A Calculus of Countable Broadcasting Systems

Yoshinao ISOBE, Yutaka SATO, Kazuhito OHMAKI

Computer Science Division, Electrotechnical Laboratory
1-1-4 Umezono, Tsukuba, Ibaraki 305, Japan
E-mail:{ isobe|ysato|ohmaki }@etl.go.jp

Abstract. In this paper we propose a process algebra named *CCB* (a Calculus of Countable Broadcasting Systems). We define an observational congruence relation in CCB after basic definitions of CCB, and give a sound and complete axiom system for the congruence relation of finite agents.
CCB is developed for analyzing a *multi-agent model* with broadcast communication. The most important property of CCB is that a broadcaster of a message can know the number of receivers of the message after broadcasting. The property is not easily described in the other process algebras.
The multi-agent model is useful for constructing extensible systems. A disadvantage of the multi-agent model is that agents must be designed very carefully because unexpected behavior may arise by interactions between the agents. Therefore we want to analyze behavior of the agents.

1 Introduction

A design of software should be divided into several program components called *agents* which may be executed concurrently and communicate with each other through *events*, in order to develop and refine them independently thus efficiently. This approach is called a *multi-agent model*[1]. The advantages of the multi-agent model can be summarized as follows:

1. *Choice of different description languages:* Each agent can be programmed by its appropriate language and communicate with other agents through some standardized protocols modeled as events.
2. *Software reusability:* If each agent is carefully designed, it can be shared and reused. The interfaces (i.e., protocols) between agents must be designed simple and flexible to achieve this goal.
3. *Machine independence:* Machine independence should be realized in recent computer systems which contain a wide variety of machines in one network. The multi-agent model will also fit to realize this machine independence.

In order to flexibly connect agents, one of the authors has developed a mechanism named VIABUS[13] which has a software bus architecture with *broadcast communication* based on the multi-agent model. The advantage of broadcast communication is that a broadcaster of an event need not be modified when new receivers of the event are attached to VIABUS, because destinations of the event are not specified. Thus broadcast communication is useful for constructing extensible software systems. For example, VIABUS has been used for an extensible User Interface Management System named VIAUIMS[13] as shown in Fig.1.

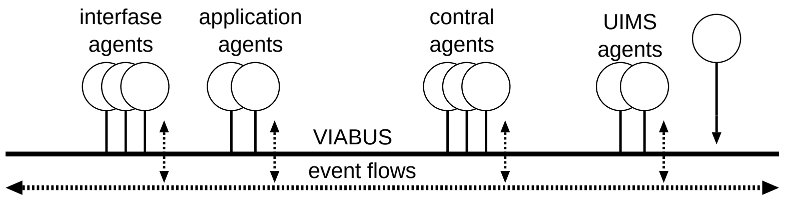

Fig. 1. The architecture of VIAUIMS

Though the multi-agent model is useful for efficient development and refinement of software, agents should be designed very carefully in order to prevent unexpected behavior such as deadlocks by interactions between agents. Thus we want a tool to analyze behavior of agents in the multi-agent model, where the tool should satisfy the following four requirements:

1. *Broadcast communication can be described.* A broadcasted event is received by *all* agents which require the event.
2. *A broadcaster of an event can know the number of receivers of the event after broadcasting.* Though a broadcaster does not specify the number of receivers of the event when broadcasting, it sometimes wants to know the number after broadcasting.
3. *Agents can dynamically change receivable events.* Agents may change their receivable events as reaction of events from the other agents.
4. *Multicast communication can be also described.* An agent sometimes wants to multicast an event to the specific number of receivers for synchronization.

We attempt to adopt a process algebra as a basic tool for analyzing agents in the multi-agent model. Many various process algebras have been proposed such as CCS[2], π-calculus[3], SCCS[4], and CBS[7, 8]. The above requirements 1 and 3 are satisfied by CBS, and the requirement 4 is satisfied by SCCS. However it is difficult for existing process algebras to satisfy the requirement 2.

If users can attach and detach agents in a system, then the number of receivers of an event can not be fixed. Furthermore one agent can change its receivable events. Therefore it is important to dynamically count the number of receivers of an event. For example, we want to describe the following behavior:

– An agent P_1 broadcasts an event st, and starts (triggers) the other agents $Q_{1,\cdots,n}$ which receive st, where P_1 must wait to continue its task until *all* the triggered agents $Q_{1,\cdots,n}$ finish their tasks.

In this case P_1 must know the number n of the triggered agents.

In this paper we propose a process algebra named *CCB* (a Calculus of Countable Broadcasting Systems). The above four requirements are satisfied by CCB. We define an observational congruence relation in CCB, and give a sound and complete axiom system for the congruence relation of finite agents.

The outline of this paper is as follows: In Section 2 we introduce process algebras, and point out some problems of existing process algebras for broadcasting systems. In Section 3 we informally introduce CCB. In Section 4 the syntax and the semantics of Core-CCB which is a base of CCB are defined. In Section 5 CCB is defined based on Core-CCB by translation. In Section 6 an observational congruence relation in Core-CCB is defined, then an axiom system is given. In Section 7 we discuss how to describe broadcast communication in the other process algebras. In Section 8 we conclude this paper.

2 Process algebras

Process algebras are well known as mathematical tools to describe and analyze concurrent and communicating systems. Behavior of agents is described as *(agent) expressions* in a process algebra, then equality of behavior of two agents is proven by rewriting their expressions according to algebraic laws in the process algebra. In general, messages are carried by *events* with *names*. When an event with a name and messages has been broadcasted, each agent decides by monitoring the name whether to read the messages in the event, or discard.

Many various process algebras have been proposed. In the rest of this section, we classify them into three groups by their communication styles, and point out some problems of existing process algebras for analyzing broadcasting systems.

2.1 Point-to-point communication

When an agent sends an event, only one agent can receive the event in CCS[2], π-calculus[3], and so on. In order to simulate broadcast communication, an event must be sent the same times as the number of agents which require the event, but it is very difficult to dynamically know the number. The following example shows that an agent $C(i)$ attempts to count the number of agents which can receive an event a but fails it, using CCS.

$$P_1 \overset{\text{def}}{=} a.\mathbf{0}, \quad P_2 \overset{\text{def}}{=} b.\mathbf{0}, \quad P_3 \overset{\text{def}}{=} a.\mathbf{0}, \quad C(i) \overset{\text{def}}{=} \overline{a}.C(i+1) + \overline{out}(i).\mathbf{0}$$
$$SYS \overset{\text{def}}{=} (C(0)|P_1|P_2|P_3)\backslash\{a,b\}$$

The agent $C(i)$ increases the local variable i by 1, when it can send the event a (in other words, an agent can receive the event a). The agent $C(i)$ also sends out an event *out* with the value of the variable i as a message. At the initial state, two agents P_1 and P_3 can receive the event a. Therefore it is expected that the value of i sent out is 2, but it can not be predicted at all which value 0, 1, or 2 is sent out, as shown the following equation[1].

$$SYS \sim \overline{out}(0).\mathbf{0} + \tau.(\overline{out}(1).\mathbf{0} + \tau.\overline{out}(2).\mathbf{0})$$

The problem is caused by possibility that $C(i)$ sends the event *out* before sending the event a to both the agents P_1 and P_3. This problem is not solved even though the event a has higher priority[11] than the event *out*. Because SYS falls into an infinite loop if P_1 is defined as $P_1 \overset{\text{def}}{=} a.P_1$.

[1] $\sim$ is strong equivalence in CCS[2].

2.2 Multicast communication

When an agent multicasts an event to n agents, just n agents can receive the event in SCCS[4], Meije[5], and so on. For example, in SCCS multicast communication is described as follows[2]:

$$P_0 \times P_1 \times P_2 \xrightarrow{1} P_0' \times P_1' \times P_2'$$

where each component is defined as follows:

$$P_0 \stackrel{\mathrm{def}}{=} a^{-2}\!:\!P_0' \qquad P_1 \stackrel{\mathrm{def}}{=} a^{1}\!:\!P_1' \qquad P_2 \stackrel{\mathrm{def}}{=} a^{1}\!:\!P_2'$$

In this example an agent P_0 multicasts an event a to explicitly *two* agents[3]. Therefore if a new agent which require the event a is attached to this example, then P_0 must be slightly modified as $P_0 \stackrel{\mathrm{def}}{=} a^{-3}\!:\!P_0'$.

In order to simulate broadcast communication, a multicaster of an event must count the number of receivers of the event, but it is very difficult by the same reason as shown in Subsection 2.1.

2.3 Broadcast communication

When an agent broadcasts an event, all agents which require it can receive it in CSP[6], CBS[7], and so on. Especially CBS has been developed for broadcasting systems. For example, in CBS broadcasting is described as follows:

$$(P_0|P_1|P_2|P_3)\backslash\{a\} \xrightarrow{\tau} (P_0'|P_1'|P_2|P_3')\backslash\{a\}$$

where each component is defined as follows:

$$P_0 \stackrel{\mathrm{def}}{=} a!.P_0', \quad P_1 \stackrel{\mathrm{def}}{=} a?.P_1' + b?.P_1'', \quad P_2 \stackrel{\mathrm{def}}{=} b?.P_2' + c?.P_2'', \quad P_3 \stackrel{\mathrm{def}}{=} a?.P_3'$$

Attributes ! and ? of events means transmitting and receiving, respectively. P_0 broadcasts an event a, then both P_1 and P_3 receive it. P_2 does not receive the event a, because it has no transitions like $\xrightarrow{a?}$, thus it dose not require a. Unfortunately, CBS is not interested in the number of receivers of an event.

3 Introduction of CCB

We propose a process algebra *CCB* which is an extension of CCS. The four requirements mentioned in Section 1 are satisfied by CCB. The requirement 2 is the most important, because the other requirements are satisfied also by the other process algebras.

An event in CCB has a form $a\theta\langle x\rangle(y)$, where a is a name, θ is an attribute, x is the number of receivers of this event (We call x the *received number*), and y means a message passed by this event. We often omit a received number $\langle 1\rangle$ and an empty message ( ). CCB has four attributes $\{!, ?, !!, ??\}$ explained as follows:

[2] : and $\times$ are a prefix and a composition combinators in SCCS, respectively.
[3] -2 in a^{-2} of P_0 means sending to two agents.

1. $a!\langle n\rangle(m)$ is used for multicasting. n is a non-negative integer (including zero). m is a message carried by this multicasting. Especially if n is zero then $a!\langle 0\rangle(m)$ is called a *silent event*. The silent event can be considered as an invisible and uncontrolled event like τ in CCS. In observational analysis, the silent event should be ignored as far as possible.
2. $a?\langle n\rangle(y)$ is used for receiving a multicasted event with the same event name a. n is a positive integer (not including zero). y is a variable bound to a message carried by the multicasted event.
3. $a!!\langle x\rangle(m)$ is used for broadcasting. x is a variable bound to the number of receivers of this event. m is a message carried by this broadcasting.
4. $a??\langle n\rangle(y)$ is used for receiving a broadcasted event with the same event name a. n is a positive integer. y is a variable bound to a message.

For example, in CCB broadcasting is described as follows:

$$(P_0|P_1|P_2|P_3)\backslash\{a\} \xrightarrow{a!\langle 0\rangle} (P_0'(2)|P_1'|P_2|P_3')\backslash\{a\}$$

where each component is defined as follows:

$$P_0 \overset{\text{def}}{=} a!!\langle x\rangle.P_0'(x),\quad P_1 \overset{\text{def}}{=} a??.P_1' + b??.P_1'',\quad P_2 \overset{\text{def}}{=} b??.P_2' + c??.P_2'',\quad P_3 \overset{\text{def}}{=} a??.P_3'$$

P_0 broadcasts an event a, then both P_1 and P_3 receive it. x is bound to the number 2 of the receivers as shown in $P_0'(2)$ after broadcasting.

4 The definition of Core-CCB

In this section, we define the syntax and the semantics of *Core-CCB* which is the base of CCB. Core-CCB is a process algebra with no *value variables*[4]. The definition of CCB is given based on Core-CCB in Section 5.

4.1 The syntax of Core-CCB

First, we assume that an infinite set of names, $\mathcal{N} = a, b, c, \cdots$, is given. The set of event *Event* is defined by using $\mathcal{N}$ as follows:

$$\mathit{Event} = \{a\theta^n : a \in \mathcal{N}, \theta \in \mathcal{T}, n \in \mathcal{I}\} - \{a?^0, a??^0 : a \in \mathcal{N}\}$$

where $\mathcal{I}$ is a set of non-negative integer $\{0, 1, 2, \cdots\}$, and $\mathcal{T}$ ranged over by θ is a set of attributes $\{!, ?, !!, ??\}$. Especially $a!^0$ is a *silent event* for any a. n of $a\theta^n$ is the received number of this event, and must be a constant even for broadcasting ($\theta =!!$). We show how to describe broadcast communication of CCB by Core-CCB in Section 5.

Furthermore we introduce a set of *agent variables*, $\mathcal{X}$ ranged over by $X, Y, \cdots$, a set of *agent constants*, $\mathcal{K}$ ranged over by $A, B, \cdots$, and a set of renaming functions, $\mathcal{F}$ ranged over by $S : \mathcal{N} \to \mathcal{N}$.

We define the set $\mathcal{E}$ of Core-CCB expressions as follows:

[4] The value variables are used for received numbers and messages in events of CCB like x, y in $a\theta\langle x\rangle(y)$.

Definition 4.1 *The set of agent expressions, $\mathcal{E}$ ranged over by $E, F, \cdots$, is defined by the following BNF expression:*

$$E ::= X \mid A \mid \mathbf{0} \mid a\theta^n.E \mid E + E \mid E|E \mid E[S] \mid E\backslash L$$

where we take $X \in \mathcal{X}$, $A \in \mathcal{K}$, $a\theta^n \in Event$, $S \in \mathcal{F}$, and $L \subseteq \mathcal{N}$. ▮

An *agent* is an agent expression with no agent variables. The set of agents is denoted by $\mathcal{P}$ ranged over by $P, Q, \cdots$. An *agent constant* is an agent whose meaning is given by a defining equation. In fact, we assume that for every agent constant A there is a defining equation of the following form: $A \overset{\text{def}}{=} P$ $(P \in \mathcal{P})$, where we also assume that A is *weakly guarded*[2] in P. The weak guard is defined as follows:

Definition 4.2 *X is weakly guarded in E if each occurrence of X is within some subexpression of E of form $a\theta^n.F$.* ▮

Agent constants which are not weakly guarded make a calculus more complex, and the behavior is indefinite. Practically, we are interested in weakly guarded agent constants in CCB. (also in CBS.)

4.2 The semantics of Core-CCB

The semantics of Core-CCB is defined by the following labelled transition system like one of CCS:

$$(\mathcal{E}, Event, \{\xrightarrow{a\theta^n} : a\theta^n \in Event\})$$

For example, $E \xrightarrow{a\theta^n} E'$ $(E, E' \in \mathcal{E})$ indicates that the agent expression E may perform the event $a\theta^n$ and thereafter become the agent expression E'. The semantics for agent expressions $\mathcal{E}$ consists in the definition of each transition relation $\xrightarrow{a\theta^n}$ over $\mathcal{E}$.

Before defining the semantics, we define a function $mon(E)$ named *monitor function* which takes an agent expression and produces a subset of names.

Definition 4.3 *For each agent expression E, we inductively define* monitor function $mon : \mathcal{E} \to 2^{\mathcal{N}}$ *as follows:*

$$\begin{array}{r|l}
mon(\mathbf{0}) = \emptyset & mon(E[S]) = \{S(a) : a \in mon(E)\} \\
mon(a\theta^n.E) = \begin{cases} \{a\} & (\theta = ⁇) \\ \emptyset & (otherwise) \end{cases} & mon(E\backslash L) = mon(E) - L \\
mon(E + F) = mon(E) \cup mon(F) & mon(A) = mon(P) \quad (A \overset{\text{def}}{=} P) \\
mon(E|F) = mon(E) \cup mon(F) & mon(X) = \emptyset
\end{array}$$

□

Since agent constants must be weakly guarded, $mon(E)$ can be effectively evaluated. Intuitively, if an agent expression E *now* requires receiving a broadcasted event, then the event name is included in $mon(E)$.

Then the semantics of Core-CCB is defined.

Definition 4.4 *The transition relation* $\xrightarrow{a\theta^n}$ *over agent expressions is the smallest relation satisfying the following inference rules. Each rule is to be read as follows: if the transition relations above the line are inferred and the side conditions are satisfied, then the transition relation below the line can be also inferred.*

$$\textbf{Event}\,\frac{}{a\theta^n.E \xrightarrow{a\theta^n} E}$$

$$\textbf{Choice}_\mathbf{1}\,\frac{E \xrightarrow{a\theta^n} E'}{E+F \xrightarrow{a\theta^n} E'}$$

$$\textbf{Choice}_\mathbf{2}\,\frac{F \xrightarrow{a\theta^n} F'}{E+F \xrightarrow{a\theta^n} F'}$$

$$\textbf{Ren}\,\frac{E \xrightarrow{a\theta^n} E'}{E[S] \xrightarrow{S(a)\theta^n} E'[S]}$$

$$\textbf{Res}_\mathbf{1}\,\frac{E \xrightarrow{a\theta^n} E'}{E\backslash L \xrightarrow{a\theta^n} E'\backslash L}\ (a \notin L)$$

$$\textbf{Res}_\mathbf{2}\,\frac{E \xrightarrow{a\theta^0} E'}{E\backslash L \xrightarrow{a!^0} E'\backslash L}\ \begin{pmatrix}\theta \in \{!, \text{‼}\}, \\ a \in L\end{pmatrix}$$

$$\textbf{Con}\,\frac{P \xrightarrow{a\theta^n} P'}{A \xrightarrow{a\theta^n} P'}\ (A \overset{\text{def}}{=} P)$$

$$\textbf{Para}_\mathbf{1}\,\frac{E \xrightarrow{a\theta^n} E'}{E|F \xrightarrow{a\theta^n} E'|F}\ \begin{pmatrix}\theta \in \{!, ?\} \text{ or} \\ a \notin mon(F)\end{pmatrix}$$

$$\textbf{Para}_\mathbf{2}\,\frac{F \xrightarrow{a\theta^n} F'}{E|F \xrightarrow{a\theta^n} E|F'}\ \begin{pmatrix}\theta \in \{!, ?\} \text{ or} \\ a \notin mon(E)\end{pmatrix}$$

$$\textbf{Para}_\mathbf{3}\,\frac{E \xrightarrow{a\theta^m} E' \qquad F \xrightarrow{a\phi^n} F'}{E|F \xrightarrow{a\theta^{(m-n)}} E'|F'}\ \begin{pmatrix}\theta \in \{!, \text{‼}\}, \\ \phi = @(\theta), \\ m \geq n\end{pmatrix}$$

$$\textbf{Para}_\mathbf{4}\,\frac{E \xrightarrow{a\phi^n} E' \qquad F \xrightarrow{a\theta^m} F'}{E|F \xrightarrow{a\theta^{(m-n)}} E'|F'}\ \begin{pmatrix}\theta \in \{!, \text{‼}\}, \\ \phi = @(\theta), \\ m \geq n\end{pmatrix}$$

$$\textbf{Para}_\mathbf{5}\,\frac{E \xrightarrow{a\theta^m} E' \qquad F \xrightarrow{a\theta^n} F'}{E|F \xrightarrow{a\theta^{(m+n)}} E'|F'}\ (\theta \in \{?, \text{⁇}\})$$

where @ *in the conditions is a function from* $\mathcal{T}$ *to* $\mathcal{T}$ *defined as follows:*

$$@(!) = ?, \quad @(?) = !, \quad @(\text{‼}) = \text{⁇}, \quad @(\text{⁇}) = \text{‼}$$ □

Then the following proposition for monitor function is proven.

Proposition 4.1 $E \overset{a\text{⁇}^n}{\not\longrightarrow}$ *for any* n **iff** $a \notin mon(E)$ □

We intuitively explain **Para$_\mathbf{i}$** rules.

1. By **Para$_\mathbf{1}$** E multicasts an event or receives a multicasted event without respect to F. While E broadcasts an events or receives a broadcasted event only if the event name is not included in $mon(F)$, thus F does not require the event. If F requires the event, then **Para$_\mathbf{3}$** or **Para$_\mathbf{5}$** msut be used instead of **Para$_\mathbf{1}$**. This restriction relates to the requirement 1 stated in Section 1. **Para$_\mathbf{2}$** is symmetric to **Para$_\mathbf{1}$**.
2. By **Para$_\mathbf{3}$** E multicasts or broadcasts an events and F receives the event, if the number of receivers within F is not greater than the received number specified by E. If the number $(m-n)$ in the conclusion of **Para$_\mathbf{3}$** is zero, then $E|F$ can no longer communicate with the other agents. This calculation of the received number relates to the requirement 2 stated in Section 1. **Para$_\mathbf{4}$** is symmetric to **Para$_\mathbf{3}$**.
3. By **Para$_\mathbf{5}$** to receive an event can be synchronized. We have no rules for synchronization of transmitting.

Then we explain an event of form $a!!^{0}$ which is not a silent event. In order to compare $a!^{0}$ with $a!!^{0}$, we show the following transitions:

$$\begin{array}{ll}(1) & ((a!^{2}.\mathbf{0}|a?^{1}.\mathbf{0})|a?^{1}.\mathbf{0})|a?^{1}.\mathbf{0} \xrightarrow{a!^{0}} ((\mathbf{0}|\mathbf{0})|\mathbf{0})|a?^{1}.\mathbf{0} \\ (2) & ((a!!^{2}.\mathbf{0}|a??^{1}.\mathbf{0})|a??^{1}.\mathbf{0})|a??^{1}.\mathbf{0} \not\xrightarrow{a!!^{0}}\end{array}$$

The transition (1) is inferred by $\mathbf{Para_3}$ and $\mathbf{Para_1}$. However $\mathbf{Para_1}$ can not be used for the transition (2), because $a \in mon(a??^{1}.\mathbf{0})$. The following transitions by broadcasting are possible:

$$\begin{array}{ll}(3) & ((a!!^{3}.\mathbf{0}|a??^{1}.\mathbf{0})|a??^{1}.\mathbf{0})|a??^{1}.\mathbf{0} \xrightarrow{a!!^{0}} ((\mathbf{0}|\mathbf{0})|\mathbf{0})|\mathbf{0} \\ (4) & ((a!!^{2}.\mathbf{0}|a??^{1}.\mathbf{0})|a??^{1}.\mathbf{0})\backslash\{a\}|a??^{1}.\mathbf{0} \xrightarrow{a!^{0}} ((\mathbf{0}|\mathbf{0})|\mathbf{0})\backslash\{a\}|a??^{1}.\mathbf{0}\end{array}$$

In the transition (3), the number specified by the broadcaster is equal to the number of receivers. In the transition (4), the scop of the event a is restricted and $a!!^{0}$ is changed to $a!^{0}$ by $\mathbf{Res_2}$.

5 The definition of CCB

In this section, we define the syntax and the semantics of CCB, and give a small example. The semantics of CCB is given by translation to Core-CCB.

5.1 The syntax of CCB

The syntax of CCB is defined as follows:

Definition 5.1 *The set of agent expressions, $\mathcal{E}^{+}$ ranged over by $E, F, \cdots$, is defined by the following BNF expression:*

$$\begin{aligned} E ::= X \mid A(e_1,\cdots,e_n) \mid \mathbf{0} \mid a[e_1]!\langle c_0\rangle(e_2).E \mid a[e_1]!!\langle x\rangle(e_2).E \mid a[e]?\langle c_1\rangle(x).E \\ \mid a[e]??\langle c_1\rangle(x).E \mid E+E \mid E|E \mid E[S] \mid E\backslash L \mid [\![b]\!]E\end{aligned}$$

where we take $X \in \mathcal{X}$, $A \in \mathcal{K}$, $a \in \mathcal{N}$, $S \in \mathcal{F}$, and $L \subseteq \mathcal{N}$. x is a bound value variable, b is a boolean expression, and e_i is an expression. c_i is also an expression, where the range of c_0 must be non-negative integer and the range of c_1 must be positive integer. All value variables in expressions b, e_i, and c_i must be bound by their occurrences on their left. □

The unconventional combinator $[\![b]\!]$ is a conditional combinator. It intuitively means that if b is true then $[\![b]\!]E$ behaves like E, otherwise it stops.

Each agent constant A with arity n has a defining equation $A(x_1,\cdots,x_n) \overset{\text{def}}{=} E$, where the right-hand side E may contain no agent variables and no free value variables except $x_1,\cdots,x_n$.

An event name $a[e]$ consists of a base event name a and a postfix $[e]$. We sometimes omit a postfix $[0]$. Intuitively an event $a[x]!$ will be received by $a[e]?$ for some e. In order to avoid ambiguous communication, x must be bound before

$a[x]!$ is multicasted. For example, the following agent INC is a procedure which takes a value y and return a value $y+1$.

$$INC \overset{\text{def}}{=} inc?(x).((in[x]?(y).out[x]!(y+1).\mathbf{0})|INC)$$

INC is duplicated just after called by a caller in order to prepare for the next caller. INC binds x to a process-id of a caller and certainly return $y+1$ to *the* caller through $out[x]$. For example a caller may be defined as follows:

$$CALLER23 \overset{\text{def}}{=} inc!(23).in[23]!(7).out[23]?(z).TASK23(z)$$

$TASK23(z)$ depends on z, and z will be bound to 8. This strategy has been used in a parallel programming language $\mathcal{M}$[2] defined in CCS.

5.2 The semantics of CCB

The semantics of CCB is given by translation. Each expression in CCB without free value variables is translated into an expression in Core-CCB by a function $\mathcal{B} : \mathcal{E}^+ \to \mathcal{E}$. We introduce a operator $\ddot{}$. It takes an expression and produce a fixed value which the expression evaluates. For example, if $e = 2+3$ then $\ddot{e} = 5$.

Definition 5.2 *The function $\mathcal{B}$ from $\mathcal{E}^+$ to $\mathcal{E}$ is defined as follows*[5]*:*

$$\begin{array}{rcl|rcl}
\mathcal{B}(X) &=& X & \mathcal{B}(E+F) &=& \mathcal{B}(E)+\mathcal{B}(F) \\
\mathcal{B}(A(e_1,\cdots,e_n)) &=& A_{e_1,\cdots,e_n} & \mathcal{B}(E|F) &=& \mathcal{B}(E)|\mathcal{B}(F) \\
\mathcal{B}(\mathbf{0}) &=& \mathbf{0} & \mathcal{B}(E[S]) &=& \mathcal{B}(E)[S'] \\
\mathcal{B}(a[e_1]!\langle c\rangle(e_2).E) &=& a_{\ddot{e}_1,\ddot{e}_2}!^{\ddot{c}}.\mathcal{B}(E) & \mathcal{B}(E\backslash L) &=& \mathcal{B}(E)\backslash L' \\
\mathcal{B}(a[e_1]!!\langle x\rangle(e_2).E) &=& \sum_{n\in\mathcal{I}} a_{\ddot{e}_1,\ddot{e}_2}!!^{n}.\mathcal{B}(E\{n/x\}) & & & \\
\mathcal{B}(a[e]?\langle c\rangle(x).E) &=& \sum_{v\in V} a_{\ddot{e},v}?^{\ddot{c}}.\mathcal{B}(E\{v/x\}) & \mathcal{B}([\![b]\!]E) &=& \begin{cases} \mathcal{B}(E) & \textit{if } b\textit{=true} \\ \mathbf{0} & \textit{otherwise} \end{cases} \\
\mathcal{B}(a[e]??\langle c\rangle(x).E) &=& \sum_{v\in V} a_{\ddot{e},v}??^{\ddot{c}}.\mathcal{B}(E\{v/x\}) & & &
\end{array}$$

where we assume that all values belong to the fixed value set V, and

$$S'(a_{v_1,v_2}) = S(a)_{v_1,v_2}, \qquad L' = \{a_{v_1,v_2} : a \in L, (v_1,v_2) \in V^2\}$$

where $\{v/x\}$ is a substitution of a value v into a variable x. Furthermore the single defining equation $A(x_1,\cdots,x_n) \overset{\text{def}}{=} E$ of an agent constant is translated into the indexed set of defining equations

$$\{A_{v_1,\cdots,v_n} \overset{\text{def}}{=} \mathcal{B}(E\{v_1/x_1,\cdots,v_n/x_n\}) : (v_1,\cdots,v_n) \in V^n\}. \qquad \square$$

We now give a simple example of translating CCB into Core-CCB. The following expression is described in CCB:

$$CHECK \overset{\text{def}}{=} a!!\langle x\rangle.([\![x=0]\!]zero!\langle 1\rangle.\mathbf{0} + [\![x \neq 0]\!]nonzero!\langle 1\rangle.\mathbf{0})$$

This agent $CHECK$ broadcasts an event $a!!\langle x\rangle$, and check the number of agents which receive $a!!\langle x\rangle$. Thereafter if the number is zero then $CHECK$ multicasts

[5] It $I \neq \emptyset$, then $\sum_{i\in I} P_i$ is a syntactic shorthand of $P_1 + P_2 + \cdots$, otherwise it is a $\mathbf{0}$.

an event $zero!\langle 1\rangle$, otherwise multicasts an event $nonzero!\langle 1\rangle$. This expression in CCB is translated into an expression in Core-CCB by $\mathcal{B}$ as follows:

$$\begin{aligned}
\mathcal{B}(CHECK) &= \mathcal{B}(a!!\langle x\rangle.([\![x=0]\!]zero!\langle 1\rangle.\mathbf{0} + [\![x\neq 0]\!]nonzero!\langle 1\rangle.\mathbf{0}))\\
&= \textstyle\sum_{n\in\mathcal{I}} a!!^{n}.\mathcal{B}([\![n=0]\!]zero!\langle 1\rangle.\mathbf{0} + [\![n\neq 0]\!]nonzero!\langle 1\rangle.\mathbf{0})\\
&= \textstyle\sum_{n\in\mathcal{I}} a!!^{n}.(\mathcal{B}([\![n=0]\!]zero!\langle 1\rangle.\mathbf{0}) + \mathcal{B}([\![n\neq 0]\!]nonzero!\langle 1\rangle.\mathbf{0}))\\
&= a!!^{0}.(\mathcal{B}(zero!\langle 1\rangle.\mathbf{0}) + \mathbf{0}) + \textstyle\sum_{n\in\mathcal{I}_1} a!!^{n}.(\mathbf{0} + \mathcal{B}(nonzero!\langle 1\rangle.\mathbf{0}))\\
&= a!!^{0}.(zero!^{1}.\mathbf{0} + \mathbf{0}) + \textstyle\sum_{n\in\mathcal{I}_1} a!!^{n}.(\mathbf{0} + nonzero!^{1}.\mathbf{0})
\end{aligned}$$

where $\mathcal{I}$ is a set of non-negative integer and $\mathcal{I}_1$ is a set of positive integer.

We show another example of translating CCB into Core-CCB by using INC and $CALLER23$ defined in Subsection 5.1.

$$\begin{aligned}
\mathcal{B}(INC) &= \mathcal{B}(inc?(x).((in[x]?(y).out[x]!(y+1).\mathbf{0})|INC))\\
&= \textstyle\sum_{v\in\mathcal{V}} inc_v?.((\sum_{u\in\mathcal{V}} in_{\ddot{v},u}?.out_{\ddot{v},\ddot{u+1}}!.\mathbf{0})|\mathcal{B}(INC))\\
&= inc_1?.((in_{1,1}?.out_{1,2}!.\mathbf{0} + in_{1,2}?.out_{1,3}!.\mathbf{0} + \cdots)|\mathcal{B}(INC))+\\
&\quad\; inc_2?.((in_{2,1}?.out_{2,2}!.\mathbf{0} + in_{2,2}?.out_{2,3}!.\mathbf{0} + \cdots)|\mathcal{B}(INC)) + \cdots
\end{aligned}$$

$$\begin{aligned}
\mathcal{B}(CALLER23) &= \mathcal{B}(inc!(23).in[23]!(7).out[23]?(z).TASK23(z))\\
&= inc_{23}!.in_{23,7}!.(\textstyle\sum_{v\in\mathcal{V}} out_{23,v}?.TASK23_v)
\end{aligned}$$

In this case, $TASK23_8$ will be choiced through an event $out_{23,8}?$.

5.3 An example in CCB

We show an example of a *distributed used-car information system.* This system consists of many agents which have information about used-cars and give the information to car-dealers, described as follows:

$$CARINFO \overset{\text{def}}{=} (INFO1|INFO2|\cdots|DEALER1|DEALER2|\cdots)$$

For example, information agents about PORSCHE-928 are defined as follows:

$$\begin{aligned}
INFO45 &\overset{\text{def}}{=} porsche928??(x).(info[x]!(\text{‘1985.RED.52940km.S34’})|INFO45)\\
INFO81 &\overset{\text{def}}{=} porsche928??(x).(info[x]!(\text{‘1991.BLUE.21380km.S16’})|INFO81)
\end{aligned}$$

A postfix $[x]$ is used for determining a destination of information as used in INC. A car-dealer wants to gather all information about a car, though he does not know the number of agents with the car information. Therefore he broadcasts an event for the car. For example, a car-dealer may be defined as follows:

$$\begin{aligned}
DEALER7 &\overset{\text{def}}{=} porsche928!!\langle x\rangle(7).LP(x,\mathtt{nil})\\
LP(n,list) &\overset{\text{def}}{=} [\![n\neq 0]\!]info[7]?(y).LP(n-1, y::list) + [\![n=0]\!]PRINT(list)
\end{aligned}$$

$DEALER7$ querys about PORSCHE-928 through an event $porsche928$, then gets the number of agents with information about PORSCHE-928. Two variables n and $list$ of $LP(n,list)$ are initially bound to the number of the information agents and `nil`, respectively. When $LP(n,list)$ receives an event $info[7]$, y is bound to car information carried by $info[7]$. Then y is concatenated into $idlist$ and n is decreased by 1. If n is zero, then $list$ is printed out by an agent $PRINT$, because $LP(n,list)$ has already gathered all information about PORSCHE-928.

6 Equivalence and congruence relations

We want observational congruence relations in Core-CCB like in CCS. It is important that *observation congruence* $=^{[2]}$ defined in CCS is *not* a congruence relation in Core-CCB as shown in the following example:

$$P_1 = P_2, \qquad P_1|Q \neq P_2|Q$$

where each component is defined as follows:

$$P_1 \overset{\text{def}}{=} a??^1.b!^0.P_1, \quad P_2 \overset{\text{def}}{=} a??^1.P_2, \quad Q \overset{\text{def}}{=} a!!^0.out0!^1.Q + a!!^1.out1!^1.Q$$

P_1 and P_2 are observation congruent because the silent event $b!^0$ is ignored, but $P_1|Q$ and $P_2|Q$ are not observation congruent because P_1 sometimes fails to receive the event a broadcasted by Q while P_2 can always receive it. Therefore we define a slightly stronger observational congruence relation in this paper than observation congruence. We prepare several notations for the definition.

The set *Event**, ranged over by $s, t, \cdots$, is a set of event sequences including an empty sequence ε, and if $t = a_1(\theta_1)^{n_1} \cdots a_k(\theta_k)^{n_k} \in \mathit{Event}^*$, then we write $E \xrightarrow{t} E'$ if $E \xrightarrow{a_1(\theta_1)^{n_1}} \cdots \xrightarrow{a_k(\theta_k)^{n_k}} E'$. Especially, if $E \xrightarrow{\varepsilon} E'$ then $E' \equiv E$. ($\equiv$ means syntactic identity.) About transitions by silent events, we sometimes write $E \xrightarrow{\tau} E'$ if $E \xrightarrow{a!^0} E'$ for *some* event name a. We now define a new transition relation as follows:

Definition 6.1 *If* $t = a_1(\theta_1)^{n_1} \cdots a_k(\theta_k)^{n_k} \in \mathit{Event}^*$, *then* $E \overset{t}{\Longrightarrow} E'$ *if*

$$E(\xrightarrow{\tau})^* \xrightarrow{a_1(\theta_1)^{n_1}} (\xrightarrow{\tau})^* \cdots (\xrightarrow{\tau})^* \xrightarrow{a_k(\theta_k)^{n_k}} (\xrightarrow{\tau})^* E'$$

where $(\xrightarrow{\tau})^*$ *means zero or more transitions by silent events.* □

Then we define an observational bisimulation relation in Core-CCB like *weak bisimulation* in CCS.

Definition 6.2 *A binary relation* $\mathcal{S} \subseteq \mathcal{P} \times \mathcal{P}$ *over agents is a* weak monitor bisimulation *if* $(P, Q) \in \mathcal{S}$ *implies, for all* $a\theta^n \in \mathit{Event}$, *that*

(i) *whenever* $P \xrightarrow{a\theta^n} P'$ *then for some* $Q', Q \overset{\widehat{a\theta^n}}{\Longrightarrow} Q'$ *and* $(P', Q') \in \mathcal{S}$,

(ii) *whenever* $Q \xrightarrow{a\theta^n} Q'$ *then for some* $P', P \overset{\widehat{a\theta^n}}{\Longrightarrow} P'$ *and* $(P', Q') \in \mathcal{S}$,

(iii) *whenever* $a \notin \mathit{mon}(P)$ *then, for some* Q', Q'',
$$Q \Rightarrow Q' \Rightarrow Q'',\ a \notin \mathit{mon}(Q'),\ \text{and } (P, Q'') \in \mathcal{S},$$

(iv) *whenever* $a \notin \mathit{mon}(Q)$ *then, for some* P', P'',
$$P \Rightarrow P' \Rightarrow P'',\ a \notin \mathit{mon}(P'),\ \text{and } (P'', Q) \in \mathcal{S}.$$

where $\widehat{t}$ *is the event sequence gained by deleting all occurrences of silent events from* t. □

Weak monitor bisimulations are obtained from weak bisimulations by adding the conditions (iii) and (iv). A relation is defined by using weak monitor bisimulations.

Definition 6.3 *P and Q are* weakly monitor equivalent*, written $P \approx_m Q$, if $(P, Q) \in \mathcal{S}$ for some weak monitor bisimulation $\mathcal{S}$.* □

Weak monitor equivalence is the *largest* relation preserved by a parallel combinator $|$ and included in observation equivalence $\approx$ defined in CCS, as shown in the following proposition.

Proposition 6.1 $P_1 \approx_m P_2$ **iff** *for any Q, $P_1|Q \approx P_2|Q$* □

We show an interesting equation in the following proposition.

Proposition 6.2 $P_1 + P_2 + a!^0.(P_1 + a!^0.(Q + P_2)) \approx_m P_1 + a!^0.(Q + P_2)$ □

This equation shows a property of weak monitor equivalence, because this equation does not held for *strong monitor equivalence* defined in Definition 6.5.

We notice a special case $P \equiv P' \equiv P''$ and $Q \equiv Q' \equiv Q''$ in Definition 6.2, and define strong monitor bisimulations as follows:

Definition 6.4 *A binary relation $\mathcal{S} \subseteq \mathcal{P} \times \mathcal{P}$ over agents is a* strong monitor bisimulation *if $(P, Q) \in \mathcal{S}$ implies, for all $a\theta^n \in Event$, that*

$$\begin{array}{l}
(i)\ \textit{whenever } P \xrightarrow{a\theta^n} P' \textit{ then for some } Q', Q \overset{\widehat{a\theta^n}}{\Longrightarrow} Q' \textit{ and } (P', Q') \in \mathcal{S},\\
(ii)\ \textit{whenever } Q \xrightarrow{a\theta^n} Q' \textit{ then for some } P', P \overset{\widehat{a\theta^n}}{\Longrightarrow} P' \textit{ and } (P', Q') \in \mathcal{S},\\
(iii)\ mon(P) = mon(Q).
\end{array}$$

□

A relation is defined by using strong monitor bisimulations like weak monitor equivalence.

Definition 6.5 *P and Q are* strongly monitor equivalent*, written $P \simeq_m Q$, if $(P, Q) \in \mathcal{S}$ for some strong monitor bisimulation $\mathcal{S}$.* □

It is helpful to consider strong monitor equivalence before weak monitor equivalence, because the definition of strong monitor equivalence is simpler than one of weak monitor equivalence and $P \simeq_m Q$ implies $P \approx_m Q$. In the rest of this paper we consider only strong monitor equivalence.

Strong monitor equivalence is an equivalence relation but is not a congruence relation because it is not preserved by a choice combinator $+$. Therefore we define a congruence relation from strong monitor equivalence by adding the weakest conditions. Before the definition of the congruence relation, an equivalence relation $\doteq$ over events is introduced for ignoring difference between many silent events.

Definition 6.6 Event equivalence $\doteq$ *is a binary relation over events defined as the following set:* $\{(a!^0, b!^0) : a, b \in \mathcal{N}\} \cup \{(a\theta^n, a\theta^n) : a\theta^n \in Event\}$. □

Then we define *strong monitor congruence* as follows:

Definition 6.7 *P and Q are strongly monitor congruent, written $P \cong_m Q$, if for all $a\theta^n \in Event$*

(i) *whenever $P \xrightarrow{a\theta^n} P'$ then for some $Q', b, a\theta^n \doteq b\theta^n, Q \overset{b\theta^n}{\Longrightarrow} Q'$, $P' \simeq_m Q'$,*
(ii) *whenever $Q \xrightarrow{a\theta^n} Q'$ then for some $P', b, a\theta^n \doteq b\theta^n, P \overset{b\theta^n}{\Longrightarrow} P'$, $P' \simeq_m Q'$,*
(iii) *$mon(P) = mon(Q)$.* □

We now extend strong monitor congruence over agents to agent expressions.

Definition 6.8 *Let agent expressions E and F contain agent variables $X_1, \cdots, X_n$ (written by $\tilde{X}$) at most. Then $E \cong_m F$ if, for all indexed sets $\tilde{P}$ of agents, $E\{\tilde{P}/\tilde{X}\} \cong_m F\{\tilde{P}/\tilde{X}\}$.*[6] □

Most of propositions for observation congruence are also held for strong monitor congruence with slight modifications. Several propositions are shown below.

Proposition 6.3 *Strong monitor congruence is a congruence relation.* □

Proposition 6.4 *$P \simeq_m Q$* **iff**
$((P \cong_m Q)$ *or* $(P \cong_m a!^0.Q + Q)$ *or* $(a!^0.P + P \cong_m Q))$ □

Proposition 6.4 is used for strengthening $\simeq_m$ to $\cong_m$, for example, in the proof of Theorem 6.6. We show two notions defined in [2].

Definition 6.9 *X is sequential in E if every subexpression of E which contains X, apart from X itself, is of the form $a\theta^n.F$ or $\sum \tilde{F}$.* □

Definition 6.10 *X is guarded in E if each occurrence of X is within some subexpression of E form $a\theta^n.F$ such that $a\theta^n$ is not a silent event.* □

Then we give a proposition which guarantees existence of unique solution P such that $P \cong_m E\{P/X\}$ under a certain condition on the agent expression E. The solution is naturally the agent A defined by $A \overset{\text{def}}{=} E\{A/X\}$. Thus this proposition is useful for proving whether two agents with recursive definition are strongly monitor congruent or not.

Proposition 6.5 *Let agent expressions $\tilde{E}$ contain the variables $\tilde{X}$ at most, and let $\tilde{X}$ are guarded and sequential in $\tilde{E}$. Then if $\tilde{P} \cong_m \tilde{E}\{\tilde{P}/\tilde{X}\}$ and $\tilde{Q} \cong_m \tilde{E}\{\tilde{Q}/\tilde{X}\}$ then $\tilde{P} \cong_m \tilde{Q}$.* □

In the rest of this section, we give an axiom system for finite agents which contains no agent constants (no recursions). We define an axiom system $\mathcal{A}$.

Definition 6.11 *We write $\mathcal{A} \vdash P = Q$ if the equality of two agents P and Q can be proven by equational reasoning from an axiom system $\mathcal{A}$, where the axiom system $\mathcal{A}$ consists of the following equations:*

M1 $P_1 + P_2 = P_2 + P_1$
M2 $(P_1 + P_2) + P_3 = P_1 + (P_2 + P_3)$
M3 $P = P + P$
M4 $P = P + \mathbf{0}$

[6] $\{\tilde{P}/\tilde{X}\}$ means a simultaneous substitution of an agent P_i into an agent variable X_i.

E1 $P|Q \equiv (\sum_{(i\in I_1)} a_i(\theta_i)^{m_i}.P_i)|(\sum_{(i\in I_2)} b_i(\phi_i)^{n_i}.Q_i)$
$= \sum_{(i\in I_1)}\{a_i(\theta_i)^{m_i}.(P_i|Q) : \theta_i \in \{!,?\}\ \mathit{or}\ a_i \notin \mathit{mon}(Q)\}$
$+ \sum_{(i\in I_2)}\{b_i(\phi_i)^{n_i}.(P|Q_i) : \phi_i \in \{!,?\}\ \mathit{or}\ b_i \notin \mathit{mon}(P)\}$
$+ \sum_{(i\in I_1)}\sum_{(j\in I_2)}\{a_i(\theta_i)^{(m_i-n_j)}.(P_i|Q_j) : a_i{=}b_j, \theta_i \in \{!,‼\}, \phi_j{=}@(\theta_i), m_i \geq n_j\}$
$+ \sum_{(i\in I_1)}\sum_{(j\in I_2)}\{b_j(\phi_j)^{(n_j-m_i)}.(P_i|Q_j) : a_i{=}b_j, \phi_j \in \{!,‼\}, \theta_i{=}@(\phi_j), n_j \geq m_i\}$
$+ \sum_{(i\in I_1)}\sum_{(j\in I_2)}\{a_i(\theta_i)^{(m_i+n_j)}.(P_i|Q_j) : a_i = b_j, \theta_i = \phi_j \in \{?,⁇\}\}$
E2 $(\sum_{(i\in I)} a_i(\theta_i)^{n_i}.P_i)\backslash L = \sum_{(i\in I)}\{a_i(\theta_i)^{n_i}.(P_i\backslash L) : a_i \notin L\}$
$\qquad + \sum_{(i\in I)}\{a_i!^0.(P_i\backslash L) : a_i \in L, \theta_i \in \{!,‼\}, n_i = 0\}$
E3 $(\sum_{(i\in I)} a_i(\theta_i)^{n_i}.P_i)[S] = \sum_{(i\in I)} S(a_i)(\theta_i)^{n_i}.(P_i[S])$
T1 $a\theta^n.(b!^0.P + P) = a\theta^n.P$
T2 $P + R + a!^0.(P + Q) = R + a!^0.(P + Q)$ $\qquad$ *if* $\mathit{mon}(P) \subseteq \mathit{mon}(R)$
T3 $a\theta^n.(P + b!^0.Q) + a\theta^n.Q = a\theta^n.(P + b!^0.Q)$
T4 $a!^0.P = b!^0.P$ $\square$

M1-4, **E1-3**, and **T1-4** correspond to monoid laws, expansion laws, and τ laws in CCS, respectively. Finally we show a theorem which means that $\mathcal{A}$ is a sound and complete axiom system for strong monitor congruence of finite agents.

Theorem 6.6 *Let P and Q are finite. Then $P \cong_m Q$* **iff** *$\mathcal{A} \vdash P = Q$.* $\square$

7 Related work

CBS has already been proposed for broadcasting systems. To introduce broadcast communication into process algebras, negative transitions by receive-events are important in general. In CBS transitions by special events (actions) named *discards* are used instead of the negative transitions. The advantage of using the discards instead of the negative transitions is to be able to use a synchronization algebra[12].

We adopt *monitor function* instead of the negative transitions, because we want to use monitor function as conditions in equations. For example, monitor function is used in equation **T2** in the axiom system $\mathcal{A}$. Monitor function causes the same effect as the discards. The important difference between CCB and CBS is the received number. In CCB a broadcaster of an event can know the number of receivers of the event after broadcasting.

Process algebras with notion of time have been proposed, for example TCCS[10]. In TCCS broadcast communication may be simulated by using a lot of loops with timeout. Thus for *development* of programming languages with broadcast communication, such timed process algebras are very useful. On the other hand, we want to *use* such languages which have been already developed, and to *simply* analyze programs in the languages. CCB is appropriate for this purpose.

We show [9] as another study of broadcast communication by existing process algebras. In [9] a translation from CBS to SCCS is presented, and the relation between CBS and SCCS is clarified. We want to also clarify relation between CCB and SCCS.

8 Conclusion and future works

In order to flexibly connect agents, one of the authors has developed a mechanism named VIABUS which has a software bus architecture with *broadcast communication.* Then π-calculus has been introduced into VIABUS[13] as a common communicating language to control agents written in different languages, but there are problems about describing broadcast communication in π-calculus.

In this paper we have proposed a process algebra CCB. The most important property of CCB is that a broadcaster of an event can know the number of receivers of the event after broadcasting. Then we have defined an observational congruence relation named *strong monitor congruence*, and have given a sound and complete axiom system for strong monitor congruence of finite agents. We consider *weak monitor equivalence* as the next subject.

In the future we shall extend CCB with a notion of space distance.

Acknowledgement

The authors wish to express our gratitude to Dr. Kimihiro Ohta, Director of Computer Science Division, ETL. They also thank Dr. Yoshiki Kinoshita in Computer Language Section and all colleagues in Information Base Section for their helpful discussions.

References

1. J.Coutaz, "Architecture Model for Interactive Software: Failures and Trends", Proc. of the IFIP TC 2/WG 2.7 Working Conference on Engineering for Human-Computer Interaction, pp.137 - 153, 1989.
2. R.Milner, "Communication and Concurrency", Prentice-Hall, 1989.
3. R.Milner, J.Parrow and D.Walker, "A Calculus of Mobile Processes, I and II", Information and Computation, 100, pp.1 - 40 and pp.41 - 77, 1992.
4. R.Milner, "Calculi for Synchrony and Asynchrony", Journal of Theoretical Computer Science, Vol.25, pp.267 - 310, 1983.
5. R.de Simone, "Higher-level Synchronizing Devices in Meije-SCCS", Journal of Theoretical Computer Science, Vol.37, pp.245 - 267, 1985.
6. C.A.R.Hoare,"Communicating Sequential Processes", Prentice-Hall, 1985.
7. K.V.S.Prasad, "A Calculus of Broadcasting Systems", TAPSOFT'91, Vol.1:CAAP, LNCS 493, Springer-Verlag, pp.338 - 358, 1991
8. K.V.S.Prasad, "A Calculus of Value Broadcasts", PARLE'93, LNCS 694, Springer-Verlag, pp.391 - 402, 1993
9. Uno Holmer, "Interpreting Broadcast Communication in SCCS", CONCUR'93, LNCS 715, Springer-Verlag, pp.188 - 201, 1993
10. F.Moller and C.Tofts,"An overview of TCCS", Proc. of EUROMICRO'92, Athens, June 1992.
11. L.Aceto, B.Bloom, and F.Vaandrager "Turning SOS Rules into Equations", Proc. 7th Annual IEEE Symposium on Logic in Computer Science, pp.113 - 124, 1988.
12. G.Winskel, "Synchronization trees", Journal of Theoretical Computer Science, Vol.34, pp.33 - 82, 1984.
13. Y.Sato and K.Ohmaki "A Flexible Inter-Agent Connection Scheme for Interactive Software", IPSJ Technical Report, SE89-7, pp.49 - 56, 1992.

This article was processed using the LaTeX macro package with LLNCS style