preorder relation.

**Proposition 3.3** Let $P_1, P_2 \in \mathcal{P}$, $L \subseteq \mathcal{L}\mathrm{oc}$, and $t, t_1, t_2 \in \mathcal{T}$ then,

(1) *If* $P_1 \preceq_t^L P_2$ *and* $t \leq t'$ *then* $P_1 \preceq_{t'}^L P_2$

(2) *If* $P_1 \preceq_t^L P_2$ *and* $L' \subseteq L$ *then* $P_1 \preceq_t^{L'} P_2$ □

This proposition is significant in revealing the meanings of the indexes of $\preceq_t^L$. Proposition (1) means that $P_1$ performs at most $t$ time units faster than $P_1$ because of $P_1 \preceq_t^L P_2$. Hence, $P_1$ can still perform at most $t'$ time units faster than $P_1$ if $t \leq t'$. (2) means that $L$ in $\preceq_t^L$ corresponds to all the locations at which the observer can send and receive messages, and two objects ordered by an observer can be still ordered by another observer having a more limited scope.

**Proposition 3.4** Let $S_1 : \ell, S_2 : \ell, P_1, P_2 \in \mathcal{P}$, $L \subseteq \mathcal{L}\mathrm{oc}$, and $t, d \in \mathcal{T}$ then,

(1) *If* $S_1 : \ell \preceq_t^L S_2 : \ell$ *then* $\downarrow a.S_1 : \ell \preceq_t^L \downarrow a.S_2 : \ell$

(2) *If* $S_1 : \ell \preceq_t^L S_2 : \ell$ *then* $\langle d\rangle.S_1 : \ell \preceq_t^L \langle d\rangle.S_2 : \ell$

(3) *If* $S_1 : \ell \preceq_t^L S_2 : \ell$ *then* $\ell' \uparrow a.S_1 : \ell \preceq_t^L \ell' \uparrow a.S_2 : \ell$

(4) *If* $P_1 \preceq_t^L P_2$ *then* $P_1 \setminus N \preceq_t^L P_2 \setminus N$ □

[12] Note that $P_2$ already precedes $P_1$ by $t$ time units. Thus, the relative difference between the arrival time of the message from $P_2$ and that from $P_1$ is $t - m_1 + m_2$ time units.

**Remark** We defined the relation where its index $t$ may not be zero in order to investigate basic properties. We will usually use $\preceq^L_0$ in verifying systems.

**Example 3.5** We present some examples to demonstrate how $\preceq^L$ works. To simplify, we assume that communication delay can be ignored.

(1) $\quad \langle 1\rangle.\ell\!\uparrow a.\langle 3\rangle.\ell\!\uparrow b.\mathbf{0}:\ell' \;\preceq^L\; \langle 2\rangle.\ell\!\uparrow a.\langle 2\rangle.\ell\!\uparrow b.\mathbf{0}:\ell' \quad$ *where* $\ell \in L$

We verify this relation. We assume the following relation $\mathcal{R}^L_0 = (\,\langle 1\rangle.\ell\!\uparrow a.\langle 3\rangle.\ell\!\uparrow b.\mathbf{0}:\ell', \langle 2\rangle.\ell\!\uparrow a.\langle 2\rangle.\ell\!\uparrow b.\mathbf{0}:\ell'\,)$.

We first prove that the relation satisfies (i) in Definition 3.1. If $\langle 1\rangle.\ell\!\uparrow a.\langle 3\rangle\,\ell\!\uparrow b.\mathbf{0}:\ell' \rightsquigarrow\!\!\!\rightarrow^1 \xrightarrow{\uparrow a}\!\!\!\rightarrow_\ell \langle 3\rangle\,\ell\!\uparrow b.\mathbf{0}:\ell'$ then

$\langle 2\rangle.\ell\!\uparrow a.\langle 2\rangle.\ell\!\uparrow b.\mathbf{0}:\ell' \rightsquigarrow\!\!\!\rightarrow^2 \xrightarrow{\uparrow a}\!\!\!\rightarrow_\ell \langle 2\rangle.\ell\!\uparrow b.\mathbf{0}:\ell'$ and $(\langle 3\rangle.\ell\!\uparrow b.\mathbf{0}:\ell', \langle 2\rangle.\ell\!\uparrow b.\mathbf{0}:\ell') \in \mathcal{R}^L_{0-1+2}$.

Moreover, $\langle 3\rangle.\ell\!\uparrow b.\mathbf{0}:\ell' \rightsquigarrow\!\!\!\rightarrow^3 \xrightarrow{\uparrow b}\!\!\!\rightarrow_\ell \mathbf{0}:\ell'$ then

$\langle 2\rangle.\ell\!\uparrow b.\mathbf{0}:\ell' \rightsquigarrow\!\!\!\rightarrow^2 \xrightarrow{\uparrow b}\!\!\!\rightarrow_\ell \mathbf{0}:\ell'$ and $(\mathbf{0}:\ell', \mathbf{0}:\ell') \in \mathcal{R}^L_{1-3+2}$.

We can verify that $\mathcal{R}^L_0$ satisfies (ii) in Definition 3.1 in the same way.

(2) $\quad \langle 1\rangle.\downarrow a.\langle 3\rangle.\ell\!\uparrow b.\mathbf{0}:\ell' \;\not\preceq^L\; \langle 2\rangle.\downarrow a.\langle 2\rangle.\ell\!\uparrow b.\mathbf{0}:\ell' \quad$ *where* $\ell \in L$

It is enough to consider a counterexample. Suppose $\langle 1\rangle.\downarrow a.\langle 3\rangle.\ell\uparrow b.\mathbf{0}:\ell' \xrightarrow{\langle 3\rangle\ell'\downarrow a}\!\!\!\rightarrow_\ell \rightsquigarrow\!\!\!\rightarrow^3 \langle 3\rangle.\ell\!\uparrow b.\mathbf{0}:\ell'$, while $\langle 2\rangle.\downarrow a.\langle 2\rangle.\ell\!\uparrow b.\mathbf{0}:\ell' \xrightarrow{\langle 3\rangle\ell'\downarrow a}\!\!\!\rightarrow_\ell \rightsquigarrow\!\!\!\rightarrow^3 \langle 2\rangle.\ell\!\uparrow b.\mathbf{0}:\ell'$. Clearly, $(\langle 3\rangle.\ell\!\uparrow b.\mathbf{0}:\ell', \langle 2\rangle.\ell\!\uparrow b.\mathbf{0}:\ell')$ is not a timed pre-bisimulation.

(3) Next, assume these objects are allocated on different processors. For all $\ell \in L$, the communication delay from $\ell_1$ to $\ell$ is three time units and that from $\ell_2$ to $\ell$ is one time units: $delay(\ell_1,\ell)=\{3\}$ and $delay(\ell_2,\ell)=\{1\}$. we have:

$$\langle 2\rangle.\ell\!\uparrow a.\langle 2\rangle.\ell\!\uparrow b.\mathbf{0}:\ell_2 \;\preceq^L\; \langle 1\rangle.\ell\!\uparrow a.\langle 3\rangle.\ell\!\uparrow b.\mathbf{0}:\ell_1$$

## Cooperationability for Distributed Real-Time Objects

It is very convenient to develop a pre-congruence with respect to speeds in order to guarantee substitutability between two ordered objects. However, there is an undesirable problem in defining such a pre-congruence with temporal inequality. Suppose three objects: $A_1:\ell_A \overset{\text{def}}{=} \langle 1\rangle.\ell_B\!\uparrow a.\mathbf{0}:\ell_A$, $A_2:\ell_A \overset{\text{def}}{=} \langle 4\rangle.\ell_B\!\uparrow a.\mathbf{0}:\ell_A$, and $B_1:\ell_B \overset{\text{def}}{=} ((\downarrow a.B' + \downarrow b.B'')\,|\,\langle 2\rangle.\ell_B\uparrow b.\mathbf{0})\setminus\{b\}:\ell_B$, $B_2:\ell_B \overset{\text{def}}{=} \langle 1\rangle.((\downarrow a.B' + \downarrow b.B'')\,|\,\langle 2\rangle.\ell_B\!\uparrow b.\mathbf{0})\setminus\{b\}:\ell_B$, where $delay(\ell_A,\ell_B)=0$. We clearly have $A_1 \preceq^L A_2$ and $B_1 \preceq^L B_2$ but cannot expect that $A_1|B_1 \preceq^L A_2|B_2$. This fact reveals that a slower object cannot always be replaced by a faster object in a parallel composition with other objects. 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 $B_1$ and $B_2$. In order to define a rational pre-congruence with respect to speed, we here define another order relation which is a little more strict than the $\preceq^L_t$ relation, and a restricted expression.

**Definition 3.6** A binary relation $\mathcal{R} \subseteq (\mathcal{P} \times \mathcal{P}) \times \mathcal{T} \times 2^{\mathcal{L}oc}$ is a *$t$-timed pre-bisimulation* on $L$ if $(P_1, P_2) \in \mathcal{R}^t_L$ ($t \in \mathcal{T}$ and $L \subseteq \mathcal{L}oc$) implies, for all $a, b \in \mathcal{M}$, $\ell, \ell', \in L$, $\dot{\ell} \in |P_1|_{Loc} \cup |P_2|_{Loc}$, and $d_1, d_2 \in \mathcal{T}$ such that $d_1 \leq d_2 + t$;

(i) $\forall m_1, \forall P_1{}' : \; P_1 \xrightarrow{\langle d_1 \rangle \dot{\ell} \downarrow a}_{\ell} \rightsquigarrow\!\!\twoheadrightarrow^{m_1} P_1{}'$ *then*
$\qquad \forall m_2, \exists P_2{}' : \; P_2 \xrightarrow{\langle d_2 \rangle \dot{\ell} \downarrow a}_{\ell} \rightsquigarrow\!\!\twoheadrightarrow^{m_2} P_2{}'$ *and* $(P_1{}', P_2{}') \in \mathcal{R}^L_{t-m_1+m_2}$

(ii) $\forall m_2, \forall P_2{}' : \; P_2 \xrightarrow{\langle d_2 \rangle \dot{\ell} \downarrow a}_{\ell} \rightsquigarrow\!\!\twoheadrightarrow^{m_2} P_2{}'$ *then*
$\qquad \forall m_1, \exists P_1{}' : \; P_1 \xrightarrow{\langle d_1 \rangle \dot{\ell} \downarrow a}_{\ell} \rightsquigarrow\!\!\twoheadrightarrow^{m_1} P_1{}'$ *and* $(P_1{}', P_2{}') \in \mathcal{R}^L_{t-m_1+m_2}$

(iii) $\exists m_1, \forall P_1{}' : \; P_1 \rightsquigarrow\!\!\twoheadrightarrow^{m_1} \xrightarrow{\uparrow b}_{\ell'} P_1{}'$ *then*
$\qquad \exists m_2, \exists P_2{}' : \; P_2 \rightsquigarrow\!\!\twoheadrightarrow^{m_2} \xrightarrow{\uparrow b}_{\ell'} P_2{}'$ *and* $(P_1{}', P_2{}') \in \mathcal{R}^L_{t-m_1+m_2}$

(iv) $\exists m_2, \forall P_2{}' : \; P_2 \rightsquigarrow\!\!\twoheadrightarrow^{m_2} \xrightarrow{\uparrow b}_{\ell'} P_2{}'$ *then*
$\qquad \exists m_1, \exists P_1{}' : \; P_1 \rightsquigarrow\!\!\twoheadrightarrow^{m_1} \xrightarrow{\uparrow b}_{\ell'} P_1{}'$ *and* $(P_1{}', P_2{}') \in \mathcal{R}^L_{t-m_1+m_2}$

For $P_1, P_2 \in \mathcal{P}$, we let $P_1 \trianglelefteq^L_t P_2$ if there exists a timed pre-bisimulation such that $(P_1, P_2) \in \mathcal{R}^L_t$. We call $\trianglelefteq^L_t$ *timed order* on $L$. We shall often abbreviate $\trianglelefteq^L_0$ as $\trianglelefteq^L$. □

This relation is basically similar to $\preceq^L_t$ except that the observer is a little strict. That is, whereas the observer of $\preceq^L_t$ sends a message to the concerned objects after the same number of time units, the observer of $\trianglelefteq^L_t$ sends a message to the second (slower) object after arbitrarily more time units than the number of time units after which it sends to the (faster) object. Also, to allow easier discussion hereafter, we define a restricted expression below.

**Definition 3.7** Let $P \in \mathcal{P}$ and $L \subseteq \mathcal{L}oc$, if $P \trianglelefteq^L_0 P$, we call $P$ a *non-timeout* expression on $L$. □

We show several basic properties of the timed order.

**Proposition 3.8** Let $P, P_1, P_2, P_3 \in \mathcal{P}$ be non-timeout expressions on $L \in \mathcal{L}oc$ then,

(1) $P \trianglelefteq^L_t P$.
(2) *If* $P_1 \trianglelefteq^L_{t_1} P_2$ *and* $P_2 \trianglelefteq^L_{t_2} P_3$ *then* $P_1 \trianglelefteq^L_{t_1+t_2} P_3$ □

From the above result, for all $P, P_1, P_2, P_3 \in \mathcal{P}$ such that $P, P_1, P_2, P_3$ are non-timeout expressions on $L$, we know that if $P_1 \trianglelefteq^L P_2$ and $P_2 \trianglelefteq^L P_3$ then $P_1 \trianglelefteq^L P_3$. Hence, $\trianglelefteq^L$ is a preorder relation on non-timeout expressions.

**Proposition 3.9** Let $P_1, P_2 \in \mathcal{P}$, $L \subseteq \mathcal{L}oc$, and $t \in \mathcal{T}$ then,

(1) *If* $P_1 \trianglelefteq^L_t P_2$ *then* $P_1 \preceq^L_t P_2$
(2) If in Proposition 3.3 every $\preceq^L_t$ is replaced by $\trianglelefteq^L_t$, the proposition still hold. □

The first above result shows that $\preceq_t^L$ at least includes $\unlhd_t^L$. From the second result, we conclude that $\unlhd_t^L$ can preserve basic properties of $\preceq_t^L$ shown in Proposition 3.3.

**Proposition 3.10** Let $P_1, P_2 \in \mathcal{P}$, $delay_1, delay_2 \in \mathcal{D}$ such that $\forall \ell \in L, \forall \ell_2 \in |P_2|_{Loc}:\ delay_1(\ell, \ell_2) \leq delay_2(\ell, \ell_2)$ and $delay_1(\ell_2, \ell) \leq delay_2(\ell_2, \ell)$,

$$\textit{If}\quad P_1 \unlhd_t^L P_2 \quad \textit{assuming}\quad delay_1 \qquad \textit{then} \qquad P_1 \unlhd_t^L P_2 \quad \textit{assuming} \quad delay_2 \quad \Box$$

This proposition shows that a slower object is still slower even from an observer residing at a further location. However, we cannot assume the similar proposition about the first expression. That is, for all $delay_1, delay_2 \in \mathcal{D}$ such that $\forall \ell \in L, \ell_1 \in |P_1|_{Loc} : delay_1(\ell, \ell_1) \geq delay_2(\ell, \ell_1)$ and $delay_1(\ell_1, \ell) \geq delay_2(\ell_1, \ell)$ then, from $P_1 \unlhd_t^L P_2$ assuming $delay_1$, $P_1 \unlhd_t^L P_2$ assuming $delay_2$ is not usually derived. This is because the observer of $\unlhd_t^L$ (and $\preceq_t^L$) cannot notice any difference between its ordered objects before it receives their first messages.

**Proposition 3.11** Let $S_1{:}\ell, S_2{:}\ell, P_1, P_2 \in \mathcal{P}$, $L \subseteq \mathcal{L}\text{oc}$, and $t \in \mathcal{T}$ then,

(1) *If* $S_1{:}\ell \ \unlhd_0^L\ S_2{:}\ell$ *then* $\downarrow a.S_1{:}\ell \ \unlhd_0^L\ \downarrow a.S_2{:}\ell$
(2) *If* $S_1{:}\ell \ \unlhd_t^L\ S_2{:}\ell$ *then* $\ell'\!\uparrow a.S_1{:}\ell \ \unlhd_t^L\ \ell'\!\uparrow a.S_2{:}\ell$
(3) *If* $S_1 : \ell \ \unlhd_t^L\ S_2 : \ell$ *and* $L \cup (|P_1|_{Loc} \cup |P_2|_{Loc}) = \emptyset$ *then* $\langle t_1 \rangle.S_1 : \ell \ \unlhd_{t+t_1-t_2}^L\ \langle t_2 \rangle.S_2{:}\ell$
(4) *If* $P_1 \ \unlhd_t^L\ P_2$ *then* $P_1 \setminus N \ \unlhd_t^L\ P_2 \setminus N$ $\Box$

When $L \cup (|P_1|_{Loc} \cup |P_2|_{Loc}) \neq \emptyset$, we have that if $S_1 : \ell \ \unlhd_t^L\ S_2 : \ell$ and $t_1 \leq t_2$ then, $\langle t_1 \rangle.S_1 : \ell \ \unlhd_t^L\ \langle t_2 \rangle.S_2 : \ell$.

We now show a significant fact for proving substitutability between two behaviorally equivalent objects with different speeds.

**Proposition 3.12** Let $P_1, P_2, Q_1, Q_2 \in \mathcal{P}$ be non-timeout expressions on $L \subseteq \mathcal{L}$ such that $|P_1|_{Loc}, |P_2|_{Loc}, |Q|_{Loc} \subseteq L$, and let $P_1|Q_1$ and $P_2|Q_2$ be non-timeout expressions on $L$.

$$\textit{If}\quad P_1 \ \unlhd_t^L\ P_2 \quad \textit{and} \quad Q_1 \ \unlhd_t^L\ Q_2 \quad \textit{then} \quad P_1 \,|\, Q_1 \ \unlhd_t^L\ P_2 \,|\, Q_2 \qquad \Box$$

We directly know that if $P_1 \ \unlhd^L\ P_2$ and $Q_1 \ \unlhd^L\ Q_2$ then $P_1|Q_1 \ \unlhd^L\ P_2|Q_2$.

Intuitively, the result shows the following theoretically and practically significant facts:

- If two objects are behaviorally equivalent and one of the objects can perform faster than the other, the faster object can behaviorally be substituted for the slower one in a parallel composition.
- A parallel composition between the faster objects can really perform faster than one between the slower ones. That is, a system when embedding the faster objects can really perform faster than when embedding the slower ones.

*Remark* The expressive power of non-timeout expressions is weaker than that of $\mathcal{P}$ because any expressions which contain the anomalous contexts mentioned above are no longer definable, for example $((\downarrow a.A_1 + \downarrow b.A_2) \,|\, \langle 2\rangle.\ell\uparrow b.\mathbf{0}) \setminus \{b\} : \ell$ and $(\langle 2\rangle.\downarrow a.\mathbf{0} \,|\, \langle 4\rangle.\downarrow a.\mathbf{0}) : \ell$. However, we insist that this is not an unreasonable restriction because we never lose the expressive capability for time-dependent operations which make some other event executable due to the passing of time, in particular this restriction never affects any required expressiveness of the language in describing the real-time systems which contain no timeout handling, including hard real-time systems.

On the contrary, every expression in $\mathcal{P}$ can satisfy all the propositions presented in this section, if we alter the fourth rule of Definition 2.6 into "$P_1 \rightsquigarrow P_1{}'$ and $P_2 \rightsquigarrow P_2{}'$ imply $P_1|P_2 \rightsquigarrow P_1{}'|P_2{}'$".[13] However, this alternation allows an executable communication to be suspended for arbitrary periods of time.

## Example of Verification

For the remainder of this section we will present an example of the verification of distributed real-time objects to demonstrate how the order relations work.

**Example 3.13** We consider two printing service systems in a distributed system. The first system consists of two remotely located objects: a printer object ($\mathit{Printer}_1$) at location $\ell_P$ and a console object ($\mathit{Console}_1$) at location $\ell_C$. We denote the location of their environment as $\ell$.

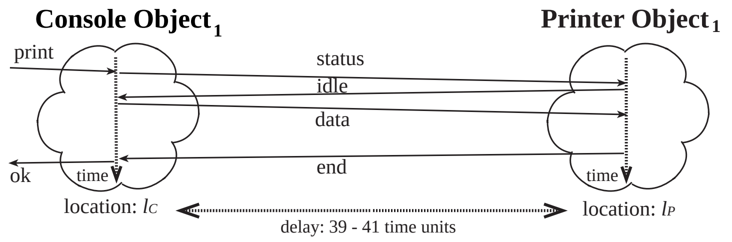

**Fig. 1.** The first printing service system

- Upon reception of a message ($\downarrow \mathit{print}$) from the environment, the console object queries the status of the printer object ($\ell_P \uparrow \mathit{status}$). If it receives a permission to send data ($\downarrow \mathit{idle}$), it sends data to the printer ($\ell_P \uparrow \mathit{data}$) after an internal execution for 5 time units and waits for a completion notice of the print ($\downarrow \mathit{end}$). After receiving the notice, it sends a message to the environment ($\ell \uparrow \mathit{ok}$).

[13] We leave further details of this alternative semantics to another paper [19].

- The printer object waits for a query message ($\downarrow status$) and then returns its status ($\ell_C \uparrow idle$) after 5 time units and waits for data transmission ($\downarrow data$). Since it takes 60 time units to print the data, it returns a print completion notice ($\ell_C \uparrow end$) after 60 time units.

The two objects are described as follows:

$$\begin{aligned} Console_1 : \ell_C &\overset{\text{def}}{=} \downarrow print \,.\, \ell_P \uparrow status \,.\, \downarrow idle \,.\, \langle 5 \rangle \,.\, \ell_P \uparrow data \,.\, \downarrow end \,.\, \ell \uparrow ok \,.\, \mathbf{0} : \ell_C \\ Printer_1 : \ell_P &\overset{\text{def}}{=} \downarrow status \,.\, \langle 5 \rangle \,.\, \ell_C \uparrow idle \,.\, \downarrow data \,.\, \langle 60 \rangle \,.\, \ell_C \uparrow end \,.\, Printer_1 : \ell_P \end{aligned}$$

The first system is described as a parallel composition of the objects as follows:

$$(Printer_1 : \ell_P \mid Console_1 : \ell_C) \setminus N_1 \ \text{ where } N_1 \ \overset{\text{def}}{=} \ \{status, idle, data, end\}$$

The second system consists of three remotely located objects: a printer object ($Printer_2$) at location $\ell_P$, an agent object ($Agent$) at location $\ell_A$, and a console object ($Console_2$) at location $\ell_C$. The agent object interacts with the printer object and the console object. The printer object is identical to that in the first system except for its message destinations.

- Upon reception of a message ($\downarrow print$), the console object sends the agent object a printing request ($\ell_P \uparrow req$) and then waits for a notice of printing start ($\downarrow started$). After receiving the notice, it sends a message to the environment ($\ell \uparrow ok$).
- The agent object receives a printing request message ($\downarrow req$). After an internal execution of 20 time units, it queries the printer about its status ($\ell_P \uparrow status$). If it receives the status ($\downarrow idle$), it sends data to the printer ($\ell_P \uparrow data$) after an internal execution of 20 time units, and then sends a message to the console ($\ell_C \uparrow started$). After that, it waits for next print request ($\downarrow req$) while waiting for a print completion notice ($\downarrow end$) from the printer object.

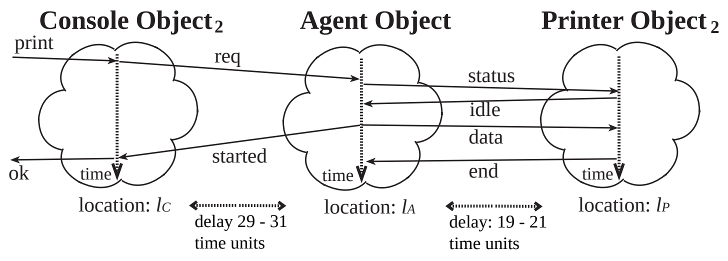

**Fig. 2.** The second printing service system

These objects are described as follows:

$$\begin{aligned}
\mathit{Console}_2\!:\!\ell_C &\overset{\text{def}}{=} \downarrow\!\mathit{print}\,.\,\ell_A\!\uparrow\!\mathit{req}\,.\,\downarrow\!\mathit{started}\,.\,\ell\!\uparrow\!\mathit{ok}\,.\,\mathbf{0}\!:\!\ell_C\\
\mathit{Agent}\!:\!\ell_A &\overset{\text{def}}{=} \downarrow\!\mathit{req}\,.\,\langle 20\rangle\,.\,\ell_P\!\uparrow\!\mathit{status}\,.\,\downarrow\!\mathit{idle}\,.\,\langle 20\rangle\,.\\
&\qquad\qquad (\ell_P\!\uparrow\!\mathit{data}\,.\,\ell_C\!\uparrow\!\mathit{started}\,.\,\downarrow\!\mathit{end}\,.\,\mathbf{0}\mid \mathit{Agent})\!:\!\ell_A\\
\mathit{Printer}_2\!:\!\ell_P &\overset{\text{def}}{=} \downarrow\!\mathit{status}\,.\,\langle 5\rangle\,.\,\ell_A\!\uparrow\!\mathit{idle}\,.\,\downarrow\!\mathit{data}\,.\,\langle 60\rangle\,.\,\ell_A\!\uparrow\!\mathit{end}\,.\,\mathit{Printer}_2\!:\!\ell_P
\end{aligned}$$

The whole second system is described as a parallel composition of the three object as follows:

$$\begin{aligned}
&(\mathit{Printer}_2\!:\!\ell_P \mid \mathit{Agent}\!:\!\ell_A \mid \mathit{Console}_2\!:\!\ell_C)\setminus N_2\\
&\qquad\qquad \text{where } N_2 \overset{\text{def}}{=} \{\mathit{req}, \mathit{status}, \mathit{idle}, \mathit{started}, \mathit{data}, \mathit{end}\}
\end{aligned}$$

We here compare the performances of these systems. We first assume that the communication delay between $\ell_C$ and $\ell_P$ is $40 \pm 1$ time units, that between $\ell_A$ and $\ell_P$ is $20 \pm 1$ time units, and that between $\ell_C$ and $\ell_A$ is $30 \pm 1$ time units.

$$\begin{aligned}
\mathit{delay}(\ell_P, \ell_C) = \mathit{delay}(\ell_C, \ell_P) = \{39, 40, 41\}\\
\mathit{delay}(\ell_A, \ell_P) = \mathit{delay}(\ell_P, \ell_A) = \{19, 20, 21\}\\
\mathit{delay}(\ell_C, \ell_A) = \mathit{delay}(\ell_A, \ell_C) = \{29, 30, 31\}
\end{aligned}$$

We also assume that the other communication channels do not exist. By using the timed order relation, the two systems are related as follows:

$$\begin{aligned}
&(\mathit{Printer}_1\!:\!\ell_P \mid \mathit{Console}_1\!:\!\ell_C)\setminus N_1\\
&\quad \trianglelefteq^L (\mathit{Printer}_2\!:\!\ell_P \mid \mathit{Agent}\!:\!\ell_A \mid \mathit{Console}_2\!:\!\ell_C)\setminus N_2\\
&\qquad\qquad \text{where } L \subseteq \mathcal{L}\text{oc such that } \ell, \ell_A, \ell_C, \ell_P \in L
\end{aligned}$$

The above result shows that the two systems are behaviorally equivalent but that the first system can perform faster than the second one.

Also, by using the timed order relation, we can verify that the systems satisfy their specification. For example, let $\mathit{Spec}\!:\!\ell_C$ be a specification for the systems.

$$\mathit{Spec}\!:\!\ell_C \overset{\text{def}}{=} \downarrow\!\mathit{req}\,.\,\langle 300\rangle\,.\,\ell\!\uparrow\!\mathit{ok}\,.\,\mathbf{0}\!:\!\ell_C$$

By using Definition 3.1, we conclude that:

$$\begin{aligned}
(\mathit{Printer}_1\!:\!\ell_P \mid \mathit{Console}_1\!:\!\ell_C)\setminus N_1 &\preceq^L \mathit{Spec}\!:\!\ell_C\\
(\mathit{Printer}_2\!:\!\ell_P \mid \mathit{Agent}\!:\!\ell_A \mid \mathit{Console}_2\!:\!\ell_C)\setminus N_2 &\preceq^L \mathit{Spec}\!:\!\ell_C
\end{aligned}$$

The above inequalities show that the two systems can execute the behaviors given in the specification faster than the required execution time in the specification.

Next we will consider a reconstruction (or an improvement) of the second system. Suppose another agent object (a new implementation) described as the following $\mathit{Agent}'$:

$$\begin{aligned}
\mathit{Agent}' &\overset{\text{def}}{=} \downarrow\!\mathit{req}\,.\,\langle 5\rangle\,.\,\ell_P\!\uparrow\!\mathit{status}\,.\,\downarrow\!\mathit{idle}\,.\,\langle 5\rangle\,.\\
&\qquad\qquad (\ell_P\!\uparrow\!\mathit{data}\,.\,\ell_C\!\uparrow\!\mathit{started}\,.\,\downarrow\!\mathit{end}\,.\,\mathbf{0}\mid \mathit{Agent}')
\end{aligned}$$

When we apply the two agent expressions to Definition 3.1, we know the following result:

$$\mathit{Agent}' : \ell_A \trianglelefteq^L \mathit{Agent} : \ell_A$$

The above inequality tells that *Agent* and *Agent*$'$ are behaviorally equivalent and *Agent* can perform faster than *Agent*$'$. In order to improve the performance of the whole second system, we replace *Agent*$'$ by *Agent* in the system. We will need to verify that the new second system really performs faster than the old one without changing any behavioral properties. From the inequality and Proposition 3.12, we can directly know the following desired fact:

$$\begin{aligned}&(\mathit{Printer}_2 : \ell_P \mid \mathit{Agent}' : \ell_A \mid \mathit{Console}_2 : \ell_C)\backslash N_2 \\ &\qquad \trianglelefteq^L (\mathit{Printer}_2 : \ell_P \mid \mathit{Agent} : \ell_A \mid \mathit{Console}_2 : \ell_C)\backslash N_2\end{aligned}$$

This demonstrates that *Agent*$'$ can be behaviorally substituted for *Agent* in the second system and that the new second system can perform faster than the old one.[14] Moreover, by using the timed order relation we can compare the new system and the first system.

$$\begin{aligned}&(\mathit{Printer}_2 : \ell_P \mid \mathit{Agent}' : \ell_A \mid \mathit{Console}_2 : \ell_C) \setminus N_2 \\ &\qquad \trianglelefteq^L (\mathit{Printer}_1 : \ell_P \mid \mathit{Console}_1 : \ell_C) \setminus N_1\end{aligned}$$

This holds for $\preceq^L$ as well as $\trianglelefteq^L$.

## 4 Related Work

A number of frameworks for specifying and verifying distributed real-time systems have already been proposed based on temporal logic, automata, and process calculi. However, although there is a close relationship between delay and asynchrony in communication, general theories for modeling both delay and asynchrony seem to have been neglected in favor of ad hoc methods. On the other hand, several formal models for concurrent object-oriented computation have previously been devised. Among these, process calculus [2, 6, 11, 12] is a well-studied theory that can naturally model concurrent objects as communicating processes. In this section we compare our work with some existing process calculi for modeling asynchronous communication and real-time.

Recently, several researchers have explored process calculi-related frameworks for asynchronous communication (inside or outside of object orientation settings). Most of the frameworks introduced auxiliary mechanisms: buffering, for example see [3, 7]. However, these extensions are not always suitable to deal with computational aspects of process calculi. On the other hand, Honda and Tokoro in [10], and Agha et al. in [1] proposed process calculi based on the notion of actor-like objects with asynchronous communication [9]. In particular, the

[14] Note that all the expressions in this example are sound.

calculus in [10] is very similar to ours, in which it expresses asynchronous messages as newly created output processes. It also provides an equivalence theory for asynchronous communicating objects. However, its purpose is to construct a purely theoretical foundation for asynchronous communication with port passing mechanism,[15] and it does not provide any support for the notion of time.

There have been many timed extensions of process calculi for synchronous communication, for example [8, 14, 16]. There is a notable work by Moller and Tofts in [13] where the authors studied a preorder relation over timed processes with respect to speed based on the bisimulation concept, like ours. However, their order relation is seriously dependent on synchronous communication and thus cannot deal with asynchronous interactions among distributed objects. Also their calculus, unlike ours, allows an executable communication to be suspended for arbitrary periods of time. As a result, the calculus cannot exactly analyze the execution time of systems, and their relation shows only that a process may *possibly* execute faster than another. On the other hand, in asynchronous communicating settings such a speed sensitive order relation does not seem to have been explored within process calculi.

Baeten and Bergstra in [3] proposed a process calculus with the ability to express asynchronous communication channel with latency and failure, based on a timed extended calculus of ACP [2]. Their calculus introduces asynchronous messages by the creation of processes corresponding to the messages and describes communication delay as a suspension of the created process for the length of the delay time, like ours. However it does not have any speed-sensitive order relation for asynchronously communicating processes.

## 5 Conclusion

In this paper we proposed a framework of specification and verification for distributed real-time object-oriented systems and studied its basic properties. The framework is formulated on the basis of two algebraic order relations for a new timed extended process calculus. The calculus is unique among existing process calculi in expressing both delay and asynchrony in communication. It provides a powerful method to describe temporal and behavioral properties of remotely located real-time objects and their asynchronous interactions with communication delay. Also we presented two speed-sensitive order relations. They are defined based on the bisimulation concept and can decide whether two real-time objects are behaviorally equivalent and whether one of them can perform its behaviors faster than the other. Also, they allow us to guarantee that a faster object can be functionally substituted for a slower one in a system and that the system embedding the faster one can perform even faster than the systems embedding the slower one. Since in asynchronous communication settings, it often is necessary only to verify that real-time objects can perform faster than what is required by

[15] For focusing temporal properties in distributed real-time objects, our calculus avoids modeling a port passing mechanism, but we can easily introduce such a mechanism into the calculus by using the approach developed in [12].

their specification, these relations offer a theoretical and practical foundation for proving the correctness and the reusability of asynchronous interacting real-time objects.

Finally, we would like to point out some further issues. Our framework is not sensitive to the order of message arrival; but in several distributed systems, arriving messages are often queued in FIFO order. We therefore are very interested in developing a process calculus to reason about the arrival order of messages. In this paper we inherently assume the existence of a global clock, but in distributed systems each processor often follows its own local clock. We have already developed a method for describing multiple local clocks in [17]. We plan to introduce the method into the framework presented in this paper. We developed some techniques to define semantics for object-oriented real-time languages with synchronous communication primitives based on a timed extended process calculus in [18]. We believe that the calculus presented here allows us to define semantics of object-oriented real-time languages with various asynchronous communication mechanisms.

**Acknowledgments** We are grateful to J. Sifakis (VERIMAG) for discussions on the topics treated here. We would like to thank to R. Pareschi, J.-M. Andreoli, K. Beesley (Rank Xerox Research Centre), and V. Vasconcelos (Universidade Nova de Lisboa) for significant comments on earlier versions of this paper, and anonymous referees for providing many constructive and valuable suggestions.

## References

1. Agha, G., Mason, I., Smith, S., and Talcott, C., *Towards a Theory of Actor Computations*, Proceedings of CONCUR'92, LNCS 630, p???-???, August, 1992.
2. Baeten, J. C. M., and Bergstra, J. A., *Process Algebra*, Cambridge University Press, 1990.
3. 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, May, 1991.
4. Bergstra, J. A., and Klop, J. W., *Process Algebra with Asynchronous Communication Mechanisms*, Seminar on Concurrency, LNCS 197, p76-95, 1985.
5. Boudol, G., Castellani, I., Hennessy, M., and Kiehn, A., *A Theory of Processes with Localities*, Proceedings of CONCUR'92, LNCS 630, p108-122, August, 1992.
6. Brinksma, E., *A tutorial on LOTOS*, Proceedings, IFIP Workshop on Protocol Specification, Testing and Verification, p73-84, North-Holland, 1986.
7. de Boer, F.S., Klop, J.W., and Palamidessi, *Asynchronous Communication in Process Algebra*, Proceedings of LICS'92, p137-147, June, 1992.
8. Hennessy, M., *On Timed Process Algebra: a Tutorial*, Technical Report 2/93, University of Sussex, 1993
9. Hewitt, C., *Viewing Control Structures as Pattern of Passing Messages*, Journal of Artificial Intelligence, Vol. 8, No.3, 1977.
10. Honda, K., and Tokoro, M., *An Object Calculus for Asynchronous Communication*, Proceedings of ECOOP'91, LNCS 512, p133-147, June, 1991.
11. Milner, R., *Communication and Concurrency*, Prentice Hall, 1989.
12. Milner, R., Parrow. J., Walker, D., *A Calculus of Mobile Processes*, Information and Computation, Vol.100, p1-77, 1992.

13. Moller, F., and Tofts, C., *Relating Processes with Respect to Speed*, Proceedings of CONCUR'91, LNCS 527, p424-438, August, 1991.
14. Nicollin. X., and Sifakis, J., *An Overview and Synthesis on Timed Process Algebras*, Proceedings of Computer Aided Verification, LNCS 575, p376-398, June, 1991.
15. Nierstrasz, O. M., and Papathomas, M., *Viewing Objects as Patterns of Communicating Agents*, Proceedings of ECOOP/OOPSLA'90, October, p38-43, 1990.
16. Satoh, I., and Tokoro, M., *A Formalism for Real-Time Concurrent Object-Oriented Computing*, Proceedings of OOPSLA'92, p315-326, October, 1992.
17. Satoh, I., and Tokoro, M., *A Timed Calculus for Distributed Objects with Clocks*, Proceedings of ECOOP'93, LNCS 707, p326-345, July, 1993.
18. Satoh, I., and Tokoro, M., *Semantics for a Real-Time Object-Oriented Programming Language*, Proceedings of IEEE International Conference on Computer Languages, p159-170, May, 1994.
19. Satoh, I., and Tokoro, M., *A Formalism for Remotely Interacting Processes*, Proceedings of Workshop on Theory and Practice of Parallel Programming, November, 1994. Also a revised version will appear in LNCS, 1995.
20. Tokoro, M., and Satoh, I., *Asynchrony and Real-Time in Distributed Systems*, Proceedings of Parallel Symbolic Computing: Languages, Systems, and Application, LNCS 748. p318-330, 1993.
21. Yonezawa, A., and Tokoro, M., editors, *Object-Oriented Concurrent Programming*, MIT Press, 1987.