プロセス代数に基づく並列プログラムの最適化

佐藤 一郎

お茶の水女子大学 理学部 情報科学科

要旨

本研究では並列プログラムの最適化を検証するための理論体系を提案していく。これはプログラム中の通信を記述するためのプロセス代数と、それに基づく代数的順序関係を中心にして定式化される。この順序関係は並列プログラムを同期及び通信のタイミングを基準にしてその実行速度を比較するものであり、並列計算の性能解析に対して理論的基礎を提供するとともに、プログラム最適化による動作内容と実行速度への影響をに解析できるようにする。本稿ではこの理論体系の初期的な成果を概説する。並列プログラムの最適化への応用を展望する。

A Process Algebra for Optimization in Parallel Programs

— Extended Abstract —

Ichiro Satoh*

Department of Information Science, Ochanomizu University†

Abstract

We propose a theoretical framework for the performance analysis and optimization of parallel programs through a time-extended process algebra with a speed-sensitive order relation. The relation is formulated based on bisimulation concept and is characterized by having the ability to order behaviorally equivalent programs with respect to their relative speeds, and useful algebraic properties. This abstract outlines some early results on the framework.‡

1 Introduction

In parallel computing systems, including distributed ones, interactions among processes, such as communication and synchronization, strongly affect the performance and correctness of these systems. Therefore, to minimize the overheads of such interactions and to optimize the timings of them are important and necessary. Many optimization techniques for communications and synchronizations have been explored and have provided a lot of significant and effective results. However, most of them often lack any qualitative and quantitative investigation about their effectiveness and validation. Moreover, parallel and distributed computation is far more complex than sequential one. We need a theoretical framework to strictly reason about optimization techniques for parallel computation.

On the other hand, over the last few years, many researchers have proposed time-extended process algebras for example see [3, 4, 10, 15]. The algebras provide widely studied frameworks for modeling and verifying real-time systems. Such theories typically consist of a simple language with the expressiveness of basic control flow of programs, communications, and time-dependent behaviors. They have a well-defined operational semantics given in terms of labeled transition systems. Some of them are also equipped with time-sensitive equivalence relations that can be used to relate implementations and specifications, which are both given as terms in the language. The relations equate two processes when both their functionally behavioral properties and their temporal properties are *completely* matched with each other. Although the algebras have the ability to deal with various temporal cost of parallel computation, the relations are not appropriate for analyzing the performance and optimization of parallel computation, since the main goal of the relations is to provide the verification of

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

†2-1-1 Otsuka Bunkyo-ku Tokyo, 112 Japan.

‡This abstract summarizes our previous work presented in [13]. Instead, we plan to present some detail results of our current work at the meeting.

real time systems.

This extended abstract outlines our basic ideas on a new theoretical framework for qualitatively and quantitatively verifying timing optimization for communications among parallel programs. The framework is formulated through a new process algebra and a speed-sensitive order relation based on the algebra.

We here present the organization of this extended abstract: In the next section we briefly present our basic ideas concerning the process algebra and define its syntax and semantics. In Section 3 we define a speed-sensitive order relation on communicating processes and study its theoretical properties. In Section 4 we compare with related work and give some concluding remarks.

2 Description Language

This section defines a process calculus for reasoning about temporal costs of parallel programs, for example execution time and synchronization time. The calculus is defined by incorporating many common features of existing process calculi and is characterized by having the ability to express time as other existing time-extended process algebras.

Before defining the syntax of the calculus, we give the notations of time values. We assume that time is interval between events instead of any absolute time, and is continuous, instead of discrete time.

Definition 2.1 Let $\mathcal{T}$ denote the set of the positive real numbers including 0. □

Time values are often denoted as positive real numbers including zero. We define symbols to present the events of processes.

Definition 2.2

- 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.

In the calculus, we describe communications of parallel programs by means of the language defined below. The syntax of the calculus is coincide with all the construction of some existing process calculi, for example CCS[6], except for 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. On the other hand, to preserve the pleasant properties of the original process calculi, all communication and internal actions are assumed to be instantaneous. Instead, various temporal costs in computation and communication can be represented by means of the new operator.

Definition 2.3 The set $\mathcal{P}$ of process expressions ranged over by $P, P_1, P_2, \ldots$ is defined recursively by the following abstract syntax:

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

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:

- $\mathbf{0}$ is a terminate process that 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 $L \cup \overline{L}$.
- $A \stackrel{\text{def}}{=} P$ 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 2.4 The syntactic sort of each process, $\mathcal{L}(P)$, is defined inductively by:

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

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

Next, we give the semantics of the calculus. 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 $\xrightarrow{\alpha}$ ($\longrightarrow \subseteq \mathcal{P} \times Act \times \mathcal{P}$) and the another defines the passage of time on processes, called *temporal transition*, written as $\xrightarrow{\langle t\rangle}$ ($\longrightarrow \subseteq \mathcal{P} \times \Gamma \times \mathcal{P}$). The advance of time is modeled as the latter transition. It 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.

Definition 2.5 The calculus is a labeled transition system $\langle\ \mathcal{P},\ Act \cup \Gamma,\ \{ \xrightarrow{\mu} \subseteq \mathcal{P} \times \mathcal{E} \mid \mu \in Act \cup \Gamma\ \}\ \rangle$. 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.

We briefly explain some important transitions shown in Definition 2.5. Time passes in all processes at the same speed. Also, all processes follow the same clock, or different ones but well-synchronized clocks. The semantics assumes that when an internal or communication action is enabled, processes must perform the action immediately without imposing unnecessary idling.[1] This assumption is the same as the notion of *maximal progress* shown in [4, 15]. It lets us exactly measure necessary time for synchronization among parallel processes, and enables the calculus to preserve the observation properties of many existing non-timed process calculus.

$$\frac{-}{\alpha.P \xrightarrow{\alpha} P} \qquad \frac{P \xrightarrow{\alpha} P',\ \alpha \notin L \cup \overline{L}}{P \setminus L \xrightarrow{\alpha} P' \setminus L}$$

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

$$\frac{P_1 \xrightarrow{\alpha} P_1'}{P_1 | P_2 \xrightarrow{\alpha} P_1' | P_2} \qquad \frac{P_2 \xrightarrow{\alpha} P_2'}{P_1 | P_2 \xrightarrow{\alpha} P_1 | P_2'}$$

$$\frac{P_1 \xrightarrow{l} P_1',\ P_2 \xrightarrow{\overline{l}} P_2'}{P_1 | P_2 \xrightarrow{\tau} P_1' | P_2'}$$

$$\frac{P \xrightarrow{\alpha} P'}{A \xrightarrow{\alpha} P'} (A \stackrel{\mathrm{def}}{=} P) \qquad \frac{P \xrightarrow{\alpha} P'}{\langle 0\rangle.P \xrightarrow{\alpha} P'}$$

Figure 1: Inference Rules for Behavioral Transition

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

$$\frac{P_1 \xrightarrow{\langle t\rangle} P_1',\ P_2 \xrightarrow{\langle t\rangle} P_2'}{P_1 + P_2 \xrightarrow{\langle t\rangle} P_1' + P_2'}$$

$$\frac{P_1 \xrightarrow{\langle t\rangle} P_1',\ P_2 \xrightarrow{\langle t\rangle} P_2'}{P_1 | P_2 \xrightarrow{\langle t\rangle} P_1' | P_2'} (P_1 | P_2 \not\xrightarrow{\tau})$$

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

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

Figure 2: Inference Rules for Temporal Transition

Example 2.6 We show some basic examples of processes in $\mathcal{P}$ 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.

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

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

(3) After three time units, $\langle 2\rangle.\overline{a}.P_1 \,|\, \langle 3\rangle.(a.P_2 + b.P_3)$ 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.

$$\langle 2\rangle.\overline{a}.P_1 \,|\, \langle 3\rangle.(a.P_2 + b.P_3)$$

[1] 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 [12].

$$\xrightarrow{(3)} \quad \overline{a}.P_1 \mid (a.P_2 + b.P_3)$$
$$\xrightarrow{\tau} \quad P_1 \mid P_2$$

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 2.7

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

In the following section, we present an algebraic inequality over process expressions in the calculus. However, in order to give a rational theory, we need to impose some certain syntactic restrictions on processes.

Definition 2.8

(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 2.9 $\quad P \in \mathcal{P}$ 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 2.3. □

Interprocess communication in real communicating systems is often realized by means of *asynchronous* style as well as *synchronous* one. The word *asynchrony* here means that sender processes can send messages without synchronizing any processes. However, most of existing process calculi, including this calculus are formulated based on synchronous communication. However, the calculus can essentially 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 $\overline{a}.\mathbf{0}|P$. We leave this extension to another paper [13].

3 Speed-Sensitive Prebisimulation

Based on time-extended process calculi, several researchers have explored time-sensitive equivalence relations that are based on trace equivalence, failure equivalence, testing equivalence, and bisimulation equivalence, for example see [4, 10, 15]. These equivalence relations equate two processes if they cannot be distinguished from each other in their temporal properties as well as their behavioral one. However, the relations 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 unpredictable 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. It is convenient to construct a 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).

This section develops such an algebraic order relation on processes with respect to their speeds based on the bisimulation concept.

Definition 3.1 A binary relation $\mathcal{R} \subseteq (\mathcal{P} \times \mathcal{P}) \times \mathcal{T}$ is a *t-prebisimulation* over communicating processes if $(P_1, P_2) \in \mathcal{R}_t$ $(t \geq 0)$ implies, for all $\alpha \in Act$;

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

(ii) $\forall d\ \forall P_2'$: $P_2 \stackrel{(d)}{\Longrightarrow}\stackrel{\alpha}{\Longrightarrow} P_2'$ *then* $\exists P_1'$: $P_1 \stackrel{(d)}{\Longrightarrow}\stackrel{(t)}{\Longrightarrow}\stackrel{\widehat{\alpha}}{\Longrightarrow} P_1'$ *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.[2] The following order relation

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

starts with a prebisimulation indexed by t (i.e., $\mathcal{R}_t$) and can change t as the bisimulation proceeds only if $t \geq 0$.

Definition 3.2 We let $P_1 \leq^t P_2$ if there exists some t-prebisimulation such that $(P_1, P_2) \in \mathcal{R}_t$. We call $\leq^t$ *speed-sensitive order* on communicating processes. We shall often abbreviate $\leq^0$ as $\leq$. □

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

Proposition 3.3 Let $P, P_1, P_2, P_3 \in \mathcal{P}$. Then,

(1) $P \leq P$

(2) $P_1 \leq P_2$ *and* $P_2 \leq P_3$ *then* $P_1 \leq P_3$ □

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

Proposition 3.4 Let $P_1, P_2 \in \mathcal{P}$, $t_1, t_2 \in \mathcal{T}$ such that $t_1 \leq t_2$. Then, $\langle t_1 \rangle.P \leq \langle t_2 \rangle.P$ □

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

Example 3.5 We show some basic examples of $\leq$ as follows:

(1) $a.P \leq \langle 1 \rangle.a.P$

(2) $\langle 1 \rangle.\overline{a}.\langle 2 \rangle.\overline{b}.P \leq \langle 1 \rangle.\overline{a}.\langle 3 \rangle.\overline{b}.P$

(3) $a.P_1|\langle 1 \rangle.b.P_2 \leq \langle 1 \rangle.a.P_1|\langle 1 \rangle.b.P_2$

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

Proposition 3.6 Let $P_1, P_2, Q \in \mathcal{P}$ such that $P_1 \leq P_2$. Then

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

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

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

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 $A_1 \leq A_2$ but cannot expect that $A_1|B \leq A_2|B$, because $A_1|B \stackrel{\langle 2 \rangle}{\longrightarrow}\stackrel{\tau}{\longrightarrow} P_1|\langle 1 \rangle.\overline{b}.\mathbf{0}$ and $A_2|B \stackrel{\langle 3 \rangle}{\longrightarrow}\stackrel{\tau}{\longrightarrow} \langle 1 \rangle.\overline{a}.\mathbf{0}|P_2$. 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. However, when we restricted processes to be included in the time-stable process set given in the previous section, they can perform executable actions in any order and thus the order relation is preserved in parallel context.

Proposition 3.7 Let $P_1, P_2, Q \in \mathcal{P}$ be time-stable processes. Then,

$$P_1 \leq P_2 \quad \textit{then} \quad P_1 \,|\, Q \leq P_2 \,|\, Q \qquad \square$$

In order to prove the above result, we need some lemmas, including a fact that any time-stable processes are confluent. However, for lack of space, we leave its detail proof to another paper.

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 Discussion

Related Work

We briefly survey related work. There have indeed been many process calculi for reasoning about temporal properties of communicating systems, for example see [3, 4, 8, 10, 14, 15]. Most of the calculi have been equipped with time-sensitive equivalence relations as verification methods. However, only a few of them intend to analyze and compare temporal costs of communicating processes, for example [5, 8, 12, 14].

Among them, Moller and Tofts in [8] 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 [14] and Jenner and Vogler in [5] presented speed-sensitive preorder relations based on testing equivalence. 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 [2]. 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 [1], and Natarajan and Cleaveland [9] propose approaches to relate processes with respect to their relative efficiencies according to the number of necessary internal actions, τ-actions through the same communication. 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.

Concluding Remarks

This abstract gives only a starting point for formulating a theoretical framework for the performance analysis and optimization of communicating processes. There are many issues that we leave in this abstract.

This abstract proposed a speed-sensitive order relation for end-to-end synchronous communication. However, 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. We are interested in formulating a speed-sensitive order relation for asynchronously communicating processes.

In synchronous communication settings, processes must be blocked until their partner processes are ready to communicate. The order relation presented in this paper 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 that it sends are received by processes. This difference between synchrony and asynchrony in communication means that a suitable speed-sensitive order relation corresponding to asynchronous communication is needed.[3] The relation has to be able to know only the arrival timings of return messages. 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. Comparisons between the two relations can reveal essential differences between synchrony and asynchrony in interactions among processes in time-sensitive contexts.

[3] We presented such a speed-sensitive order relation for asynchronously communicating processes in [13].

References

[1] Arun-Kumar, S. and Hennessy, M., *An Efficiency Preorder for Processes*, Acta Informatica, Vol.29, 737-760, 1992.

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

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

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

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

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

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

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

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

[10] 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.

[11] Satoh, I., and Tokoro, M., *Semantics for a Real-Time Object-Oriented Programming Language*, Proceedings of IEEE International Conference on Computer Languages (ICCL'94), p159-170, May, 1994.

[12] 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, 1994.

[13] Satoh, I., *Speed-Sensitive Orders for Communicating Processes* Concurrency Theory and Application'96, (an extended version appeared in Research Institute for Mathematical Sciences Report 996, kyoto Unversity, 1997.

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

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