数理解析研究所講究録
996 巻 1997 年 82-95

RIMS Workshop in Computing
Concurrency Theory and Applications '96
Kyoto University,
July 22-24, 1996

A Domain for Concurrent Semantics of Mobile Processes

Masaki Murakami

Faculty of Engineering, Okayama University
3-1-1 Tsushima-Naka, Okayama 700, Japan
e-mail: murakami@momo.it.okayama-u.ac.jp

Abstract: This paper studies a domain for a non-interleaving denotational semantics for mobile processes that are defined as a subset of π-calculus. A domain of *dual bracket structures (DBS's)* is presented. A DBS is a collection of partially ordered events with the branching structure of a process. The partial order denotes the causality between events. The branching structure of a process is denoted using two types of nested brackets '$[\![$', '$]\!]$' and '$\langle$', '$\rangle$'. A semantic mapping from a subset of π-calculus to the domain of DBS is presented.

1 Introduction

π-calculus[Mil92, Mil89a] is an extension of CCS that is equipped with the feature of *name passing* and can formally represent concurrent systems consists of "mobile" processes which have dynamically reconfigurable communication connections. It was intended to provide a theoretical framework based on a CCS-like processes algebra and the theory of equivalence relations are investigated as classical process algebras. A number of results are reported on bisimulation equivalences of processes with name passing. One of the motivations of these studies is that the bisimilarity defined in the conventional manner as that of CCS is not congruent for prefixing. A typical example is as follows. Consider two processes $P \equiv x(y)|\overline{v}z$ and $Q \equiv x(y).\overline{v}z+\overline{v}z.x(y)$. It is easy to show $P \approx Q$ in conventional sense using the expansion rule. However $w(x).P$ is not bisimilar to $w(x).Q$ because $w(x).P \stackrel{w(a)}{\rightarrow}\stackrel{\tau}{\rightarrow} \mathbf{0}$ but $w(x).Q \stackrel{w(a)}{\rightarrow}\stackrel{\tau}{\not\rightarrow}$.

This informs us that P and Q are different processes in the sense that $x(y)$ and $\overline{v}z$ can happen concurrently in P, on the other hand, $x(y)$ and $\overline{v}z$ must happen sequentially in Q though we do not mind the order.

This reminds us a limit of interleaving models in mobile processes. Interleaving models are justified by the assumption that *any concurrent process can be regarded as a sequential non-deterministic process.* But as the above example shows, two events in a concurrent process may be executed simultaneously in a certain context. Thus a process with a composition operator cannot be regard as a process with no composition operator without considering its context, if it is equipped with name passing facility. We need a new equivalence that distinguish $a|b$ and $ab+ba$. That is one of the motivation for a concurrent (non-interleaving) model of mobile processes.

It was reported that *true concurrency* can be a helpful idea to obtain a congruence relation of mobile processes [Oda]. It was reported that we can obtain a congruence relation that is defined in the manner defining conventional bisimulation relation except using the notion of *multi action.* Multi action is an extension of the notion of actions in conventional π-calculus.

This paper studies a domain for a non-interleaving denotational semantics for mobile processes that are defined as a subset of π-calculus. A domain of *dual bracket structures (DBS's)* is presented. A DBS is a collection of partially ordered events with the branching structure of a process. The partial order denotes the causality between events. The branching structure of a process is denoted using two types of nested brackets '$[\![$', '$]\!]$' and '$\langle$', '$\rangle$'. If two events a and b are in the scope of '$[\![$' and '$]\!]$' , then a and b are in the common branch of the process and are executed concurrently or sequentially in one particular history. In this case, if there is an order between a and b then they are executed sequentially. On the other hand, if they are incomparable, they can happen concurrently. If a and b are in the scope of '$\langle$' and '$\rangle$', they are in the different blanch of the process and only one of them happens in its execution. If a and b are in both '$[\![$' , '$]\!]$' and '$\langle$' , '$\rangle$' as $[\![...\langle...a, ...b, ...\rangle, ...]\!]$, the innermost pair has priority. For example, consider a DBS: $[\![a, b, \langle c, d\rangle]\!]$ where $a \prec b, a \prec c$ and $a \prec d$. In this example, a happens before others. c and b can happen concurrently, and d and b also do. But c or d happens exclusively. Thus this DBS denotes a process such as $a.(b|(c + d))$.

We present a preorder relation on the set of DBS's. The preorder is defined using the idea similar to (bi)simulation of processes in CCS or π-calculus. Intuitively, for DBS's δ_1, δ_2, δ_1 is smaller than or equal to δ_2 if and only if for any set of events $\hat{\alpha}$, if $\hat{\alpha}$ can happen in δ_1 initially, then $\hat{\alpha}$ can also happen in δ_2 initially and the derivative of δ_1 by $\hat{\alpha}$ is also smaller than the derivative of δ_2 by $\hat{\alpha}$. Note that "the rest of δ_i" does not mean that "a process that invoked *after* α" because events in α and events in the rest of δ_i can happen concurrently. That means "a process that must be invoked if we commit to do all events in α". We define a partial order and an equivalence relation between DBS's using the preorder .

We present a semantic mapping from a set of mobile processes that is a fragment of π-calculus to the set of DBS's.

2 Dual Bracket Structures

2.1 Basic Idea

In this section, we define the notion of *dual bracket structures: DBS.* A DBS is a collection of partially ordered events with the branching structure of a process. In that sense, the notion of DBS is very much like to the notion of sets. But a DBS is not just a 'flat' collection of elements. It has nested brackets unlike sets.

The difference between a partially ordered set and a DBS is similar to the difference between a sequence and a tree. A sequence of events is a set of occurrences of actions that are linearly ordered, such as $a \prec b \prec c$... A tree is not just a set of sequence but it has the branching structure. For example a set of sequences $\{ab, ac\}$ may be the set of traces of a tree of Fig 1 or a tree of Fig 2 that are not equivalent.

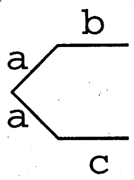

Fig.1

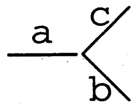

Fig.2

Thus a tree is a collection of traces with the branching structure. Using the branching structure, we can define finer equivalences between processes than the trace equivalence that is too coarse.

The basic idea of the notion of DBS is an extension of the idea that *a tree is a collection of linearly ordered sets with the branching structure* to partially ordered sets. The branching structure of a process is denoted by two type of brackets $[\![,]\!]$ and $\langle,\rangle$ that appears in a collection of events. The basic rules are as following.

- For two events a and b, if the innermost pair of brackets that parenthesis them is $[\![,]\!]$, then both of them may occur concurrently (if a and b are not ordered) or sequentially (if they are ordered). It never happens that only one of them occurs exclusively.
- If the innermost pair is $\langle,\rangle$, then at most only one of them occurs. It never happens that both of a and b occur.

To make it simple, we start with examples of DBS that are syntactic codings of trees. Consider the tree of Fig 2 in which $a \prec b$ and $a \prec c$ and c or b occurs exclusively. The tree is denoted as,

$$[\![a,\ \langle b,\ c\rangle]\!].$$

Note that the innermost pair of brackets that has b and c is $\langle,\rangle$. It means that b or c occurs exclusively. On the other hand the innermost brackets which has both of a and b (or c) is $[\![,]\!]$, that means both of a and b (or a and c) occur. Thus we can reconstruct the tree of Fig. 2 from the DBS $[\![a,\ \langle b,\ c\rangle]\!]$ and the orderings $a \prec b$ and $a \prec c$.

For the case of Fig. 1, we must distinguish two occurrences of a's. We denote the first a that proceeds b as a_1 and the second a that proceeds c as a_2.

The DBS that denotes the tree as Fig 2 is

$$\langle[\![a_1,\ b]\!],\ [\![a_2,\ c]\!]\rangle$$

where $a_1 \prec b$ and $a_2 \prec c$. In this case, the innermost brackets enclosing a_1 and a_2 is $\langle,\rangle$, then a_1 and a_2 occurs exclusively.

The formal definition of DBS is presented in the next section.

2.2 Definition of DBS

Let *Act* be a set of actions that consists of *silent action* τ, *input actions* such as $x(y)$, *free output actions* such as $\overline{x}y$ and *bound output actions* such as $\overline{x}(y)$. We consider a partially ordered set S such that each element of S is labeled with an element of *Act*. S is intended to denote the set of occurrences of actions. Each element of S is called an *event*.

Definition 1

Let S be a set of events with a partial order $\prec$ and $\mathcal{D}_S$ be a set defined from S as the greatest X that satisfies the equation:

$$X = \{\langle\rangle, \llbracket\rrbracket\} \times 2^{(S \cup X)}.$$

In other words, $\delta \in \mathcal{D}_S$ is a tuple of a pair of brackets ($\langle, \rangle$ or $\llbracket\rrbracket$) and a number of elements from $S \cup \mathcal{D}_S$.

For example, $(\llbracket\rrbracket, \{a, (\langle\rangle, \{b, c\})\}) \in \mathcal{D}_S$ if $a, b, c \in S$. We denote this as $\llbracket a, \langle b, c\rangle\rrbracket$ for convenience. If there is no confusion, we drop S from $\mathcal{D}_S$ and write as $\mathcal{D}$.

We are interested in the greatest fixpoint of the equation because of interests to infinite executions of processes. However not all of elements in $\mathcal{D}$ make sense as the meanings of processes. For example, an infinitely nested brackets such as $\langle\llbracket\langle\llbracket\cdots\rrbracket\rangle\rrbracket$ is in $\mathcal{D}$ but we cannot say what it means. (A finitely nested brackets such as $\langle\llbracket\rrbracket\rangle$ means *no action.*)

We need more restrictions for the elements of $\mathcal{D}_S$ to define DBS.

Definition 2

For $\delta \in \mathcal{D}_S$, $\lessdot$, $\sqsubset$ and $\mathrel{-\!\!\!<}$ ($\subset S \cup \mathcal{D}_S \times \mathcal{D}_S$) and $\in^*$ ($\subset S \times \mathcal{D}_S$) are defined as follows respectively.

1. If $\delta = (\langle\rangle, S_1)$, $s \lessdot \delta$ for any $s \in S_1$.
2. If $\delta = (\llbracket\rrbracket, S_1)$, $s \sqsubset \delta$ for any $s \in S_1$.
3. If $s \sqsubset \delta$ or $s \lessdot \delta$, then $s \mathrel{-\!\!\!<} \delta$.
4. The relation $\in^* \subset S \times \mathcal{D}_S$ is defined as follows.
   (a) If $s \mathrel{-\!\!\!<} \delta$ and $s \in S$ then $s \in^* \delta$.
   (b) If $\delta' \mathrel{-\!\!\!<} \delta$ and $s \in^* \delta'$ then $s \in^* \delta$.

Example 1

If $a \in S$ and $\delta \in \mathcal{D}_S$,

1. $a \lessdot \langle\ldots, a, \ldots\rangle$, $\delta \lessdot \langle\ldots, \delta \ldots\rangle$
2. $a \sqsubset \llbracket\ldots, a, \ldots\rrbracket$, $\delta \sqsubset \llbracket\ldots, \delta \ldots\rrbracket$
3. $a \mathrel{-\!\!\!<} \langle\ldots, a, \ldots\rangle$, $\delta \mathrel{-\!\!\!<} \langle\ldots, \delta \ldots\rangle$
4. $a \mathrel{-\!\!\!<} \llbracket\ldots, a, \ldots\rrbracket$, $\delta \mathrel{-\!\!\!<} \llbracket\ldots, \delta \ldots\rrbracket$
5. $a \in^* \llbracket\ldots, a, \ldots\rrbracket$
6. $a \in^* \langle\ldots, \llbracket\ldots, \langle\ldots, a, \ldots\rangle, \ldots\rrbracket, \ldots\rangle$

For $s_1, s_2 \in^* \delta$ $(s_1, s_2) \in S)$, if the innermost pair of brackets that has both of s_1 and s_2 is "$\llbracket, \rrbracket$", then we say that s_1 and s_2 are *compatible* in δ, and if the innermost pair is "$\langle, \rangle$" then we say that s_1 and s_2 are *exclusive.*

Note that the partial order "$\prec$" of two events s_1 and s_2 makes sense only if s_1 and s_2 are compatible, because if they are exclusive then only one of them occurs. We need not to consider the causality between two exclusive events. Thus we can assume that for any s_1 and s_2, if $s_1 \prec s_2$ then they are compatible. (If not, we redefine "$\prec$" as the intersection of "$\prec$" and the compatibility.)

Definition 3

δ is a *finitely nested empty pair* if,

- $\delta \in \{(\langle\rangle, \emptyset), ([\![]\!], \emptyset)\}$ or
- for any $\delta' \prec\!\!\!\!- \ \delta$, δ' is a finitely nested empty pair.

Definition 4.

An equivalence relation $\simeq$ on $\mathcal{D}$ is defined as the smallest equivalence relation satisfying:

1. $\delta \simeq \delta$.
2. $(\langle\rangle, D) \simeq (\langle\rangle, D \setminus \delta)$ and $([\![]\!], D) \simeq ([\![]\!], D \setminus \delta)$ if δ is an finitely nested empty pair.
3. $(\langle\rangle, \{(\langle\rangle, D_1)\} \cup D_2) \simeq (\langle\rangle, D_1 \cup D_2)$ and $([\![]\!], \{([\![]\!], D_1)\} \cup D_2) \simeq ([\![]\!], D_1 \cup D_2)$.
4. $(\langle\rangle, \{\delta\}) \simeq \delta$ and $([\![]\!], \{\delta\}) \simeq \delta$ if $\delta \notin S$.
5. $(\langle\rangle, \{s\}) \simeq ([\![]\!], \{s\})$ if $s \in S$.

The second equations mean that empty process can be ignored. Namely,

$$\langle [\![]\!], \delta_1, \ldots, \delta_j, \ldots \rangle \simeq \langle \delta_1, \ldots, \delta_j, \ldots \rangle$$

and

$$[\![\langle\rangle, \delta_1, \ldots, \delta_j, \ldots]\!] \simeq [\![\delta_1, \ldots, \delta_j, \ldots]\!].$$

The equations listed in 3. can be rewritten as:

$$\langle \langle \delta_1, \delta_2, \ldots, \delta_i, \ldots \rangle, \delta'_1, \delta'_2, \ldots, \delta'_j, \ldots \rangle \simeq \langle \delta_1, \delta_2, \ldots, \delta_i, \ldots, \delta'_1, \delta'_2, \ldots, \delta'_j, \ldots \rangle$$

and

$$[\![[\![\delta_1, \delta_2, \ldots, \delta_i, \ldots,]\!], \delta'_1, \delta'_2, \ldots, \delta'_j, \ldots]\!] \simeq [\![\delta_1, \delta_2, \ldots, \delta_i, \ldots, \delta'_1, \delta'_2, \ldots, \delta'_j, \ldots]\!].$$

These equations reminds us that the parallel composition and the sum of processes are associative.

The set of equations 4. means that if a pair of brackets have one δ only the the brackets can be removed. For example, consider a DBS: $\delta = [\![\langle \ldots \rangle]\!]$. It is obvious that there is no pair of elements in δ that the outermost $[\![,]\!]$ is the innermost pair of brackets which have both of the elements in their scope, because any element of δ is at least in the scope of $\langle, \rangle$ that appear outermost but one. Thus removing the outermost pair does not affect any pair of elements in δ. Thus

$$[\![\langle \ldots \rangle]\!] \simeq \langle \ldots \rangle.$$

Intuitively, this equality means that if the number of processes composed by parallel composition is one then we need not to consider the parallel composition. We can also justify the next equation, namely

$$\langle [\![\ldots]\!] \rangle \simeq [\![\ldots]\!].$$

Intuitively, this means if the choice is only one at a branching point then it is equal to no branching there.

The equation 5. can be rewritten as,

$$\langle s \rangle \simeq [\![s]\!]$$

if s is an event. It can be regarded as a special case of the previous equations.

Not that for any events $s_1, s_2 (s_1 \neq s_2) \in^* \delta$ and for any δ' such that $\delta' \simeq \delta$, if s_1 and s_2 are compatible (exclusive) then they are also compatible (exclusive) in δ'.

We define $\hat{\mathcal{D}} \equiv \mathcal{D}_S / \simeq$.

Definition 5 (*Dual Brackets Structure, DBS*)

For $\delta \in \hat{\mathcal{D}}$, δ is a *dual bracket structure (DBS)* on S if for all $s \in^* \delta$, $s \in S$ or δ is a finitely nested empty pair.

This condition is for avoiding infinite nesting brackets without containing any events.

The set of all DBS's defined on S is denoted as $\bar{\mathcal{D}}_S (\subset \hat{\mathcal{D}})$. We drop S and denote $\bar{\mathcal{D}}$ if there is no confusion.

Definition 6 *(prefix)*

A finite set of events $\hat{\alpha} (\subset S)$ is a *prefix* of $\delta (\in \bar{\mathcal{D}})$ if :

- for any $s_1, s_2 \in^* \delta$, if $s_1 \prec s_2$ and and $s_2 \in \hat{\alpha}$, then $s_1 \in \hat{\alpha}$.
- for all $s_1, s_2 \in \hat{\alpha}$, s_1 and s_2 are compatible in δ.

Example 2

Consider a DBS $\langle [\![a, b]\!], [\![c, \langle [\![d, e]\!], [\![f, g]\!] ... \rangle, ...]\!], ... \rangle$ where $a \prec b,\ c \not\prec\!\!\not\succ d,\ c \not\prec\!\!\not\succ f,\ d \prec e,\ f \prec g$, then:

- $\{a\}$ is a prefix.
- $\{c\}, \{d\}, \{f\}, \{c, d\}$ and $\{c, f\}$ are prefixes.
- $\{b\}, \{a, b\}, \{c, e\}, \{g\}$ are not prefixes.
- $\{a, c\}, \{d, f\}$ are not prefixes.

Definition 7 (sub DBS)

Let δ be a DBS. The *sub-DBS's* of δ: $sub(\delta)$ is the largest set that is defined as follows. For any $\delta' \in sub(\delta)$, one of the followings holds.

- δ' is δ,
- $\delta = \langle \delta_1, \ldots, \delta_i, \ldots \rangle$ and $\delta' \in sub(\delta_i)$ for some i,

- $\delta = [\![\delta_1, \ldots, \delta_i, \ldots]\!]$ and $\delta' = [\![\delta'_1, \ldots, \delta'_i, \ldots]\!]$ where $\delta'_i \in sub(\delta_i)$ for all i and for any $s \in^* \delta'$, $\forall s', (s' \in^* \delta \wedge s' \notin^* \delta' \Rightarrow s \not\prec s')$, or

- $\delta = ([\![]\!], S_1)$ $(= [\![s_1, s_2, \ldots, s_i, \ldots]\!])$ for $S_1 = \{s_1, s_2, \ldots, s_i, \ldots\} \subset S$, $\delta' = ([\![]\!], S'_1)$ $(= [\![s'_1, s'_2, \ldots, s'_j, \ldots]\!])$ where $S'_1 \subset S_1$ and $\forall s'_j \sqsubseteq \delta', \forall s_i \sqsubseteq \delta, (s'_j \prec s_i \Rightarrow s_i \sqsubseteq \delta')$.

For a set of DBS's Δ, δ is *maximal* in Δ when for any $\delta' \in \Delta$ if $\delta \in sub(\delta')$ then $\delta' = \delta$.

Definition 8 Derivative of a DBS

Let δ be a DBS and $\hat{\alpha}$ be a prefix of δ. δ' is a *derivative* of δ by $\hat{\alpha}$ iff δ' is a maximal DBS in the set of DBS's that satisfy the following conditions.

- $\delta' \in sub(\delta)$,
- for all $s \sqsubseteq \delta'$, $s \notin \hat{\alpha}$,
- for all $s \sqsubseteq \delta'$ and $s' \in \hat{\alpha}$, s and s' are compatible in δ.

Intuitively, δ' should be done if we committed to perform $\hat{\alpha}$ initially. Note that δ' do not have to be done *after* $\hat{\alpha}$ but may done with $\hat{\alpha}$ in parallel.

Example 3

Let $\delta = \langle [\![a, b]\!], [\![c, \langle [\![d, e]\!], [\![f, g]\!] ... \rangle]\!], ... \rangle$ where $a \prec b$, $c \not\prec\!\not\succ d$, $c \not\prec\!\not\succ f$, $d \prec e$, $f \prec g$, as example 2.

- $[\![b]\!]$ is a derivative of δ by $\{a\}$.
- $\langle [\![d, e]\!], [\![f, g]\!] ... \rangle$ is a derivative of δ by $\{c\}$.
- $[\![e]\!]$ is a derivative of δ by $\{c, d\}$.

Definition 9 (simulation)

Let $\mathcal{R} \subset \bar{\mathcal{D}} \times \bar{\mathcal{D}}$, $\mathcal{R}$ is a *simulation* iff the following condition holds.

> For any $(\delta_1, \delta_2) \in \mathcal{R}$, if for any prefix $\hat{\alpha}$ of δ_1 and any derivative δ'_1 of δ_1 by $\hat{\alpha}$, there exists a derivative δ'_2 of δ_2 by $\hat{\alpha}'$ and $(\delta'_1, \delta'_2) \in \mathcal{R}$.

where α' is a prefix of δ_2 such that there is a one-to one mapping f from α to α' and $f(s)$ is an occurrence of the same action to s and f is order preserving.

Definition 10

The binary relation $\sqsubset \subset \bar{\mathcal{D}} \times \bar{\mathcal{D}}$ is defined as follows.

$$\sqsubset \; \equiv \; \bigcup_{i \in \mathcal{I}} \mathcal{R}_i$$

where $\{\mathcal{R}_i | \; i \in \mathcal{I}\}$ is the collection of all simulations.

Example 4

Let $a \prec b$, $c \not\prec\!\not\succ d$, $c \not\prec\!\not\succ f$, $d \prec e$, $f \prec g$, as example 2.

$$[\![a, b]\!] \sqsubset \langle [\![a, b]\!], [\![c, \langle [\![d, e]\!], [\![f, g]\!] ... \rangle]\!], ... \rangle$$

$$[\![c,d,e]\!] \sqsubset \langle [\![a,b]\!], [\![c, \langle [\![d,e]\!], [\![f,g]\!] ... \rangle]\!], ... \rangle.$$

Definition 11
The binary relation $\approx \subset \bar{\mathcal{D}} \times \bar{\mathcal{D}}$ is defined as follows.

$$\delta_1 \approx \delta_2 \text{ iff } \delta_1 \sqsubset \delta_2 \text{ and } \delta_2 \sqsubset \delta_1.$$

Obviously $\sqsubset$ is transitive and reflexive. Thus we have the following proposition.

Proposition 1
$\approx$ is an equivalence relation.

Definition 12

1. Let $\mathcal{DBS} \equiv \bar{\mathcal{D}}/\approx$.
2. Let $\sqsubseteq \subset \mathcal{DBS} \times \mathcal{DBS}$ be the partial order defined from $\sqsubset$ in standard way.

2.3 CPO of DBS

Now we can show that every $\delta \in \bar{\mathcal{D}}$ has a *normal form* wrt $\approx$.

Definition 13
Let δ be a DBS. For $\hat{\alpha} \subset S$ and a DBS: δ', if δ is $([\![]\!], \hat{\alpha} \cup \{\delta'\})$ where $\hat{\alpha}$ is a prefix of δ and δ' is the unique derivative of δ, then we denote δ as $[\![\hat{\alpha} \ ; \ \delta']\!]$. Furthermore, let $\delta = [\![\hat{\alpha} \ ; \ \delta]\!]$ for $\hat{\alpha} \subset S$ and a DBS: δ'. For any $\hat{\alpha}' \subset S$ and δ' such that $\delta = [\![\hat{\alpha}' \ ; \ \delta']\!]$, if $\alpha \subset \alpha'$ implies $\alpha = \alpha'$ then $[\![\hat{\alpha} \ ; \ \delta]\!]$ is a *maximal prefix form of* δ and denoted as $[\![\hat{\alpha} \ ;; \ \delta]\!]$.

Definition 14 (normal form)
For $\bar{\delta} \in \bar{\mathcal{D}}$ is in *normal form* if the following condition holds.

$\bar{\delta}$ is

$$\langle [\![\hat{\alpha}_1 \ ;; \ \bar{\delta}_1]\!], \ldots [\![\hat{\alpha}_i \ ;; \ \bar{\delta}_i]\!], \ldots \rangle.$$

(or $[\![\hat{\alpha}_1 \ ;; \ \bar{\delta}_1]\!]$ if $\bar{\delta} = \langle [\![\hat{\alpha}_1 \ ;; \ \bar{\delta}_1]\!] \rangle$.) and each $\bar{\delta}_i$ is also in normal form.

Proposition 2
For all $\delta \in \bar{\mathcal{D}}$, there exists a DBS $\bar{\delta}$ such that

$$\bar{\delta} \approx \delta$$

and is in *normal form*

Now we can assume that every $\delta \in \mathcal{DBS}$ is in normal form. We define the GLB operation on $\mathcal{DBS}$.

Definition 15 (GLB operation)
Let $\delta_1, \delta_2 \in \mathcal{DBS}$ (and they are normal form). The binary operation $\sqcap : \mathcal{DBS} \times \mathcal{DBS} \rightarrow \mathcal{DBS}$ is defined as follows.

1. $\delta_1 \sqcap \delta_2$ is a normal form of $\langle \ldots, \delta_{1i} \sqcap \delta_{2j}, \ldots \rangle$ where $\delta_k \equiv \langle \delta_{k1}, \delta_{k2}, \ldots \delta_{kl}, \ldots \rangle$.

2. $\delta_1 \sqcap \delta_2$ is a normal form of $[\![\hat{\alpha}_1 \cap \hat{\alpha}_2,\ ;;\ \delta'_1 \sqcap \delta'_2]\!]$ if $\delta_k \equiv [\![\hat{\alpha}_k\ ;;\ \delta'_k]\!]$.

Proposition 3
For any $\delta_1, \delta_2 \in \mathcal{DBS}$,

1. $\delta_1 \sqcap \delta_2 \sqsubseteq \delta_1$ and $\delta_1 \sqcap \delta_2 \sqsubseteq \delta_2$.
2. If $\delta \sqsubseteq \delta_1$ and $\delta \sqsubseteq \delta_2$ then $\delta \sqsubseteq \delta_1 \sqcap \delta_2$.

Now we can show that $\mathcal{DBS}$ is a CPO with $\sqsubseteq$.

Proposition 4

1. For any directed subset Δ of $\mathcal{DBS}$, there exists a GLB of Δ.

   Proof: (outline)

   $\langle \delta_1, \delta_2, \ldots, \delta_i, \ldots \rangle$ is the GLB of Δ where $\delta_1, \delta_2, \ldots, \delta_i, \ldots$ is the collection of all lower bounds of Δ.

2. $\mathcal{DBS}$ has the greatest element.

   Proof: (outline)

   $\langle [\![\alpha_1\ ;;\ \delta_1]\!], \ldots, [\![\alpha_i\ ;;\ \delta_j]\!], \ldots \rangle$ is the maximum element of $\mathcal{DBS}$ where $\alpha, \ldots \alpha_i, \ldots$ is the collection of all subset of S and $\delta_1, \delta_2, \ldots, \delta_i, \ldots$ is the collection of all DBS's.

3 Semantics of Mobile Processes

This sections presents a set of rules that maps mobile processes to a DBS. Processes are given using a subset of π-calculus [Mil89a].

3.1 Syntax

This section presents the syntax of π-calculus. In this paper, we adopt a subset that consists of nill, prefix, sum, composition, hiding and constants.

Let $\mathcal{N}$ be a set of *names*, and use $x, y, z, u, v, w, \ldots$ for names. We denote processes using meta-variables $P, Q, R, \ldots$. *An action* is *a silent action* denoted as τ , *a free output action* $\overline{x}y$, *a bound output action* $\overline{x}(y)$ or an input action $x(y)$.

The set of processes are defines as follows.

nill $\mathbf{0}$

prefix $\overline{x}y.P, \quad x(y).P, \quad \tau.P$

sum $P + Q$

composition $P|Q$

hiding $(x)P$

constant $A(y_1, \ldots, y_n)$

We omit replication !, but use defining equations:

$$A(y_1, \ldots, y_n) \stackrel{\text{def}}{=} P$$

for recursive definitions. We also rule out *match* $[x = y]P$ operations because, we consider the match operation is introduced with the motivation to maintain the validity of the expansion rule in the framework of interleaving approach.

The intuitive operational semantics of processes is similar to the conventional one [Mil89a]. $\mathbf{0}$ is a process of *no-action.* $\overline{x}y.P$ outputs y using a port x first and then behaves like P. $\tau.P$ is similar to that of CCS. $x(z).P$ inputs a name y using the port x first then behaves like $P\{y/z\}$ where $P\{y/z\}$ is the process that is obtained form P by replacing all z with y. $P + Q$ behaves like P or like Q. $(x)P$ behaves like P but does not use the port x for communications with outside. $A(y_1, \ldots y_n)$ behaves like P if $A(y_1, \ldots, y_n) \stackrel{\text{def}}{=} P$.

As the purpose of this paper is to define a semantics that distinguish $a|b$ and $a.b + b.a$, we will define a slightly different semantics for $P|Q$ from the conventional semantics. $P|Q$ is concurrent execution of P and Q. Thus $P|Q$ may behave like the interleaving of P and Q, and actions in P and actions in Q may arise simultaneous.

We introduce the conventional syntactic "$\equiv$" relation define as follows and we identify two processes P and Q if $P \equiv Q$.

- If P is $\alpha-$convertible to Q, then $P \equiv Q$.
- $P|0 \equiv P$、$P|Q \equiv Q|P$、$P|(Q|R) \equiv (P|Q)|R$
- $(x)0 \equiv 0$、$(x)(y)P \equiv (y)(x)P$
- If x is bound in P then $(x)(P|Q) \equiv P|(x)Q$

We alos use the notion of *free names* and *boud names* as usual. Notations such as $fn(P)$, $bn(P)$, $fn(\alpha)$ and $bn(\alpha)$ are used in conventional manner.

We define the syntactic derivative of processes.

Definition 16

For an action α and a process P, the *syntactic derivative of P by α* is a set of processes defined as follows and denoted as P/α.

1. $\mathbf{0} \;/\; \alpha = \emptyset$
2. $x(z).P \;/\; \alpha = \begin{cases} \{P\{y/z\}\} & \text{if } \alpha = x(y) \\ \emptyset & \text{otherwise} \end{cases}$
3. $\overline{x}z.P \;/\; \alpha = \begin{cases} \{P\} & \text{if } \alpha = \overline{x}z \\ \emptyset & \text{otherwise} \end{cases}$
4. $\tau.P \;/\; \alpha = \begin{cases} \{P\} & \text{if } \alpha = \tau \\ \emptyset & \text{otherwise} \end{cases}$
5. $(P + Q) \;/\alpha = P \;/\; \alpha \cup Q \;/\; \alpha$

$$6.\ (x)P\ /\ \alpha = \begin{cases} \emptyset & \text{if } \alpha = x(y) \text{ or } \overline{x}y \\ \{P'|P' \in P\ /\alpha\} & \text{if } \alpha = \overline{y}(x) \\ \{(x)P'|P' \in P\ /\alpha\} & \text{otherwise} \end{cases}$$

$$7.\ (P|Q)\ /\alpha = \begin{cases} \{P'|Q\ |P' \in P/\alpha\} \cup \{P|Q'\ |Q' \in Q/\alpha\} & \text{if } \alpha \neq \tau \\ \\ \{P'|Q\ |P' \in P/\tau\} \cup \{P|Q'\ |Q' \in Q/\tau\} \cup & \\ \{P'|Q'\ |P' \in P/\overline{x}y, Q' \in Q/x(y), y \in \mathcal{N} \setminus bn(Q)\} \cup & \\ \{(w)(P'|Q')\ |P' \in P/\overline{x}(w), Q' \in Q/x(w)\} & \text{if } \alpha = \tau \end{cases}$$

This syntactic derivative is defined in the manner that is very similar to the rules of labeled transition system in the interleave semantics of π-calculus. Namely $P' \in P/\alpha$ is almost equivalent to $P \xrightarrow{\alpha} P'$. However the intuitive meaning of the syntactic derivative defined here is slightly different from the notion of derivative in interleaving semantics. $P \xrightarrow{\alpha} P'$ means that P becomes P' *after* α in the interleaving semantics. On the other hand, $P' \in P/\alpha$ means that if you decided to do α then P' my be a process that should be done. There is no requirement for the order of α and P' in this notation. Sometimes actions in P' can be done before α.

3.2 Semantic mapping

In the following definition, the subscript of an event (for example α of $init_\alpha$) denotes the label of the event.

Definition 17

Let S be a partially ordered set such that each element is labeled with an action on $\mathcal{N}$. $\lceil \cdot \rceil$ is a function form the set of processes on $\mathcal{N}$ to $\mathcal{DBS}$ defined as follows.

- nill : $\lceil \mathbf{0} \rceil = [\![]\!]\ \ (= \langle\rangle)$
- sum: $\lceil P + Q \rceil = \langle \lceil P \rceil, \lceil Q \rceil \rangle$
- prefix:
  - $\lceil \overline{x}y.P \rceil = [\![\mathit{first}_{\overline{x}y},\ \lceil P \rceil]\!]$
    where $\mathit{first}_{\overline{x}y}$ is an event such that for any $s \in^* \lceil P \rceil, \mathit{first}_{\overline{x}y} \prec s$.
  - $\lceil \tau.P \rceil = [\![\mathit{first}_\tau,\ \lceil P \rceil]\!]$
    where first_τ is an event such that for any $s \in^* \lceil P \rceil, \mathit{first}_\tau \prec s$.
  - $[\![\mathit{first}_{x(v)},\ \lceil P\{v/y\} \rceil]\!] \lessdot \lceil x(y).P \rceil$
    where $\mathit{first}_{x(v)}$ is the event such that for all $v \in \mathcal{N} \setminus bn(P)$ and for any $s \in^* \lceil P\{v/y\} \rceil, \mathit{first}_{x(y)} \prec s$.
- hiding:
  - $[\![init_\alpha, \lceil P' \rceil]\!] \prec\!\!\!\!\cdot\ \lceil (x)P \rceil$
    where $P' \in (x)P/\alpha$, the subject of α is not x and $init_\alpha$ is an event such that for any $s \in^* \lceil P' \rceil, s \not\prec init_\alpha$.
  - $[\![init_{\overline{x}(y)},\ \lceil P' \rceil]\!] \prec\!\!\!\!\cdot\ \lceil (y)P \rceil$
    where $P' \in P/\overline{x}(y)$ and $init_{\overline{x}(y)}$ is an event such that for any $s \in^* \lceil P' \rceil, s \not\prec init_{\overline{x}(y)}$ and for any $s' \in^* \lceil P' \rceil$ if s' is labeled with an action using y as the subject then $init_{\overline{x}(y)} \prec s'$.

- defining equation

  $\lceil P \rceil \sim \lceil E \rceil$ if $P \stackrel{\text{def}}{=} E$

- composition:

  - $[\![init_\alpha, \lceil P'|Q \rceil]\!] \lessdot \lceil P|Q \rceil$ where $P' \in P/\alpha$ where $init_\alpha$ is an event such that for any $s \in^* \lceil P'|Q \rceil, s \not\lessdot init_\alpha$.
  - $[\![init_\alpha, \lceil P|Q' \rceil]\!] \lessdot \lceil P|Q \rceil$ where $Q' \in Q/\alpha$ where $init_\alpha$ is an event such that for any $s \in^* \lceil P|Q' \rceil, s \not\lessdot init_\alpha$.
  - $[\![init_\tau, \lceil P'|Q' \rceil]\!] \lessdot \lceil P|Q \rceil$ where $P' \in P/\overline{\alpha}, Q' \in Q/\alpha$ and $init_\tau$ is an event such that for any $s \in^* \lceil P'|Q' \rceil, s \not\lessdot init_\tau$.
  - $[\![init_\tau, \lceil (w)(P'|Q') \rceil]\!] \lessdot \lceil P|Q \rceil$ where $P' \in P/\overline{x}(w), Q' \in Q/x(w)$ and $init_\tau$ is an event such that for any $s \in^* \lceil P'|Q' \rceil, s \not\lessdot init_\tau$.

4 Conclusion

This paper presented a domain of Dual Bracket Structures that are collections of partially ordered events and provide branching structures of processes denoted with two types of brackets, and showed the domain $\mathcal{DBS}$ forms a CPO. Furthermore, a semantic mapping from a subset of π-calculus to the domain is presented. DBS models concurrent computations in truly concurrent manner. One of the motivations to adopt the truly concurrent approach to semantics of mobile processes is that we consider that works to obtain congruence relations of processes wrt operators including input prefixing. Thus, there remain discussions for congruence property of relations defined by the semantics as future works.

Another future work is a weak equivalence version of the DBS semantics that ignores occurrence of τ actions.

Acknowledgement: The author gratefully thank Professor Yamasaki, Mr. Toru Kato, Mr. Yukihiro Oda and all members in his laboratory for their support and encouragements.

References

[Amadio] Amadio R. M. , On the Reduction of Chocs Bisimulation to π-calculus Bisimulation, CONCUR'93, Lecture Notes in Computer Science 715 (1993) pp. 112-126

[Boreal92] Boreale M. and R. De Nicola , Testing Equivalence for Morbile Processes, CONCUR'92, Lecture Notes in Computer Science 630 (1992) pp. 2-16

[Boreal94] Boreale M. and R. De Nicola, A Symbolic Semantics for π-calculus, CONCUR'94, Lecture Notes in Computer Science 836 (1994) pp. 299-314

[Boreal95] Boreal M. and D. Sangiorgi, A fully abstract semantics for causality in the π-calculus, Proc. of STACS '95, Lecture Notes in Comp. Sci. 900, (1995) pp. 243-254

[Boreal96] Boreal M. and D. Sangiorgi, Some Congruence Properties for π-calculus Bisimularities, Tech. Rep. RR-2870, INRIA-Sophia Antipolis, `ftp://ftp.dcs.ed.ac.uk/pub/sad/congruence96.ps.gz` (1996)

[Boud87] Boudol G. and Castellani I, On the Semantics of Concurrency: Partial Orders and Transition Systems, TAPSOFT 87, Vol. I, Lecture Notes in Computer Science 249 (1987) pp. 123-137

[Dega88] Degano P., R. De Nicola and U. Montanari, On the Consistency of "Truly Concurrent" Operational and Denotational Semantics, Proc. of LICS'88, IEEE Computer Society Press (1988) pp. 133-141

[Deg95] Degano P. and C. Proami, Causality for Mobile Processes, Proc. of ICALP' 95, Lecture Notes in Comp. Sci. 944, (1995) pp. 660-671

[Busi95] Busi N. and R. Gorrieri, A Petri Semantics for π-calcurus, CONCUR'95, Lecture Notes in Computer Science 962 (1995) pp. 145-159

[Enge93] Engelfriet J., A Multiset Semantics for the pi-calculus with Replication, CONCUR'93, Lecture Notes in Computer Science 715 (1993) pp. 7-21

[deBakk89] de Bakker J. W., W. -P. de Rover and G. Rozenberg, editors, *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, Lecture Notes in Computer Science 354 (1989)

[Ferr95] Ferrari G-L., U. Montanari and P. Quaglia, The Weak Late π-calculus Semantics as Observation Equivalence CONCUR'95, Lecture Notes in Computer Science 962 (1995) pp. 57-71

[Fior96] Fiore M. P., E. Moggi, and D. Sangiorgi, A Fully-Abstract Model for the π-calculus, Proc. of 11th Annual IEEE Symp. on Logic in Computer Sci. (LICS'96), (1996) pp 43-55

[Gaif87] Gaifman H., and V. Pratt, Partial Order Models of Concurrency and the Computation of Functions, Proc. of LICS'87, IEEE Computer Society Press (1987) pp72-85

[Henn92] Hennessy M., Concurrent Testing of Processes, CONCUR'92, Lecture Notes in Computer Science 630 (1992) pp. 94-107

[Kiehn94] Kiehn A. and M. Hennessy, On the Decidability of Non-Interleaving Process Equivalences, CONCUR'94, Lecture Notes in Computer Science 836 (1994) pp. 18-33

[Liu94] Liu X., Characterizing Bisimulation Congruence in the π-calculus,CONCUR'94, Lecture Notes in Computer Science 836 (1994) pp. 331-350

[Mil89a] Milner R., J. Parrow and D. Walker, A Calculus of Morbile Processes, Part1, Part2, LFCS Report Series, ECS-LFCS-89-85 and 86, University of Edinburgh, (1989)

[Mil89] Milner R. , *Communication and Concurrency*, Prentice Hall (1989)

[Mil92] Milner R. , Functions as Processes, *Mathematical Structure in Computer Science*, vol. 2, (1992) pp. 119-141

[Mont5] Montanari U. and M. Pistore, Checking Bisimilarity for Finitely π-calculus, CONCUR'95, Lecture Notes in Computer Science 962 (1995) pp. 42-56

[Mont95] Montanari U. and M. Pistore, Concurrent Semantics for the π-calculus, Electronic Notes in Theoretical Computer Science 1, Elsiver Science B. V. (1995)

[NPW81] Nielsen M., G. Plotkin, and G. Winskel, Petri Nets, Event Structures and Domains, Part 1. Theoretical Computer Science, 13, (1981) pp. 85-108

[NSW93] Nielsen M. V. Sassone and G. Winskel, Relationship between Models of Concurrency, J.W. de Bakker, W.-P. de Rover and G. Rozenberg (Eds.), *A Decade of Concurrency: Reflections and Perspectives*, Lecture Notes in Computer Science 803 (1993) pp. 425-476

[Oda] Oda, Y. and M. Murakami, Multi-action π-calculus, RIMS Workshop in Computing, Concurrency Theory and Applications '96, (1996)

[Parr93] Parrow J. and D. Sangiorgi, Algebraic Theories for Name-Passing Calculi, J.W. de Bakker, W.-P. de Rover and G. Rozenberg (Eds.), *A Decade of Concurrency: Reflections and Perspectives*, Lecture Notes in Computer Science 803 (1993) pp. 509-529

[Rens92] Rensink A. Posets for Configurations!, CONCUR'92, Lecture Notes in Computer Science 630 (1992) pp. 269-285

[Ren95] Rensink A. A Complete Theory of Deterministic Event Structure, CONCUR'95, Lecture Notes in Computer Science 962 (1995) pp. 160-174

[Sang93] Sangiorgi D., A Theory of Bisimulation for π-calculus, CONCUR'93, Lecture Notes in Computer Science 715 (1993) pp. 127-142

[Sang94] Sangiorgi D., Locality and True-Concurrency in Calculi for Mobile Processes, TACS'94, Lecture Notes in Computer Science 789 (1994) pp. 405-424

[Stark96] Stark I., A Fully Abstract Domain Model for the π-calculus, Proc. of 11th Annual IEEE Symp. on Logic in Computer Sci. (LICS'96), (1996) pp 36-42

[Walk94] Walker D., On Bisimulation in the π-calculus, CONCUR'94, Lecture Notes in Computer Science 836 (1994) pp. 315-330

[Zwier93] Zwiers J. and W. Janssen, Partial Order Based Design of Concurrent Systems, J.W. de Bakker, W.-P. de Rover and G. Rozenberg (Eds.), *A Decade of Concurrency: Reflections and Perspectives*, Lecture Notes in Computer Science 803 (1993) pp. 622-684