Multi-action $\pi$-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 $\pi$-calculus\cite{1}. We extend the labelled transition system of the $\pi$-calculus to use multi-sets of actions as labels. We call this extension multi-action $\pi$-calculus. The multi-action $\pi$-calculus can describe concurrent behavior of agents. Strong bisimilarity defined in the multi-action $\pi$-calculus is closed under input-prefixings and parallel compositions.

1 Introduction

In recent years, studies of process algebra, especially the $\pi$-calculus\cite{1}, have a lot of attention. The $\pi$-calculus is an extension of CCS\cite{2}. Agents of the $\pi$-calculus can modify their linkage structures dynamically via name-passing mechanism. The observation equivalence is proposed as a semantics, an equivalence relation over the $\pi$-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\cite{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 $\pi$-calculus, the observation equivalence (even strong ground bisimilarity) is not closed under input prefixings. That is, the expansion law is not correct in the $\pi$-calculus. For example, $\overline{x}z\tau|y(w) \sim \overline{x}z.y(w) + y(w).\overline{x}z$ but $x'(y).(\overline{x}z|y(w)) \not\sim x'(y).((\overline{x}z.y(w) + y(w).\overline{x}z)$. Because the left-side agent can perform a $\tau$ 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 $\pi$-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 $a.P|b.Q|(c+R)|z\rightarrow a[b]\tau_{\backslash }P|Q$. We extend the labelled transition system of the $\pi$-calculus to use multi-actions as labels and define bisimilarities over the extended $\pi$-calculus. We call this extension of the $\pi$-calculus multi-action $\pi$-calculus.

Multi-action approaches are already proposed for non-mobile calculi[12]. But we cannot adopt these works simply for the $\pi$-calculus. Because the special treatment for name-extrusion is required. In the $\pi$-calculus, restriction operators, denoted by $(\nu z)$, also work as sequentializers by the different way to the CCS. For example, let us consider an agent $P \overset{def}{=} \overline{x}.Q_1|z(y).Q_2|w(y)$. 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 $\pi$-calculus, $(\nu z)P$ performs action $w(y)$ and $\overline{x}(z)$ followed by $z(y)$. Because $\overline{x}(z)$ eliminates $(\nu z)$ from the agent. But $z(y)$ cannot 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 $\pi$-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 $\pi$-calculus and its bisimilarity in section 2. We propose the multi-actions and the multi-action $\pi$-calculus in section 3. And we propose strong bisimilarity for the multi-action $\pi$-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 $\pi$-calculus

In this section, we define the $\pi$-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 $\text{Act}$ defined as follows:

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

We let $x, y, \cdots$ range over $\mathcal{N}$ and $a, b, \cdots$ range over $\text{Act}$. 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 $\text{P}$ of all terms generated by the following rule:

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

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 $bn(\cdot)$, $fn(\cdot)$, $n(\cdot)$, $\text{obj}(\cdot)$, $\text{sub}(\cdot)$ 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 $\theta : \mathcal{N} \rightarrow \mathcal{N}$. We write application of a name-substitution $\theta$ to name $x$ as $x\theta$. Let $\iota$ be the identity name-substitution and $\{x'/x\}$ is the same name-substitution to $\iota$ 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 $L = (\mathcal{P}, \text{Act}, \rightarrow)$ where $\rightarrow \subset \mathcal{P} \times \text{Act} \times \mathcal{P}$ is the relation given by the transitive closure of
the following inference rules.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Premise</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Prefix</strong></td>
<td>$\alpha.P \xrightarrow{a} P$</td>
<td>$x(y).P \xrightarrow{z} P[z/y]$</td>
</tr>
<tr>
<td><strong>Sum</strong></td>
<td>$P \xrightarrow{a} P'$</td>
<td>$P + Q \xrightarrow{a} P'$</td>
</tr>
<tr>
<td><strong>Input</strong></td>
<td>$P \xrightarrow{a} P'$</td>
<td>$z \notin n(a)$</td>
</tr>
<tr>
<td><strong>Restriction</strong></td>
<td>$\frac{P \xrightarrow{a} P'}{(\nu z)P \xrightarrow{a} (\nu z)P'}$</td>
<td>$\frac{(\nu z)Parrow(a \nu z)P'}{(\nu z)Parrow(a \nu z)P'}$</td>
</tr>
<tr>
<td><strong>Communication</strong></td>
<td>$P \xrightarrow{(\nu z)Parrow(a \nu z)P'}$</td>
<td>$Q \xrightarrow{z} Q'$</td>
</tr>
<tr>
<td><strong>Close</strong></td>
<td>$P \xrightarrow{(\nu z)Parrow(a \nu z)P'}$</td>
<td>$Q \xrightarrow{z} Q'$</td>
</tr>
<tr>
<td><strong>Open</strong></td>
<td>$P \xrightarrow{(\nu w)Parrow(a \nu w)P'}$</td>
<td>$Q \xrightarrow{(\nu w)Parrow(a \nu w)P'}$</td>
</tr>
<tr>
<td><strong>Structure</strong></td>
<td>$P \xrightarrow{(\nu z)Parrow(a \nu z)P'}$</td>
<td>$Q \xrightarrow{(\nu z)Parrow(a \nu z)P'}$</td>
</tr>
<tr>
<td><strong>Parallel</strong></td>
<td>$P \xrightarrow{a} P'$</td>
<td>$P</td>
</tr>
</tbody>
</table>

**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 $R \subseteq P \times P$ is a strong ground bisimulation if and only if for every $(P, Q) \in R$,

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

**Definition 2.9** $\sim \overset{\text{def}}{=} \text{(the largest strong ground bisimulation).}$

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

$$P_0 \overset{\text{def}}{=} \overline{xz}y(w)$$
$$Q_0 \overset{\text{def}}{=} \overline{xz}y(w) + y(w).\overline{xz}$$

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

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

3 Multi-action $\pi$-calculus

In this section, we extend the $\pi$-calculus. The extended calculus, multi-action $\pi$-calculus, uses the same syntax to the $\pi$-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 $\mathcal{A}_{\text{Act}}$ of terms generated from the rule:

$$ A ::= \text{Act} | A | A. $$

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

Let $=_{s}$ be the smallest binary relation over $\mathcal{A}_{\text{Act}}$ 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 $\mathcal{A}_{\text{Act}}/_{=_{s}}$ (the quotient set of $\mathcal{A}_{\text{Act}}$ by $=_{s}$) as $A$, and $[A]_{=_{s}}$ as $A$.

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

$$ A \mid_{A} B \overset{\text{def}}{=} \begin{cases} A & B = \emptyset,\\ B & A = \emptyset,\\ A|B & \text{otherwise.} \end{cases} $$

More specifically, the final line means that $[A]_{=_{s}} \mid_{A} [B]_{=_{s}} \overset{\text{def}}{=} [A|B]_{=_{s}}$.

**Definition 3.3** Let $a, b \in \text{Act}$. $a$ 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**

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

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

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

Intuitively, $A \triangleleft B$ invokes interactions between communicatable actions in $A$ and $B$ simultaneously. TIE($A, B$) gives names required to bind after interactions between bound actions in $A$ and $B$. 
**Example 3.5**

\[
\overline{a}(z)|b(w)|c\overline{y}|d \lor \overline{b}(w)|a(z)|\overline{c}y = \tau|\tau|d
\]

\[
\mathrm{TIE}( \overline{a}(z)|b(w)|c\overline{y}|d, \overline{b}(w)|a(z)|\overline{c}y ) = \{w, z\}
\]

**Lemma 3.6** Let \( M_{\text{strong}} \) be defined as \((\mathcal{A} \cup \emptyset, \emptyset, |_{\mathcal{A}})\). Then, \( M_{\text{strong}} \) is a commutative monoid and \( \emptyset \) is the unit element of \( M_{\text{strong}} \).

**Definition 3.7 (Multi-action)** We call each element of \( M_{\text{strong}} \) 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 \( \Pi_{i=1}^{n} A_{i} \) 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. \( \{a|b, a|a|b, a|b|\tau\} \), \( \{a|a, \tau, \tau|\tau\} \) and \( \{a|\tau, a\} \) are sets of different multi-actions.

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

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

\( fi(A) \) 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:

\[
bo2fo(A) \triangleq \begin{cases} 
\overline{xy}|bo2fo(A') & \text{if } \exists x, y. \ A = \overline{x}(y)|A', \\
A & \text{otherwise.}
\end{cases}
\]

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

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

Prefix  \[
\begin{align*}
\alpha.P \xrightarrow{\alpha} P
\end{align*}
\]

Input  \[
\begin{align*}
x(y).P \xrightarrow{\alpha} P\{z/y\}
\end{align*}
\]

Sum  \[
\begin{align*}
P \xrightarrow{\alpha} P' \\
P + Q \xrightarrow{\alpha} P'
\end{align*}
\]

Restriction  \[
\begin{align*}
P \xrightarrow{\alpha} P' \\
\phi \notin n(A) \quad z \notin n(A)
\end{align*}
\]

Communication  \[
\begin{align*}
P | Q \xrightarrow{\alpha} P' Q' \\
\phi \notin n(A) \quad (\nu z)P \xrightarrow{\alpha} (\nu z)P'
\end{align*}
\]

Open  \[
\begin{align*}
P \equiv Q \\
Q \xrightarrow{\alpha} Q' \\
Q' \equiv P' \quad \frac{P \equiv Q}{P | Q \xrightarrow{\alpha} P' | Q'}
\end{align*}
\]

Parallel  \[
\begin{align*}
P \equiv Q \\
P \xrightarrow{\alpha} P' \\
Q \xrightarrow{\alpha} P' \quad \frac{P \equiv Q}{P | Q \xrightarrow{\alpha} P' | Q'}
\end{align*}
\]

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

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 $a|b(w).P|(\nu z)(\overline{b}z|z.Q)$ by the following inference:

\[
\begin{align*}
\frac{a \xrightarrow{\alpha}}{b(w).P \xrightarrow{\alpha} P\{z/w\}x|z.Q} \\
\frac{\overline{b}z \xrightarrow{\alpha}}{b(w).P \xrightarrow{\alpha} P\{z/w\}x|z.Q} \\
\frac{(\nu z)(\overline{b}z|z.Q) \xrightarrow{\alpha}}{(\nu z)(\overline{b}z|z.Q) \xrightarrow{\alpha}}
\end{align*}
\]

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

4 Strong Bisimilarity

In this section, we propose strong ground bisimilarity for the multi-action $\pi$-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 $(P, Q) \in \mathcal{R}$,

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

**Definition 4.2** $\overset{\text{def}}{\sim}_M$ (the largest strong ground multi-action bisimulation).

We also define strong ground multi-action bisimulation up to $\overset{\text{def}}{\sim}_M$ in usual way. We can show that if a relation $\mathcal{R}$ is a strong ground multi-action bisimulation up to $\overset{\text{def}}{\sim}_M$ then $\overset{\text{def}}{\sim}_M \mathcal{R} \overset{\text{def}}{\sim}_M$ is a strong ground multi-action bisimulation.

**Theorem 4.3** $\overset{\text{def}}{\sim}_M$ is closed under $\tau$-prefixings, output-prefixings, restrictions and sum compositions.

**Theorem 4.4** $\overset{\text{def}}{\sim}_M$ is closed under parallel compositions.

**Discussion 4.5** By the side-conditions of the Open Rule in the **Definition 3.10**, $\overset{\text{def}}{\sim}_M$ is preserved for parallel compositions. The side-condition $w \notin sub(A) \cup obj(f_i(A))$ 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.

<table>
<thead>
<tr>
<th>dependency</th>
<th>actions depending on $\overline{x}(a)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>link $\sqsubseteq_{\text{lnk}}$</td>
<td>$\overline{a}z, \overline{a}(z), az, a(z)$</td>
</tr>
<tr>
<td>object $\ll_{\text{obj}}$</td>
<td>$\overline{y}a, ya$</td>
</tr>
<tr>
<td>enabling $\sqsubseteq_{\text{lnk}}'$</td>
<td>$\overline{a}z, \overline{a}(z), az, a(z), \overline{y}a, \overline{y}(a), ya$</td>
</tr>
<tr>
<td>multi-action</td>
<td>$\overline{a}z, \overline{a}(z), az, a(z), ya$</td>
</tr>
</tbody>
</table>

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

**Discussion 4.6** In the previous **Discussion 4.5**, we focus the transition $P|(x_1(y)|x_2(y)) \not\xrightarrow{M}$. This transition raise a question whether we should identify $\tau$ and $\tau|\tau$ in a weak bisimilarity.
Let us consider the following systems:

\[
\begin{align*}
\text{Mem} & \overset{\text{def}}{=} \overline{\text{read~}v.\text{Mem~}+~\text{write~}v.\text{Mem}} \\
\text{Exec} & \overset{\text{def}}{=} \text{exec}(v).\tau.\text{write~}v.\text{Exec}
\end{align*}
\]

\[
\begin{align*}
\begin{cases}
\text{Fetch}_1 & \overset{\text{def}}{=} \text{read}(v).\text{decode~}v.\text{Fetch}_1 \\
\text{Decode}_1 & \overset{\text{def}}{=} \text{decode}(v).\text{exec~}v.\text{Decode}_1
\end{cases}
\end{align*}
\]

\[
\begin{align*}
\text{Sys} & \overset{\text{def}}{=} (\nu\{\text{decode, read}\})(\text{Fetch}_1|\text{Decode}_1|\text{Mem}) \\
\begin{cases}
\text{Fetch}_2 & \overset{\text{def}}{=} \text{read}(v).(\text{decode~}v\mid\text{Fetch}_2) \\
\text{Decode}_2 & \overset{\text{def}}{=} \text{decode}(v).(\text{exec~}v\mid\text{Decode}_2)
\end{cases}
\end{align*}
\]

\[
\begin{align*}
\text{PLSys} & \overset{\text{def}}{=} (\nu\{\text{decode, read}\})(\text{Fetch}_2|\text{Decode}_2|\text{Mem}).
\end{align*}
\]

If we adopt the view of \(\tau = \tau|\tau\)", then system \(\text{Sys}\) and \(\text{PLSys}\) are weakly multi-action bisimilar, furthermore weakly multi-action congruent. Thus whole system \(\text{Sys}|\text{Exec}\) and \(\text{PLSys}|\text{Exec}\) are regarded as same. But we consider the system \(\text{PLSys}|\text{Exec}\) is more desirable because \(\text{PLSys}\) is pipe-lined. \(\text{Sys}\) performs fetch-decode stage sequentially as the transition \(\frac{\tau_M}{\tau_{\chi}}\). On the other hand, \(\text{PLSys}\) can perform fetch-decode stage concurrently (or in parallel) as the transition \(\frac{\tau_M^1}{\tau_{\chi}^1}\). If we refine an agent via its degree of the concurrency, then we should distinguish \(\tau\) and \(\tau|\tau\).

**Theorem 4.7** \(\sim_M\) is closed under input-prefixings.

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

**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:

\[
\frac{P \xrightarrow{a_M} P'}{P \triangleright Q \xrightarrow{a_M} P'|Q}. \quad \text{Priority}
\]
In double-action framework, agent $a|b|c$ and $(a|b)|c+(c|b)|a+(a|c)|b$ are strongly bisimilar and its bisimilarity is preserved in any context. But the former agent performs $a$, $b$, $c$ 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 $\rightarrow_{M}$. And we define another multi-action transition relation $\overrightarrow{M}$ using a notion of confluence. Finally, we obtain that $\overrightarrow{M} = \overleftarrow{M}$.

**Definition 5.1** Let $\equiv_{b}$ be the smallest binary relation over $A_{Act|}$ that satisfies the following conditions

- $A \equiv_{b} B \implies A \equiv_{b} B$, and
- $\overline{x_{1}}(z)|\overline{x_{2}}z|A \equiv_{b} \overline{x_{1}}z|\overline{x_{2}}(z)|A$.

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

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

Now, we re-define the transition relation $\rightarrow_{M}$ using $M_{\text{strong}}^{i/e}$.

**Definition 5.3** $\overrightarrow{M} \overset{\text{def}}{=} \{(P, [A] \equiv_{b}, P') \mid P \xrightarrow{A} P' \}$

**Theorem 5.4** $P \xrightarrow{\overline{x_{1}}(z)|\overline{x_{2}}z|A} P' \implies P \xrightarrow{\overline{x_{1}}z|\overline{x_{2}}(z)|A} P'$.

**Remark 5.5** Theorem 5.4 means that $\overrightarrow{M}$ and $\overleftarrow{M}$ 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 $\overleftarrow{M}$ 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 $\overrightarrow{M}$ as the $\overrightarrow{M}$.

**Notation 5.6** Let $[A] \equiv_{b}$ be an element of $M_{\text{strong}}^{i/e}$. We denote this element as $(\nu Z)A'$ where $Z = \text{obj}(bo(A))$ and $A' = \text{bo}2\text{fo}(A)$. A $(\nu Z)A$ is valid if and only if $Z \subseteq M \text{obj}(\text{fo}(A))$.

We define another multi-action transition relation $\overrightarrow{M}$ using the confluence of transitions.
**Definition 5.7 (link independence predicate)** We define a binary predicate $\triangleright$ over $\mathcal{A}$. Let $(\nu Z_1)A_1, (\nu Z_2)A_2 \in \mathcal{A}$.

$$(\nu Z_1)A_1 \triangleright (\nu Z_2)A_2 \iff \text{obj}(fo((\nu Z_1)A_1)) \cap Z_2 = \emptyset$$

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

**Example 5.8** $(\nu z)(\overline{x_1}z|\overline{x_2}z|\overline{x_3}y) \triangleright \overline{x_1}y|\overline{x_2}z$ and $(\nu Z, Z)(\overline{X_1}Z|\overline{X_2}z|\overline{X_3}y) \triangleright (\nu z)(\overline{X_1}y|\overline{X_2}z)$.

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

For every $Z_{11}, Z_{12}, Z_{21}, Z_{22}$, those satisfy $Z_{11}|Z_{12} = Z_{21}|Z_{22} = Z$, $(\nu Z_{11})A_1 \triangleright (\nu Z_{12})A_2$ and $(\nu Z_{21})A_2 \triangleright (\nu Z_{22})A_1$, the following diagram is commuted.

$$
\begin{array}{c}
C[P_1, P_2] \\
(\nu Z_{11})A_1 & (\nu Z_{21})A_2 \\
C_1[P'_1, P'_2] & C_2[P_1, P'_2] \\
(\nu Z_{12})A_2 & (\nu Z_{22})A_1 \\
C'[P'_1, P'_2]
\end{array}
$$

where $C, C', C_1$ and $C_2$ are two-hole contexts. The transition $C[P_1, P_2] \xrightarrow{\nu Z_{11}} A_1$ $C_1[P'_1, P'_2]$ must be caused by the internal transition $P_1 \xrightarrow{\nu Z_{11}} A_1$, 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**

$$P \xrightarrow{\nu Z} P' \overset{\text{def}}{=} \left\{ P \xrightarrow{a} P', \ \text{if } (\nu Z)A = a \right\}$$

for any $A_1|A_2 = A$. $A_1 \vdash_{P, P', Z} A_2$, otherwise

**Theorem 5.11** $\xrightarrow{\mathcal{M}} = \xrightarrow{\mathcal{M}'}$.

**Remark 5.12** By Theorem 5.11, $P \xrightarrow{a|b} P' \implies P \xrightarrow{a | b} P'$ and $P \xrightarrow{b | a} P'$. This is similar to the expansion law. But please note that $P \xrightarrow{a | b} P'$ and $P \xrightarrow{b | a}$ $P'$. 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:

\[
P_1 \overset{\text{def}}{=} (\nu z)(\overline{x}z.z)
\]

\[
Q_1 \overset{\text{def}}{=} (\nu z)(\overline{x}z.z).
\]

The location bisimulation distinguishes \(P_1\) and \(Q_1\). Because \(Q_1\) has a computation \(Q_1 \overset{\overline{x}(z)}{\overset{l_0}{\rightarrow}} l_1\), but \(P_1\) does not. \(P_1\) has a computation \(P_1 \overset{\overline{x}(z)}{\overset{l_0}{\rightarrow}} 0\) 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 \(z\) concurrently (the locations of these actions are not related) but \(P_1\) performs these actions sequentially (the location \(l_0\) is the prefix of \(l_0l_1\)". But, in fact, both \(P_1\) and \(Q_1\) perform \(\overline{x}(z)\) and \(z\) sequentially. We have no need to distinguish \(P_1\) and \(Q_1\). Another example is shown:

\[
P_2 \overset{\text{def}}{=} a|b
\]

\[
Q_2 \overset{\text{def}}{=} (\nu c)(a.z|c.b) + (\nu c)(b.c|z.a)
\]

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 misidentification 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) prefixes.

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 \(\overline{x}(z)\). 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.

<table>
<thead>
<tr>
<th>agent</th>
<th>transitions</th>
<th>dependencies</th>
<th>concurrency</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>$P_1 \xrightarrow{\overline{x(z)}} \overline{y}$</td>
<td>$\overline{x(z)} \subseteq \overline{y}$</td>
<td>$-$</td>
</tr>
<tr>
<td>$Q_1$</td>
<td>$Q_1 \xrightarrow{\overline{x(z)}} \overline{y}$</td>
<td>$\overline{x(z)} \subseteq \overline{y}$</td>
<td>$-$</td>
</tr>
<tr>
<td>$P_2$</td>
<td>$P_2 \xrightarrow{a \rightarrow b}$ or $P_2 \xrightarrow{b \rightarrow a}$</td>
<td>identity</td>
<td>$a \asymp b$</td>
</tr>
<tr>
<td>$Q_2$</td>
<td>$Q_2 \xrightarrow{a \rightarrow b}$ or $Q_2 \xrightarrow{b \rightarrow a}$</td>
<td>$a \subseteq b$ or $b \subseteq a$</td>
<td>$-$</td>
</tr>
</tbody>
</table>

(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 $(\nu z)$ 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 $R \equiv (a, b)\mid (c, d)$. We can extract the dependencies and concurrency from transitions of $R$.

<table>
<thead>
<tr>
<th>agent</th>
<th>dependencies</th>
<th>concurrency</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(a, b)\mid (c, d)$</td>
<td>$a \subseteq b$, $c \subseteq d$</td>
<td>$a \asymp c$, $a \asymp d$, $b \asymp c$, $b \asymp d$</td>
</tr>
</tbody>
</table>

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

$R \xrightarrow{a \mid c, b \rightarrow d}$, $R \xrightarrow{a \rightarrow b \mid c, d}$, $R \xrightarrow{c \rightarrow a \mid d, b}$, $\ldots$.

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 $\pi$-calculus with the slight syntactic restriction.

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

### 7 Conclusion

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

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


