プログラミング－言語・基礎・実践－ 15－7
（1994. 1. 21）

プロセス代数によるプロセス生成機能をもつ並行システムの解析

磯部 祥尚, 小島 功, 大蒔 和仁

電子技術総合研究所 情報アーキテクチャ部 情報ベース研究室

本稿では、(1) 子プロセスを生成して、(2) プロセス木を構築し、(3) 親プロセスと子プロセス間でマルチウェイ局所通信ができる並行システムを解析する。我々はこのようなシステムを **GP-** システムと呼ぶ。例えば、アクティブデータベースシステムはこの GP- システム上に実現できる。アクティブデータベースでは、利用者が自由にデータの更新時期、更新条件、更新方法を記述することができるが、並行動作するプロセス間の通信や、子プロセスの反復的生成により、その設計は困難である。

並行システムを記述し解析するために、CCS や CSP のようなプロセス代数が知られている。プロセス代数は、(1) 並行システムの動作がその仕様に等価であるかどうかをチェックできる、または (2) 2 つの等価な小さいサブシステムを用いて、2 つの等価な大きいシステムを設計できるなどの利点をもつ。しかし、従来のプロセス代数を直接 GP- システムに適用することは困難であると思われる。

我々は、CCS を拡張した **CCSGP** を提案する。CCSGP は GP- システムの記述に適したプロセス代数である。本稿では、GP- システムがどのように CCSGP によって記述されるかをアクティブデータベースの例をもとに説明した後、CCSGP の形式的な定義を与え、最後に GP- システムの設計に有効な定理を示す。

Analysis of Communicating Systems with Generating Processes by Process Algebra

Yoshinao ISOBE, Isao KOJIMA, Kazuhito OHMAKI

Information Base Section, Computer Science Division, Electrotechnical Laboratory
1-1-4 Umezono, Tsukuba, Ibaraki 305, Japan

We analyze concurrent and communicating systems which (1) can *generate child-processes*, (2) construct *process trees*, and (3) have *multi-way local communications* between a parent-process and child-processes. In this paper, such systems with generating processes are called *GP-systems*. For example, *active database systems* are implemented in GP-systems. In active database systems, though users can freely design occasions, conditions, and actions for updating datas, it is difficult to design them because of the communication, the concurrency, the cascading generation, and so on.

In order to describe and analyze concurrent and communicating systems, *process algebras* such as CCS, CSP, and ACP, are well known. Process algebras are useful for (1) checking whether behavior of a concurrent system is equal to its specification or not, (2) constructing equal large systems from equal small subsystems, and so on. But it seems hard to directly use these process algebras for GP-systems.

In this paper, we propose a process algebra named *CCSGP* (Calculus of Communicating Systems with Generating Processes), which is an extension of CCS. GP-systems can be directly described in CCSGP. We explain how GP-systems are described in CCSGP by an example of an active database system, give a formal definition of CCSGP, and show one important proposition useful for design of GP-systems.

isobe@etl.go.jp, kojima@etl.go.jp, ohmaki@etl.go.jp

1 Introduction

We analyze concurrent and communicating systems which (1) can *generate child-processes*, (2) construct *process trees*, and (3) have *multi-way local communications* between a parent-process and child-processes. In this paper, such systems are called *GP-systems*. A GP-system is explained by Figure 1. P_i in Figure 1(a) is a process running concurrently, communicating with other processes. Q_i is a process reserved as a resource, for example, in disks. The running process such as P_i can generate *child-processes* Q_j's from the resources. Furthermore, a child-process Q_i also can generate child-processes Q_j's, and then a *process tree* is constructed as shown in Figure 1(b). The process tree shows the relations of a parent and children, and is used for multi-way local communications between them.

For example, *active database systems* are implemented in GP-systems. It fundamentally consists of *rules* and *managers*. In HiPAC[1], the each rule is a program including an *event*, *conditions*, and *actions*. When a rule is triggered by the event, the managers generate transactions for the condition evaluation and the action execution. If the conditions are satisfied, then the action is executed. The action can include operations which trigger another rules as nested transactions (also called subtransactions). Thus, cascading triggers produce a tree of nested transactions. In active database systems, users can make new rules, exchange rules, and modify rules if necessary. It may cause unexpected interactions among rules. For example, infinite cascading triggers may happen because of rules with triggering each other. Hence, it is important to analyze the behavior of the systems before they are executed.

In order to describe and analyze concurrent and communicating systems, *process algebras* such as CCS[2], CSP[3], and ACP[4], are well known. Process algebras are useful for (1) checking whether behavior of a concurrent system is equal to its specification or not, (2) constructing equal large systems from equal small subsystems, and so on. For (1), a specification may be a list of formulae of the form {`if (event,state) then (new state)`}. If both a concurrent system and its specification are interpreted to descriptions in a language followed a process algebra, then their equality can be checked by the algebraic laws as shown in Figure 2. For (2), it is possible to check equality of large systems by separating into small subsystems, if the operators preserve the equality. For example see Figure 3.

But it seems hard to directly use the process algebras such as CCS, CSP, and ACP, for GP-systems. We tried to describe GP-systems in π-calculus[5]

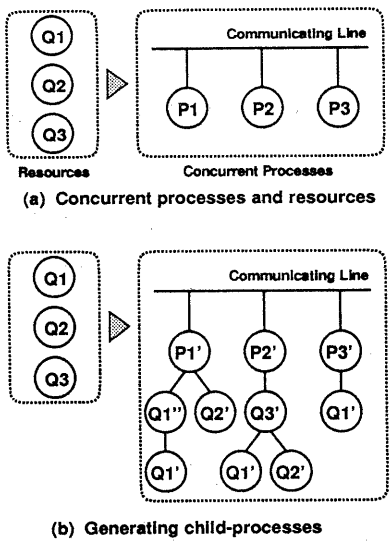

Figure 1: An example of GP-system.

which is an extension of CCS. For example, generating processes can be described as follows in π-calculus:

$$R \stackrel{\text{def}}{=} a(x).(R \mid NEWPR)$$
$$SYSTEM \stackrel{\text{def}}{=} (a)(R \mid \overline{a}(pr1).PR1 \mid PR2)$$
$$SYSTEM \xrightarrow{\tau}$$
$$(a)((pr1)((R \mid NEWPR)\{pr1/x\} \mid PR1) \mid PR2)$$

where the operator '|' means a concurrent composition, and the operator '(a)' means a restriction of communication scope for 'a'. In this example, $SYSTEM$ can generate a new process $NEWPR$ by an internal event 'τ' through 'a' and '$\overline{a}$'. And a *private link* between $PR1$ and $NEWPR$ through '$pr1$' is made by passing the event '$pr1$' from $PR1$ to $NEWPR$. The '$pr1$' works as a *process-id* of $PR1$ and it shows that the parent of $NEWPR$ is not $PR2$, but $PR1$. Hence, it is possible to have local communications between the parent-process $PR1$ and the child-process $NEWPR$.

On the other hand, multi-way communications are used in CSP or SCCS[2] as follows:

$$(a.P1 \| a.P2 \| a.P3) \xrightarrow{a} (P1 \| P2 \| P3) \qquad (CSP)$$
$$(a.P1 \times a.P2 \times a.P3) \xrightarrow{a^3} (P1 \times P2 \times P3) \quad (SCCS)$$

For active database systems, the transition such as in CSP is required because it is independent of the number of synchronized processes (Note '$\xrightarrow{a}$' in (CSP) and '$\xrightarrow{a^3}$' in (SCCS)).

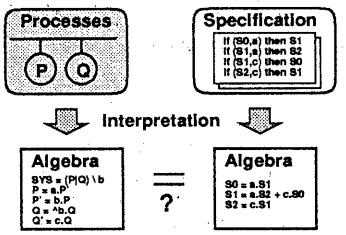

Figure 2: A proof of equality between a concurrent processes and its specification.

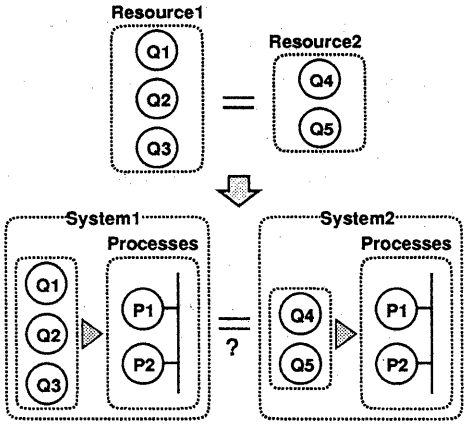

Figure 3: A proof of equality of a system based on equality of subsystems.

For describing GP-systems, a process algebra such as π-calculus must be extended to describe multi-way communications such as in CSP. However, it is difficult, because an *alphabet*[1] must be given before the execution in CSP and the *alphabet* may be variable by passing events in π-calculus. Furthermore, passing events among processes forces a calculus to be complex, though it is too useful.

In this paper, we propose a process algebra named *CCSGP*. CCSGP is a 'Calculus of Communicating Systems with Generating Processes', and is an extension of CCS, not of π-calculus. In CCSGP a new operator '$\rangle$' is used for determining the relation between a parent-process and child-processes instead of a *process-id*. For example, the two state (a) and (b) of the system in Figure 1 are described in CCSGP as follows:

[1] An *alphabet* is a set of all event which a process can occur in the future. It is also called a *sort* in CCS.

$$SYS(a) \equiv (\{\!|Q1|\!\} :: \{\!|Q2|\!\} :: \{\!|Q3|\!\}) \rhd (P1|P2|P3)$$

$$\begin{aligned} SYS(b) \equiv (\{\!|Q1|\!\} :: \{\!|Q2|\!\} :: \{\!|Q3|\!\}) \rhd \\ ((P1')((Q1'')Q1')|Q2')) \\ |(P2')(Q3')(Q1'|Q2'))) \\ |(P3')Q1')) \end{aligned}$$

where the operator '$\rangle$' means a subordinate concurrent composition. In this way, GP-systems can be directly described in CCSGP.

Section 2 informally introduce the different points of CCSGP from CCS, and shows an application of CCSGP for an active database system. The definition of CCSGP is formally given in Section 3. In Section 4 we explain some properties of strong equivalence in CCSGP.

2 Introduction to CCSGP

In this section, extended points of CCSGP are stated, and examples of descriptions are shown.

2.1 The new operators of CCSGP compared with CCS

CCS (Calculus of Communicating Systems) is a fundamental process algebra proposed by R.Milner[2]. We extend CCS to a process algebra *CCSGP* adding seven basic operators '$\|$, $\rangle$, $[\![\ ]\!]$, $/$, $\rhd$, $\{\!|\ |\!\}$, and $::$' for describing GP-systems. Local-events '$\lceil a \rceil, \lfloor a \rfloor$' are added for local communications and Call-events '$\lceil a \rceil$' are added for calling child-processes from resources. The function of each new operator is informally explained as follows:

- **Synchronous composition** '$\|$'. It is similar to an Asynchronous composition '$|$' in CCS except for synchronization of only local-up-events '$\lfloor a \rfloor$'. For example:

  $$((\lfloor a \rfloor.P1) \,\|\, (\lfloor a \rfloor.P2)) \xrightarrow{\lfloor a \rfloor} (P1 \| P2)$$

  It is more similar to a concurrency operator '$\|$' in CSP.

- **Subordinate composition** '$\rangle$'. It is similar to an Asynchronous composition '$|$' in CCS except for local communications between a parent-process and child-processes. For example:

  $$((\lceil a \rceil.P0) \rangle (\lfloor a \rfloor.P1 \| \lfloor a \rfloor.P2)) \xrightarrow{\tau} (P0 \rangle (P1 \| P2))$$

  where $P0$ is a parent-process and $P1, P2$ are child-processes of $P0$.

- **Renaming** '$[\![\ ]\!]$'. It is similar to a Relabelling '$[\ ]$' in CCS except for renaming only call-events $\lceil a \rceil$. For example:

  $$(\lceil a \rceil.P3)[\![b/a]\!] \xrightarrow{\lceil b \rceil} P3[\![b/a]\!]$$

- **Hiding** '$/$'. It is exactly same to a Hiding '$/$' in CSP. For example:

  $$(\lfloor a \rfloor.P3)/a \xrightarrow{\tau} P3/a$$

In CCS, since a Hiding can be defined by Asynchronous compositions and Restrictions, it is removed from basic operators. But in CCSGP it is needed for local-events.

- **Supplying '$\triangleright$'.** It supplys a new process from resources of left side of '$\triangleright$' to processes of right side through call-events. For example:

$$\{\!|b.Q1|\!\} \triangleright (([a].P3)[b/a]) \xrightarrow{\tau} \{\!|b.Q1|\!\} \triangleright ((P3[b/a]))Q1)$$

where $\{\!|\ |\!\}$ is a Resource operator as explained next.

- **Resourcing '$\{\!|\ |\!\}$'.** It builds a resource from a process. A process in a resource is called out by initial events of the process. For example:

$$\{\!|b.Q1|\!\} \xrightarrow{b} \{\!|b.Q1|\!\} \triangleright Q1$$

- **Union '::'.** It unites two resources into one resource. If processes in resources have same initial events, then such processes are simultaneously generated. For example:

$$\{\!|b.Q1|\!\} :: \{\!|a.Q2 + b.Q3|\!\} :: \{\!|c.Q4|\!\} \xrightarrow{b} (\{\!|b.Q1|\!\} :: \{\!|a.Q2 + b.Q3|\!\} :: \{\!|c.Q4|\!\}) \triangleright (Q1 \| Q3)$$

2.2 Penetration of Supplying operators

Notice that Supplying operators '$\triangleright$' can *penetrate* other operators, and can supply processes from resources. For example, the transition

$$\{\!|a.Q1|\!\} \triangleright ([a].P1|[b].P2) \xrightarrow{\tau} \{\!|a.Q1|\!\} \triangleright ((P1)Q1)|[b].P2)$$

is possible. In this case, '$\{\!|a.Q1|\!\} \triangleright$' penetrates '|', and supplys the $Q1$ to $P1$. Furthermore, it is important to notice that the call-events '$[a]$' are prohibited by all operators except for a Prefix, a Summation, and a Renaming. For example, the transition

$$[a].P1 + [b].P2 \xrightarrow{[a]} P1$$

is possible, but the transition

$$([a].P1|[b].P2) \xrightarrow{[a]} (P1|[b].P2)$$

is impossible. So, the transition

$$\{\!|a.Q1|\!\} \triangleright ([a].P1|[b].P2) \xrightarrow{\tau} \{\!|a.Q1|\!\} \triangleright ((P1|[b].P2))Q1) \qquad (*)$$

is impossible. In principle, child-processes have one parent-process, to avoid undeterminism of relations of parent-processes and child-processes, in other words, undeterminism of process trees.[2] In the case $(*)$, the child-process $Q1$ has the two parent-processes $P1$ and $[b].P2$. Of course, one parent-process can have many child-processes. If a parent-process $P1$ has already had a child-process $Q1$ and $P1$ calls $Q2$, then $Q2$ are running at the same level as $Q1$.

$$\{\!|a.Q2|\!\} \triangleright ([a].P1)Q1) \xrightarrow{\tau} \{\!|a.Q2|\!\} \triangleright (P1)(Q1|Q2))$$

[2] We are also thinking an extension for describing two or more parent-processes, for example, using a *Strong composition* '$\|$' which prohibits the penetration of Supplying operators.

2.3 An application of CCSGP for an active database system

In this subsection, we apply CCSGP to an active database system called a Securities Analyst's Assistant (SAA)[1]. (It is slightly different from SAA in [1].) The purpose of this application is to deliver information to an analyst's display, and to automatically execute trades according to the analyst's instructions. On the other hand, the analyst can also buy securities manually. The SAA consists of four rules as follows:

1. **DISP** displays the new price, if a current price of a security is updated.

   Event : `update`
   Condition : `new value` $\neq$ `old value`
   Action : `display new value`

2. **TR** buys a security, if the current price of the security is updated and if the new value is 50.

   Event : `update`
   Condition : `new value = 50`
   Action : `buy the security from account1`

3. **BUY** buys the security and decreases account1, if 'buy' is requested.

   Event : `buy`
   Condition : `true`
   Action : `decrease account1`

4. **CHK** checks whether account1 is minus or not, if one bought from account1.

   Event : `buy`
   Condition : `account1 < 0`
   Action : `display warning`

The **DISP** and the **TR** can run *separately*. The **TR** calls the **BUY** and the **CHK**, and waits to commit until they commit. The **BUY** is *immediately* executed when it is called. However, the execution of the **CHK** is *deffered* unitl just prior to the top level transaction committing, because it is insignificant to be minus temporally if the last results are plus. Thus, we expect that SAA is behavior as shown in Figure 4. The *immediate*, the *separate*, and the *deferred* are called *coupling modes* in active database systems[1].

The system SAA is described in CCSGP as follows:

$$\begin{aligned}
\mathtt{SAA} &\stackrel{\mathrm{def}}{=} \mathtt{RES0} \triangleright (\mathtt{M1}|\mathtt{M2}) \\
\mathtt{RES0} &\stackrel{\mathrm{def}}{=} \{\!|\mathtt{DISP}|\!\} :: \{\!|\mathtt{TR}|\!\} :: \{\!|\mathtt{BUY}|\!\} :: \{\!|\mathtt{CHK}|\!\} \\
\mathtt{DISP} &\stackrel{\mathrm{def}}{=} \mathtt{update}.\mathtt{DISP}' \\
\mathtt{DISP}' &\stackrel{\mathrm{def}}{=} (\lfloor \mathtt{com} \rfloor.\lfloor \mathtt{stdef} \rfloor.\lfloor \mathtt{comdef} \rfloor.0) \mid \mathtt{DISP}'' \\
\mathtt{DISP}'' &\stackrel{\mathrm{def}}{=} \{\mathtt{nv} \neq \mathtt{ov}\}.\overline{\mathtt{display}}.0 + \{\mathtt{nv} = \mathtt{ov}\}.0 \\
\mathtt{TR} &\stackrel{\mathrm{def}}{=} \mathtt{update}.\mathtt{TR}' \\
\mathtt{TR}' &\stackrel{\mathrm{def}}{=} \{\mathtt{nv} = 50\}.\lceil \mathtt{buy} \rceil.\mathtt{TR}'' \\
&\quad + \{\mathtt{nv} \neq 50\}.\lfloor \mathtt{com} \rfloor.\lfloor \mathtt{stdef} \rfloor.\lfloor \mathtt{comdef} \rfloor.0 \\
\mathtt{TR}'' &\stackrel{\mathrm{def}}{=} \lceil \mathtt{com} \rceil.\lfloor \mathtt{com} \rfloor.\lfloor \mathtt{stdef} \rfloor.\lceil \mathtt{stdef} \rceil.\lceil \mathtt{comdef} \rceil.\lfloor \mathtt{comdef} \rfloor.0 \\
\mathtt{BUY} &\stackrel{\mathrm{def}}{=} \mathtt{buy}.\mathtt{BUY}' \\
\mathtt{BUY}' &\stackrel{\mathrm{def}}{=} \overline{\mathtt{buysecu}}.\overline{\mathtt{decrease}}.\lfloor \mathtt{com} \rfloor.\lfloor \mathtt{stdef} \rfloor.\lfloor \mathtt{comdef} \rfloor.0
\end{aligned}$$

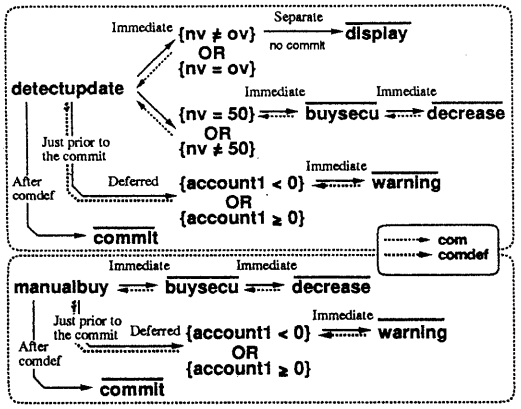

Figure 4: The sketch of the behavior of the SAA

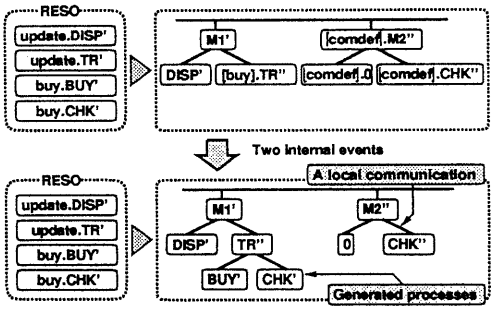

Figure 5: One example of transitions in SAA

$$
\begin{aligned}
\texttt{CHK} &\stackrel{\text{def}}{=} \texttt{buy.CHK'} \\
\texttt{CHK'} &\stackrel{\text{def}}{=} \lfloor \texttt{com} \rfloor.\lfloor \texttt{stdef} \rfloor.\texttt{CHK''} \\
\texttt{CHK''} &\stackrel{\text{def}}{=} \{\texttt{account1} < 0\}.\overline{\texttt{warning}}.\lfloor \texttt{comdef} \rfloor.\texttt{0} \\
&\quad + \{\texttt{account1} \geq 0\}.\lfloor \texttt{comdef} \rfloor.\texttt{0} \\
\texttt{M1} &\stackrel{\text{def}}{=} \texttt{detectupdate.[update].M1'} \\
\texttt{M1'} &\stackrel{\text{def}}{=} \lceil \texttt{com} \rceil.\lceil \texttt{stdef} \rceil.\lceil \texttt{comdef} \rceil.\texttt{M1''} \\
\texttt{M1''} &\stackrel{\text{def}}{=} \overline{\texttt{commit}}.\texttt{M1} \\
\texttt{M2} &\stackrel{\text{def}}{=} \texttt{manualbuy.[buy].M2'} \\
\texttt{M2'} &\stackrel{\text{def}}{=} \lceil \texttt{com} \rceil.\lceil \texttt{stdef} \rceil.\lceil \texttt{comdef} \rceil.\texttt{M2''} \\
\texttt{M2''} &\stackrel{\text{def}}{=} \overline{\texttt{commit}}.\texttt{M2}
\end{aligned}
$$

We can also describe the behavior of Figure 4 in CCSGP, and can prove that the behavior of SAA defined above is equivalent to one of Figure 4. One example of transitions in SAA is shown as follows:

$$
\begin{aligned}
&\texttt{RESO} \rhd ((\texttt{M1'})(\texttt{DISP'} \| (\texttt{[buy].TR''})) \\
&\quad |(\lceil \texttt{comdef} \rceil.\texttt{M2''})(\lfloor \texttt{comdef} \rfloor.\texttt{0} \| \lfloor \texttt{comdef} \rfloor.\texttt{CHK''}))) \xrightarrow{\tau}\xrightarrow{\tau} \\
&\texttt{RESO} \rhd ((\texttt{M1'})(\texttt{DISP'} \| (\texttt{TR''})(\texttt{BUY'} \| \texttt{CHK'}))) \\
&\quad |(\texttt{M2''})(\texttt{0} \| \texttt{CHK''})))
\end{aligned}
$$

This transition is illustrated in Figure 5. One internal event 'τ' occurs for generating new processes BUY' and CHK' under TR''. The other 'τ' occurs for a local communication between the parent ($\lceil$comdef$\rceil$.M2'') and the children ($\lfloor$comdef$\rfloor$.0, $\lfloor$comdef$\rfloor$.CHK'').

3 The definitions of CCSGP

In this section, we formally define the event, the syntax and the semantics of CCSGP.

3.1 Event

We first assume that an infinite set $\mathcal{N} = \{a, b, c, \cdots\}$ of names is given. It is ranged over by 'a'. Then we define sets of events, where $\tau \notin \mathcal{N}$ and 'τ' is a special event called an *internal event.*

Definition 3.1 *We define the following four sets of events:*

- $E_G = \{a, \overline{a} : a \in \mathcal{N}\}$ *is a set of* global-events, *and is ranged over by 'ρ'.*
- $E_L = \{\lfloor a \rfloor, \lceil a \rceil : a \in \mathcal{N}\}$ *is a set of* local-events, *and is ranged over by 'ν'.*
- $E_C = \{[\rho] : \rho \in E_G\}$ *is a set of* call-events, *and is ranged over by 'μ'.*
- $Event = E_G \cup E_L \cup E_C \cup \{\tau\}$ *is a set of* events, *and is ranged over by 'ω'.* ∎

The set $E_G \cup \{\tau\}$ and $E_G \cup E_L \cup \{\tau\}$ are ranged over by 'α' and 'σ', respectively.

3.2 Syntax

We define the syntax of CCSGP as follows:

Definition 3.2 *The set of process expressions, $\mathcal{E}$ ranged over by $E, F, \cdots$, is the smallest set including the following expressions:*

$$
\begin{aligned}
X &: \text{Variable } (X \in \mathcal{X}) \\
A &: \text{Constant } (A \in \mathcal{K}) \\
R &: \text{Resource } (R \in \mathcal{R}) \\
\omega.E &: \text{Prefix } (\omega \in Event) \\
\Sigma_{i \in I} E_i &: \text{Summation } (I \text{ is an indexing set}) \\
E_1 \| E_2 &: \text{Synchronous composition} \\
E_1 | E_2 &: \text{Asynchronous composition} \\
E_1 \rangle E_2 &: \text{Subordinate composition} \\
E[f] &: \text{Relabelling } (f \text{ is relabelling function}) \\
E[\![f]\!] &: \text{Renaming } (f \text{ is renaming function}) \\
E \backslash L &: \text{Restriction } (L \subseteq \mathcal{N}) \\
E/L &: \text{Hiding } (L \subseteq \mathcal{N}) \\
R \rhd E &: \text{Supplying}(R \in \mathcal{R})
\end{aligned}
$$

where E, E_i are already in $\mathcal{E}$. $\mathcal{X}$ and $\mathcal{K}$ are sets of process variables and process constants, respectively.

The set of Resources, $\mathcal{R}$ ranged over by $R, R_i, \cdots$, is the smallest set including the following expressions:

$$\begin{array}{l} \{\!|P|\!\} : \text{Resourcing}(P \in \mathcal{P}) \\ R_1 :: R_2 : \text{Union} \end{array}$$

where R, R_i are already in $\mathcal{R}$. The set of process, $\mathcal{P}$ ranged over by $P, Q, \cdots$, is the smallest set including the following expressions:

$$\begin{array}{llllll} A(\in \mathcal{K}), & R(\in \mathcal{R}), & \omega.P, & \Sigma_{i\in I} P_i, & P_1 \| P_2, & P_1 | P_2, \\ P_1 \rangle P_2, & P[f], & P[\![f]\!], & P \backslash L, & P/L, & R \rhd P, \end{array}$$

where P, P_i are already in $\mathcal{P}$. ∎

The relabelling function '$f : E_G \cup E_L \cup \{\tau\} \to E_G \cup E_L \cup \{\tau\}$' satisfies the following conditions:

- $f(\omega) \in E_G$ iff $\omega \in E_G$
- $f(\omega) \in E_L$ iff $\omega \in E_L$
- $\overline{f(\rho)} = f(\overline{\rho})$
- $f(\tau) = \tau$
- $\lfloor f(a) \rfloor = f(\lfloor a \rfloor)$
- $\lceil f(a) \rceil = f(\lceil a \rceil)$

The renaming function '$f : E_C \to E_C$' satisfies the condition $[f(\rho)] = f([\rho])$.

A *Constant* is an process whose meaning is given by defining equation. In fact, we assume that for every Constant 'A' there is a defining equation of the following form:

$$A \stackrel{\text{def}}{=} P \qquad (P \in \mathcal{P})$$

where each occurrence of A in P is within some subexpression $\omega.P'$. In other words, A is *weakly guarded* in P. Constants which are not weakly guarded make a calculus more complex, and the behavior is indefinite[7]. Hence, we treat only weakly guarded Constants in CCSGP.

A special process *inaction* '**0**' is defined by using *Summation* as follows:

$$0 \stackrel{\text{def}}{=} \Sigma_{i\in\emptyset} E_i$$

3.3 Semantics

The semantics of CCSGP is defined by the following labelled transition system like one of CCS:

$$(\mathcal{E}, Event, \{\stackrel{\omega}{\longrightarrow} : \omega \in Event\})$$

For example, '$E \stackrel{\omega}{\longrightarrow} E'$ $(E, E' \in \mathcal{E})$' means that the process expression E becomes the process expression E' due to the occurrence of the event 'ω'. The semantics of the process expressions consists of the definition of the transition relations '$\stackrel{\omega}{\longrightarrow}$' over $\mathcal{E}$.

Before defining the semantics, we define a set of global-events for each process P.

Definition 3.3 *We define a set '$ev(P)$' of* syntactic global-events *of a process P as follows:*

$$ev(\omega.P) = \begin{cases} \{\rho\} & (\omega = \rho) \\ \emptyset & (\omega \neq \rho) \end{cases}$$

$$\begin{array}{rcl} ev(\Sigma_{i\in I} P_i) & = & \bigcup_{i\in I} ev(P_i) \\ ev(P \| Q) & = & ev(P) \cup ev(Q) \\ ev(P | Q) & = & ev(P) \cup ev(Q) \\ ev(P \rangle Q) & = & ev(P) \cup ev(Q) \\ ev(P[f]) & = & \{f(\rho) : \rho \in ev(P)\} \\ ev(P[\![f]\!]) & = & ev(P) \\ ev(P \backslash L) & = & ev(P) - \{a, \overline{a} : a \in L\} \\ ev(P/L) & = & ev(P) - \{a, \overline{a} : a \in L\} \\ ev(A) & = & ev(P) \qquad (A \stackrel{\text{def}}{=} P) \\ ev(\{\!|P|\!\}) & = & ev(P) \\ ev(R_1 :: R_2) & = & ev(R_1) \cup ev(R_2) \\ ev(R \rhd P) & = & ev(P) \end{array}$$ ∎

Since a Constant 'A' must be weakly guarded, this definition always stops. Then, the semantics of CCSGP is defined as follows:

Definition 3.4 *The transition relation '$\stackrel{\omega}{\longrightarrow}$' over process expressions is the smallest relation satisfying the following inference rules, where the rule means that if the transition relation above the line exists and the side conditions are satisfied, then the transition relation below the line also exists.*

$$\textbf{Event}\ \frac{}{\omega.E \stackrel{\omega}{\longrightarrow} E}$$

$$\textbf{Sum}_j\ \frac{E_j \stackrel{\omega}{\longrightarrow} E'_j}{\Sigma_{i\in I} E_i \stackrel{\omega}{\longrightarrow} E'_j} \quad (j \in I)$$

$$\textbf{Sync}_1\ \frac{E \stackrel{\lfloor a \rfloor}{\longrightarrow} E' \qquad F \stackrel{\lfloor a \rfloor}{\longrightarrow} F'}{E \| F \stackrel{\lfloor a \rfloor}{\longrightarrow} E' \| F'}$$

$$\textbf{Sync}_2\ \frac{E \stackrel{\alpha}{\longrightarrow} E'}{E \| F \stackrel{\alpha}{\longrightarrow} E' \| F}$$

$$\textbf{Sync}_3\ \frac{F \stackrel{\alpha}{\longrightarrow} F'}{E \| F \stackrel{\alpha}{\longrightarrow} E \| F'}$$

$$\textbf{Sync}_4\ \frac{E \stackrel{\rho}{\longrightarrow} E' \qquad F \stackrel{\overline{\rho}}{\longrightarrow} F'}{E \| F \stackrel{\tau}{\longrightarrow} E' \| F'}$$

$$\textbf{Com}_1\ \frac{E \stackrel{\sigma}{\longrightarrow} E'}{E | F \stackrel{\sigma}{\longrightarrow} E' | F} \quad (\sigma \neq \lceil a \rceil)$$

$$\textbf{Com}_2\ \frac{F \stackrel{\sigma}{\longrightarrow} F'}{E | F \stackrel{\sigma}{\longrightarrow} E | F'} \quad (\sigma \neq \lceil a \rceil)$$

$$\textbf{Com}_3\ \frac{E \stackrel{\rho}{\longrightarrow} E' \qquad F \stackrel{\overline{\rho}}{\longrightarrow} F'}{E | F \stackrel{\tau}{\longrightarrow} E' | F'}$$

$$\textbf{Subo}_1\ \frac{E \stackrel{\lfloor a \rfloor}{\longrightarrow} E'}{E \rangle F \stackrel{\lfloor a \rfloor}{\longrightarrow} E' \rangle F}$$

$$\textbf{Subo}_2\ \frac{F \stackrel{\lceil a \rceil}{\longrightarrow} F'}{E \rangle F \stackrel{\tau}{\longrightarrow} E \rangle F'}$$

$$\textbf{Subo}_3\ \frac{E \stackrel{\lceil a \rceil}{\longrightarrow} E' \qquad F \stackrel{\lfloor a \rfloor}{\longrightarrow} F'}{E \rangle F \stackrel{\tau}{\longrightarrow} E' \rangle F'}$$

$$\textbf{Subo}_4\ \frac{E \stackrel{\rho}{\longrightarrow} E'}{E \rangle F \stackrel{\rho}{\longrightarrow} E' \rangle F}$$

$$\textbf{Subo}_5\ \frac{F \xrightarrow{\alpha} F'}{E\rangle F \xrightarrow{\alpha} E\rangle F'}$$

$$\textbf{Subo}_6\ \frac{E \xrightarrow{\rho} E' \quad F \xrightarrow{\overline{\rho}} F'}{E\rangle F \xrightarrow{\tau} E'\rangle F'}$$

$$\textbf{Rel}\ \frac{E \xrightarrow{\sigma} E'}{E[f] \xrightarrow{f(\sigma)} E'[f]}$$

$$\textbf{Ren}_1\ \frac{E \xrightarrow{[\rho]} E'}{E[\![f]\!] \xrightarrow{f([\rho])} E'[\![f]\!]}$$

$$\textbf{Ren}_2\ \frac{E \xrightarrow{\sigma} E'}{E[\![f]\!] \xrightarrow{\sigma} E'[\![f]\!]}\quad (\sigma \neq \tau)$$

$$\textbf{Res}\ \frac{E \xrightarrow{\sigma} E'}{E\backslash L \xrightarrow{\sigma} E'\backslash L}\quad (\sigma \notin \langle L\rangle)$$

$$\textbf{Hide}_1\ \frac{E \xrightarrow{\sigma} E'}{E/L \xrightarrow{\tau} E'/L}\quad (\sigma \in \langle L\rangle)$$

$$\textbf{Hide}_2\ \frac{E \xrightarrow{\sigma} E'}{E/L \xrightarrow{\sigma} E'/L}\quad (\sigma \notin \langle L\rangle)$$

$$\textbf{Reso}\ \frac{P \xrightarrow{\rho} P'}{\{\!|P|\!\} \xrightarrow{\rho} \{\!|P|\!\} \rhd P'}$$

$$\textbf{Uni}_1\ \frac{R_1 \xrightarrow{\rho} R_1 \rhd P}{(R_1 :: R_2) \xrightarrow{\rho} (R_1 :: R_2) \rhd P}\quad (\rho \notin ev(R_2))$$

$$\textbf{Uni}_2\ \frac{R_2 \xrightarrow{\rho} R_2 \rhd P}{(R_1 :: R_2) \xrightarrow{\rho} (R_1 :: R_2) \rhd P}\quad (\rho \notin ev(R_1))$$

$$\textbf{Uni}_3\ \frac{R_1 \xrightarrow{\rho} R_1 \rhd P \quad R_2 \xrightarrow{\rho} R_2 \rhd Q}{(R_1 :: R_2) \xrightarrow{\rho} (R_1 :: R_2) \rhd (P\|Q)}$$

$$\textbf{Con}\ \frac{P \xrightarrow{\omega} P'}{A \xrightarrow{\omega} P'}\quad (A \stackrel{\text{def}}{=} P)$$

$$\textbf{Supp}_1\ \frac{E \xrightarrow{[\rho]} E' \quad R \xrightarrow{\rho} R \rhd P}{R \rhd E \xrightarrow{\tau} R \rhd (E'\rangle P)}$$

$$\textbf{Supp}_2\ \frac{E \xrightarrow{[\rho]} E' \quad R \xrightarrow{\rho} R \rhd P}{R \rhd (E\rangle F) \xrightarrow{\tau} R \rhd (E'\rangle (F|P))}$$

$$\textbf{S.Event}\ \frac{}{R \rhd (\omega.E) \xrightarrow{\omega} R \rhd E}\quad (\omega \neq [\rho])$$

$$\textbf{S.Sum}_j\ \frac{R \rhd E_j \xrightarrow{\sigma} R \rhd E_j'}{R \rhd (\Sigma_{i\in I} E_i) \xrightarrow{\sigma} R \rhd E_j'}\quad (j \in I)$$

$$\textbf{S.Sync}_1\ \frac{R \rhd E \xrightarrow{[a]} R \rhd E' \quad R \rhd F \xrightarrow{\lfloor a \rfloor} R \rhd F'}{R \rhd (E\|F) \xrightarrow{\tau} R \rhd (E'\|F')}$$

$$\textbf{S.Sync}_2\ \frac{R \rhd E \xrightarrow{\alpha} R \rhd E'}{R \rhd (E\|F) \xrightarrow{\alpha} R \rhd (E'\|F)}$$

$$\textbf{S.Sync}_3\ \frac{R \rhd F \xrightarrow{\alpha} R \rhd F'}{R \rhd (E\|F) \xrightarrow{\alpha} R \rhd (E\|F')}$$

$$\textbf{S.Sync}_4\ \frac{R \rhd E \xrightarrow{\rho} R \rhd E' \quad R \rhd F \xrightarrow{\overline{\rho}} R \rhd F'}{R \rhd (E\|F) \xrightarrow{\tau} R \rhd (E'\|F')}$$

$$\textbf{S.Com}_1\ \frac{R \rhd E \xrightarrow{\sigma} R \rhd E'}{R \rhd (E|F) \xrightarrow{\sigma} R \rhd (E'|F)}\quad (\sigma \neq [a])$$

$$\textbf{S.Com}_2\ \frac{R \rhd F \xrightarrow{\sigma} R \rhd F'}{R \rhd (E|F) \xrightarrow{\sigma} R \rhd (E|F')}\quad (\sigma \neq [a])$$

$$\textbf{S.Com}_3\ \frac{R \rhd E \xrightarrow{\rho} R \rhd E' \quad R \rhd F \xrightarrow{\overline{\rho}} R \rhd F'}{R \rhd (E|F) \xrightarrow{\tau} R \rhd (E'|F')}$$

$$\textbf{S.Subo}_1\ \frac{R \rhd E \xrightarrow{[a]} R \rhd E'}{R \rhd (E\rangle F) \xrightarrow{[a]} R \rhd (E'\rangle F)}$$

$$\textbf{S.Subo}_2\ \frac{R \rhd F \xrightarrow{[a]} R \rhd F'}{R \rhd (E\rangle F) \xrightarrow{\tau} R \rhd (E\rangle F')}$$

$$\textbf{S.Subo}_3\ \frac{R \rhd E \xrightarrow{[a]} R \rhd E' \quad R \rhd F \xrightarrow{\lfloor a \rfloor} R \rhd F'}{R \rhd (E\rangle F) \xrightarrow{\tau} R \rhd (E'\rangle F')}$$

$$\textbf{S.Subo}_4\ \frac{R \rhd E \xrightarrow{\rho} R \rhd E'}{R \rhd (E\rangle F) \xrightarrow{\rho} R \rhd (E'\rangle F)}$$

$$\textbf{S.Subo}_5\ \frac{R \rhd F \xrightarrow{\alpha} R \rhd F'}{R \rhd (E\rangle F) \xrightarrow{\alpha} R \rhd (E\rangle F')}$$

$$\textbf{S.Subo}_6\ \frac{R \rhd E \xrightarrow{\rho} R \rhd E' \quad R \rhd F \xrightarrow{\overline{\rho}} R \rhd F'}{R \rhd (E\rangle F) \xrightarrow{\tau} R \rhd (E'\rangle F')}$$

$$\textbf{S.Rel}\ \frac{R \rhd E \xrightarrow{\sigma} R \rhd E'}{R \rhd (E[f]) \xrightarrow{f(\sigma)} R \rhd (E'[f])}$$

$$\textbf{S.Ren}\ \frac{R \rhd E \xrightarrow{\sigma} R \rhd E'}{R \rhd (E[\![f]\!]) \xrightarrow{\sigma} R \rhd (E'[\![f]\!])}\quad (\sigma \neq \tau)$$

$$\textbf{S.Res}\ \frac{R \rhd E \xrightarrow{\sigma} R \rhd E'}{R \rhd (E\backslash L) \xrightarrow{\sigma} R \rhd (E'\backslash L)}\quad (\sigma \notin \langle L\rangle)$$

$$\textbf{S.Hide}_1\ \frac{R \rhd E \xrightarrow{\sigma} R \rhd E'}{R \rhd (E/L) \xrightarrow{\tau} R \rhd (E'/L)}\quad (\sigma \in \langle L\rangle)$$

$$\textbf{S.Hide}_2\ \frac{R \rhd E \xrightarrow{\sigma} R \rhd E'}{R \rhd (E/L) \xrightarrow{\sigma} R \rhd (E'/L)}\quad (\sigma \notin \langle L\rangle)$$

$$\textbf{S.Reso}\ \frac{R_1 \xrightarrow{\rho} R_1 \rhd P}{R_2 \rhd R_1 \xrightarrow{\rho} R_2 \rhd (R_1 \rhd P)}$$

$$\textbf{S.Con}\ \frac{R \rhd P \xrightarrow{\sigma} R \rhd P'}{R \rhd A \xrightarrow{\sigma} R \rhd P'}\quad (A \stackrel{\text{def}}{=} P)$$

$$\textbf{S.Supp}\ \frac{R_1 \rhd E \xrightarrow{\sigma} R_1 \rhd E'}{R_2 \rhd (R_1 \rhd E) \xrightarrow{\sigma} R_2 \rhd (R_1 \rhd E')}$$

where,

$$\langle L\rangle = \{a, \overline{a}, \lfloor a \rfloor, \lceil a \rceil : a \in L\}$$

4 Equivalence in CCSGP

We define equivalence relations in CCSGP like in CCS, for example, strong equivalence and observation equivalence. In this paper, we state some properties of CCSGP for only strong equivalence,

because we have not completed the full research of the equivalence yet.

Strong equivalence is defined by strong bisimulations as follows[2].

Definition 4.1 *A binary relation $\mathcal{S} \subseteq \mathcal{P} \times \mathcal{P}$ over processes is a* strong bisimulation *if $(P,Q) \in \mathcal{S}$ implies, for all $\omega \in Event$, that*

(i) whenever $P \xrightarrow{\omega} P'$ then, for some Q', $Q \xrightarrow{\omega} Q'$ and $(P',Q') \in \mathcal{S}$,

(ii) whenever $Q \xrightarrow{\omega} Q'$ then, for some P', $P \xrightarrow{\omega} P'$ and $(P',Q') \in \mathcal{S}$. ∎

Definition 4.2 *P and Q are strongly equivalent, written $P \sim Q$, if $(P,Q) \in \mathcal{S}$ for some strong bisimulation $\mathcal{S}$.* ∎

For the strong equivalence, the following important proposition holds. This proposition is illustrated in Figure 3.

Proposition 4.1 *For any $R_1, R_2 \in \mathcal{R}$, $P \in \mathcal{P}$,*

$$\text{If } R_1 \sim R_2, \text{ then } R_1 \rhd P \sim R_2 \rhd P.$$

Proof *The proof is accomplished by showing that the $\mathcal{S}$ defined below is a strong bisimulation, using induction over 'n' and structures of processes.*

$$\mathcal{S}^{(1)} = \{ (R_1 \rhd P, R_2 \rhd P), (R_1 \rhd Q_1, R_2 \rhd Q_2) : P, Q_1, Q_2 \in \mathcal{P}, \quad R_1, R_2 \in \mathcal{R}, \; R_1 \sim R_2, R_1 \rhd Q_1 \sim R_2 \rhd Q_2 \}$$

($n \geq 2$)

$$\begin{aligned}\mathcal{S}^{(n)} = \{ & (R_1 \rhd (P)P_I), R_2 \rhd (P)P_{II})), \\ & (R_1 \rhd (P_I \| P'_I), R_2 \rhd (P_{II} \| P'_{II})), \\ & (R_1 \rhd (P_I | P'_I), R_2 \rhd (P_{II} | P'_{II})), \\ & (R_1 \rhd (P_I[f]), R_2 \rhd (P_{II}[f])), \\ & (R_1 \rhd (P_I \setminus L), R_2 \rhd (P_{II} \setminus L)), \\ & (R_1 \rhd (P_I / L), R_2 \rhd (P_{II} / L)) \\ & (R_1 \rhd P_I, R_2 \rhd P_{II}) \\ & : (R_1 \rhd P_I, R_2 \rhd P_{II}) \in \mathcal{S}^{(n-1)}, \\ & (R_1 \rhd P'_I, R_2 \rhd P'_{II}) \in \mathcal{S}^{(n-1)}, \\ & P \in \mathcal{P}, \forall f, \forall L, R_1 \sim R_2 \quad \}\end{aligned}$$

$$\mathcal{S} = \lim_{i \to \infty} \mathcal{S}^{(i)}$$

∎

Unfortunately, '$\sim$' is not a congruence relation in CCSGP, because $P \sim Q$ does not imply $R \rhd P \sim R \rhd Q$. For example,

$$([a].0)|(b.0) \sim b.0$$

since call-events are prohibited by '|'. And

$$\{\!\{a.0\}\!\} \rhd (([a].0)|(b.0)) \not\sim \{\!\{a.0\}\!\} \rhd b.0$$

since the following transition is possible,

$$\{\!\{a.0\}\!\} \rhd (([a].0)|(b.0)) \xrightarrow{\tau} \{\!\{a.0\}\!\} \rhd ((0)0)|(b.0))$$

but '$\{\!\{a.0\}\!\} \rhd b.0$' have no τ-transition.

So, we prove some auxiliary propositions to make up for the above defect. For example,

- If $R \rhd P_1 \sim R \rhd P_2$ then $R \rhd (P_1|P) \sim R \rhd (P_2|P)$.
- If $P_1 \sim P_2$ then $R \rhd (P_1)P) \sim R \rhd (P_2)P)$,
- If $R \rhd P_1 \sim R \rhd P_2$ then $R \rhd (P)P_1) \sim R \rhd (P)P_2)$.

5 Conclusion

We have stated the difficulty of describing GP-systems in π-calculus, and we have proposed a process algebra CCSGP for describing GP-systems. In Section 2, some new operators of CCSGP have been informally introduced, and an application of CCSGP for an active database system has been shown. In Section 3 the definition of CCSGP has been given, and in Section 4 an important proposition have given as follows: if a resource $R1$ is strongly equivalent to another resource $R2$, then we can exchange $R1$ for $R2$, preserving the behavior of the whole system. The features of CCSGP are summarized as follows.

- Processes can generate other processes from resources,
- Process trees are constructed automatically,
- Multi-way local communications between a parent-process and child-processes are possible.

Yet, we remains many works on CCSGP about equivalence relations. We hope to have algebraic laws enough to prove equivalence between processes. Furthermore, CCSGP may be modified to describe systems with two or more parent-processes for one child-process.

Acknowledgement

The authors wish to express our gratitude to Dr. Kimihiro Ohta, Director of Computer Science Division, Electrotechnical Laboratory. They thank all colleagues in Information Base Section for their helpful discussions.

References

[1] D.McCarthy and U.Dayal. "The Architecture Of An Active Data Base Management System", Proc. of the 1989 ACM SIGMOD Conference, pp.215 - 224, 1989.

[2] R.Milner. "Communication and Concurrency", Prentice-Hall, 1989.

[3] C.A.R.Hoare. "Communicating Sequential Processes", Prentice-Hall, 1985.

[4] J.C.M.Baeten and W.P.Weijland. "Process Algebra", Cambridge Tracts in Theoretical Computer Science 18, Cambridge University Press, 1990.

[5] Robin Milner, Joachim Parrow and David Walker. "A Calculus of Mobile Processes, I and II", Information and Computation, 100, pp.1 - 40 and pp.41 - 77, 1992.

[6] Luca Aceto, Bard Bloom, and Frits Vaandrager. "Turnig SOS Rules into Equations", Proc. of 7th annual IEEE symposium on Logic in Computer Science, pp.113 - 124, 1992.

[7] D.J.Walker. "Bisimulations and Divergence", Proc. of 3th annual IEEE symposium on Logic in Computer Science, pp.186 - 192, 1988.