A Modal Proof System for Mobile Processes

ATSUSHI TOGASHI† and FUMIAKI KANEZASHI††

To promote rapid development of computer systems, concurrent / parallel processing is one of the promising research items. For the mathematical description for concurrent processes, there have been several proposal as process calculi, formal systems for concurrent processes, such as CCS, CSP, ACP, LOTOS, π-calculus. In this paper, we will propose a modal proof system for the π-calculus, introduced by Milner, Parrow and Walker in 1989. π-calculus is an extension of CCS or CSP, to express mobile agents. Purpose in this paper is to offer a proof system for the π-calculus based on propositional modal logics. A partial soundness and completeness is discussed. An extension of the resulting system is also considered.

1. Introduction

In this paper, we will propose a modal proof system for the π-calculus, one of the promising calculus for mobile processes, introduced by Milner, Parrow and Walker in 1989. π-calculus is an extension of CCS or CSP, to express mobility for agents. The main purpose in this paper is to offer a rigorous proof system for the π-calculus based on propositional modal logics.

The *π-calculus*[13] has achieved a remarkable simplification by focusing on naming and allowing the communicated data along channels (names) to be names themselves. The calculus is sufficiently expressive to describe mobile systems and the ability of natural embeddings of both lazy and call-by-value λ-calculi into the π-calculus[11] suggests that it may form an appropriate foundation for the design of new programming languages. It has been shown that higher-order processes can be faithfully encoded in the π-calculus[15]. The *polyadic π-calculus* by Milner[12] is a straightforward generalization of the monadic π-calculus[13], in which finite tuples of names, instead of single names, are the atomic unit of communication. Furthermore, the fact that a tuple of names is exchanged at each communication step suggests a natural discipline of sorting.

Let us consider the following processes:

$$\begin{aligned} A &\overset{\text{def}}{=} \bar{l}C.a(x).0 \\ B &\overset{\text{def}}{=} l(X).(X|m(z).\overline{m}msg(z).0) \\ C &\overset{\text{def}}{=} \overline{m}r.m(y).\bar{a}y.0 \end{aligned}$$

In this example, a process A communicates with a process B along a logical link l, a process C is transmitted from A to B. The entire behavior of the communication is expressed as the following transitions:

$$\begin{aligned} & A|B \\ = \;& \bar{l}C.a(x) \mid l(X).(X|m(z).\overline{m}msg(z)) \\ \longrightarrow \;& a(x) \mid (C|m(z).\overline{m}msg(z)) \\ = \;& a(x) \mid \overline{m}r.m(y).\bar{a}y \mid m(z).\overline{m}msg(z) \\ \longrightarrow \;& a(x) \mid m(y).\bar{a}y \mid \overline{m}msg(r) \\ \longrightarrow \;& a(x) \mid \bar{a}msg(r) \\ \longrightarrow \;& \mathbf{0} \end{aligned}$$

In this paper, we will propose a proof system for mobile agents based on the propositional modal logic. The properties of processes are expressed as formulae. As an example, let us consider the problem that the process

$$P = \bar{x}y.Q + \bar{w}y.R$$

satisfies the property

† Department of Computer Science, Shizuoka University
†† Graduate school of Science and Engineering, Shizuoka University

$A = \langle \bar{x}y \rangle true \wedge [\bar{z}y]\, false.$

This can be proved via the proof

$$\frac{\dfrac{\dfrac{P \vdash true}{\bar{x}y.P \vdash \langle \bar{x}y \rangle true}}{\bar{x}y.P + \bar{w}y.Q \vdash \langle \bar{x}y \rangle true} \qquad \dfrac{\bar{x}y.P \vdash [\bar{z}y]\, false \quad \bar{w}y.Q \vdash [\bar{z}y]\, false}{\bar{x}y.P + \bar{w}y.Q \vdash [\bar{z}y]\, false}}{\bar{x}y.P + \bar{w}y.Q \vdash \langle \bar{x}y \rangle true \wedge [\bar{z}y]\, false}$$

The proof is constructed by inference rules.

The outline of the paper is as follows: Section 2 presents the monadic π-calculus to a certain extent needed for the study. Section 3 introduces a modal proof system system and discusses sound and completeness of the system. An extension of the resulting system is considered in Section 4. This paper is concluded in Section 5.

2. π-calculus

This section introduces the π-calculus[13] to a certain extent needed for the study.

Let $\mathcal{N}$ be a possibly infinite set of *names* throughout this paper. The basic syntactic categories of the π-calculus we consider in this paper are defined by the following definition:

Definition2.1 A *process* in the $\pi - calculus$ is defined by the following grammar:

$$\begin{array}{rcll}
P & ::= & \mathbf{0} & \textit{inaction} \\
 & \mid & \tau.P & \textit{silent prefix} \\
 & \mid & x(y).P & \textit{input prefix} \\
 & \mid & \bar{x}y.P & \textit{output prefix} \\
 & \mid & P + P & \textit{summation} \\
 & \mid & P|P & \textit{composition} \\
 & \mid & [x = y]P & \textit{match} \\
 & \mid & (\nu x)P & \textit{restriction} \\
 & \mid & !P & \textit{replication}
\end{array}$$

Let $\mathcal{P}$ denote the set of all processes. □

We use the metavariables P, Q, and R for processes; a, b, c, x, y, z, etc for names.

Following Milner[13], we adopt the replication operator ! instead of allowing systems of recursive definitions of process expressions; $!P$ stands for the parallel composition of infinite number of copies of P. We don't use the notational devices of *abstractions* and *concretions*[13] to simplify the discussion.

Definition2.2 For a process P, the set $fn(P)$ of names with free occurrences (simply called *free names*) and the set $bn(P)$ of names with bound occurrences (*bound names*) are defined in the usual way as follows:

P	$fn(P)$	$bn(P)$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$
$\tau.P'$	$fn(P')$	$bn(P')$
$x(y).P'$	$(fn(P') \cup \{x\}) - \{y\}$	$bn(P') \cup \{y\}$
$\bar{x}y.P'$	$fn(P') \cup \{x, y\}$	$bn(P')$
$P_1 + P_2$	$fn(P_1) \cup fn(P_2)$	$bn(P_1) \cup bn(P_2)$
$P_1 \mid P_2$	$fn(P_1) \cup fn(P_2)$	$bn(P_1) \cup bn(P_2)$
$[x = y]P'$	$fn(P') \cup \{x = y\}$	$bn(P')$
$(\nu x)P'$	$fn(P') - \{x\}$	$bn(P') \cup \{x\}$
$!P'$	$fn(P')$	$bn(P')$

□

We formally identify processes P up to renaming bound names in P, so that it is assumed that $fn(P) \cap bn(P) = \emptyset$. This implies the usual conventions about substitutions to avoid capturing free names during substitution, α-conversion, side-condition concerning freshness of names, etc. For example, the substitution $\sigma = \{y/x\}$ on a process P, denoted by $P\sigma$, is defined in the usual way.

$$\begin{array}{lcl}
\mathbf{0}\{y/x\} & \stackrel{\text{def}}{=} & \mathbf{0} \\
(\tau.P)\{y/x\} & \stackrel{\text{def}}{=} & \tau.P\{y/x\} \\
(x(y).P')\{y/x\} & \stackrel{\text{def}}{=} & x(y).P'\{y/x\} \\
(\bar{x}y.P')\{y/x\} & \stackrel{\text{def}}{=} & fn(P') \cup \{x, y\} \\
(P_1 + P_2)\{y/x\} & \stackrel{\text{def}}{=} & P_1\{y/x\} + P_2\{y/x\} \\
(P_1|P_2)\{y/x\} & \stackrel{\text{def}}{=} & P_1\{y/x\}|P_2\{y/x\} \\
([x = y]P')\{y/x\} & \stackrel{\text{def}}{=} & [x = y]P'\{y/x\} \\
((x)P')\{y/x\} & \stackrel{\text{def}}{=} & (x)P'\{y/x\} \\
(!P')\{y/x\} & \stackrel{\text{def}}{=} & !P'\{y/x\}
\end{array}$$

The substitution is applied only to the free names. The substitution does not modify bound names; to avoid that a name free in P become bound in $P\sigma$ we assume that the bound names of P have been previously α-converted to fresh names, so that $bn(P) \cap \sigma(fn(P)) = \emptyset$.

Following Milner[13], we present the operational semantics of the calculus using two relations: a *structural congruence* on processes that permits the rearrangement of summations, parallel compositions, replications, and restriction so that the participants in a potential communication can be brought into immediate proximity; and a *reduction relation* that describes the act of communication itself.

Definition2.3 We define a *structural congruence relation* $\equiv$ to be the smallest congruence relation over processes which satisfies the axiom schemes listed below.

1. If $P \equiv_\alpha Q$ then $P \equiv Q$: Processes are identified if they differ only by a change of bound names.
2. $M + (N + L) \equiv (M + N) + L$.
   $M + N \equiv N + M$.
   $M + \mathbf{0} \equiv M$.
3. $P \mid (Q \mid R) \equiv (P \mid Q) \mid R$.
   $P \mid Q \equiv Q \mid P$.
   $P \mid \mathbf{0} \equiv P$.
4. $[x = x]P \equiv P$.
5. $(\nu x)P \equiv P \quad$ if $x \notin fn(P)$.
   $(\nu x)(\nu y)P \equiv (\nu y)(\nu x)P$.
   $(\nu x)P \mid Q \equiv (\nu x)(P \mid Q) \quad$ if $x \notin fn(Q)$.
6. $!P \equiv !P \mid P$.

$\square$

Note that the first axiom scheme in 5. induces the usual axiom schemes:

$$(\nu x)\mathbf{0} \equiv \mathbf{0}$$
$$(\nu x)(\nu x)P \equiv (\nu x)P$$

Note also that the side condition of the last axiom scheme in 5. can be viewed as a consequence of our convention of regarding bound names.

labels	Kind	$fn(\alpha)$	$bn(\alpha)$	
τ	silent	$\emptyset$	$\emptyset$	L, E
xy	free input	$\{x, y\}$	$\emptyset$	E
$x(y)$	bound input	$\{x\}$	$\{y\}$	L
$\bar{x}y$	free output	$\{x, y\}$	$\emptyset$	E,L
$\bar{x}(y)$	bound output	$\{x\}$	$\{y\}$	E,L

Definition2.4 The set of inference rules LATE (*late instantiation*) consists of the following:

TAU $\quad \dfrac{}{\tau.P \xrightarrow{\tau} P}$

L-INPUT $\quad \dfrac{}{x(y).P \xrightarrow{x(y)} P}$

OUTPUT $\quad \dfrac{}{\bar{x}y.P \xrightarrow{\bar{x}y} P}$

SUM $\quad \dfrac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$

PAR $\quad \dfrac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q} \; bn(\alpha) \cap fn(Q) = \emptyset$

L-COM $\quad \dfrac{P \xrightarrow{\bar{x}y} P' \quad Q \xrightarrow{x(z)} Q'}{P|Q \xrightarrow{\tau} P'|Q'\{y/z\}}$

RES $\quad \dfrac{P \xrightarrow{\alpha} P'}{(\nu x)P \xrightarrow{\alpha} (\nu x)P'} \; x \notin n(\alpha)$

REST $\quad \dfrac{P \xrightarrow{\bar{x}y} P'}{(\nu x)P \xrightarrow{\bar{x}(y)} (\nu x)P'} \; x \neq y$

STRUCT $\quad \dfrac{P' \equiv P \quad P \xrightarrow{\alpha} Q \quad Q \equiv Q'}{P' \xrightarrow{\alpha} Q'}$

$\square$

We write $P \xrightarrow{\alpha}_L Q$ to mean that the transition $P \xrightarrow{\alpha} Q$ can be inferred from LATE.

Definition2.5 The set of rule EARLY (*early instantiation*) is obtained from LATE by replacing the rules L-INPUT and L-COM with the following two rules:

E-INPUT $\quad \dfrac{}{x(y).P \xrightarrow{xw} P\{w/y\}}$

E-COM $\quad \dfrac{P \xrightarrow{\bar{x}y} P' \quad Q \xrightarrow{xy} Q'}{P|Q \xrightarrow{\tau} P'|Q'}$

$\square$

We write $P \xrightarrow{\alpha}_E Q$ to mean that the transition $P \xrightarrow{\alpha} Q$ can be inferred from EARLY.

Lemma2.1

(1) $P \xrightarrow{xz}_E P'$ iff $P \xrightarrow{x(y)}_L P''$ for some P'' such that $P' = P''\{z/y\}$.
(2) $P \xrightarrow{\bar{x}y}_E P'$ iff $P \xrightarrow{\bar{x}y}_L P'$.
(3) $P \xrightarrow{\bar{x}(y)}_E P'$ iff $P \xrightarrow{\bar{x}(y)}_L P'$.
(4) $P \xrightarrow{\tau}_E P'$ iff $P \xrightarrow{\tau}_L P'$.

Proof: A standard induction over LATE and EARLY. □ □

Definition2.6 A binary relation $\mathcal{S}$ on $\mathcal{P}$ is a *late simulation* if $P\mathcal{S}Q$ implies that

1. if $P \xrightarrow{\alpha} P'$ for some $\alpha = \tau$, $\bar{x}y$, or $\bar{x}(y)$ with $y \notin f(P,Q)$, then for some Q', $Q \xrightarrow{\alpha} Q'$ and $P'\mathcal{S}Q'$;
2. if $P \xrightarrow{x(y)} P'$ and $y \notin f(P,Q)$, then for some Q', $Q \xrightarrow{x(y)} Q'$ and for all z, $P'\{z/y\}\mathcal{S}Q'\{z/y\}$.

The relation $\mathcal{S}$ is a *late bisimulation* if both $\mathcal{S}$ and $\mathcal{S}^{-1}$ are late simulations. We define *late bisimilarity* $P \dot{\sim}_L Q$ to mean that $P\mathcal{S}Q$ for some late bisimulation $\mathcal{S}$. □

Definition2.7 A binary relation $\mathcal{S}$ on $\mathcal{P}$ is an *early simulation* if $P\mathcal{S}Q$ implies that
if $P \xrightarrow{\alpha} P'$ for some $\alpha = \tau$, xy, $\bar{x}y$, or $\bar{x}(y)$ with $y \notin f(P,Q)$, then for some Q', $Q \xrightarrow{\alpha} Q'$ and $P'\mathcal{S}Q'$;
The relation $\mathcal{S}$ is an *early bisimulation* if both $\mathcal{S}$ and $\mathcal{S}^{-1}$ are early simulations. We define *early bisimilarity* $P \dot{\sim}_E Q$ to mean that $P\mathcal{S}Q$ for some early bisimulation $\mathcal{S}$. □

Definition2.8 A binary relation $\mathcal{S}$ on $\mathcal{P}$ is an *alternative simulation* if $P\mathcal{S}Q$ implies that

1. if $P \xrightarrow{\alpha} P'$ for some $\alpha = \tau$, $\bar{x}y$, or $\bar{x}(y)$ with $y \notin f(P,Q)$, then for some Q', $Q \xrightarrow{\alpha} Q'$ and $P'\mathcal{S}Q'$;
2. if $P \xrightarrow{x(y)} P'$ and $y \notin f(P,Q)$, then for any z, there is Q' such that $Q \xrightarrow{x(y)} Q'$ and $P'\{z/y\}\mathcal{S}Q'\{z/y\}$.

The relation $\mathcal{S}$ is an *alternative bisimulation* if both $\mathcal{S}$ and $\mathcal{S}^{-1}$ are alternative simulations. We define *alternative bisimilarity* $P \dot{\sim}' Q$ to mean that $P\mathcal{S}Q$ for some alternative bisimulation $\mathcal{S}$. □

3. Modal Proof System

In this section we introduce a modal proof system for mobile processes based on the modal logic introduced by Milner-Parrow-Walker92[13]. Before stating the proof system we introduce the modal logic.

Definition3.1 Let $\mathcal{L}$ denote the set of all formulae defined by the following grammar:

$$\begin{array}{lcll}
A & ::= & \textbf{true} & \\
 & \mid & \textbf{false} & \\
 & \mid & A \wedge B & \\
 & \mid & A \vee B & \\
 & \mid & \langle\alpha\rangle A & \alpha = \tau,\ xy,\ x(y),\ \bar{x}y, \text{ or } \bar{x}(y) \\
 & \mid & [\alpha]\,A & \alpha = \tau,\ xy,\ x(y),\ \bar{x}y, \text{ or } \bar{x}(y) \\
 & \mid & \langle x(y)\rangle^{l} A & l = L, \text{ or } E \\
 & \mid & [x(y)]^{l} A & l = L, \text{ or } E \\
 & \mid & [x=y]A &
\end{array}$$
□

In each of $\langle\bar{x}(y)\rangle A, \langle x(y)\rangle A, \langle x(y)\rangle^{L} A, \langle x(y)\rangle^{E} A$, the occurrence of y in parentheses is a binding occurrence whose scope is A. The set of names occurring free in A is written $fn(A)$.

Definition3.2 The satisfaction relation between processes and formulae is defined by structural induction as follows:

1. $P \models \textbf{true}$, for all P.
2. $P \models A \wedge B$
   $\Leftrightarrow\ P \models A$ and $P \models B$.
3. $P \models A \vee B$
   $\Leftrightarrow\ P \models A$ or $P \models B$.
4. $P \models \langle\alpha\rangle A$, $(\alpha = \tau, xy, \bar{x}y, \bar{x}(y))$
   $\Leftrightarrow\ \exists P'. (P \xrightarrow{\alpha} P' \ \&\ P' \models A)$.
5. $P \models [\alpha]\,A$, $(\alpha = \tau, xy, \bar{x}y, \bar{x}(y))$
   $\Leftrightarrow\ \forall P'. (P \xrightarrow{\alpha} P' \Rightarrow P' \models A)$.
6. $P \models \langle x(y)\rangle A$
   $\Leftrightarrow\ \exists P'. (P \xrightarrow{x(y)} P' \ \&\ \exists z.\ P'\{z/y\} \models A\{z/y\})$.
7. $P \models [x(y)]\,A$
   $\Leftrightarrow\ \forall P'. (P \xrightarrow{x(y)} P' \Rightarrow \forall z.\ P'\{z/y\} \models A\{z/y\})$.
8. $P \models \langle x(y)\rangle^{L} A$
   $\Leftrightarrow\ \exists P'. (P \xrightarrow{x(y)} P' \ \&\ \forall z.\ P'\{z/y\} \models A\{z/y\})$.

9. $P \models [x(y)]^L A$
$\Leftrightarrow \quad \forall P'.\ (P \overset{x(y)}{\rightarrow} P' \Rightarrow \exists z.\ P'\{z/y\} \models A\{z/y\})$.

10. $P \models \langle x(y)\rangle^E A$
$\Leftrightarrow \quad \forall z.\exists P'.\ (P \overset{xz}{\rightarrow} P' \ \&\ P' \models A\{z/y\})$.

11. $P \models [x(y)]^E A$
$\Leftrightarrow \quad \exists z.\ \forall P'.\ (P \overset{xz}{\rightarrow} P' \Rightarrow P' \models A\{z/y\})$.

12. $P \models [x = y]A$
$\Leftrightarrow \quad (x = y \Rightarrow P \models A)$.

□

Definition3.3 A *modal proof system* for *π − calculus* $\mathcal{NL}$ is defined by the rules below, where we use the following notational abbreviations :

$\alpha ::= \tau | x(y) | \bar{x}y$
$\beta ::= \alpha | xy | \bar{x}(y)$

(TRUE) $\dfrac{}{P \vdash \mathbf{true}}$ (0) $\dfrac{}{\mathbf{0} \vdash [\beta]A}$

(ACT) $\dfrac{}{\alpha.P \vdash [\beta]A}\ sub(\alpha) \neq sub(\beta)$

$(\wedge I)$ $\dfrac{P \vdash A \quad P \vdash B}{P \vdash A \wedge B}$

$(\vee I)$ $\dfrac{P \vdash A}{P \vdash A \vee B}$ $(\vee I)$ $\dfrac{P \vdash B}{P \vdash A \vee B}$

$(\langle\tau\rangle I)$ $\dfrac{P \vdash A}{\tau.P \vdash \langle\tau\rangle A}$ $(\langle\bar{x}y\rangle I)$ $\dfrac{P \vdash A}{\bar{x}y.P \vdash \langle\bar{x}y\rangle A}$

$([\tau]I)$ $\dfrac{P \vdash A}{\tau.P \vdash [\tau]A}$ $([\bar{x}y]I)$ $\dfrac{P \vdash A}{\bar{x}y.P \vdash [\bar{x}y]A}$

$(\langle x(y)\rangle I)$ $\dfrac{P\{a/y\} \vdash A\{a/y\}}{x(y).P \vdash \langle x(y)\rangle A}$

$([x(y)]I)$

$$\frac{P\{a_1/y\} \vdash A\{a_1/y\} \quad \cdots P\{a_n/y\} \quad \vdash A\{a_n/y\}}{x(y).P \vdash [x(y)]A}$$

provided that $fn(P) \cup fn(A) = \{a_1, \cdots, a_n\}$.

$(\langle x(y)\rangle^L I)$

$$\frac{P\{a_1/y\} \vdash A\{a_1/y\} \quad \cdots \quad P\{a_n/y\} \vdash A\{a_n/y\}}{x(y).P \vdash \langle x(y)\rangle^L A}$$

provided that $fn(P) \cup fn(A) = \{a_1, \cdots, a_n\}$.

$([x(y)]^L I)$ $\dfrac{P\{a/y\} \vdash A\{a/y\}}{x(y).P \vdash [x(y)]^L A}$

$(\langle x(y)\rangle^E I)$

$$\frac{P\{a_1/y\} \vdash A\{a_1/y\} \quad \cdots \quad P\{a_n/y\} \vdash A\{a_n/y\}}{x(y).P \vdash \langle x(y)\rangle^E A}$$

provided that $fn(P) \cup fn(A) = \{a_1, \cdots, a_n\}$.

$([x(y)]^E I)$ $\dfrac{P\{a/y\} \vdash A\{a/y\}}{x(y).P \vdash [x(y)]^E A}$

$(\langle xy\rangle I)$ $\dfrac{P\{z/y\} \vdash A\{z/y\}}{x(y).P \vdash \langle xz\rangle A}$

$([xy]I)$ $\dfrac{P\{z/y\} \vdash A\{z/y\}}{x(y).P \vdash [xz]A}$

$(\langle\bar{x}(y)\rangle\nu I)$ $\dfrac{P \vdash \langle\bar{x}y\rangle A}{(\nu y)P \vdash \langle\bar{x}(y)\rangle A}\ x \neq y$

$([\bar{x}(y)]\nu I)$ $\dfrac{P \vdash [\bar{x}y]A}{(\nu y)P \vdash [\bar{x}(y)]A}\ x \neq y$

$(\langle\beta\rangle + I)$ $\dfrac{P \vdash \langle\beta\rangle A}{P + Q \vdash \langle\beta\rangle A}$

$(\langle\beta\rangle + I)$ $\dfrac{Q \vdash \langle\beta\rangle A}{P + Q \vdash \langle\beta\rangle A}$

$([\beta] + I)$ $\dfrac{P \vdash [\beta]A \quad Q \vdash [\beta]A}{P + Q \vdash [\beta]A}$

$(!I)$ $\dfrac{P \vdash A}{!P \vdash A}$

$(\equiv I)$ $\dfrac{P \equiv P' \quad P' \vdash A}{P \vdash A}$

(MI) $\dfrac{P \vdash A}{[x = y]P \vdash [x = y]A}$

(NMI) $\dfrac{}{P \vdash [x = y]A}\ x \neq y$

□

Theorem3.1 (Soundness and Completeness) For a composition free process P and a formula F, we have $P \vdash F$ *iff* $P \models F$. □

4. Extension

In this section, an extension of the resulting proof system is considered for agents instead of processes.

4.1 The Polyadic π-calculus

Definition4.1 An *agent* in the polyadic π-calculus is defined by the following grammar:

P	$::=$	$\mathbf{0}$	*inaction*
	$\mid$	$\tau.A$	*silent prefix*
	$\mid$	$a.A$	*input prefix*
	$\mid$	$\bar{a}.A$	*output prefix*
	$\mid$	$(\lambda x)A$	*abstraction*
	$\mid$	Ax	*application*
	$\mid$	$[y]A$	*concretion*
	$\mid$	$A+A$	*summation*
	$\mid$	$A \mid A$	*composition*
	$\mid$	$[x=y]A$	*match*
	$\mid$	$(\nu x)A$	*restriction*
	$\mid$	$!A$	*replication*
		$(\mathit{fix}\ D.A)$	*recursive agent*

Let $\mathcal{P}$ denote the set of all processes. □

An input prefix and an output prefix in the monadic $\pi - calculus$ can be describe by agents, e.g.

$$x(y).P = x.(\lambda y)P$$
$$\bar{x}y.P = \bar{x}.[y]P$$

We need a notion of arity to distinguish abstractions and concretions.

Definition4.2 Assignment of arities for agents can be done by the following rules:

$$\frac{}{\mathbf{0}:0} \qquad \frac{A:0}{\tau.A:0}$$

$$\frac{A:n \quad n\leq 0}{a.A:0} \qquad \frac{A:n \quad n\geq 0}{\bar{a}.A:0}$$

$$\frac{A:n \quad n\leq 0}{(\lambda x)A:n-1} \qquad \frac{A:n \quad n<0}{Ax:n+1}$$

$$\frac{A:n \quad n\geq 0}{[x]A:n+1} \qquad \frac{A:0 \quad B:0}{A+B:0}$$

$$\frac{A:0 \quad B:0}{A\mid B:0} \qquad \frac{A:n}{[x=y]A:n}$$

$$\frac{A:n}{(\nu x)A:n} \qquad \frac{A:n}{A!:n} \qquad \frac{D:n \quad A:n}{\mathit{fix}\ D.A:n}$$

A *process* is an agent with arity 0, i.e. an agent such that $A:0$ can be proved. In the similar way, we can say an abstraction is an agent with negative arity; a concretion is an agent with positive arity. □

According to the extension, the corresponding structural congruence relation is changed. The *structural congruence relation* $\equiv$ over agents can be defied by the following equalities.

(1) If $A \equiv_\alpha B$ then $A \equiv B$:
Processes are identified if they differ only by a change of bound names.

(2) $((\lambda x)A)y \equiv A\{y/x\}$.

(3) $A+(B+C) \equiv (A+B)+C$.
$A+B \equiv B+A$.
$A+\mathbf{0} \equiv A$.

(4) $A \mid (B \mid C) \equiv (P \mid B) \mid C$.
$A \mid B \equiv B \mid A$.
$A \mid \mathbf{0} \equiv A$.

(5) $[x=x]A \equiv A$.

(6) $(\nu x)A \equiv A \quad$ if $x \notin fn(A)$.
$(\nu x)(\nu y)A \equiv (\nu y)(\nu x)A$.
$(\nu x)A \mid B \equiv (\nu x)(A \mid B) \quad$ if $x \notin fn(B)$.

(7) if $x \neq y$ then,
$(\nu x)(\lambda y)A \equiv (\lambda y)(\nu x)A$.
$[x](\nu y)A \equiv (\nu y)[x]A$.

(8) $!A \equiv !A \mid A \quad (\mathit{fix}\ D.A \equiv A\{\mathit{fix}\ D.A/D\})$.

Proposition4.1 Given any well-formed agent A, there is a normal form B such that $A \equiv B$, where an agent A is a normal form if it is either

(1) an abstraction of the form $(\lambda x)B$;
(2) a concretion of the form $(\nu x)[x]B$ or $[x]B$;
(3) a process P generated by the grammar

P	$::=$	$\mathbf{0}$	*inaction*
	$\mid$	$\alpha.P$	*action prefix*
	$\mid$	$P+P$	*summation*
	$\mid$	$P \mid P$	*composition*
	$\mid$	$[x=y]P$	*match*
	$\mid$	$(\nu x)P$	*restriction*
	$\mid$	$!P$	*replication.*

□

We need a commitment relation instead of the transition relation.

Definition4.3 The commitment relation over agents can be defined by the rules:

$$\text{ACT}\ \frac{}{\alpha.A \succ \alpha.A} \qquad \text{SUM}\ \frac{A_1 \succ B}{A_1 + A_2 \succ B}$$

$$\text{COMM}\ \frac{A_1 \succ a.B_1 \qquad A_2 \succ \bar{a}.B_2}{A_1 \mid A_2 \succ \tau.(B_1 \cdot B_2)}$$

$$\text{PAR}\ \frac{A_1 \succ \alpha.B}{A_1 \mid A_2 \succ \alpha.(B \mid A_2)}$$

$$\text{RES}\ \frac{A \succ \alpha.B}{(\nu x)A \succ \alpha.(\nu x)B}\ x \notin n(\alpha)$$

$$\text{STRUCT}\ \frac{A \equiv A' \quad A' \succ \alpha.B' \quad B' \equiv B}{A \succ \alpha.B}$$

□

4.2 Modal Logic

We need the extension to the modal logic.

Definition4.4 Let $\mathcal{M}$ denote the set of all formulae defined by the following grammar:

F	$::=$	**true**	true
	$\mid$	**false**	false
	$\mid$	$x = y$	match
	$\mid$	$x \neq y$	dismatch
	$\mid$	$F \wedge G$	
	$\mid$	$F \vee G$	
	$\mid$	$\langle\alpha\rangle F$	
	$\mid$	$[\alpha]A$	
	$\mid$	X	propositional variable
	$\mid$	$\nu Z.F$	greatest fixed point
	$\mid$	$\mu Z.F$	least fixed point
	$\mid$	$\lambda x.F$	abstraction
	$\mid$	Fx	application
	$\mid$	$\Sigma x.F$	dependent sum
	$\mid$	$\forall x.F$	dependent product
	$\mid$	$\exists x.F$	

□

Definition4.5 The assignment of arities for formulae is defined by the rules:

$$\frac{}{\textbf{true} : 0} \qquad \frac{}{\textbf{false} : 0}$$

$$\frac{}{x = y : 0} \qquad \frac{}{x \neq y : 0}$$

$$\frac{F : 0 \quad G : 0}{F \wedge G : 0} \qquad \frac{F : 0 \quad G : 0}{F \vee G : 0}$$

$$\frac{F : 0}{\langle\alpha\rangle F : 0} \qquad \frac{F : 0}{[\alpha]F : 0}$$

$$\frac{Z : n \quad F : n}{\sigma Z.F : 0}\ \sigma \in \{\nu, \mu\}$$

$$\frac{F : n}{\lambda x.F : n+1} \qquad \frac{F : n+1}{Fx : n}$$

$$\frac{F : n+1}{\Sigma x.F : n} \qquad \frac{F : n+1}{\forall x.F : n} \qquad \frac{F : n+1}{\exists x.F : n}$$

□

Let ρ be a *proposition environment*, i.e. a function which given a propositional variable Z of arity m, an m-vector of names $x_1, \ldots, x_m$ gives a set $\rho Z x_1, \cdots, x_m \subset \mathcal{A}$ of agents. Given a proposition environment ρ and a a sequence of names $\tilde{x}$, the semantics of a formula F is defined as $[\![F]\!]\rho\tilde{x} \subset \mathcal{A}$ by structural induction over formulae F:

$$[\![\textbf{true}]\!]\rho\tilde{x} = \mathcal{A}$$
$$[\![\textbf{false}]\!]\rho\tilde{x} = \emptyset$$
$$[\![x = y]\!]\rho\tilde{x} = \begin{cases} \mathcal{A} & \text{if } x = y \\ \emptyset & \text{otherwise} \end{cases}$$
$$[\![x \neq y]\!]\rho\tilde{x} = \begin{cases} \emptyset & \text{if } x = y \\ \mathcal{A} & \text{otherwise} \end{cases}$$
$$[\![F \wedge G]\!]\rho\tilde{x} = [\![F]\!]\rho\tilde{x} \cap [\![G]\!]\rho\tilde{x}$$
$$[\![F \vee G]\!]\rho\tilde{x} = [\![F]\!]\rho\tilde{x} \cup [\![G]\!]\rho\tilde{x}$$
$$[\![\langle\alpha\rangle F]\!]\rho\tilde{x} = \{A \mid \exists B.A \succ \alpha.B,\ B \in [\![F]\!]\rho\tilde{x}\}$$
$$[\![[\alpha]A]\!]\rho\tilde{x} = \{A \mid \forall B.A \succ \alpha.B \text{ implies } B \in [\![F]\!]\rho\tilde{x}\}$$
$$[\![Z]\!]\rho\tilde{x} = \rho Z\tilde{x}$$
$$[\![\nu Z.F]\!]\rho\tilde{x} = \cup\{\Phi \mid \Phi \subset [\![F]\!]\rho[Z \mapsto \Phi]\tilde{x}\}$$
$$[\![\mu Z.F]\!]\rho\tilde{x} = \cap\{\Phi \mid [\![F]\!]\rho[Z \mapsto \Phi]\tilde{x} \subset \Phi\}$$
$$[\![\lambda x.F]\!]\rho y\tilde{x} = [\![F\{y/x\}]\!]\rho\tilde{x}$$
$$[\![Fy]\!]\rho\tilde{x} = [\![F]\!]\rho y\tilde{x}$$
$$[\![\Sigma x.F]\!]\rho\tilde{x} = \{A \mid A \equiv [y]B \text{ or } (\nu y)[y]B, \text{ with } y \notin fn(F) \cup \{\tilde{x}\}, \text{ and } B \in [\![F]\!]\rho y\tilde{x}\}$$
$$[\![\forall x.F]\!]\rho\tilde{x} = \{A \mid \forall y.Ax \in [\![F]\!]\rho y\tilde{x}\}$$
$$[\![\exists x.F]\!]\rho\tilde{x} = \{A \mid \exists y.Ax \in [\![F]\!]\rho y\tilde{x}\}$$

4.3 Proof System

A *definition list* is a sequence of assertions

Delta $= U_1 \mapsto F_1, \ldots, U_n \mapsto F_n$

such that

1. each U_i is unique;
2. each constant occurring in A_i is $\{U_1, \ldots, U_{i-1}\}$

Let consider the inference rules:

$$(\text{TRUE})\ \frac{}{A \vdash_\Delta \textbf{true}} \qquad (\text{EQ})\ \frac{}{A \vdash_\Delta [x = y]}$$

$$(\wedge I)\ \frac{A \vdash_\Delta F \quad A \vdash_\Delta G}{A \vdash_\Delta F \wedge G}$$

$$(\vee I)\ \frac{A \vdash_\Delta F}{A \vdash_\Delta F \vee G} \qquad (\vee I)\ \frac{A \vdash_\Delta G}{A \vdash_\Delta F \vee G}$$

$$(\langle\alpha\rangle I)\quad \frac{B \vdash_{\Delta} F}{A \vdash_{\Delta} \langle\alpha\rangle F}\ A \succ \alpha.B$$

$$([\alpha] I)\quad \frac{B_1 \vdash_{\Delta} F \quad \cdots \quad B_n \vdash_{\Delta} F}{A \vdash_{\Delta} [\alpha] F}$$

provided that $\{B \mid\succ \alpha.B\} = \{B_1, \ldots, B_n\}$

$$(\text{FIX})\quad \frac{A \vdash_{\Delta, U \mapsto \sigma Z.F} U\tilde{x}}{A \vdash_{\Delta} (\sigma Z.F)\tilde{x}}$$

$$(\text{FOLD})\quad \frac{A \vdash_{\Delta} F\{U/Z\}\tilde{x}}{A \vdash_{\Delta} U\tilde{x}}\ \Delta(U) = \sigma Z.F$$

$$(\text{ABST})\quad \frac{A \vdash_{\Delta} F\{y/x\}\tilde{x}}{A \vdash_{\Delta} (\lambda x.F)y\tilde{x}}$$

$$(\text{APP})\quad \frac{A \vdash_{\Delta} Fy\tilde{x}}{A \vdash_{\Delta} (Fy)\tilde{x}}$$

$$(\text{CONC-1})\quad \frac{A \vdash_{\Delta} Fy\tilde{x}}{[y]A \vdash_{\Delta} (\Sigma x.F)\tilde{x}}$$

$$(\text{CONC-2})\quad \frac{A \vdash_{\Delta} Fy\tilde{x}}{(\nu y)[y]A \vdash_{\Delta} (\Sigma x.F)\tilde{x}}\ y:\ \text{fresh}$$

$$(\text{PROD})\quad \frac{Ay \vdash_{\Delta} Fy\tilde{x}}{A \vdash_{\Delta} (\forall x.F)\tilde{x}}\ y:\ \text{fresh}$$

$$(\text{EXIST})\quad \frac{A \vdash_{\Delta} Fy\tilde{x}}{A \vdash_{\Delta} (\exists x.F)\tilde{x}}$$

Theorem4.1 Let $A \vdash_{\Delta} F$ be a sequent with A an agent of finite control. $A \vdash_{\Delta} F$ is derivable iff $A \vdash_{\Delta} F$ is true. □

5. Conclusion

In this paper, a modal proof system for processes in the π-calculus was proposed with the discussion on soundness and completeness. An extension of the resulting system was also proposed for mobile agents.

Acknowledgment

This paper was partially supported by The Grand-In-Aid for Scientific Research (C)08680343, The Grant-In-Aid for Scientific Research on Priority Areas 09245214, The Telecommunications Advancement Foundation, Tokai Foundation for Technology, and KAYAMORI Foundation of Information Science Advancement.

References

1) Henrik Reif Andersen, Coling Stirling, Glynn Winskel, A compositional proof system for the modal μ-calculus, Proc. of ninth Annual IEEE Symposium on Logic in Computer Science, 1994.
2) Model checking mobile processes, Information and Computation, 129, pp.34–51, 1996.
3) Dexter Kozen, Results on the propositional μ-calculus, Theoretical Computer Science 27, pp.333-354, 1983.
4) Coling Stirling, A proof-theoretic characterization of observational equivalence, Theoretical Computer Science 39, pp.27-45, 1985.
5) Coling Stirling, Modal logics for communicating systems, Theoretical Computer Science 49, pp.311-347, 1987.
6) Coling Stirling, David Walker, Local model checking in the modal mu-calculus, Theoretical Computer Science 89, pp.161-177, 1991.
7) Glynn Winskel, A complete proof system for SCCS with modal assertions, Lecture Notes in Computer Science 206, pp.392-410, 1985.
8) Coling Stirling, An introduction to modal and temporal logic for CCS, Lecture Notes in Computer Science 491, pp.2-20, 1991.
9) Coling Stirling, David Walker, Local model checking in the modal mu-calculus, Theoretical Computer Science 89, pp.161-177, 1991.
10) Milner, R., *Communication and Concurrency*, Prentice Hall, 1989.
11) Milner, R., Functions as processes, ICALP'90, Lecture Notes in Computer Science 443, 1990.
12) Milner, R., The polyadic π-calculus: A tutorial, Technical Report ECS-LFCS-91-180, LFCS, Dept. of Comput. Sci., Edinburgh Univ., 1991.
13) Robin Milner, Joachim Parrow, David Walker, A calculus of mobile processes, Information and Computation 100, pp.1-77, 1992.
14) Robin Milner, Joachim Parrow, David Walker, Modal logics for mobile processes, Theoretical Computer Science 114, pp.149-171, 1993.
15) Sangiorgi, D., Expressing mobility in process algebras: first-order and higher-order paradigms, Ph.D. Thesis, Edinburgh University, 1992.
16) Coling Stirling, Modal and temporal logics for processes, Dept. of Computer Science University of Edinburgh, 1995.
17) Togashi, A., On typing systems for the polyadic π-calculus, to appear in *Technical Report*, COGS, University of Sussex, 1996.