数理解析研究所講究録
996 巻 1997 年 111-124

Multi-action π-calculus

Yukihiro ODA Masaki MURAKAMI

Okayama University

E-mail: {oda,murakami}@momo.it.okayama-u.ac.jp

Abstract

We propose a new truly-concurrent semantics for the π-calculus[1]. We extend the labelled transition system of the π-calculus to use multi-sets of actions as labels. We call this extension *multi-action π-calculus.* The multi-action π-calculus can describe concurrent behavior of agents. Strong bisimilarity defined in the multi-action π-calculus is closed under input-prefixings and parallel compositions.

1 Introduction

In recent years, studies of process algebra, especially the *π-calculus*[1], have a lot of attention. The π-calculus is an extension of CCS[2]. Agents of the π-calculus can modify their linkage structures dynamically via *name-passing* mechanism. The *observation equivalence* is proposed as a semantics, an equivalence relation over the π-calculus agents. This semantics is called *interleaving semantics*, because this semantics is based on the notion of interleaving . In the view of the interleaving, concurrent (or parallel) executions are described by non-deterministic alternations of sequential executions. For example, agents $a|b$ and $a.b+b.a$ are regarded as same. This property is characterized as the *expansion law*[2]. The expansion law is very convenient to use CCS-like languages as formal specification languages.

On the other hand, semantics those are not based on the interleaving are proposed. These semantics are called non-interleaving semantics or *truly-concurrent semantics.* These semantics focus the following problems:

- CCS-like languages can describe concurrent systems of agents. But the interleaving can not describe the concurrent behavior of agents truly. Any concurrent agent is sequentialized in the interleaving semantics.

- In the π-calculus, the observation equivalence (even strong ground bisimilarity) is not closed under *input prefixings.* That is, the expansion law is not correct in the π-calculus. For example, $\overline{x}z|y(w) \stackrel{\cdot}{\sim} \overline{x}z.y(w)+y(w).\overline{x}z$ but $x'(y).(\overline{x}z|y(w)) \not\stackrel{\cdot}{\sim} x'(y).(\overline{x}z.y(w)+y(w).\overline{x}z)$. Because the left-side agent can perform a τ action but the right-side agent can not. This problem is caused via identification of concurrent executions and sequential executions without consideration to communication ability given by name-passing.

Location[3][4][5], causality[6][7][8][9], Petri-Net[10], graph rewriting system[11] semantics are already proposed as truly-concurrent semantics. These semantics improved the concurrency and the inconvenience involved with the name-passing. But these semantics have the following problems:

- We must analyze semantics of each agent to capture the concurrent behavior of such agents. The concurrent behavior of agents should be captured from its semantics directly.
- Especially in the location based semantics, the concurrency that is captured by observation and that is determined by semantics are not equivalent. That is, concurrent executions of an agent are regarded as sequential executions by semantics. This ruins congruence of semantics by the similar way to the expansion law.

We discuss these problems again in Section 6.

We propose another truly-concurrent semantics for the π-calculus to improve the above two problems. The idea of our new semantics is to re-define the notion of actions using *multi-actions.* One multi-action means that two or more actions (or events) are performed concurrently and observed. For example, we infer a transition $a.P|b.Q|(c+R)|\overline{c} \xrightarrow{a|b|\tau} P|Q$. We extend the labelled transition system of the π-calculus to use multi-actions as labels and define bisimilarities over the extended π-calculus. We call this extension of the π-calculus *multi-action π-calculus.*

Multi-action approaches are already proposed for non-mobile calculi[12]. But we can not adopt these works simply for the π-calculus. Because the special treatment for *name-extrusion* is required. In the π-calculus, restriction operators, denoted by (νz), also work as *sequentializers* by the different way to the CCS. For example, let us consider an agent $P \stackrel{def}{=} \overline{x}z.Q_1|z(y).Q_2|w(y)$. Intuitively, P performs action $w(y)$, $\overline{x}z$ and $z(y)$ concurrently. Suppose an agent $(\nu z)P$. $(\nu z)P$ reaches deadlock after $w(y)$ in the framework of non-mobile calculi. Because executions of actions containing name z are prohibited. On the other hand, in the π-calculus, $(\nu z)P$ performs action $w(y)$ and $\overline{x}(z)$ followed by $z(y)$. Because $\overline{x}(z)$ eliminates (νz) from the agent. But $z(y)$ can not be performed before $\overline{x}(z)$ and these two actions are no longer performed concurrently. On the other hand, $w(y)$ and $\overline{x}(z)$ are concurrent. We must determine that which actions are concurrently executable with a name-extrusion, when we use the Open Rule. This requirement of special treatment is similar to the case of the causality for the π-calculus[6]. We can consider many kinds of treatments of name-extrusion. Congruence of semantics depends on the choice of the treatments.

Outline of this paper: We introduce the π-calculus and its bisimilarity in section 2. We propose the multi-actions and the multi-action π-calculus in section 3. And we propose strong bisimilarity for the multi-action π-calculus and prove its congruence (outline) in section 4. We discuss the concurrency via multi-action in section 5. Finally, we compare multi-action based semantics with other truly-concurrent semantics in section 6.

2 π-calculus

In this section, we define the π-calculus and its (observational) strong ground bisimilarity. Furthermore, we show an example to demonstrate the problems mentioned in the previous section.

Definition 2.1 (Actions) Let $\mathcal{N}$ be an infinite set of names. We define a set *Act* defined as follows:

$$Act \stackrel{def}{=} \{xy, x(y), \overline{x}y, \overline{x}(y) | x, y \in \mathcal{N}\} \cup \{\tau\}$$

We let x, y, $\cdots$ range over $\mathcal{N}$ and a, b, $\cdots$ range over *Act*. We identify x and $\overline{\overline{x}}$.

When we have no interest in the object-part of an action, we omit it.

Definition 2.2 (Agents) We define a set $\mathcal{P}$ of all terms generated by the following rule:

$$P ::= 0 \,\big|\, N(N).P \,\big|\, \overline{N}N.P \,\big|\, \tau.P \,\big|\, (\nu N)P. \,\big|\, P + P \,\big|\, P|P$$

where N is an element of $\mathcal{N}$. We call each element of $\mathcal{P}$ an *agent*. We abbreviate $(\nu z_1)(\nu z_2)\cdots(\nu z_n)P$ as $(\nu Z)P$ with the multi-set $Z = \{z_1, z_2, \cdots, z_n\}$. And we identify P and $(\nu\emptyset)P$.

We define $bn(\cdot)$, $fn(\cdot)$, $n(\cdot)$, $obj(\cdot)$, $sub(\cdot)$ and *structural congruence* $\equiv$ in usual way.

Remark 2.3 Replication, denoted by $!P$ usually, is not considered.

Definition 2.4 (name substitution) A *name-substitution* is a full function $\theta : \mathcal{N} \longrightarrow \mathcal{N}$. We write application of a name-substitution θ to name x as $x\theta$. Let ι be the identity name-substitution and $\{x'/x\}$ is the same name-substitution to ι except $x\{x'/x\} = x'$.

We also define application of a name-substitution to agents in usual way.

Assumption 2.5 In order to simplify the following discussion, we assume that any bound name in an agent is distinct from any free name in such agent. That is, bound names are renamed *fresh* names automatically in the agent.

Definition 2.6 (Transition System) We define the labelled transition system $L = (\mathcal{P}, Act, \rightarrow)$ where $\rightarrow \subset \mathcal{P} \times Act \times \mathcal{P}$ is the relation given by the transitive closure of the following inference rules.

$$\frac{}{\alpha.P \xrightarrow{\alpha} P}\text{ Prefix} \qquad \frac{}{x(y).P \xrightarrow{xz} P\{z/y\}}\text{ Input}$$

$$\frac{P \xrightarrow{a} P'}{P+Q \xrightarrow{a} P'}\text{ Sum} \qquad \frac{P \xrightarrow{a} P' \quad z \notin n(a)}{(\nu z)P \xrightarrow{a} (\nu z)P'}\text{ Restriction}$$

$$\frac{P \xrightarrow{\overline{x}y} P' \quad Q \xrightarrow{xy} Q'}{P|Q \xrightarrow{\tau} P'|Q'}\text{ Communication} \qquad \frac{P \xrightarrow{\overline{x}(z)} P' \quad Q \xrightarrow{x(z)} Q'}{P|Q \xrightarrow{\tau} (\nu z)(P'|Q')}\text{ Close}$$

$$\frac{P \xrightarrow{\overline{x}w} P' \quad w \neq x}{(\nu w)P \xrightarrow{\overline{x}(w)} P'}\text{ Open}$$

$$\frac{P \equiv Q \quad Q \xrightarrow{a} Q' \quad Q' \equiv P'}{P \xrightarrow{a} P'}\text{ Structure} \qquad \frac{P \xrightarrow{a} P'}{P|Q \xrightarrow{a} P'|Q}\text{ Parallel}$$

Remark 2.7 Without Assumption 2.5, the side condition $bn(a) \cap fn(Q) = \emptyset$ is needed for the Parallel Rule.

Definition 2.8 (strong ground bisimilarity) A *symmetric* relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a *strong ground bisimulation* if and only if for every $(P, Q) \in \mathcal{R}$,

$$P \xrightarrow{a} P' \text{ (where } bn(a) \cap fn(P, Q) = \emptyset \text{)} \implies \exists Q'.\ Q \xrightarrow{a} Q' \text{ and } P'\mathcal{R}Q'.$$

Definition 2.9 $\dot{\sim} \stackrel{def}{=}$ (the largest strong ground bisimulation).

Example 2.10 Please recall these two agent appeared in the Section 1.

$$P_0 \stackrel{def}{=} \overline{x}z|y(w)$$
$$Q_0 \stackrel{def}{=} \overline{x}z.y(w) + y(w).\overline{x}z$$

P_0 performs actions concurrently but Q_0 performs sequentially. $\dot{\sim}$ can not distinguish agents via their concurrent behavior. Because both P_0 and Q_0 have same interleaved (that is, sequential) transitions. For instance, $\xrightarrow{\overline{x}z}\xrightarrow{yw}$, $\xrightarrow{yw}\xrightarrow{\overline{x}z}$, $\cdots$.

Suppose the following two agent $x'(y).P_0$ and $x'(y).Q_0$. These two agent no longer identified by $\dot{\sim}$. Because we have a transition $x'(y).P_0 \xrightarrow{x'x}\xrightarrow{\tau}$ but we have $x'(y).Q_0 \xrightarrow{x'x}\not\xrightarrow{\tau}$ always. Thus, $\dot{\sim}$ is not closed under input-prefixings.

3 Multi-action π-calculus

In this section, we extend the π-calculus. The extended calculus, *multi-action π-calculus*, uses the same syntax to the π-calculus. But its labelled transition system uses *multi-actions* as labels. A multi-action is a cluster of actions. Intuitively, the multi-action means that actions in it are performed concurrently and observed. We describe the concurrent behavior of agents directly using multi-actions.

Definition 3.1 Let $|$ be an infix 2-ary operator symbol. We define a set $\mathcal{A}_{Act,|}$ of terms generated from the rule:

$$A ::= Act \,\Big|\, A|A.$$

We let A, B, $\cdots$ range over $\mathcal{A}_{Act,|}$.

Let $=_s$ be the smallest binary relation over $\mathcal{A}_{Act,|}$ that satisfies the following conditions:

- It is an equivalence relation,
- $(A|B)|C =_s A|(B|C)$, and
- $A|B =_s B|A$.

We abbreviate $\mathcal{A}_{Act,|}/=_s$ (the quotient set of $\mathcal{A}_{Act,|}$ by $=_s$) as $\mathcal{A}$, and $[A]_{=_s}$ as A.

Definition 3.2 We define a binary operator $|_{\mathcal{A}}$ over $\mathcal{A} \cup \{\emptyset\}$

$$A|_{\mathcal{A}}B \stackrel{def}{=} \begin{cases} A & B = \emptyset, \\ B & A = \emptyset, \\ A|B & \text{otherwise.} \end{cases}$$

More specifically, the final line means that $[A]_{=_s}|_{\mathcal{A}}[B]_{=_s} \stackrel{def}{=} [A|B]_{=_s}$.

Definition 3.3 Let $a, b \in Act$. a and b are *communicatable* if and only if $a = \overline{b}$. Please note that any bound action and free action are *not* communicatable.

Definition 3.4

$$A \bowtie B \stackrel{def}{=} \begin{cases} \tau|_{\mathcal{A}}(A' \bowtie B') & \text{if } \exists a, b \in Act.\ A = a|A',\ B = b|B' \text{ and} \\ & \qquad a \text{ and } b \text{ are communicatable ,} \\ A|_{\mathcal{A}}B & \text{otherwise.} \end{cases}$$

$$\mathsf{TIE}(A, B) \stackrel{def}{=} \begin{cases} \{y\} \uplus \mathsf{TIE}(A', B') & \text{if } \exists x, y \in \mathcal{N}.\ (A = \overline{x}(y)|A' \text{ and } B = x(y)|B') \text{ or} \\ & \qquad (A = x(y)|A' \text{ and } B = \overline{x}(y)|B'), \\ \emptyset & \text{otherwise.} \end{cases}$$

where $\uplus$ is the multi-set union operator. Thus, $\mathsf{TIE}(A, B)$ takes two elements of $\mathcal{A}$ and returns a multi-set of names.

Intuitively, $A \bowtie B$ invokes interactions between communicatable actions in A and B simultaneously. $\mathsf{TIE}(A, B)$ gives names required to bind after interactions between bound actions in A and B.

Example 3.5

$$\begin{aligned} \overline{a}(z)|b(w)|cy|d \bowtie \overline{b}(w)|a(z)|\overline{c}y &= \tau|\tau|\tau|d \\ \mathsf{TIE}(\ \overline{a}(z)|b(w)|cy|d,\ \overline{b}(w)|a(z)|\overline{c}y\ ) &= \{w, z\} \end{aligned}$$

Lemma 3.6 Let $M_{\text{strong}} \stackrel{def}{=} (\mathcal{A} \cup \{\emptyset\}, \emptyset, |_{\mathcal{A}})$. Then, M_{strong} is a commutative monoid and $\emptyset$ is the unit element of M_{strong}.

Definition 3.7 (Multi-action) We call each element of M_{strong} a *multi-action.*

We can use any structure as the multi-actions if it follows the definitions. For example, we can use the *multi-set of actions and multi-set union* as the multi-actions.

We abbreviate $|_{\mathcal{A}}$ as $|$. For convenience, we write $\prod_{i \leq n} A_i$ as $A_1|A_2|\cdots|A_n$.

Example 3.8 $\{a|b|c,\ c|a|b,\ b|a|c\}$ is a set of same multi-actions. $\{a|b,\ a|a|b,\ a|b|\tau\}$, $\{a|\overline{a},\ \tau,\ \tau|\tau\}$ and $\{a|\tau,\ a\}$ are sets of different multi-actions.

Definition 3.9 We define a function $fi(\cdot)$ over multi-actions:

$$fi(A) \stackrel{def}{=} \begin{cases} xy|fi(A') & \text{if } \exists x, y.\ A = xy|A', \\ \emptyset & \text{otherwise.} \end{cases}$$

$fi(A)$ is the multi-action constructed from free-input actions in A. We also define $fo(A)$ (extraction of free-output from A), $bi(A)$ (extraction of bound-input from A) and $bo(A)$ (extraction of bound-output from A).

We define a function $bo2fo(\cdot)$ as follows:

$$bo2fo(A) \stackrel{def}{=} \begin{cases} \overline{x}y|bo2fo(A') & \text{if } \exists x, y.\ A = \overline{x}(y)|A', \\ A & \text{otherwise.} \end{cases}$$

$bo2fo(A)$ is the multi-action obtained by replacing each bound-output $\overline{x}(y)$ of A with corresponding free-output $\overline{x}y$.

Definition 3.10 (Multi-action Transition System) We define the labelled transition system $L_M = (\mathcal{P}, \mathcal{A}, \underset{\mathrm{M}}{\rightarrow})$ where $\underset{\mathrm{M}}{\rightarrow} \subset \mathcal{P} \times \mathcal{A} \times \mathcal{P}$ is the relation given by the transitive closure of the following inference rules.

$$\frac{}{\alpha.P \xrightarrow[\mathrm{M}]{\alpha} P}\ \text{Prefix} \qquad \frac{}{x(y).P \xrightarrow[\mathrm{M}]{xz} P\{z/y\}}\ \text{Input}$$

$$\frac{P \xrightarrow[\mathrm{M}]{A} P'}{P{+}Q \xrightarrow[\mathrm{M}]{A} P'}\ \text{Sum} \qquad \frac{P \xrightarrow[\mathrm{M}]{A} P' \quad z{\notin}n(A)}{(\nu z)P \xrightarrow[\mathrm{M}]{A} (\nu z)P'}\ \text{Restriction}$$

$$\frac{P \xrightarrow[\mathrm{M}]{A_P} P' \quad Q \xrightarrow[\mathrm{M}]{A_Q} Q'}{P|Q \xrightarrow[\mathrm{M}]{A_P \bowtie A_Q} (\nu \mathsf{TIE}(A_P, A_Q))(P'|Q')}\ \text{Communication}$$

$$\frac{P \xrightarrow[\mathrm{M}]{A|\overline{x}w} P' \quad w{\neq}x \quad w{\notin}sub(A){\cup}obj(fi(A))}{(\nu w)P \xrightarrow[\mathrm{M}]{A|\overline{x}(w)} P'}\ \text{Open}$$

$$\frac{P{\equiv}Q \quad Q \xrightarrow[\mathrm{M}]{A} Q' \quad Q'{\equiv}P'}{P \xrightarrow[\mathrm{M}]{A} P'}\ \text{Structure} \qquad \frac{P \xrightarrow[\mathrm{M}]{A} P'}{P|Q \xrightarrow[\mathrm{M}]{A} P'|Q}\ \text{Parallel}$$

Remark 3.11 Without Assumption 2.5, the Communication Rule must have complexed side conditions. That is, "bound actions of $A_P{\bowtie}A_Q$ come from A_P do not bind any free name in agent Q and Q', and vise versa. Furthermore, names in $\mathsf{TIE}(A_P, A_Q)$ are distinct from names in $A_P{\bowtie}A_Q$".

Of course, the side condition "$bn(A){\cap}fn(Q) = \emptyset$" is needed for the Parallel Rule.

Example 3.12 We can obtain a transition $a|b(w).P|(\nu z)(\overline{b}z|z.Q) \xrightarrow[\mathrm{M}]{a|\tau} (\nu z)(P\{z/w\}|z.Q)$ by the following inference:

$$\frac{\dfrac{a \xrightarrow[\mathrm{M}]{a} 0 \quad b(w).P \xrightarrow[\mathrm{M}]{b(z)} P\{z/w\}}{a|b(w).P \xrightarrow[\mathrm{M}]{a|b(z)} P\{z/w\}} \qquad \dfrac{\dfrac{\overline{b}z \xrightarrow[\mathrm{M}]{\overline{b}z} 0}{\overline{b}z|z.Q \xrightarrow[\mathrm{M}]{\overline{b}z} z.Q} \quad z \neq b \quad z \in \emptyset}{(\nu z)(\overline{b}z|z.Q) \xrightarrow[\mathrm{M}]{\overline{b}(z)} z.Q}}{a|b(w).P|(\nu z)(\overline{b}z|z.Q) \xrightarrow[\mathrm{M}]{a|\tau} (\nu z)(P\{z/w\}|z.Q).}$$

On the other hand, we can not obtain $a|b(w).P|(\nu z)(\overline{b}z|z.Q) \xrightarrow[\mathrm{M}]{a|\tau|z} (\nu z)(P\{z/w\}|Q)$ because the transition $(\nu z)(\overline{b}z|z.Q) \xrightarrow[\mathrm{M}]{\overline{b}(z)|z} Q$ is prohibited.

4 Strong Bisimilarity

In this section, we propose strong ground bisimilarity for the multi-action π-calculus and we show congruent results for this bisimilarity.

Definition 4.1 (strong ground multi-action bisimilarity) A *symmetric* relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a *strong ground multi-action bisimulation* if and only if for every $(P, Q) \in \mathcal{R}$,

$$P \xrightarrow[\mathrm{M}]{A} P' \text{ (where } bn(A) \cap fn(P, Q) = \emptyset \text{) } \Longrightarrow \exists Q'.\ Q \xrightarrow[\mathrm{M}]{A} Q' \text{ and } P'\mathcal{R}Q'.$$

Definition 4.2 $\underset{\mathrm{M}}{\sim} \stackrel{def}{=}$ (the largest strong ground multi-action bisimulation).

We also define *strong ground multi-action bisimulation up to* $\underset{\mathrm{M}}{\sim}$ in usual way. We can show that if a relation $\mathcal{R}$ is a strong ground multi-action bisimulation up to $\underset{\mathrm{M}}{\sim}$ then $\underset{\mathrm{M}}{\sim} \mathcal{R} \underset{\mathrm{M}}{\sim}$ is a strong ground multi-action bisimulation.

Theorem 4.3 $\underset{\mathrm{M}}{\sim}$ is closed under τ-prefixings, output-prefixings, restrictions and sum compositions.

Theorem 4.4 $\underset{\mathrm{M}}{\sim}$ is closed under parallel compositions.

Discussion 4.5 By the side-conditions of the Open Rule in the Definition 3.10, $\underset{\mathrm{M}}{\sim}$ is preserved for parallel compositions. The side-condition $w \notin sub(A) \cup obj(fi(A))$ checks each action's *independency* of extruded name w. We can consider many kinds of independency (or *dependency*) involved with name-extrusion. For example, *link dependency*[6], *enabling dependency*[6] and *object dependency*[6][8] are considered. But these dependencies are not appropriate.

dependency	actions depending on $\overline{x}(a)$
link $\sqsubset_{lnk}$	$\overline{a}z, \overline{a}(z), az, a(z)$
object $\ll_{obj}$	$\overline{y}a, ya$
enabling $\sqsubset'_{lnk}$	$\overline{a}z, \overline{a}(z), az, a(z), \overline{y}a, \overline{y}(a), ya$
multi-action	$\overline{a}z, \overline{a}(z), az, a(z), ya$

The link dependency is not appropriate in the sense of the concurrency. Because it allows transitions like $(\nu z)(\overline{x_1}z.z|x_2(y).\overline{y} + R) \xrightarrow[\mathrm{M}]{\overline{x_1}(z)|x_2 z}$. This transition do not satisfy confluence (see Section 5). Furthermore, such transitions do not contribute to preserve $\underset{\mathrm{M}}{\sim}$ for parallel compositions. Because agent $(\nu z)(\overline{x_1}z.z|x_2(y).\overline{y} + R)$ always can perform $\overline{x_1}(z)|x_2(z')$ with fresh name z', $\underset{\mathrm{M}}{\sim}$ is preserved. The object dependency and the enabling dependency can not make $\underset{\mathrm{M}}{\sim}$ congruence for parallel compositions. For example, let $P \stackrel{def}{=} (\nu z)(\overline{x_1}z|\overline{x_2}z)$ and $Q \stackrel{def}{=} (\nu z)(\overline{x_1}z.\overline{x_2}z) + (\nu z)(\overline{x_2}z.\overline{x_1}z)$. Both dependencies prohibit agent P to perform $\overline{x_1}(z)|\overline{x_2}z$. Thus, $P \underset{\mathrm{M}}{\sim} Q$. But parallel composed agent $P|(x_1(y)|x_2(y))$ is not strongly bisimilar to $Q|(x_1(y)|x_2(y))$. Because $Q|(x_1(y)|x_2(y)) \overset{\tau|\tau}{\underset{\mathrm{M}}{\not\longrightarrow}}$.

Discussion 4.6 In the previous Discussion 4.5, we focus the transition $P|(x_1(y)|x_2(y)) \overset{\tau|\tau}{\underset{\mathrm{M}}{\not\longrightarrow}}$. This transition raise a question whether we should identify τ and $\tau|\tau$ in a weak bisimilarity.

Let us consider the following systems:

$$
\begin{aligned}
& Mem \stackrel{def}{=} \overline{read}\ v.Mem + write(v).Mem \\
& Exec \stackrel{def}{=} exec(v).\tau.\overline{write}\ v.Exec \\
& \begin{cases} Fetch_1 \stackrel{def}{=} read(v).\overline{decode}\ v.Fetch_1 \\ Decode_1 \stackrel{def}{=} decode(v).\overline{exec}\ v.Decode_1 \end{cases} \\
& Sys \stackrel{def}{=} (\nu\{decode, read\})(Fetch_1|Decode_1|Mem) \\
& \begin{cases} Fetch_2 \stackrel{def}{=} read(v).(\overline{decode}\ v|Fetch_2) \\ Decode_2 \stackrel{def}{=} decode(v).(\overline{exec}\ v|Decode_2) \end{cases} \\
& PLSys \stackrel{def}{=} (\nu\{decode, read\})(Fetch_2|Decode_2|Mem).
\end{aligned}
$$

If we adopt the view of "$\tau = \tau|\tau$", then system Sys and $PLSys$ are weakly multi-action bisimilar, furthermore weakly multi-action congruent. Thus whole system $Sys|Exec$ and $PLSys|Exec$ are regarded as same. But we consider the system $PLSys|Exec$ is more desirable because $PLSys$ is pipe-lined. Sys performs fetch–decode stage sequentially as the transition $\xrightarrow[\mathrm{M}]{\tau}\xrightarrow[\mathrm{M}]{\tau}$. On the other hand, $PLSys$ can perform fetch–decode stage concurrently (or in parallel) as the transition $\xrightarrow[\mathrm{M}]{\tau|\tau}$. If we refine an agent via its *degree of the concurrency*, then we should distinguish τ and $\tau|\tau$.

Theorem 4.7 $\underset{\mathrm{M}}{\sim}$ is closed under input-prefixings.

Outline of Proof: Let a relation $\mathcal{R} \stackrel{def}{=} \{((\nu Z)(P\theta), (\nu Z)(Q\theta)) \mid P \underset{\mathrm{M}}{\sim} Q\}$. We show that $\mathcal{R}$ is strong ground multi-action bisimilar up to $\underset{\mathrm{M}}{\sim}$. Consider $P\theta$. We prove that $P\theta \xrightarrow[\mathrm{M}]{A} P' \implies \exists Q'.\ Q\theta \xrightarrow[\mathrm{M}]{A} Q'$ and $P' \underset{\mathrm{M}}{\sim} \mathcal{R} \underset{\mathrm{M}}{\sim} Q'$. In the case analysis of A, the most significant case is $\exists A''.\ A = \tau|A''$. When $P\theta \xrightarrow[\mathrm{M}]{A} P'$, there always exists a multi-action A' and a multi-set of names Z that satisfy $P \xrightarrow[\mathrm{M}]{A'} P''$ and $(\nu Z)(P''\theta) \equiv P'$. A' and Z are obtained from the inference tree of the transition $P\theta \xrightarrow[\mathrm{M}]{\tau} P'$ constructively. By the definition of $\mathcal{R}$, $P \underset{\mathrm{M}}{\sim} Q$. Thus, there exists Q'' that satisfies $Q \xrightarrow[\mathrm{M}]{A'} Q''$ and $P'' \underset{\mathrm{M}}{\sim} Q''$. By the definition of $\mathcal{R}$, $(\nu Z)(P''\theta)\mathcal{R}(\nu Z)(Q''\theta)$. On the other hand, for same A, A' and Z, when $Q \xrightarrow[\mathrm{M}]{A'} Q''$, there always exists Q' that satisfies $Q\theta \xrightarrow[\mathrm{M}]{A} Q'$ and $(\nu Z)(Q''\theta) \equiv Q'$. $\equiv \subset \underset{\mathrm{M}}{\sim}$. Therefore, when $P\theta \xrightarrow[\mathrm{M}]{A} P'$, there exists Q' that satisfies $Q\theta \xrightarrow[\mathrm{M}]{A} Q'$ and $P' \underset{\mathrm{M}}{\sim} \mathcal{R} \underset{\mathrm{M}}{\sim} Q'$. $\square$

Discussion 4.8 If we introduce the notion of multi-action for congruence of (strong) bisimulation, then it is sufficient to introduce double-actions. A double-action is a multi-action the length of that is one or two. But double-actions can not describe the concurrency. For example, suppose *priority* composition (denoted by $\triangleright$) and the following transition rule:

$$
\frac{P \xrightarrow{a} P'}{P \triangleright Q \xrightarrow{a} P'|Q.}\ Priority
$$

In double-action framework, agent $a|\overline{b}|c$ and $(a|\overline{b})\triangleright c+(c|\overline{b})\triangleright a+(a|c)\triangleright\overline{b}$ are strongly bisimilar and its bisimilarity is preserved in any context. But the former agent performs a, $\overline{b}$, c concurrently and the later agent can not perform those actions concurrently. Similar example can be shown for weak bisimilarity without priority compositions.

5 Concurrency

The concurrency described via multi-actions is characterized by *confluence* of transitions. To show that, we refine the multi-action transition relation $\xrightarrow[\mathrm{M}]{}$. And we define another multi-action transition relation $\xrightarrow[\mathrm{M}]{}{}''$ using a notion of confluence. Finally, we obtain that $\xrightarrow[\mathrm{M}]{} = \xrightarrow[\mathrm{M}]{}{}''$.

Definition 5.1 Let $=_b$ be the smallest binary relation over $\mathcal{A}_{Act,|}$ that satisfies the following conditions

- $A =_s B \Longrightarrow A =_b B$, and
- $\overline{x_1}(z)|\overline{x_2}z|A =_b \overline{x_1}z|\overline{x_2}(z)|A$.

We can consider $\mathcal{A}_{Act,|}/_{=_b}$ and the operation $|_{\mathcal{A}'}$ on this set in the similar way to $|_{\mathcal{A}}$. In the following, we abbreviate $\mathcal{A}_{Act,|}/_{=_b}$ as $\mathcal{A}'$

Lemma 5.2 Let $M^{\mathrm{i/e}}_{\mathrm{strong}} \stackrel{def}{=} (\mathcal{A}' \cup \{\emptyset\}, \emptyset, |_{\mathcal{A}'})$. Then, $M^{\mathrm{i/e}}_{\mathrm{strong}}$ is a commutative monoid and $\emptyset$ is the unit element.

Now, we re-define the transition relation $\xrightarrow[\mathrm{M}]{}$ using $M^{\mathrm{i/e}}_{\mathrm{strong}}$.

Definition 5.3 $\xrightarrow[\mathrm{M}]{}{}' \stackrel{def}{=} \{(P, [A]_{=_b}, P') \mid P \xrightarrow[\mathrm{M}]{A} P'\}$

Theorem 5.4 $P \xrightarrow[\mathrm{M}]{\overline{x_1}(z)|\overline{x_2}z|A} P' \Longrightarrow P \xrightarrow[\mathrm{M}]{\overline{x_1}z|\overline{x_2}(z)|A} P'$.

Remark 5.5 Theorem 5.4 means that $\xrightarrow[\mathrm{M}]{}{}'$ and $\xrightarrow[\mathrm{M}]{}$ have same computations. And the information "which action extrude the name" in a multi-action are not needed for our purpose. If these information make some sense, then $\underset{\mathrm{M}}{\sim}$ is not preserved for parallel compositions (see Discussion 4.5). Required information for a multi-action are "which names are extruded". To emphasis this property, we introduce the notation defined in Notation 5.6.

In the following, we use the $\xrightarrow[\mathrm{M}]{}$ as the $\xrightarrow[\mathrm{M}]{}{}'$.

Notation 5.6 Let $[A]_{=_b}$ be an element of $M^{\mathrm{i/e}}_{\mathrm{strong}}$. We denote this element as $(\nu Z)A'$ where $Z = obj(bo(A))$ and $A' = bo2fo(A)$. A $(\nu Z)A$ is *valid* if and only if $Z \subseteq_M obj(fo(A))$.

We define another multi-action transition relation $\xrightarrow[\mathrm{M}]{}{}''$ using the confluence of transitions.

Definition 5.7 (link independency predicate) We define a binary predicate $\blacktriangleright$ over $\mathcal{A}'$. Let $(\nu Z_1)A_1, (\nu Z_2)A_2 \in \mathcal{A}'$.

$$(\nu Z_1)A_1 \blacktriangleright (\nu Z_2)A_2 \Longleftrightarrow obj(fo((\nu Z_1)A_1)) \cap Z_2 = \emptyset$$

where $(\nu Z_1)A_1$ and $(\nu Z_2)A_2$ are valid.

Example 5.8 $(\nu z)(\overline{x_1}z|\overline{x_2}z|\overline{x_3}y) \blacktriangleright \overline{x_1}y|\overline{x_2}z$. And $(\nu z, z)(\overline{x_1}z|\overline{x_2}z|\overline{x_3}y) \blacktriangleright (\nu z)(\overline{x_1}y|\overline{x_2}z)$. But $(\nu z, z)(\overline{x_1}z|\overline{x_2}z|\overline{x_3}y) \not\blacktriangleright (\nu z, y)(\overline{x_1}y|\overline{x_2}z)$.

Definition 5.9 (Confluence) Let A_1, A_2 be multi-actions, P, P' be agents and Z be a multi-set of names. $A_1 \smile_{P,P',Z} A_2$ if and only if the following condition is satisfied:

> For every $Z_{11}, Z_{12}, Z_{21}, Z_{22}$, those satisfy $Z_{11}|Z_{12} = Z_{21}|Z_{22} = Z$, $(\nu Z_{11})A_1 \blacktriangleright (\nu Z_{12})A_2$ and $(\nu Z_{21})A_2 \blacktriangleright (\nu Z_{22})A_1$, the following diagram is commuted.
>
> $$\begin{tikzcd}
> & \mathcal{C}[P_1, P_2] \arrow[dl, "(\nu Z_{11})A_1"'] \arrow[dr, "(\nu Z_{21})A_2"] & \\
> \mathcal{C}_1[P_1', P_2] \arrow[dr, "(\nu Z_{12})A_2"'] & & \mathcal{C}_2[P_1, P_2'] \arrow[dl, "(\nu Z_{22})A_1"] \\
> & \mathcal{C}'[P_1', P_2'] &
> \end{tikzcd}$$
>
> where $\mathcal{C}$, $\mathcal{C}'$, $\mathcal{C}_1$ and $\mathcal{C}_2$ are two-hole contexts. The transition $\mathcal{C}[P_1, P_2] \xrightarrow[\text{M}]{(\nu Z_{11})A_1} \mathcal{C}_1[P_1', P_2]$ must be caused by the internal transition $P_1 \xrightarrow[\text{M}]{(\nu Z_{11}')A_1} P_1'$, where Z_{11}' is a (multi-set) subset of Z_{11}. The other transitions must be also restricted to transitions caused by P_1 or P_2.

Definition 5.10

$$P \xrightarrow[\text{M}]{(\nu Z)A}{}'' P' \stackrel{def}{=} \begin{cases} P \xrightarrow{a} P', & \text{if } (\nu Z)A = a \\ \text{for any } A_1|A_2 = A.\ A_1 \smile_{P,P',Z} A_2, & \text{otherwise} \end{cases}$$

Theorem 5.11 $\xrightarrow[\text{M}]{} = \xrightarrow[\text{M}]{}''$.

Remark 5.12 By Theorem 5.11, $P \xrightarrow[\text{M}]{a|b} P' \Longrightarrow P \xrightarrow[\text{M}]{a}\xrightarrow[\text{M}]{b} P'$ and $P \xrightarrow[\text{M}]{b}\xrightarrow[\text{M}]{a}$. This is similar to the expansion law. But please note that $P \xrightarrow[\text{M}]{a}\xrightarrow[\text{M}]{b} P'$ and $P \xrightarrow[\text{M}]{b}\xrightarrow[\text{M}]{a} \not\Longrightarrow P \xrightarrow[\text{M}]{a|b} P'$. The confluence defined by Definition 5.9 requires some kind of *locations* to be preserved.

6 Comparing with other approaches

In this section, we compare the multi-action approach with other approaches, *location*[3][4][5] and *causality*[6][7][8][9]. For other approaches, for instance, *Petri-Net*[10] and *graph-rewriting*[11], we can adopt the same discussion to the causality.

We mention the recent truly-concurrent congruence result[9].

6.1 Versus Locations

One agent is made by parallel composition of some sub-agents. A *location* is the information which sub-agents an action occurred at. That is, the location is the "birthplace" of the action. The location and location based semantics (location bisimulation) are proposed to distinguish agents via the *spatial distribution* essentially.

On the other hand, actions whose locations are not related each other have possibility to be performed concurrently. Thus, we can consider to describe the concurrent behavior of agents via locations. But such actions can not be performed concurrently in general. Let us see this example:

$$\begin{aligned} P_1 &\stackrel{def}{=} (\nu z)(\overline{x}z.\overline{z}) \\ Q_1 &\stackrel{def}{=} (\nu z)(\overline{x}z|\overline{z}). \end{aligned}$$

The location bisimulation distinguishes P_1 and Q_1. Because Q_1 has a computation $Q_1 \xrightarrow[l_0]{\overline{x}(z)} \xrightarrow[l_1]{\overline{z}} 0$, but P_1 does not. P_1 has a computation $P_1 \xrightarrow[l_0]{\overline{x}(z)} \xrightarrow[l_0 l_1]{\overline{z}} 0$ only. In the sense of the concurrency via locations, we can explain that "P_1 and Q_1 are distinguished because Q_1 can perform action $\overline{x}(z)$ and $\overline{z}$ concurrently (the locations of these actions are not related) but P_1 performs these actions sequentially (the location l_0 is the prefix of $l_0 l_1$)". But, in fact, both P_1 and Q_1 perform $\overline{x}(z)$ and $\overline{z}$ sequentially. We have no need to distinguish P_1 and Q_1. Another example is shown:

$$\begin{aligned} P_2 &\stackrel{def}{=} a|b \\ Q_2 &\stackrel{def}{=} (\nu c)(a.\overline{c}|c.b) + (\nu c)(b.c|\overline{c}.a) \end{aligned}$$

The location bisimulation identifies P_2 and Q_2 because both of these agents can perform action a and b concurrently. But, in fact, Q_2 can not perform these actions concurrently. We should distinguish these agents. This miss-identification is more serious than the previous example. Because agent $x(b).P_2$ and $x(b).Q_2$ are no longer identified by the location bisimulation. This shows that the location bisimulation is not closed under (input) prefixings.

In the multi-action bisimulation, these problems do not occur. The multi-action bisimulation identifies P_1 and Q_1. Because both of these agent can only perform $\xrightarrow[\mathrm{M}]{\overline{x}(z)} \xrightarrow[\mathrm{M}]{\overline{z}}$. These are sequential agents. On the other hand, agents P_2 and Q_2 are distinguished. Because P_2 can perform $a|b$ but Q_2 can not. That is, P_2 is a concurrent agent but Q_2 is a sequential agent.

6.2 Versus Causality

Causality is a set of dependencies between transitions. The causality is denoted by a labelled tree or a partial order introduced over transitions. By causality based semantics, we consider that transitions those have no dependencies each other can be performed concurrently. For the agents appeared in Section 6.1, we can extract the following causalities (

dependencies) and concurrency.

agent	transitions	dependencies	concurrency
P_1	$P_1 \xrightarrow{\overline{x}(z)}\xrightarrow{\overline{z}}$	$\overline{x}(z) \sqsubseteq \overline{z}$	–
Q_1	$Q_1 \xrightarrow{\overline{x}(z)}\xrightarrow{\overline{z}}$	$\overline{x}(z) \sqsubseteq \overline{z}$	–
P_2	$P_2 \xrightarrow{a}\xrightarrow{b}$ or $P_2 \xrightarrow{b}\xrightarrow{a}$	identity	$a \asymp b$
Q_2	$Q_2 \xrightarrow{a}\xrightarrow{b}$ or $Q_2 \xrightarrow{b}\xrightarrow{a}$	$a \sqsubseteq b$ or $b \sqsubseteq a$	–

(where a dependency is denoted as a partial order over transitions and each transition is displayed by whose labels. $a \asymp b$ means that the transition labelled a and the transition labelled b is concurrent. The dependencies of Q_1 and Q_2 occur because restriction (νz) or prefix $c, \overline{c}$ work as *sequentializers.*) Like this, the concurrency based on the causality is same to the concurrency based on multi-actions. We can use both semantics to capture the concurrent behavior of agents. But the causality requires analysis of dependencies of agents to do that. On the other hand, multi-actions can treat that directly. For example, let us suppose agent $R \stackrel{def}{=} (a.b)|(c.d)$. We can extract the dependencies and concurrency from transitions of R.

agent	dependencies	concurrency
$(a.b)\|(c.d)$	$a \sqsubseteq b,\ c \sqsubseteq d$	$a \asymp c,\ a \asymp d,\ b \asymp c,\ b \asymp d$

The analysis of semantics is required to capture the following concurrent behavior of R:

$$R \xrightarrow{a|c}\xrightarrow{b}\xrightarrow{d},\ R \xrightarrow{a}\xrightarrow{b|c}\xrightarrow{d},\ R \xrightarrow{c}\xrightarrow{a|d}\xrightarrow{b},\cdots.$$

This analysis is not easy. On the other hand, multi-actions collect these transitions directly.

6.3 Versus Other Truly-Concurrent Congruence Relation

In the recent paper[9], it is shown that the weak causality semantics (weak causal bisimulation) is a congruence relation in the π-calculus with the slight syntactic restriction.

We can define weak ground multi-action bisimilarity by employing τ as the unit element of M_{strong}, instead of $\emptyset$. We call this commutative monoid M_{weak}. We can also define weak multi-action congruence.

7 Conclusion

We extended π-calculus to use the multi-actions as labels. The multi-action π-calculus can describe the concurrent behavior of agents and its transitions are characterized by confluence. We proposed a new truly-concurrent semantics for π-calculus, strong bisimilarity defined over multi-action π-calculus. Strong bisimilarity defined over multi-action π-calculus is closed under input-prefixings and parallel compositions.

In the future works, we propose the weak multi-action bisimilarity and congruence.

References

[1] Robin Milner, Joachim Parrow, and David Walker. A Calculus of Mobile Processes, Parts I and II. *Information and Computation*, 100:1–77, 1992.

[2] Robin Milner. *Communication and Concurency.* Prentice Hall, 1989.

[3] G. Boudol, I. Castellani, Matthew Hennessy, and A. Kiehn. A Theory of Processes with Localities (Extended Abstract). In *Proc. of 3rd CONCUR*, volume 630 of LNCS. Springer-Verlag, 1992.

[4] G. Boudol, I. Castellani, M. Hennessy, and A. Kiehn. Observing localities. *Theoretical Computer Science*, 114:31–61, 1993.

[5] Davide Sangiorgi. Locality and True-concurrency in calculi for mobile processes. In *Proc. of TACS*, volume 789 of LNCS. Springer-Verlag, 1994.

[6] Pierpaolo Degano and Corrado Priami. Causality for Mobile Processes. In *Proc. of ICALP*, 1995.

[7] Michele Boreale and Davide Sangiorgi. A fully abstract semantics for causality in the π-calculus. ECS-LFCS-94 297, Univ. of Edinburgh, 1994.

[8] Michele Boreale and Davide Sangiorgi. A fully abstract semantics for causality in the π-calculus. In *Proc. of 12th STACS*, volume 900 of LNCS. Springer-Verlag, 1995.

[9] Michele Boreale and Davide Sangiorgi. Some Congruence Properties for π-calculus Bisimilarities. Technical Report RR-2870, INRIA, Sophia-Antipolis, 1996.

[10] Nadia Busi and Roberto Gorrieri. A Petri Net Semantics for π-calculus. In *Proc. of 6th CONCUR*, volume 962 of LNCS. Springer-Verlag, 1995.

[11] Ugo Montanari and Marco Pistore. Concurrent Semantics for the π-calculus. In *Proc. of MFPS*, volume ENCS 1. Elsevier, 1995.

[12] J.C.M. Baeten and J.A. Bergstra. Non Interleaving Process Algebra. In *Proc. of 4th CONCUR*, volume 715 of LNCS. Springer-Verlag, 1993.