数理解析研究所講究録
992 巻 1997 年 126-133

モービルプロセス計算の型システムについて

静岡大学 情報学部 情報科学科　富樫 敦 (Atsushi Togashi)

1 Introduction

In the literature, there have been intensive studies on typing (sorting) systems for the polyadic π-calculus, originated by Milner's sorting discipline [7] based on name matching. The proposed systems, so far, are categorized into the two groups — systems by name matching and ones by structure matching (possibly with subtyping) — and obtain similar results. A natural question arises "Is there any relationship between the two paradigms ?". With this motivation, the present paper gives deeper investigations on typing systems between the two approaches. For this purpose, a sorting system by name matching, a quite similar to the system in [4], and a typing system by structure matching with subtyping, a slight extension of the system in [9], are presented, along with several basic properties. Then, correspondence between the sorting system and the typing system is investigated via transformations both form sortings to typings and from typings to sortings. It is shown that if a process is well-sorted *w.r.t.* a safe sorting in the sorting system, then it is well-typed *w.r.t.* the transformed typing in the typing system, but not vice versa. This result can be straightforwardly extended to Liu and Walker's consistent sortings. Under a certain condition, we can show the reverse implication. Furtheremore, on the other direction from typings to sortings, it is shown that the derived typing from the sorting which is the result of applying transformation to a typing coincides with the original typing. However, the derived sorting from the typing which is the result of applying transformation to a sorting is proved to be a proper specialization of the original sorting.

The outline of the paper is as follows: Section 2 presents the polyadic π-calculus to a certain extent needed for the study. Section 3 and 4 introduce a sorting system and a typing system, respectively. Section 5, the main part of this paper, relates the sorting system and the typing system via both-directional transformations. This paper is concluded in Section 6 with some concluding remarks.

2 The Polyadic π-Calculus

Let $\mathcal{N}$ be a possibly infinite set of *names.* The basic syntax of *processes* we consider in this paper is defined by the following grammar:

$$P ::= \mathbf{0} \mid a(x_1,\ldots,x_n).P \mid \bar{a}\langle b_1,\ldots,b_n\rangle.P \mid P \mid Q \mid (\nu x)P \mid !P$$

where $\mathbf{0}$ is the *nil* process; $a(x_1,\ldots,x_n).P$ and $\bar{a}\langle b_1,\ldots,b_n\rangle.P$ are *input-prefixes* and *output-prefixes*, respectively; $P \mid Q$ are *parallel compositions*; $(\nu x)P$ are *restrictions*; $!P$ are *replications.* A *structural congruence relation* $\equiv$ is defined to be the smallest congruence relation over processes which satisfies the axiom schemes.

1. If $P \equiv_\alpha Q$ then $P \equiv Q$: Processes are identified if they differ only by a change of bound names.
2. $P \mid (Q \mid R) \equiv (P \mid Q) \mid R$; $P \mid Q \equiv Q \mid P$; $P \mid \mathbf{0} \equiv P$.
3. $!P \equiv !P \mid P$.
4. $(\nu x)P \equiv P$ if $x \notin fn(P)$*; $(\nu x)(\nu y)P \equiv (\nu y)(\nu x)P$;
   $(\nu x)P \mid Q \equiv (\nu x)(P \mid Q)$ if $x \notin fn(Q)$†.

*This induces the usual axiom schemes: $(\ x)\mathbf{0} \equiv \mathbf{0}$; $(\ x)(\ x)P \equiv (\ x)P$.
†Note that the side condition can be viewed as a consequence of our convention of regarding bound names.

Now, we define a *reduction relation* $\rightarrow$ over processes to be the smallest relation satisfying the following rules:

$$\textsc{Comm}\ \frac{}{a(\tilde{x}).P \mid \bar{a}\langle\tilde{b}\rangle.Q \rightarrow P\{\tilde{b}/\tilde{x}\} \mid Q}\ |\tilde{x}| = |\tilde{b}| \qquad \textsc{Par}\ \frac{P \rightarrow P'}{P \mid Q \rightarrow P' \mid Q}$$

$$\textsc{Rest}\ \frac{P \rightarrow P'}{(\ x)P \rightarrow (\ x)P'} \qquad \textsc{Struct}\ \frac{Q \equiv P \quad P \rightarrow P' \quad P' \equiv Q'}{Q \rightarrow Q'}$$

3 A Sorting System by Name Matching

Let Σ be a finite set of (*subject*) *sorts*. Σ^* denotes the set of all finite sequences of elements in Σ. An element in Σ^* is called an *object sort*, denoted by $(s_1, \ldots, s_n)$, or simplely $(\tilde{s})$ if the number of the sequence is not important. We use u and v to range over Σ^*. A *subject sorting* Γ on Σ is a finite set of *subject sort assignments* $a : s$, where $a \in \mathcal{N}$ and $s \in \Sigma$, such that $a : s$, $a : t \in \Gamma$ implies $s = t$. An *object sorting* Ω on Σ is a finite set of *object sort assignments* either of the form $s^+ : u$ or $s^- : u$, where $s \in \Sigma$ and $u \in \Sigma^*$, such that $s^\star : u$, $s^\star : v \in \Omega$ implies $u = v$, for $\star \in \{+, -\}$. A *sorting* on Σ is a pair $\Gamma; \Omega$ of a subject sorting Γ and an object sorting Ω. Ω is *safe* in s if $s^+ : u$, $s^- : v \in \Omega$ implies $u = v$. If Ω is safe in all s in Σ then Ω is called *safe*. A sorting $\Gamma; \Omega$ is *safe* if its object sorting Ω is safe. A *sorting judgment* (*by name matching*) on Σ is an expression of the form: $\Gamma; \Omega \vdash P : ()$, where $\Gamma; \Omega$ is a sorting, P is a process, and $()$ is the special symbol standing for well-behavedness of a process.

Definition 3.1 is a *sorting system* (*by name matching*) consisting of the following inference rules:

$$\textsc{S-Nil}\ \frac{}{\Gamma; \Omega \vdash \mathbf{0} : ()} \qquad \textsc{S-Comp}\ \frac{\Gamma; \Omega \vdash P : () \quad \Gamma; \Omega \vdash Q : ()}{\Gamma; \Omega \vdash P \mid Q : ()}\ .$$

$$\textsc{S-In}\ \frac{\Gamma, a : s, \tilde{x} : \tilde{t};\ \Omega, s^+ : (\tilde{t}) \vdash P : ()}{\Gamma, a : s;\ \Omega, s^+ : (\tilde{t}) \vdash a(\tilde{x}).P : ()} \qquad \textsc{S-Out}\ \frac{\Gamma, a : s, \tilde{b} : \tilde{t};\ \Omega, s^- : (\tilde{t}) \vdash P : ()}{\Gamma, a : s, \tilde{b} : \tilde{t};\ \Omega, s^- : (\tilde{t}) \vdash \bar{a}\langle\tilde{b}\rangle.P : ()}$$

$$\textsc{S-Repl}\ \frac{\Gamma; \Omega \vdash P : ()}{\Gamma; \Omega \vdash !P : ()} \qquad \textsc{S-Rest}\ \frac{\Gamma, x : s;\ \Omega \vdash P : ()}{\Gamma; \Omega \vdash (\ x)P : ()}\ .$$

□

The interesting cases are the rules for input and output. In order to be sure that the input prefix $a(\tilde{x}).P$ is well-behaved in a given sorting, we must check that first the object sort for the subject sort of a with the positive polarity matches the sort of the sequence of names read from a; secondly the continuation P is well-behaved in the augmented sorting by the sort assignments $\tilde{x} : \tilde{t}$. The case for the output prefix is analogous. The notation $\Gamma; \Omega \vdash\ \ P : ()$ indicates that a sorting judgment $\Gamma; \Omega \vdash P : ()$ is provable in the system .

Proposition 3.1 *If* $\Gamma; \Omega \vdash\ \ P : ()$ *and* $P \equiv Q$ *then* $\Gamma; \Omega \vdash\ \ Q : ()$. □

Proposition 3.2 *If* $\Gamma; \Omega \vdash\ \ P : ()$ *for a safe sorting* $\Gamma; \Omega$ *and* $P \rightarrow Q$ *then* $\Gamma; \Omega \vdash\ \ Q : ()$. □

In the inference rule Comm of the reduction relation, it is required that the arities of the input-prefix and the output prefix must be equal. If a process P contains *unguarded* prefixes $a(\tilde{x}).Q$ and $\bar{a}\langle\tilde{b}\rangle.R$ with $|\tilde{x}| \neq |\tilde{b}|$, then P is said to *contain a communication mismatch* [4] or a *run-time type error* [9,13]. P is *free from communication mismatch* if whenever $P \overset{*}{\rightarrow} P'$ then P' does not contain a communication mismatch.

4 Typing Systems by Subtyping

Let $I \leq J$ be the least preorder on the *tags* $\{\mathsf{r}, \mathsf{w}, \mathsf{b}\}$ containing $\mathsf{b} \leq \mathsf{r}$ and $\mathsf{b} \leq \mathsf{w}$. A *type*, ranged over by T or S, is defined by the grammar:

$$\begin{array}{rcl} T & ::= & \alpha \mid \top \mid \bot \mid (T_1, \ldots, T_n)^I \mid \mu\alpha.S \\ I & ::= & \mathsf{r} \mid \mathsf{w} \mid \mathsf{b}, \end{array}$$

where α is a *type-variables*; $\top$ and $\bot$ are *constant types top* and *bottom*, respectively; $(T_1, \ldots, T_n)^I$ is a *tagged tuple*; $\mu\alpha.S$ is a *recursive type*. Let $\boldsymbol{T}$ and $\boldsymbol{T}^c$ denote the set of all (open) types and the set of all closed types, respectively, where α-convergent types are identified. Let Λ be a sequence of pairs of types $S \leq T$. A *subtyping judgment* is an expression of the form $\Lambda \vdash S \leq T$, pronounced as S is a *subtype* of T under the assumption Λ.

Definition 4.1 is a *subtyping system* consisting of the following rules[‡]:

$$\text{ASMP}\ \frac{}{\Lambda, S \leq T \vdash S \leq T} \qquad \text{REF}\ \frac{}{\Lambda \vdash T \leq T}$$

$$\text{TOP}\ \frac{}{\Lambda \vdash S \leq \top} \qquad \text{BTM}\ \frac{}{\Lambda \vdash \bot \leq T}$$

$$\text{BB}\ \frac{\text{for each } i,\ \ \Lambda \vdash S_i \leq T_i \quad \Lambda \vdash T_i \leq S_i}{\Lambda \vdash (S_1, \ldots, S_n)^{\mathsf{b}} \leq (T_1, \ldots, T_n)^{\mathsf{b}}}$$

$$\text{RB-R}\ \frac{I \leq \mathsf{r} \quad \text{for each } i,\ \ \Lambda \vdash S_i \leq T_i}{\Lambda \vdash (S_1, \ldots, S_n)^I \leq (T_1, \ldots, T_n)^{\mathsf{r}}} \qquad \text{WB-W}\ \frac{I \leq \mathsf{w} \quad \text{for each } i,\ \ \Lambda \vdash T_i \leq S_i}{\Lambda \vdash (S_1, \ldots, S_n)^I \leq (T_1, \ldots, T_n)^{\mathsf{w}}}$$

$$\text{REC-L}\ \frac{\Lambda, \mu\alpha.S \leq T \vdash S\{\mu\alpha.S/\alpha\} \leq T}{\Lambda \vdash \mu\alpha.S \leq T} \qquad \text{REC-R}\ \frac{\Lambda, S \leq \mu\alpha.T \vdash S \leq T\{\mu\alpha.T/\alpha\}}{\Lambda \vdash S \leq \mu\alpha.T}$$

□

A *typing judgment* (*by subsorting*) is an expression of the form $\Delta \vdash P : \circ$, where Δ is a set of *type assignments* $a : T$, P is a process, and $\circ$ is the special symbol standing for well-behavedness of the process.

Definition 4.2 is a *typing system* (*by subtyping*) consisting of the following rules:

$$\text{T-NIL}\ \frac{}{\Delta \vdash \mathbf{0} : \circ} \qquad \text{T-COMP}\ \frac{\Delta \vdash P : \circ \quad \Delta \vdash Q : \circ}{\Delta \vdash P \mid Q : \circ}$$

$$\text{T-IN}\ \frac{\vdash \Delta(a) \leq (\tilde{T})^{\mathsf{r}} \quad \Delta, \tilde{x} : \tilde{T} \vdash P : \circ}{\Delta \vdash a(\tilde{x}).P : \circ} \qquad \text{T-OUT}\ \frac{\vdash \Delta(a) \leq (\Delta(\tilde{b}))^{\mathsf{w}} \quad \Delta \vdash P : \circ}{\Delta \vdash \bar{a}\langle\tilde{b}\rangle.P : \circ}$$

$$\text{T-REPL}\ \frac{\Delta \vdash P : \circ}{\Delta \vdash !P : \circ} \qquad \text{T-REST}\ \frac{\Delta, x : T \vdash P : \circ}{\Delta \vdash (\ x)P : \circ}\ .$$

□

[‡]In the subtyping system regarding closed types only as in [9], the rule REF is derivable by well-founded induction on subtyping judgments. See [11]. However, it is no more derivable when open types are concerned.

Pierce and Sangiorgi [9] have formulated their typing system in the Church style (á la Church), where typing information for the input parameters and restricted names are given explicitly. As there is often a simple relationship between the two styles in the typed λ-calculi there is a simple relationship between the type system in this paper and the one by Pierce and Sangiorgi [9]. This will be explained below: Let $|\ |$ be the function mapping process terms with type ornamentations into the ordinary processes in this paper by erasing the all type information.

Proposition 4.1

1. *Let Q be a process with type annotations. If $\Delta \vdash Q : \circ$ is provable in the Pierce and Sangiorgi's Church style typing system* [9], *then $\Delta \vdash\ |Q| : \circ$.*
2. *Let P be a process. If $\Delta \vdash\ P : \circ$ then there is a process Q with type annotations such that $\Delta \vdash Q : \circ$ is provable in the Pierce and Sangiorgi's system and $|Q| = P$.* □

5 Relating Sortings and Typings

With each sorting judgment $\Gamma; \Omega \vdash P : ()$ we will associate a typing judgment $[\![\Gamma]\!]_\Omega \vdash P : \circ$ such that hopefully we expect $\Gamma; \Omega \vdash\ P : ()$ iff $[\![\Gamma]\!]_\Omega \vdash\ P : \circ$. For this purpose, given an object sorting Ω and an environment $\rho : \Sigma \to \boldsymbol{T}^c$, mapping (free) sorts to closed types, for each sort s in Σ the corresponding type $[\![s]\!]^\rho_\Omega$ of s with respect to Ω and ρ is defined as follows:

$$\begin{array}{rcl}
[\![s]\!]^\rho_\Omega & \triangleq & {}^{\rho}_{\Omega}(s, \emptyset); \\
{}^{\rho}_{\Omega}(s, X) & \triangleq & \begin{cases} s & \text{if } s \in X \\ \mu s.({}^{\rho}_{\Omega}(s^+, X \cup \{s\}) \wedge {}^{\rho}_{\Omega}(s^-, X \cup \{s\})) & \text{otherwise;} \end{cases} \\
{}^{\rho}_{\Omega}(s^\star, X) & \triangleq & \begin{cases} ({}^{\rho}_{\Omega}(\tilde{t}, X))^{\mathsf{r}} & \text{if } \star = + \text{ and } s^+ : (\tilde{t}) \in \Omega \\ ({}^{\rho}_{\Omega}(\tilde{t}, X))^{\mathsf{w}} & \text{if } \star = - \text{ and } s^- : (\tilde{t}) \in \Omega \\ \rho(s) & \text{otherwise.} \end{cases}
\end{array}$$

In the definition we use the notational convention ${}^{\rho}_{\Omega}(\tilde{t}, X)$ to denote the sequence ${}^{\rho}_{\Omega}(t_1, X), \ldots, {}^{\rho}_{\Omega}(t_n, X)$, for $\tilde{t} = t_1, \ldots, t_n$.

Let $\Gamma; \Omega$ be a sorting then the corresponding typing $[\![\Gamma]\!]^\rho_\Omega$ is defined by

$$[\![\Gamma]\!]^\rho_\Omega \triangleq \{a : [\![s]\!]^\rho_\Omega \mid a : s \in \Gamma\}.$$

Usually, the environment $\rho_\top$, $\rho_\top(s) \triangleq \top$ for each $s \in \Sigma$, is used to assign types to sorts. However, almost results stated in this section hold for any environment ρ. So that $[\![s]\!]_\Omega$ and $[\![\Gamma]\!]_\Omega$ are the abbreviations of $[\![s]\!]^\rho_\Omega$ and $[\![\Gamma]\!]^\rho_\Omega$ for any environment ρ, respectively, when ρ is not very important. Note that $[\![s]\!]_\Omega =_t \bot$ if Ω possesses object assignments to s having mismatch in number with the I/O parameters.

Lemma 5.1 *Let Ω be a safe object sorting and ρ be an environment. If $s^+ : (\tilde{t}) \in \Omega$ ($s^- : (\tilde{t}) \in \Omega$) then*

$$\vdash [\![s]\!]^\rho_\Omega =_{sub} ([\![\tilde{t}]\!]^\rho_\Omega)^I,$$

for some I such that $I \leq \mathsf{r}$ ($I \leq \mathsf{w}$). Thus, $[\![s]\!]^\rho_\Omega =_t ([\![\tilde{t}]\!]^\rho_\Omega)^I$.

Proof: Suppose $s^+ : (\tilde{t}) \in \Omega$ ($s^- : (\tilde{t}) \in \Omega$). By the definition of ${}^{\rho}_{\Omega}(s, X)$, the safety property of Ω implies that $[\![s]\!]^\rho_\Omega$ can be expressed as $[\![s]\!]^\rho_\Omega = \mu s.({}^{\rho}_{\Omega}(\tilde{t}, \{s\}))^I$, for some I, where $I \leq \mathsf{r}$ ($I \leq \mathsf{w}$). Since unfolding of the recursive definition preserves the identity, see Corollary 2.4.6 in [9],

$$\vdash [\![s]\!]^\rho_\Omega =_{sub} ({}^{\rho}_{\Omega}(\tilde{t}, \{s\})\{[\![s]\!]^\rho_\Omega / s\})^I.$$

Let t_i be the i-th element in the sequence $\tilde{t}$. If $t_i = s$ then ${}^{\rho}_{\Omega}(t_i, \{s\})\{[\![s]\!]^{\rho}_{\Omega}/s\} = [\![s]\!]^{\rho}_{\Omega} = [\![t_i]\!]^{\rho}_{\Omega}$. If $t_i \neq s$ then ${}^{\rho}_{\Omega}(t_i, \{s\})\{[\![s]\!]^{\rho}_{\Omega}/s\} = [\![t_i]\!]^{\rho}_{\Omega}$. □

Theorem 5.1 *If $\Gamma; \Omega \vdash P : ()$ for a safe sorting $\Gamma; \Omega$ then $[\![\Gamma]\!]_{\Omega} \vdash P : \circ$.*

Proof: By induction on the proof of $\Gamma; \Omega \vdash P : ()$ in . Interesting case is the one when the last inference is by S-IN or S-OUT.

Case S-IN: Suppose

$$\frac{\Gamma, a : s, \tilde{x} : \tilde{t}; \Omega, s^{+} : (\tilde{t}) \vdash P : ()}{\Gamma, a : s; \Omega, s^{+} : (\tilde{t}) \vdash a(\tilde{x}).P : ()}$$

is the last inference by applying S-IN. Let $\Omega' = \Omega \cup \{s^{+} : (\tilde{t})\}$. By the induction hypothesis, $[\![\Gamma]\!]_{\Omega'}, a : [\![s]\!]_{\Omega'}, \tilde{x} : [\![\tilde{t}]\!]_{\Omega'} \vdash P : \circ$. It remains to show that $\vdash [\![s]\!]_{\Omega'} \leq ([\![\tilde{t}]\!]_{\Omega'})^{\mathsf{r}}$ to deduce the typing judgment $[\![\Gamma']\!]_{\Omega'}, a : [\![s]\!]_{\Omega'} \vdash a(\tilde{x}).P : \circ$. This can be obtained by Lemma 5.1. The case by S-OUT is similar. □

The theorem insists that if a process is well-sorted with respect to a safe sorting in the system employing name matching, then it is well-typed as well in the system employing structure matching with subtyping. Theorem 5.1 can be extended to a consistent sorting in a straightforward way.

Corollary 5.1 *Let $\Gamma; \Omega$ be a consistent sorting. Let $\Gamma_0; \Omega_0$ be the unique most general safe sorting, its existence is guaranteed by the discussions in section 2. Then for a process P, $\Gamma; \Omega \vdash P : ()$ implies $[\![\Gamma_0]\!]_{\Omega_0} \vdash P : \circ$.* □

Example 5.1 The converse of Theorem 5.1 is not true in general. The following simple counter example illustrates the fact: Let us consider the process $P_1 = \bar{a}\langle b \rangle.\mathbf{0}$ under the safe sorting $\Gamma_1 = \{a : s, b : r\}$; $\Omega_1 = \{s^{-} : (t), t^{+} : (), r^{+} : (), r^{-} : ()\}$ on $\Sigma_1 = \{s, t, r\}$. By the transformation, $[\![s]\!]_{\Omega_1} = (\mathsf{r})^{\mathsf{w}}$, $[\![t]\!]_{\Omega_1} = \mathsf{r}$, $[\![r]\!]_{\Omega_1} = \mathsf{b}$, and $[\![\Gamma_1]\!]_{\Omega_1} = \{a : (\mathsf{r})^{\mathsf{w}}, b : \mathsf{b}\}$. Then, trivially we have $[\![\Gamma_1]\!]_{\Omega_1} \vdash P_1 : \circ$. But, $\Gamma_1; \Omega_1 \not\vdash P_1$ because $t \neq r$. □

If the transformation defined by a safe object sorting Ω from sorts into types satisfies a certain condition, then the converse of Theorem 5.1 holds.

Theorem 5.2 *Let $\Gamma; \Omega$ be a safe sorting on Σ such that $[\![s]\!]^{\rho\top}_{\Omega} \leq_{sub} [\![t]\!]^{\rho\top}_{\Omega}$ implies $s = t$, for any sorts $s, t \in \Sigma$*[§]*. Then, $[\![\Gamma]\!]^{\rho\top}_{\Omega} \vdash P : \circ$ implies $\Gamma; \Omega \vdash P : ()$, for any process P.*

Proof: By induction on the proof $[\![\Gamma]\!]^{\rho\top}_{\Omega} \vdash P : \circ$ and by case analysis of the applied rules. For detailed proof, refer to [11]. □

We will define a sorting $\Delta^{@}; \Delta^{\#}$ in terms of a typing Δ. To this end, we need some preliminaries. Given an open type T, let $Sub(T)$ be the set obtained from the set of all the subterms of T by replacing each bound type variable appearing in a subterm by its definition, formally $Sub(T)$ is defined inductively as follows:

$$\begin{aligned}
Sub(\alpha) &\triangleq \{\alpha\}; \\
Sub(\top) &\triangleq \{\top\}; \\
Sub(\bot) &\triangleq \{\bot\}; \\
Sub((T_1, \ldots, T_n)^{I}) &\triangleq \{(T_1, \ldots, T_n)^{I}\} \cup Sub(T_1) \cup \cdots \cup Sub(T_n); \\
Sub(\mu\alpha.T) &\triangleq \{\mu\alpha.T\} \cup \{S\{\mu\alpha.T/\alpha\} \mid S \in Sub(T)\}.
\end{aligned}$$

[§]This condition means that Ω represents the unique object sorting up to renaming of sorts such that no distinct sorts represent the same type where the type equality by forgetting the tags is used as the identity of types. Under this condition, $[\![\Gamma]\!]^{\rho\top}_{\Omega} \vdash P : \circ$ means P is well-typed with respect to $[\![\Gamma]\!]^{\rho\top}_{\Omega}$, where ***only structure matching without subtyping*** is used.

From definition it is easy to see that $Sub(T)$ is finite for any type T. In fact, $Sub(T)$ can have no more elements than the number of distinct subterms of T.

With an open type in canonical form $T \in \boldsymbol{T}$ we associate a tuple $\langle \Sigma(T), T^{\#} \rangle$ consisting of the set $\Sigma(T)$ of sorts and the object sorting $T^{\#}$. The sorts are defined by

$$\Sigma(T) \triangleq \{ [S] \mid S \in Sub(T) \}$$

for a type T, where $[T] \triangleq \{S \mid T =_t S\}$ is the congruence class of T with respect to the identity relation $=_t$ on $\boldsymbol{T}$. The object sorting $T^{\#}$ is defined by structural induction on T.

$$T^{\#} \triangleq \begin{cases} \emptyset & \text{if } T = \alpha, \text{ or } \top \\ \Omega_\bot & \text{if } T = \bot \\ \Omega_t \cup T_1^{\#} \cup \cdots \cup T_n^{\#} & \text{if } T = (T_1, \ldots, T_n)^I \\ \Omega_r \cup (T_1^{\#} \cup \cdots \cup T_n^{\#})\{[T]/[\alpha]\} & \text{if } T = \mu\alpha.(T_1, \ldots, T_n)^I, \end{cases}$$

where

$$\begin{array}{lcl} \Omega_\bot & \triangleq & \{[\bot]^+ : (), [\bot]^- : ([\bot])^{\mathsf{w}}\} \\ \Omega_t & \triangleq & \begin{cases} \{[T]^+ : ([T_1], \ldots, [T_n])\} & \text{if } I = \mathsf{r} \\ \{[T]^- : ([T_1], \ldots, [T_n])\} & \text{if } I = \mathsf{w} \\ \{[T]^+ : ([T_1], \ldots, [T_n]),\ [T]^- : ([T_1], \ldots, [T_n])\} & \text{if } I = \mathsf{b}. \end{cases} \\ \Omega_r & \triangleq & \begin{cases} \{[T]^+ : ([T_1\{T/\alpha\}], \ldots, [T_n\{T/\alpha\}])\} & \text{if } I = \mathsf{r} \\ \{[T]^- : ([T_1\{T/\alpha\}], \ldots, [T_n\{T/\alpha\}])\} & \text{if } I = \mathsf{w} \\ \{[T]^+ : ([T_1\{T/\alpha\}], \ldots, [T_n\{T/\alpha\}]), & \text{if } I = \mathsf{b}. \\ \quad [T]^- : ([T_1\{T/\alpha\}], \ldots, [T_n\{T/\alpha\}])\} & \end{cases} \end{array}$$

Let Δ be a typing. The corresponding set of subject sorts is defined by

$$\Sigma(\Delta) \triangleq \cup \{\Sigma(T) \mid x : T \in \Delta, \text{ for some } x\}.$$

The associated sorting $\Delta^{@}; \Delta^{\#}$ on $\Sigma(\Delta)$ with Δ is defined as follows:

$$\begin{array}{lcl} \Delta^{@} & \triangleq & \{x : [T] \mid x : T \in \Delta\}; \\ \Delta^{\#} & \triangleq & \cup \{T^{\#} \mid x : T \in \Delta, \text{ for some } x\}. \end{array}$$

We hope that for instance $\Delta \vdash P : \mathsf{o}$ implies $\Delta^{@}; \Delta^{\#} \vdash P : ()$. But, unfortunately there is a simple counter example. Let us consider the context $\Delta = \{a : (\top)^{\mathsf{w}}, b : (\top)^{\mathsf{r}}\}$ and the process $P = \bar{a}\langle b \rangle.\mathbf{0}$. P is well-typed under the context Δ.

$$\frac{(\top)^{\mathsf{w}} \leq ((\top)^{\mathsf{r}})^{\mathsf{w}} \quad a : (\top)^{\mathsf{w}}, b : (\top)^{\mathsf{r}} \vdash \mathbf{0} : \mathsf{o}}{\bar{a} : (\top)^{\mathsf{w}}, b : (\top)^{\mathsf{r}} \vdash a\langle b \rangle.\mathbf{0} : \mathsf{o}}$$

Thus, $\Delta \vdash P : \mathsf{o}$. By definition, $\Sigma(\Delta) = \{\mathsf{w}(\top), \mathsf{r}(\top), \top\}$; $\Delta^{@} = \{a : \mathsf{w}(\top), b : \mathsf{r}(\top)\}$; $\Delta^{\#} = \{\mathsf{w}(\top)^- : (\top), \mathsf{r}(\top)^+ : (\top)\}$. Because $\top \neq \mathsf{r}(\top)$, $\Delta^{@}; \Delta^{\#} \not\vdash P : ()$.

Proposition 5.1 *Let T be a type and ρ an environment, then $[\![[S]]\!]^{\rho}_{T^{\#}} =_t [\![[S]]\!]^{\rho}_{S^{\#}}$, for any $S \in sub(T)$.*
□

Lemma 5.2 *Let σ be any function mapping type variables α to closed types $\sigma(\alpha) \in \boldsymbol{T}^c$ and T be any type. Define the environment $\rho : \Sigma(T) \to \boldsymbol{T}^c$ by*

$$\rho(\,[S]\,) \triangleq \begin{cases} \sigma(\alpha) & \textit{if } [S] = [\alpha] \textit{ for some } \alpha \\ S & \textit{otherwise,} \end{cases}$$

for $[S] \in \Sigma(T)$. Then, we have $[\![\,[T]\,]\!]^{\rho}_{T\#} =_t \sigma(T)$. □

Theorem 5.3

1. *$\Gamma; \Omega \sqsubseteq [\![\Gamma]\!]^{@}_{\Omega}; [\![\Gamma]\!]^{\#}_{\Omega}$, for any safe sorting $\Gamma; \Omega$ on Σ.*
2. *$\Gamma; \Omega \neq [\![\Gamma]\!]^{@}_{\Omega}; [\![\Gamma]\!]^{\#}_{\Omega}$, for some safe sorting $\Gamma; \Omega$ on Σ.*
3. *$\Delta = [\![\Delta^{@}]\!]_{\Delta^{\#}}$, for any typing Δ.*

Proof: 1. Let $\theta : \Sigma \to \Sigma([\![\Gamma]\!]_{\Omega})$ be the function defined by $\theta(s) \triangleq [[\![s]\!]_{\Omega}]$, for $s \in \Sigma$. Suppose $x : s \in \Gamma$ then $x : [[\![s]\!]_{\Omega}] \in [\![\Gamma]\!]^{@}_{\Omega}$ by definition. It remains to show that θ is a homomorphism from Ω to $[\![\Gamma]\!]^{\#}_{\Omega}$. Suppose $s^{\star} : (t_1, \ldots, t_n) \in \Omega$, where $\star = +$ (or $\star = -$). By Lemma 5.1, $[\![s]\!]_{\Omega} =_t ([\![t_1]\!]_{\Omega}, \ldots, [\![t_n]\!]_{\Omega})^I$, where $I \leq \mathsf{r}$ (or $I \leq \mathsf{w}$). Thus by construction, we have $[[\![s]\!]_{\Omega}]^{\star} : (\,[[\![t_1]\!]_{\Omega}], \ldots, [[\![t_n]\!]_{\Omega}]\,) \in [\![\Gamma]\!]^{\#}_{\Omega}$, as required.

2. Consider the safe sorting $\Gamma_0 = \{a : t, b : s\}$; $\Omega_0 = \{t^+ : (),\ s^+ : ()\}$ on $\{s, t\}$. The inequality is obvious from the followings: $[\![\Gamma_0]\!]_{\Omega_0} = \{a : \mathsf{r}, b : \mathsf{r}\}$; $[\![\Gamma_0]\!]^{@}_{\Omega_0} = \{a : [\mathsf{r}], b : [\mathsf{r}]\}$; $[\![\Gamma_0]\!]^{\#}_{\Omega_0} = \{\,[\mathsf{r}]^+ : ()\}$.

3. The proof is by Lemma 5.2 since any type in Δ is closed. □

Corollary 5.2

1. *If $\Gamma; \Omega \vdash\ P : ()$ for a safe sorting $\Gamma; \Omega$ then $[\![\Gamma]\!]^{@}_{\Omega}; [\![\Gamma]\!]^{\#}_{\Omega} \vdash\ P : ()$.*
2. $[\![\Gamma]\!]_{\Omega} = [\![\,[\![\Gamma]\!]^{@}_{\Omega}]\!]_{[\![\Gamma]\!]^{\#}_{\Omega}}$. □

6 Concluding Remarks

In this paper, the sorting system by name matching and the typing system by structure matching with subtyping were related via the transformations. The introduced sorting (typing) system is quite closed to the typing system by Liu and Walker [4] (by Pierce and Sangiorgi [9]). So the results obtained in this paper are applicable to the investigation of the correspondence between them. If we forget the polarities (the tags and subtyping), then the resulting sorting (typing) system turns out to coincide with a variant of Milner's sorting system [7] (the typing system by Vasconcelos and Honda [13]). Thus, our results interpret the relationship between both the systems as well.

The correspondence between Milner's sorting and the typing system [13] is informally discussed with the illustrative example in [13] and more formally discussed in [12]. The idea is that a set of basic sorts and sorting defines a regular system of equations; such a system has a unique solution whose components are represented as regular trees; then derive a typing from the solution. Conversely, trees in a finite set of regular trees are components of the unique solution of a single system of equation; from such a system the set of sorts and sorting are obtained.

The transformation from sortings to typings has a similar flavor to the one from regular system equations in canonical form to recursive types discussed in [1, 11]. From typings without subtyping to

Milner's sortings, as stated in [13], well-typing induces well-sorting. But, as illustrated in Section 5, in general well-typing doesn't always implies well-sorting along the given translation. But, we convince that the following conjecture must hold.

Conjecture 6.1 *If* $\Delta \vdash P : \circ$ *then there exists a typing* Δ_0 *such that* $\Delta_0 \preceq \Delta$ — $dom(\Delta_0) \supset dom(\Delta)$ *and* $\Delta_0(x) \leq \Delta(x)$ *for all* $x \in dom(\Delta)$ — *and* $\Delta_0^{@}; \Delta_0^{\#} \vdash P : ()$. *Note that* $\Delta_0 \vdash P : \circ$. *See* [9,11]. $\square$

Finally, the relations between incremental systems and non-incremental systems are discussed in both sorting and typing in [11].

参考文献

[1] Amadio, R.M., Cardelli, L., Subtyping recursive types, *TOPLAS*, **15**, No. 4, pp.575–631, 1993.

[2] Hindley, R., The completeness theorem for typing λ-terms, *TCS*, 22, pp.1–7, pp.127–133, 1983.

[3] Gay, S.J., A sort inference algorithm for the polyadic π-calculus, in 20th *POPL*, 1993.

[4] Liu, X., Walker, D., A polymorphic type system for the polyadic π-calculus, *CONCUR'95*, *LNCS*, **962**, pp.103–116, 1995.

[5] Milner, R., *Communication and Concurrency*, Prentice Hall, 1989.

[6] Milner, R., Functions as processes, *th ICALP* **'90**, *LNCS*, **443**, 1990.

[7] Milner, R., The polyadic π-calculus: A tutorial, Technical Report ECS-LFCS-91-180, LFCS, Dept. of Comput. Sci., Edinburgh Univ., 1991.

[8] Milner, R., Parrow, J., Walker, D., A calculus of mobile processes, Part I and II, *Info. & Comp.*, **100**, No.1, pp.1–40, pp.41–77, 1992.

[9] B., Pierce, D. Sangiorgi, Typing and subtyping for mobile processes, Dep. of Computer Science, University of Edinburgh, 1994.

[10] Sangiorgi, D., Expressing mobility in process algebras: first-order and higher-order paradigms, Ph.D. Thesis, Edinburgh University, 1992.

[11] Togashi, A., On typing systems for the polyadic π-calculus, *Technical Report* 1/96, COGS, University of Sussex, 1996.

[12] Vasconcelos, V.T., Honda, K., Principal typing-schemes in a polyadic π-calculus, CS 92-4, Keio University, 1992.

[13] Vasconcelos, V.T., Honda, K., Principal typing schemes in a polyadic π-calculus, CONCUR'93, *LNCS*, **715**, pp.524–538, 1993.