On the Expressiveness of Monadic Higher Order Safe Ambient Calculus

Zining Cao

1State Key Laboratory for Civil Aircraft Flight Simulation
Shanghai Aircraft Design and Research Institute
Shanghai 201210, China
2Department of Computer Science and Research Technology
Nanjing University of Aeronautics & Astronautics
Nanjing 210016, China
Email: caozn@nuaa.edu.cn

Abstract—In this paper, we propose a monadic higher order safe ambient calculus. The expressiveness of this calculus is studied. We showed that polyadic higher order safe ambient calculus, first order safe ambient calculus with capability-passing, first order safe ambient calculus with name-passing, and polyadic π-calculus can all be encoded in monadic higher order ambient calculus. At last, we show that synchronous monadic higher order ambient calculus can be encoded in asynchronous monadic higher order ambient calculus.

Keywords: Process Calculus; Higher Order Safe Ambient Calculus; Expressiveness.

1. Introduction

Mobile Ambients was proposed and studied intensively in [3]. The calculus of Mobile Ambients (MA) is proposed both as a core programming language for the Web and as a model for reasoning about properties of mobile processes, including security. In contrast with previous formalisms for mobile processes such as the π-calculus, whose computational model is based on the notion of communication, the MA computational model is based on the notion of movement. An ambient, which may be thought of as a named location, is the unit of movement. Processes within the same ambient may exchange messages; ambients may be nested, so to form a hierarchical structure. The three primitives for movement allow: an ambient to enter another ambient; an ambient to exit another ambient; a process to dissolve an ambient boundary thus obtaining access to its content. Elegant type systems for MA have been given; they control the type of values exchanged within an ambient and the mobility of ambients. A few variants of MA were proposed in literatures [1], [5], [8]. In the Safe Ambients calculus (SA) [8], for example, CCS-style co-actions are introduced into the calculus to control potential interferences from other ambients. Recently, there are several works about MA with higher order communication. In [2], authors proposed an extension of the ambient calculus in which processes can be passed as values. A filter model for this calculus was presented. This model was proved to be fully abstract with respect to the notion of contextual equivalence where the observables are ambients at top level. In [4], we present a higher order ambient calculus, which is a higher order extension of Safe Ambients calculus with passwords [10]. Furthermore, we present late bisimulation, quasi late bisimulation, concise quasi late bisimulation and quasi normal bisimulation for the higher order ambient calculus and study the relation between these bisimulations. In this paper, we propose a higher order extension of Safe Ambients calculus, named MHSA, and study its expressive power.

This paper is organized as follows: Section 2 gives a brief view of syntax and operational semantics of higher order ambient calculus. Then we also give the reduction barbed congruence. Section 3 we show that polyadic higher order ambient calculus can be encoded in monadic higher order ambient calculus. In Section 4 we show that first order ambient calculus with capability-passing can be encoded in monadic higher order ambient calculus. In Section 5 we show that first order ambient calculus with name-passing can be encoded in monadic higher order ambient calculus. In Section 6, we show that polyadic π-calculus can be encoded in monadic higher order ambient calculus. In Section 7, we show that the synchronous monadic higher order ambient calculus can be encoded in the asynchronous monadic higher order ambient calculus, which means that all process calculi in this paper can be encoded in the asynchronous monadic higher order ambient calculus. The paper is concluded in Section 8.

2. Monadic Higher Order Safe Ambient Calculus

In this section, we present a monadic higher order safe ambient calculus (named as MHSA), which is an extension of safe ambients by adding capability of higher order
communication. Mobile capabilities (in, out, and their co-capabilities) make ambient calculi born with the higher order property. But these mobile capabilities are linear, i.e., only one copy of a process can move, whereas higher order communication ((X) and ⟨P⟩) are non-linear higher order operators since more than one copy of a process can be communicated. Therefore MHSA extends SA with non-linear higher order communication capabilities.

2.1 Syntax and Labelled Transition System of MHSA

The formal definition of process is given as follows:

\[ P ::= 0 \mid X \mid (X).P \mid \langle P_1, P_2 \rangle \mid \text{in}(n).P \mid \text{out}(n).P \mid \text{open}(n).P \mid \text{in}(n).P \mid \text{out}(n).P \mid \text{open}(n).P \mid P_1|P_2 \mid (vn)P \mid n[P] \mid recX.P, \text{ where } n \in \mathbb{N} \text{ of names, } X \in \text{Var} \text{ of process variables.} \]

Informally, 0 denotes inaction. X is a process variable.

c.P can perform action c, where c is in the form of in(n), out(n), open(n), \overline{n}(n), \overline{o}(n), P_1|P_2, (vn)P, n[P], recX.P, where n \in \mathbb{N} of names, X \in Var of process variables.

Now we can give the concept of reduction barbed congruence for SA. In the remainder of this paper, we abbreviate P\{R\} as P(R). In the following, we use P \Longrightarrow P' to abbreviate P \Longrightarrow \cdots \Longrightarrow P'.

**Definition 1.** For each name n, the observability predicate \( \Downarrow_n \) is defined by

\[ P \Downarrow_n P' \text{ if } \exists P'' \text{ such that } P \Longrightarrow P' \equiv (\nu k)(n.c.P_1.P_2) \text{, where } k \in \{ k \} \]

**Definition 2.** A symmetric relation R \( \subseteq P_r^c \times P_r^c \) is a weak reduction barbed congruence if P R Q implies:

1. (C[P] R C[Q]) for any C[];
2. whenever P \Longrightarrow P' then there exists Q' such that \( Q \equiv Q' \) and P' R Q';
3. P \Downarrow_n Q implies Q \Downarrow_n.

We write P \( \approx_B \) Q if P and Q are weakly reduction barbed congruent.

3. Encoding of Polyadic Higher Order Safe Ambient Calculus

Now we introduce a polyadic higher order safe ambient calculus, then give an encoding which transforms this polyadic higher order safe ambient calculus into the monadic calculus. At last, we prove the full abstraction property of this encoding.

3.1 Syntax and Labelled Transition System

Intuitively, polyadic higher order safe ambient calculi are ambients which can send or receive many processes contemporaneously. The formal definition of processes of polyadic higher order safe ambients is given as follows:

\[ P ::= 0 \mid X \mid (X_1, \ldots, X_k).P \mid \langle P_1, \ldots, P_k \rangle.P \mid \text{in}(n).P \mid \text{out}(n).P \mid \text{open}(n).P \mid \text{in}(n).P \mid \text{out}(n).P \mid \text{open}(n).P \mid P_1|P_2 \mid (vn)P \mid n[P] \mid recX.P, \text{ where } n \in \mathbb{N} \text{ of names, } X \in Var \text{ of process variables.} \]
The operational semantics of processes is similar as Table 1 except that $COM$ is replaced by the following rule:

$COM : (X_1, \ldots, X_k).P | (Q_1, \ldots, Q_k).R \rightarrow P|Q_1/X_1, \ldots, Q_k/X_k|R$

### 3.2 Encoding Polyadic Higher Order Safe Ambient Calculus in MHSA

Now we show that polyadic higher order safe ambients can be simulated by monadic higher order safe ambients.

**Definition 3.** We give a mapping $Tr_{PS}\{\cdot\}^a$ with respect to name $a$ which transforms every polyadic higher order safe ambient $P$ into the monadic higher order safe ambient $Tr_{PS}\{P\}^a$. The mapping is defined inductively on the structure of $P$.

1. $Tr_{PS}\{\emptyset\}^a = 0$;
2. $Tr_{PS}\{X\}^a = X$;
3. $Tr_{PS}\{(X_1, \ldots, X_k).P\}^a = (X).a[X|\overline{\text{open}}(a)](X_1)\ldots(X_k).Tr_{PS}\{P\}^a$;
4. $Tr_{PS}\{(\nu p)((in(p)).0).\overline{\text{in}}(p),\overline{\text{open}}{\langle a\rangle}, Tr_{PS}\{P\}^a\} = \overline{\text{in}}(p).Tr_{PS}\{\nu p|\overline{\text{in}}(p)|\overline{\text{open}}\langle a\rangle|\overline{\text{rec}}X.P\}^a$;
5. $Tr_{PS}\{in(n).P\}^a = in(n).Tr_{PS}\{P\}^a$;
6. $Tr_{PS}\{out(n).P\}^a = out(n).Tr_{PS}\{P\}^a$;
7. $Tr_{PS}\{\overline{\text{open}}(n).P\}^a = \overline{\text{open}}(n).Tr_{PS}\{P\}^a$;
8. $Tr_{PS}\{\overline{\text{in}}(n).P\}^a = \overline{\text{in}}(n).Tr_{PS}\{P\}^a$;
9. $Tr_{PS}\{\overline{\text{out}}(n).P\}^a = \overline{\text{out}}(n).Tr_{PS}\{P\}^a$;
10. $Tr_{PS}\{\overline{\text{open}}(n).P\}^a = \overline{\text{open}}(n).Tr_{PS}\{P\}^a$;
11. $Tr_{PS}\{\overline{\text{in}}(n).P\}^a = in(n).Tr_{PS}\{P\}^a$;
12. $Tr_{PS}\{\overline{\text{out}}(n).P\}^a = out(n).Tr_{PS}\{P\}^a$;
13. $Tr_{PS}\{\nu p|\overline{\text{in}}(p)|\overline{\text{open}}\langle a\rangle|\overline{\text{rec}}X.P\}^a$;
14. $Tr_{PS}\{\overline{\text{rec}}X.P\}^a = \overline{\text{rec}}X.Tr_{PS}\{P\}^a$.

Now we can give the full abstraction property of encoding $Tr_{PS}\{\cdot\}^a$.

**Lemma 1.** For any polyadic higher order ambients $P$ and $Q_1, \ldots, Q_k$, $Tr_{PS}\{P\}^a|\overline{\text{rec}}X.P | Tr_{PS}\{Q\}^a|\overline{\text{rec}}X.P \approx_{ba} Tr_{PS}\{P|Q_1/X_1, \ldots, Q_k/X_k\}^a$, where $a \notin fn(P, Q_1, \ldots, Q_k)$.

**Lemma 2.** For any polyadic higher order ambients $P$ and $Q$, $P \implies Q \Leftrightarrow (\nu a)Tr_{PS}\{P\}^a \Leftrightarrow_{ba} (\nu a)Tr_{PS}\{Q\}^a$, where $a \notin fn(P) \cup fn(Q)$.

**Lemma 3.** For any polyadic higher order ambients $P$, $P \Downarrow n \Leftrightarrow (\nu a)Tr_{PS}\{P\}^a \Downarrow n$, where $a \notin fn(P)$.

The definition (of weak) reduction barbed congruence $\approx_{ba}$ for polyadic higher order ambients calculus is the same as Definition 2.

**Proposition 1.** For any polyadic higher order ambients $P$ and $Q$, $P \approx_{ba} Q \Leftrightarrow (\nu a)Tr_{PS}\{P\}^a \approx_{ba} (\nu a)Tr_{PS}\{Q\}^a$, where $a \notin fn(P) \cup fn(Q)$.

**Proof:** By Lemmas 1, 2 and 3.

The above results show that the polyadic higher order ambients calculus can be encoded in the monadic higher order ambients calculus.

### 4. Encoding of First Order Ambient Calculus with Capability-Passing

For first order ambient calculus with communication, there are two kinds of communication, one is capability-passing, i.e., processes can send or receive capabilities; another is name-passing, i.e., processes can send or receive names. We will prove that both these calculus can be encoded in monadic higher order safe ambient calculus.

In this section, we prove that ambient calculus with capability-passing can be encoded in monadic higher order safe ambient calculus.

#### 4.1 Syntax and Labelled Transition System of Polyadic Calculus and Monadic Calculus

The formal definition of processes of polyadic first order ambient calculus with capability-passing is given as follows:

$$P ::= 0 \mid X \mid (x_1, \ldots, x_k).P \mid \langle c_1, \ldots, c_k \rangle.P \mid x.P \mid \text{in}(n).P \mid \text{out}(n).P \mid \text{open}(n).P \mid \text{in}(n).P \mid \text{out}(n).P \mid \text{open}(n).P \mid \text{rec}X.P \mid \nu p|\text{in}(p)|\text{out}(p)|\text{open}(p)\langle a\rangle|\text{rec}X.P, n \in \mathbb{N}$$

where $x_i$ is a variable, $c_i$ is a capability.

The operational semantics of processes is similar as Table 1 except that $COM$ is replaced by the following rule:

$COM : (x_1, \ldots, x_k).P|\langle c_1, \ldots, c_k \rangle.Q \rightarrow P|\langle c_1, \ldots, c_k/x_k \rangle|Q$

#### 4.2 Encoding Polyadic Ambient Calculus in Monadic Ambient Calculus

To prove that polyadic first order ambient calculus can be encoded by polyadic higher order ambient calculus, we approach this aim by two steps: firstly, we show that polyadic first order ambient calculus can be encoded by monadic first order ambient calculus; secondly, we show that monadic first order ambient calculus can be encoded by monadic higher order ambient calculus.

Now we first prove that polyadic first order ambient calculus can be simulated by monadic first order ambient calculus.

**Definition 4.** We give an encoding of the polyadic ambient calculus in the monadic ambient calculus. The mapping is defined inductively on the structure of $P$, where $a$ is a fresh name.

1. $Tr_{PFC}\{\emptyset\}^a = 0$;
2. $Tr_{PFC}\{X\}^a = X$;
3. $Tr_{PFC}\{(x_1, \ldots, x_k).P\}^a = (x).a[x,\overline{\text{open}}(a)](x_1)\ldots(x_k).Tr_{PFC}\{P\}^a$;
4. $Tr_{PFC}\{\nu p|\text{in}(p)|\text{open}(p)\langle a\rangle|\text{rec}X.P\}^a = \nu p|\text{in}(p)|\text{open}(p)\langle a\rangle|\text{rec}X.P|\text{open}(p).0$;
(5) \( T_{PFC}\{\alpha.P\}^a = \alpha.T_{PFC}\{P\}^a \), where \( \alpha \) is not in the form of \((x_1, \ldots, x_k)\) and \((c_1, \ldots, c_k)\);
(6) \( T_{PFC}\{P_1|P_2\}^a = T_{PFC}\{P_1\}^aT_{PFC}\{P_2\}^a \);
(7) \( T_{PFC}\{(vn).P\}^a = (vn)T_{PFC}\{P\}^a \);
(8) \( T_{PFC}\{n[P]\}^a = n[T_{PFC}\{P\}^a] \);
(9) \( T_{PFC}\{\text{rec}X.P\}^a = \text{rec}X.T_{PFC}\{P\}^a \).

In the following, we give the full abstraction property of \( T_{PFC}\{\cdot\}^a \).

**Lemma 4.** For any polyadic ambient \( P, T_{PFC}\{P\}^a \)
\((c_1/x_1, \ldots, c_k/x_k) \equiv_{Ba} T_{PFC}\{P[c_1/x_1, \ldots, c_k/x_k]\}^a\), where \( a \not\in fn(P, c_1, 0, \ldots, c_k, 0) \).

**Lemma 5.** For any polyadic ambients \( P \) and \( Q \), \( P \Rightarrow Q \Leftrightarrow (\nu a)T_{PFC}\{P\}^a \Rightarrow \equiv_{Ba} (\nu a)T_{PFC}\{Q\}^a \), where \( a \not\in fn(P) \cup fn(Q) \).

**Lemma 6.** For any polyadic ambient \( P, P \Downarrow n \Rightarrow (\nu a)T_{PFC}\{P\}^a \Downarrow n \), where \( a \not\in fn(P) \).

The definition of (weak) reduction barb congruence \( \equiv_{Ba} \) for polyadic/monadic ambients with capability-passing is the same as Definition 2.

**Proposition 2.** For any polyadic ambients \( P \) and \( Q \), \( P \equiv_{Ba} Q \Leftrightarrow (\nu a)T_{PFC}\{P\}^a \equiv_{Ba} (\nu a)T_{PFC}\{Q\}^a \), where \( a \not\in fn(P) \cup fn(Q) \).

**Proof:** By Lemmas 4, 5 and 6.

### 4.3 Encoding Monadic Ambient Calculus in MHSA

Now we show that every monadic ambient can be encoded in a monadic higher order safe ambient.

**Definition 5.** We give a mapping \( T_{PFC}\{\cdot\}^a \) with respect to name \( a \) which transforms every monadic ambient \( P \) into monadic higher order safe ambient \( T_{PFC}\{P\}^a \). The mapping is defined inductively on the structure of \( P \):

1. \( T_{PFC}\{\cdot\}^a(0) = 0 \);
2. \( T_{PFC}\{\cdot\}^a(X) = X \);
3. \( T_{PFC}\{\cdot\}^a((x).P)^a = (x).T_{PFC}\{P\}^a \);
4. \( T_{PFC}\{\cdot\}^a((in(n)).P)^a = (in(n), open(a), 0).T_{PFC}\{P\}^a \);
5. \( T_{PFC}\{\cdot\}^a((out(n)).P)^a = (out(n), open(a), 0).T_{PFC}\{P\}^a \);
6. \( T_{PFC}\{\cdot\}^a((\text{open}(n)).P)^a = (\text{open}(n), open(a), 0).T_{PFC}\{P\}^a \);
7. \( T_{PFC}\{\cdot\}^a((\text{in}(n)).P)^a = (\text{in}(n), open(a), 0).T_{PFC}\{P\}^a \);
8. \( T_{PFC}\{\cdot\}^a((\text{out}(n)).P)^a = (out(n), open(a), 0).T_{PFC}\{P\}^a \);
9. \( T_{PFC}\{\cdot\}^a((\text{open}(n)).P)^a = (\text{open}(n), open(a), 0).T_{PFC}\{P\}^a \);
10. \( T_{PFC}\{\cdot\}^a((\text{in} \cdot n).P)^a = \text{in}(n).T_{PFC}\{P\}^a \);
11. \( T_{PFC}\{\cdot\}^a((\text{out} \cdot n).P)^a = \text{out}(n).T_{PFC}\{P\}^a \);
12. \( T_{PFC}\{\cdot\}^a((\text{open} \cdot n).P)^a = \text{open}(n).T_{PFC}\{P\}^a \);
13. \( T_{PFC}\{\cdot\}^a((\text{in} \cdot n).P)^a = \text{in}(n).T_{PFC}\{P\}^a \);
14. \( T_{PFC}\{\cdot\}^a((\text{out} \cdot n).P)^a = \text{out}(n).T_{PFC}\{P\}^a \);
15. \( T_{PFC}\{\cdot\}^a((\text{open} \cdot n).P)^a = \text{open}(n).T_{PFC}\{P\}^a \);
16. \( T_{PFC}\{\cdot\}^a((\text{in} \cdot n).P)^a = \text{in}(n).T_{PFC}\{P\}^a \);

(17) \( T_{PFC}\{P_1|P_2\}^a = T_{PFC}\{P_1\}^aT_{PFC}\{P_2\}^a \);
(18) \( T_{PFC}\{(vn).P\}^a = (vn)T_{PFC}\{P\}^a \);
(19) \( T_{PFC}\{n[P]\}^a = n[T_{PFC}\{P\}^a] \);
(20) \( T_{PFC}\{\text{rec}X.P\}^a = \text{rec}X.T_{PFC}\{P\}^a \).

The following lemmas and propositions state the full abstraction property of \( T_{PFC}\{\cdot\}^a \).

**Lemma 7.** For any monadic ambient \( P, T_{PFC}\{P\}^a(c, \text{open}(a), 0/X) \equiv_{Ba} T_{PFC}\{P[c/x]\}^a \), where \( a \not\in fn(P, c, 0) \).

**Lemma 8.** For any monadic ambients \( P \) and \( Q \), \( P \Rightarrow Q \Leftrightarrow (\nu a)T_{PFC}\{P\}^a \Rightarrow \equiv_{Ba} (\nu a)T_{PFC}\{Q\}^a \), where \( a \not\in fn(P) \cup fn(Q) \).

**Lemma 9.** For any monadic ambient \( P, \ P \Downarrow n \Rightarrow (\nu a)T_{PFC}\{P\}^a \Downarrow n \), where \( a \not\in fn(P) \).

**Proposition 3.** For any monadic ambients with capability-passing \( P \) and \( Q \), \( P \equiv_{Ba} Q \Leftrightarrow (\nu a)T_{PFC}\{P\}^a \equiv_{Ba} (\nu a)T_{PFC}\{Q\}^a \), where \( a \not\in fn(P) \cup fn(Q) \).

**Proof:** By Lemmas 7, 8 and 9.

We can indirectly encode polyadic first order ambient calculus with capability-passing in monadic higher order ambient calculus since polyadic first order ambient calculus with capability-passing can be encoded in monadic first order ambient calculus with capability-passing and monadic first order ambient calculus with capability-passing can be encoded in monadic higher order safe ambient calculus.

### 5. Encoding of First Order Ambient Calculus with Name-Passing

In this section, we will give an encoding of ambient calculus with name-passing, then we will prove the full abstraction of this encoding.

#### 5.1 Syntax and Labelled Transition System of Polyadic Calculus and Monadic Calculus

The formal definition of processes of polyadic first order ambient calculus with name-passing is given as follows:

\[
P ::= 0 \mid X \mid (x_1, \ldots, x_k).P \mid (n_1, \ldots, n_k).P \mid in(n).P \mid out(n).P \mid open(n).P \mid \overline{in}(n).P \mid \overline{out}(n).P \mid PC(n).P \mid P_1 | P_2 \mid (\nu n).P \mid n[P] \mid \text{rec}X.P \mid x[P] \mid in(x).P \mid out(x).P \mid open(x).P \mid \overline{in}(x).P \mid \overline{out}(x).P \mid PC(x).P, \text{ where } n \in \text{ set } N \text{ of names, } x_i \text{ is a variable.}
\]

The operational semantics of processes is similar as Table 1 except that \( COM \) is replaced by the following rule:

\[
COM : (x_1, \ldots, x_k).P|n_1, \ldots, n_k| R \rightarrow P[n_1/x_1, \ldots, n_k/x_k]|R
\]

Similarly, monadic calculus is a subcalculus of polyadic calculus where only one parameter can be exchanged in one communication. The syntax and labelled transition system of monadic calculus is similar to polyadic calculus except that the number of parameters in communications is one.
5.2 Encoding Polyadic Ambient Calculus with Name-Passing in Monadic Calculus

At first we show that the polyadic ambient calculus with name-passing can be simulated by the monadic ambient calculus with name-passing.

**Definition 6.** We give an encoding of the polyadic ambient calculus with name-passing in the monadic ambient calculus with name-passing. The mapping is defined inductively on the structure of $P$, where $\alpha$ is a fresh name.

1. $TR_{FN}\{0\}^\alpha = 0$
2. $TR_{FN}\{X\}^\alpha = X$
3. $TR_{FN}\{x_1, \ldots, x_k\}.P^\alpha = (x).a[\overline{m}(x).\overline{open}(a)](x_1).\ldots.(x_k).TR_{FN}\{P\}^\alpha$
4. $TR_{FN}\{(\alpha_1, \ldots, \alpha_n).P\}^\alpha = \nu p((p).p[\overline{m}(p).\overline{open}(a)])(\langle p \rangle_1).\ldots.(\langle p \rangle_n).TR_{FN}\{P\}^\alpha$.
5. $TR_{FN}\{\alpha.P\}^\alpha = a.TR_{FN}\{P\}^\alpha$, where $\alpha$ is not in the form of $(x_1, \ldots, x_k)$ and $\langle \alpha_1, \ldots, \alpha_n \rangle$
6. $TR_{FN}\{P_1; P_2\}^\alpha = TR_{FN}\{P_1\}^\alpha[TR_{FN}\{P_2\}^\alpha$.
7. $TR_{FN}\{\nu m.P\}^\alpha = \nu m.TR_{FN}\{P\}^\alpha$.
8. $TR_{FN}\{n[\overline{m}(P)]\}^\alpha = n.TR_{FN}\{P\}^\alpha$.
9. $TR_{FN}\{\overline{x}.P\}^\alpha = x.TR_{FN}\{P\}^\alpha$.
10. $TR_{FN}\{\overline{rec.X}.P\}^\alpha = \overline{rec.X}.TR_{FN}\{P\}^\alpha$.

The full abstraction of $TR_{FN}\{\}^\alpha$ is stated in the following lemmas and propositions.

**Lemma 10.** For any monadic ambient $P$, $TR_{FN}\{n[\overline{m}(n).\overline{open}(a)].0/Z_1, \overline{m}(n).0/Z_2, \overline{m}(n).\overline{open}(a).0/Z_3, out(\overline{open}(a).0)/Z_4, \overline{open}(a).0/Z_5, \overline{open}(a).0/Z_6, \overline{m}(n).\overline{open}(a).0/Z_7 \} \approx_{Ba} TR_{FN}\{P[\overline{n}(x)]\}^\alpha$, where $a \notin fn(P) \cup \{n\}$.

**Lemma 11.** For any monadic ambient $P$ and $Q$, $P \leftrightarrow Q \equiv (\overline{v}a)TR_{FN}\{P\}^\alpha \leftrightarrow_{Ba} (\overline{v}a)TR_{FN}\{Q\}^\alpha$, where $a \notin fn(P)$.

**Lemma 12.** For any monadic ambient $P$, $P \downarrow n \equiv (\overline{v}a)TR_{FN}\{P\}^\alpha \downarrow n$, where $a \notin fn(P)$.

The definition of (weak) reduction barbed congruence $\approx_{Ba}$ for monadic ambients with name-passing is the same as Definition 2.

**Proposition 4.** For any monadic ambients with name-passing $P$ and $Q$, $P \approx_{Ba} Q \iff (\overline{v}a)TR_{FN}\{P\}^\alpha \approx_{Ba} (\overline{v}a)TR_{FN}\{Q\}^\alpha$, where $a \notin fn(P) \cup fn(Q)$.

**Proof:** By Lemmas 10, 11 and 12.

We can indirectly encode polyadic first order ambient calculus with name-passing in monadic higher order ambient calculus since polyadic first order ambient calculus with name-passing can be encoded in monadic first order ambient calculus with name-passing, monadic first order ambient calculus with name-passing can be encoded in polyadic higher order safe ambient calculus, and polyadic higher order safe ambient calculus can be encoded in monadic higher order safe ambient calculus.

6. Encoding of Polyadic $\pi$-Calculus

In this section, we will show that polyadic $\pi$-calculus can be encoded in monadic higher order ambient calculus.

6.1 Syntax and Labelled Transition System of Polyadic $\pi$-Calculus

Now we briefly recall the syntax and labelled transition system of the polyadic $\pi$-calculus.

We use $a, b, c, ..., x, y, z, ...$ to range over the class of names. The class $Pr$ of the polyadic $\pi$-calculus processes is built up using the operators of prefixing, sum, parallel composition, restriction and replication in the grammar below: $P ::= 0 | x(y_1, ..., y_k).P | \overline{x}(y_1, ..., y_k).P | P_1 ; P_2 | (\nu x).P$.

In each process of the form $(\nu y).P$ or $x(y).P$ the occurrence of $y$ is bound within the scope of $P$. An occurrence of $y$ in a process is said to be free iff it does not lie within the scope of a bound occurrence of $y$. The set of names
occurring free in $P$ is denoted $fn(P)$. An occurrence of a name in a process is said to be bound if it is not free, we write the set of bound names as $bn(P)$. Process $P$ and $Q$ are $\alpha$-convertible, $P \equiv_{\alpha} Q$, if $Q$ can be obtained from $P$ by a finite number of changes of bound names. The set of all processes is denoted as $Pr_{e_{\pi}}$.

Structural congruence of polyadic $\pi$-calculus is a congruence relation including the following rules:

$$P \equiv Q \text{ if } P \equiv_{\alpha} Q; \quad P|Q \equiv P|P; \quad (P|Q)|R \equiv P|(Q|R); \quad P|0 \equiv P; \quad (vn|0) \equiv 0; \quad (vn)(vn)P \equiv (vn)(vn)P; \quad (vn)(P|Q) \equiv P|(vn)Q \text{ if } n \notin fn(P).$$

The operational semantics of processes is given in Table 2. We have omitted the symmetric of the parallelism and communication.

Table 2: Labelled transition system of polyadic $\pi$-calculus

| $COM : \pi(y_1, ..., y_k).P|x(z_1, ..., z_k).Q \rightarrow $ |
| $P|Q[y_1/z_1, ..., y_k/z_k] $ |
| $ALP : \overline{Q} \rightarrow Q \quad P \equiv Q, P' \equiv Q' $ |
| $PAR : \overline{P|Q} \rightarrow P'|Q $ |
| $RES : (\nu x)P \rightarrow (\nu x)P' $ |
| $REP : |P|P \rightarrow P'|P $ |

6.2 Encoding Polyadic $\pi$-Calculus in First Order Ambient Calculus with Name-Passing

Now we show that polyadic $\pi$-calculus [12] can be encoded in first order ambient calculus with name-passing.

**Definition 8.** We give a mapping $Tr_{\pi}\{\}$ which transforms every polyadic $\pi$-calculus process $P$ into the first order ambient calculus with name-passing process $Tr_{\pi}\{P\}$. The mapping is defined inductively on the structure of $P$.

1. $Tr_{\pi}\{0\} = 0$;
2. $Tr_{\pi}\{x|y_1, ..., y_k).P\} = x|\overline{in}(x).open(x).\pi(y_1, ..., y_k).
3. $Tr_{\pi}\{y_1, ..., y_k).P\} | open(x).0$.
4. $Tr_{\pi}\{P|x|y_1, ..., y_k).Tr_{\pi}\{P\}| open(x).0$.
5. $Tr_{\pi}\{(\nu x)P\} = (\nu x)Tr_{\pi}\{P\}$.
6. $Tr_{\pi}\{P\}^n = !Tr_{\pi}\{P\}^n$.

By the above Proposition 4, we can also get a full abstract encoding from polyadic $\pi$-calculus in the monadic higher order ambient calculus. Therefore, $\pi$-calculus can be expressed in the monadic higher order ambient calculus.

**Definition 9.** A symmetric relation $R \subseteq Pr_{e_{\pi}} \times Pr_{e_{\pi}}$ is a weakly reduction barbed congruence if $P \not\equiv Q$ implies:

1. $P|C \not\equiv Q|C$ for any $C$;
2. Whenever $P \not\rightarrow P'$ then $Q \not\rightarrow Q'$ and $P' \not\equiv Q'$;
3. For any name $n$, if $P \downarrow n$, then also $Q \downarrow n$. Here $P \downarrow n$ means $\exists P'$, $P \not\equiv P' \equiv (\nu k)(\alpha.P)|P_2$ where $n = x$ if $\alpha = x(y_1, ..., y_k), n = \pi$ if $\alpha = \pi(y_1, ..., y_k)$ and $\{n, \pi\} \cap \tilde{k} = \emptyset$.

We write $P \equiv_{Ba} Q$ if $P$ and $Q$ are weakly reduction barbed congruent.

**Lemma 13.** For any polyadic $\pi$-calculus process $P$, $Tr_{\pi}\{P\}|z_1/y_1, ..., z_k/y_k) \equiv_{Ba} Tr_{\pi}\{P(z_1)/y_1, ..., z_k/y_k)\}.

**Lemma 14.** For any polyadic $\pi$-calculus processes $P$ and $Q$, $P \not\rightarrow Q \not\rightarrow Tr_{\pi}\{P\} \not\rightarrow Tr_{\pi}\{Q\}$.

**Lemma 15.** For any polyadic $\pi$-calculus process $P$, $P \downarrow n \not\equiv Tr_{\pi}\{P\} \downarrow n$.

**Proposition 5.** For any polyadic $\pi$-calculus processes $P$ and $Q$, $P \equiv_{Ba} Q \not\rightarrow Tr_{\pi}\{P\} \equiv_{Ba} Tr_{\pi}\{Q\}$.

**Proof:** By Lemmas 13, 14 and 15.

7. Asynchronous vs. Synchronous Communication

In the above sections, we study the expressiveness of synchronous calculi. But many ambients calculi are asynchronous calculi [3], [9]. So in this section, we will exploit the expressiveness of asynchronous calculi.

For asynchronous calculi, message emission is non-blocking. Asynchronous communications are interesting from the point of view of concurrent and distributed programming languages, because they are closer to the communication primitives offered by available distributed systems. Asynchronous calculi are usually achieved, syntactically, by disallowing output prefix (that is, continuations underneath output messages) and choice. In [7], authors showed that higher order asynchronous calculations can be encoded in asynchronous higher order $\pi$-calculus.

In this section, we will show that the similar result also holds for ambient calculi: synchronous monadic higher order safe ambient calculi can be encoded in asynchronous monadic higher order safe ambient calculi, named asynchronous MHSA. This result implies that all process calculi in this paper can be encoded in asynchronous monadic higher order ambient calculus.

7.1 Syntax and Labelled Transition System of Asynchronous MHSA

The formal definition of processes of asynchronous monadic higher order safe ambients is given as follows:

$$P ::= 0 \mid X \mid \langle X \rangle.P \mid \langle P \rangle \mid in(n).P \mid out(n).P \mid \overline{in}(n).P \mid \overline{out}(n).P \mid \overline{\pi}(n).P \mid P_1|P_2 \mid (\nu n)P \mid n[P] \mid recX.P$$

where $n \in set Var$ of process variables.

The operational semantics of processes is similar as Table 1 except that $COM$ is replaced by the following rule:

$$COM : \langle X \rangle.P|(Q) \rightarrow P\langle Q/X \rangle$$
7.2 Encoding MHSA in Asynchronous MHSA

In this section we present an encoding from synchronous monadic higher order safe ambients to asynchronous monadic higher order safe ambients, then prove the full abstraction property of this encoding.

Definition 10. We give a mapping $Tr_S\{a; b; c\}$ with respect to names $a, b, c$ which transforms every synchronous monadic higher order safe ambient $P$ into the asynchronous monadic higher order safe ambient $Tr_S\{P\}$. The mapping is defined inductively on the structure of $P$.

1. $Tr_S\{0\} = 0$;
2. $Tr_S\{X\} = X$;
3. $Tr_S\{(X).P\} = a(in(a)\cdot open(b)\cdot (X)\cdot open(c)\cdot open(a)\cdot open(b)\cdot open(c))$;
4. $Tr_S\{(P_1)_{P_2}\} = b(in(a), open(a)\cdot open(b)\cdot open(c))$;
5. $Tr_S\{in(n)\} = in(n)\cdot Tr_S\{P\}$;
6. $Tr_S\{out(n)\} = out(n)\cdot Tr_S\{P\}$;
7. $Tr_S\{open(n)\} = open(n)\cdot Tr_S\{P\}$;
8. $Tr_S\{in(n)\cdot P\} = in(n)\cdot Tr_S\{P\}$;
9. $Tr_S\{out(n)\cdot P\} = out(n)\cdot Tr_S\{P\}$;
10. $Tr_S\{open(n)\cdot P\} = open(n)\cdot Tr_S\{P\}$;
11. $Tr_S\{P_1\cdot P_2\} = Tr_S\{P_1\}\cdot Tr_S\{P_2\}$;
12. $Tr_S\{rec\cdot X\cdot P\} = rec\cdot X\cdot Tr_S\{P\}$.

We give a mapping $Tr_S\{a; b; c\}$ for asynchronous monadic higher order ambient calculus. In [6], authors showed that a higher order $\pi$-calculus with $n$-adic communication can be encoded in a calculus with $n - 1$-adic communication. We have shown that polyadic $\pi$-calculus can be encoded in monadic higher order ambient calculus in this paper. Since it was proved that higher order $\pi$-calculus can be encoded in $\pi$-calculus in [11], we can conclude that the monadic higher order ambient calculus cannot be encoded in higher order $\pi$-calculus with $n$-adic communication for any $n$. Therefore, this result means that the expressive power of monadic higher order ambient calculus is strictly stronger than higher order $\pi$-calculus.

Acknowledgment

This work was supported by the Aviation Science Fund of China under Grant No. 20128052064 and the National Natural Science Foundation of China under Grant No. 60873025.

References