Notes on Linear Typing for Free Output

Kohei Honda*

Abstract. This note presents an alternative linear typing system (in the sense of [3]). Unlike the system in [3], this system can type processes with free name outputs. Even though the set of typable terms in the new typing system is strictly greater than those in the original system, all these additional terms are semantically equivalent to their image in the original system via a compositional mapping which is essentially identity on the original set of terms.

1 Introduction

In [1, 3], we presented two typing systems for the π-calculus, called *affine* and *linear*, in which functional calculi are fully abstractly embeddable. In the systems in [1, 3], the outputs are restricted to those which pass only bound names, i.e. those of form $(\boldsymbol{\nu}\,\vec{y})(\overline{x}\langle\vec{y}\rangle|P)$. Using bound names has significance in making the representation of computational behaviour as tight as possible: given some behaviour which we wish to model, the way of representing it in the typed calculus becomes strongly constrained and thus, for example, we own a fairly tractable notion of inhabitants in each type. A natural question however arises: can we impose behavioural constraints of the similar kind on terms with free name passing, i.e. in the unconstrained (asynchronous) π-calculus? And if we can, does it add any expressive power? This is not only intellectual curiosity. Apart from the simplicity of the presentation (by moving to free name passing we can get rid of a couple of added structural rules), free name passing makes the computation more tractable: it also has technical advantages in the higher-order setting, which we are currently investigating.

The present note studies these questions, extending the syntax in [3] to free outputs while using precisely the same type structures. The typing rules do not change except for those free outputs. The rules for free outputs are suggested by a natural representation of free outputs in the original system (for details see Section 2). In fact, this representation gives a simple mapping of free outputs to the terms which only use bound outputs, folding all terms of the system with free outputs onto those of the system with bound outputs. On the other hand, the embedding of terms of the system with bound outputs into the system with free outputs is essentially the injection. After presenting these translations, we shall show that these two maps not only preserve types but also preserve semantics: they do not change the behaviour of processes up to the canonical equalities. In fact they form a retract in both ways up to the respective semantic equalities, showing the universe of linear terms with free outputs are semantically equivalent to its strict subset which use only bound outputs, thus answering the questions

*Queen Mary and Westfield College, University of London, U.K.

we posed at the outset. These and other results also give a firm basis on our ongoing study on the use of typed free name passing with more complex type structure.

Another possible, albeit secondary, interest of the present note is that we use one-sided sequents in typing (which we write $\vdash P \triangleright A$), unlike the two-sided ones in [1, 3] (which we write $\Gamma \vdash P \triangleright A$). This is equivalent to the original presentation, while enabling more concise, but arguably less illustrative and sometimes inconvenient, presentation. It may also be useful when we compare the present system with other constructions such as Linear Logic. Our motivations in studying these typing systems is to offer a set of basic syntactic tools for the study of computational behaviours using the π-calculus. We hope the one-sided sequent is a useful addition to such a tool set.

The next section, Section 2, offers basic syntactic preliminaries and introduces the typing system for processes with free name passing. Section 3 presents the main results, showing the fully abstract mutual embeddings between two systems. We also discuss other syntactic results and open questions.

2 Two Systems

In this section we review the linear π-calculus as presented in [3] and introduce its free name passing variant. We work with the syntax without branching and selection. While they play an essential role in both expressiveness and semantics, the central ideas are better presented in a simpler, unary, setting.

2.1 Types

The set of *action modes* is given by $\{\downarrow, \uparrow, !, ?, \bot\}$, ranged over by $p, q, \ldots$. The duality is defined on these modes except on $\bot$, by: $\overline{\downarrow} = \uparrow$, $\overline{!} = ?$ and their converse. We set a partial symmetric algebra $p \odot q$ by (1) $\downarrow \odot \uparrow = \bot$, (2) $? \odot ? = ?$ and (3) $! \odot ? = !$. Using these action modes, the set of *channel types* is given as follows.

$$\alpha ::= \langle \tau,\ \overline{\tau} \rangle \qquad \tau ::= \tau_{\mathtt{I}} \mid \tau_{\mathtt{O}} \qquad \tau_{\mathtt{I}} ::= (\vec{\tau}_{\mathtt{O}})^{\downarrow} \mid (\vec{\tau}_{\mathtt{O}})^{!} \qquad \tau_{\mathtt{O}} ::= (\vec{\tau}_{\mathtt{I}})^{\uparrow} \mid (\vec{\tau}_{\mathtt{I}})^{?}$$

In the first line, $\overline{\tau}$ denotes the *dual* of τ, which is the result of dualising all action modes. A type of form $\langle \tau, \overline{\tau} \rangle$, called a *pair type*, is an unordered pair of mutually dual types. We write $\delta, \delta', \ldots$ for the union of both non-pair and pair channel types. We define $\mathsf{md}(\delta)$ by (1) $\mathsf{md}((\vec{\tau})^p) = p$ and (2) $\mathsf{md}(\langle \tau, \overline{\tau} \rangle) = \mathsf{md}(\delta) \odot \overline{\mathsf{md}(\delta)}$. The partial symmetric algebra $\delta_1 \odot \delta_2$ is defined following $p_1 \odot p_2$: (i) $\tau \odot \overline{\tau} = \langle \tau, \overline{\tau} \rangle$, (ii) $\tau \odot \tau = \tau$ when $\mathsf{md}(\tau) = ?$ and (iii) $\langle \tau, \overline{\tau} \rangle \odot \overline{\tau} = \langle \tau, \overline{\tau} \rangle$ when $\mathsf{md}(\tau) = !$.

Finally an *action type*, denoted $A, B, \ldots$, is a finite directed graph with nodes of the form x^{δ}, such that:

- no names occur twice; and
- if there is an edge, say $x^{\delta} \rightarrow y^{\delta'}$, then either (1) $\mathsf{md}(\delta) = \downarrow$ and $\mathsf{md}(\delta') = \uparrow$, or (2) $\mathsf{md}(\delta) = !$ and $\mathsf{md}(\delta') = ?$.

If x^δ is in A and for no y we have $y^{\delta'} \to x^\delta$, then we say x is *active in* A. $|A|$ (resp. $\mathsf{fn}(A)$, $\mathsf{sbj}(A)$, $\mathsf{md}(A)$) denotes the set of nodes (resp. names, active names, modes) in A. $A\backslash\vec{x}$ is the result of taking off nodes with names in $\vec{x}$ from A. $A \uplus B$ is the graph union of A and B. $A \odot B$ is defined (denoted $A \asymp B$) iff: (1) $x^\delta \in |A|$ and $x^{\delta'} \in |B|$ implies $\delta \odot \delta'$ is defined, and (2) It does not induce circular dependency (i.e. there is a sequence of edges from x to x) when composing all edges. When defined, $A \odot B$ is the result of composed nodes connected by the composed edges (for the latter neglecting all intermediate nodes, cf. [3]).

The action types above combine information of channel types with action types in the sense of [3]. Thus, instead of $\Gamma \vdash P \triangleright A$, we can write the sequent as $\vdash P \triangleright A$. This is more concise but is less convenient when we consider e.g. typed transition. The latter however does not matter in the present exposition.

2.2 Linear Typing with Bound Name Passing

We first offer the syntax with bound name output.

$$P \ ::= \ x(\vec{y}).P \ \mid \ !x(\vec{y}).P \ \mid \ \overline{x}(\vec{y})P \ \mid \ P|Q \ \mid \ (\boldsymbol{\nu}\, x)P \ \mid \ \mathbf{0}$$

The structural equality includes those which make the output prefix asynchronous, e.g. $(\overline{x}(\vec{y})P)|Q \equiv \overline{x}(\vec{y})(P|Q)$ when $\mathsf{fn}(P) \cap \{\vec{y}\} = \emptyset$. The reduction is given starting from the following rules:

$$x(\vec{y}).P|\overline{x}(\vec{y})Q \longrightarrow (\boldsymbol{\nu}\,\vec{y})(P|Q) \qquad !x(\vec{y}).P|\overline{x}(\vec{y})Q \longrightarrow !x(\vec{y}).P|(\boldsymbol{\nu}\,\vec{y})(P|Q).$$

The typing system for this syntax is given in Appendix A, using the sequent $\vdash_{\mathrm{b}} P \triangleright A$. The system, which we call BNP, is equivalent to the one in [3] in the following sense.

(i) If $\Gamma \vdash P \triangleright A$ in the system in [3] then $\vdash_{\mathrm{b}} P \triangleright \Gamma(A)$. Here $\Gamma(A)$ assigns channel types to names in A as far as their modes conform to the action modes given in A.
(ii) If $\vdash_{\mathrm{b}} P \triangleright A$ then $|A| \vdash P \triangleright \langle A \rangle$ in the system in [3]. Here $|A|$ forgets causality from A, while $\langle A \rangle$ extracts the action mode assignment in $\mathsf{fn}(A)$ from the channel type assignment in A.

By this correspondence we immediately know that whenever $\vdash_{\mathrm{b}} P \triangleright A$ we have P is church rosser and strongly normalising.

2.3 Linear Typing with Free Name Passing

The syntax of processes with free name passing is simply the standard asynchronous polyadic π-calculus.

$$P; ::= \ x(\vec{y}).P \ \mid \ !x(\vec{y}).P \ \mid \ \overline{x}\langle\vec{y}\rangle \ \mid \ P|Q \ \mid \ (\boldsymbol{\nu}\, x)P \ \mid \ \mathbf{0}$$

Note the bound output $\overline{x}(\vec{y})P$ can be recovered as $(\boldsymbol{\nu}\,\vec{y})(\overline{x}\langle\vec{y}\rangle|P)$, so that the second syntax in fact subsumes the first one. The typing rules for these processes are exactly the same as those of BNP except the sequent is now written $\vdash_{\mathrm{b}} P \triangleright A$ and that we use the following rules for outputs.

$$\frac{(\mathsf{Out}^{\uparrow}) \quad (\forall i.\rho_i = \overline{\tau_i}) \qquad -}{\vdash_{\mathrm{f}} \overline{x}\langle \vec{y} \rangle \triangleright x^{(\vec{\tau})^{\uparrow}} \otimes \vec{y}^{\vec{\rho}}} \qquad\qquad \frac{(\mathsf{Out}^{\uparrow}) \quad (\forall i.\rho_i = \overline{\tau_i}) \qquad -}{\vdash_{\mathrm{f}} \overline{x}\langle \vec{y} \rangle \triangleright x^{(\vec{\tau})^{\mathbf{?}}} \otimes \vec{y}^{\vec{\rho}}}$$

Note the types for object names in the above two rules are dualised. The resulting system is denoted FNP.

The rules for outputs given above, are best understood in terms of the following representation of free outputs in the realm of bound outputs.

$$\overline{x}\langle \vec{y}^{\vec{\tau}} \rangle^{\circ} \stackrel{\mathrm{def}}{=} \overline{x}(\vec{w})\Pi_i[w_i \to y_i]^{\tau_i}$$

Here $[w_i \to y_i]^{\tau_i}$ is the standard *copy-cat agent*, whose definition is left to [1, 3]. The above representation says that a free name output is a bound name output in which all exported names are "equated" with the mentioned free names. Note we used type annotations. This shows how the construction essentially depends on type structures.[1] In this representation, w_i is used as τ_i and, as a result, y_i is used as $\overline{\tau}_i$. Thus, in fact, we have the following equi-typability:

$$\vdash_{\mathrm{f}} \overline{x}\langle \vec{y} \rangle \triangleright A \quad \Leftrightarrow \quad \vdash_{\mathrm{b}} \overline{x}\langle \vec{y} \rangle^{\circ} \triangleright A$$

(the types of $\vec{y}$ are omitted since they are inferable from A). The same holds when we extend the encoding compositionally, which we also write P°. We shall be using this encoding for relating the two systems.

It is easy to observe that the new system satisfies such basic properties as the subject reduction. Further we note:

Proposition 1. *If $\vdash_f P \triangleright A$ then P is church-rosser and strongly normalising.*

The proof of CR follows [3]. For strong normalisability, we use the following extended reduction:

$$\begin{aligned} x(\vec{y}).P|C[\overline{x}(\vec{y})Q] &\mapsto C[(\boldsymbol{\nu}\, \vec{y})(P|Q)] \\ !x(\vec{y}).P|C[\overline{x}(\vec{y})Q] &\mapsto !x(\vec{y}).P|C[(\boldsymbol{\nu}\, \vec{y})(P|Q)] \\ (\boldsymbol{\nu}\, x)!x(\vec{y}).P &\mapsto \mathbf{0} \end{aligned}$$

We then use the following correspondence in extended reduction via the mapping: whenever $P \longrightarrow^{*} Q$ we have $P^{\circ} \longrightarrow^{*} Q'$ where Q' is the result of adding copy-cats to Q° in the way e.g. $(\boldsymbol{\nu}\, y)(Q^{\circ}\{y/x\}|[y \to x]$ where y is fresh. By this we can show $P \mapsto^{n}$ implies $P^{\circ} \mapsto^{m}$ such that $m \geq n$. Since we already know, by [3], that P° is strongly normalising w.r.t. $\mapsto$, so is P. Since $\longrightarrow \subset \mapsto$ we are done.

[1] This however does not mean that a similar idea does not work in the untyped setting. We can in fact use *equators* (which are recently studied by, among others, Sangiorgi, Boreale, Merro, Gardner) to realise essentially the same effects.

3 Mutual Embeddings

3.1 Preliminaries

We use the following construction to obtain a decent equality over typed processes in each system. First we extend the syntax to include branching (of forms $x[\&_{i\in I}(\vec{y}_i).P_i]$ and $!x[\&_{i\in I}(\vec{y}_i).P_i]$ with I non-empty) and selection (of form $\overline{x}\mathtt{in}_i(\vec{y})P$ in BNP and $\overline{x}\mathtt{in}_i\langle\vec{y}\rangle$ in FNP). The reduction is then given (in the case of unary free output) as

$$x[\&_i(\vec{y}_i).P_i] \mid \overline{x}\mathtt{in}_i\langle\vec{v}\rangle \longrightarrow P\{\vec{v}/\vec{y}_i\}$$

The typing rules are given as in [3], which we omit. We then have the following notion of convergence (we show the case of BNP: the definition for FNP is given accordingly). Below we say A' is *closed* if $x^{\delta} \in |A|$ implies δ is paired (thus $A' = \otimes x_i^{\alpha_i}$).

$$\vdash_{\mathrm{b}} P \triangleright A \Downarrow_x^i \quad \stackrel{\text{def}}{\Leftrightarrow} \quad P \longrightarrow^* \overline{x}\mathtt{in}_i(\vec{y})P|Q,\ A = A' \otimes x^{\tau},\ A' \text{ closed and } \mathsf{md}(\tau) = \uparrow$$

Now consider the set of typed terms in BNP. A typed congruence $\cong$ over these terms is *sound*, cf.[2], iff: (1) $\cong$ is *consistent*, i.e. $\equiv \subset \cong \subsetneq \mathbf{U}$ ($\mathbf{U}$ is the universal typed congruence); (2) $\cong$ is *reduction-closed*, i.e. whenever $\vdash_{\mathrm{b}} P \cong Q \triangleright A$ and $P \longrightarrow P'$ we have $Q \longrightarrow^* Q'$ and $\vdash_{\mathrm{b}} P' \cong Q' \triangleright A$; and (3) P *respects insenstivity*, i.e. whenever P is insensitive (i.e. P's reduction derivatives never own an active input and output) we have $\vdash_{\mathrm{b}} P \cong \mathbf{0} \triangleright \emptyset$. We can then show there is the maximum sound equality, via the *observability lemma*, cf.[2]: for each sound $\cong$, whenever $\vdash_{\mathrm{b}} P \cong Q \triangleright A \otimes x^{\tau}$ with A closed and $\mathsf{md}(\tau) = \uparrow$, we have $P \Downarrow_x^i$ iff $Q \Downarrow_x^i$. This immediately allows us to take the union of all sound congruences to obtain the maximum congruence. We write this maximum congruence $\cong_{\mathrm{b}}$. Similarly we write $\cong_{\mathrm{f}}$ for the maximum sound congruence in FNP. We can easily prove the following.

Proposition 2. *$P \cong_b Q$ iff for each typed $C[P]$ and for each name x we have $C[P] \Downarrow_x^i \Leftrightarrow C[Q] \Downarrow_x^i$. Similarly for $\cong_f$.*

PROOF: The "only if" direction is by the observability lemma mentioned above. For the "if" direction, we show the congruence thus characterised is reduction-closed, which is immediate from the Church-Rosser property. ■

The following equation plays a central role in the embedding results in the next subsection. Below we regard $|A|$ as the map from free names to the assigned channel types.

Lemma 1. *For each $\vdash_b P \triangleright A$ with $\mathsf{md}(|A|(x)) \in \{\uparrow, \mathbf{?}\}$ and a fresh name y, we have $(\boldsymbol{\nu}\, x)(P|[x \to y]) \cong_b P\{y/x\}$. Similarly, for each $\vdash_f P \triangleright A$ with $\mathsf{md}(|A|(x)) \in \{\uparrow, \mathbf{?}\}$, we have $(\boldsymbol{\nu}\, x)(P|[x \to y]) \cong_f P\{y/x\}$.*

The proof is immediate by $(\boldsymbol{\nu}\, x)(P|[x \to y]) \mapsto^* P\{y/x\}$ and by noting $\mapsto \subset \cong_{\mathrm{b}}$ (resp. $\mapsto \in \cong_{\mathrm{f}}$). The latter in turn is proved by showing the congruent closure of $\mapsto$ is sound (which holds by the commutativity of $\mapsto$ w.r.t. $\longrightarrow$). Alternatively we can use the technique by Boreale and Sangiorgi, giving a bound-name transition to FNP, whose associated bisimilarity is easily a sound congruence.

3.2 Mutual Fully Abstract Embeddings

Let $\vdash_{\mathrm{b}} P \triangleright A$. Then we write $P^\star$ for the result of translating P into processes with free name passing by turning $\overline{x}(\vec{y})P$ into $(\boldsymbol{\nu}\, \vec{y})(\overline{x}\langle\vec{y}\rangle|P)$. We observe:

Proposition 3. (type preservation) $\vdash_b P \triangleright A$ *iff* $\vdash_f P^\star \triangleright A$. *Similarly* $\vdash_f P \triangleright A$ *iff* $\vdash_f P^\circ \triangleright A$.

Both statements are easy to show by observing how the polarity of object names in outputs are treated in these maps. We can now state one of the main results of this note.

Theorem 1. (both-way retracts) *For each* $\vdash_b P \triangleright A$, *we have* $P^{\star\circ} \cong_b P$. *Similarly, for each* $\vdash_f P \triangleright A$, *we have* $P^{\circ\star} \cong_f P$.

Proof: For the first half, we use induction. The only non-trivial case is bound output. Let $\vdash_{\mathrm{b}} \overline{x}(\vec{y})P \triangleright A$. Then

$$\begin{aligned}
(\overline{x}(\vec{y})P)^{\star\circ} &\stackrel{\text{def}}{=} (\boldsymbol{\nu}\, \vec{y})((\overline{x}(\vec{w})\Pi_i[w_i \to y_i])|P^{\star\circ}) \\
&\equiv \overline{x}(\vec{w})(\boldsymbol{\nu}\, \vec{y})(\Pi_i[w_i \to y_i]|P^{\star\circ}) \\
&\cong_{\mathrm{b}} \overline{x}(\vec{y})(P^{\star\circ}) \\
&\cong_{\mathrm{b}} \overline{x}(\vec{y})P
\end{aligned}$$

The last two equations are by Lemma 1 and by induction hypothesis, respectively. The second half is also by induction, which boils down to the establishment of the equality $\overline{x}\langle\vec{y}\rangle^{\circ\star} \cong \overline{x}\langle\vec{y}\rangle$. In fact, using Lemma 1, we have:

$$\begin{aligned}
\overline{x}\langle\vec{y}\rangle^{\circ\star} &\stackrel{\text{def}}{=} (\boldsymbol{\nu}\, \vec{w})(\overline{x}\langle\vec{w}\rangle|\Pi_i[w_i \to y_i]) \\
&\cong \overline{x}\langle\vec{y}\rangle
\end{aligned}$$

as required. ■

Theorem 1 shows, among others, that all additional terms in FNP which do not exist in BNP are semantically equivalent to their image in BNP, so that FNP does not add anything to BNP semantically. Further it says that this map is semantically the identity on BNP as embedded in FNP via $(\ )^\star$. We now conclude the present note with the following result.

Corollary 1. (full abstraction) $\vdash_b P \cong_b Q \triangleright A$ *iff* $\vdash_f P^\star \cong_f Q^\star \triangleright A$. *Similarly* $\vdash_f P \cong_f Q \triangleright A$ *iff* $\vdash_b P^\circ \cong_b Q^\circ \triangleright A$.

For the proof we use the characterisation in Proposition 2. Suppose $P \cong_{\mathrm{b}} Q$ and $C[P^\star] \Downarrow^i_x$. By Theorem 1, $R^\circ \cong_{\mathrm{f}} R$, so that $(C[P^\star])^\circ \Downarrow^i_x$, that is $C^\circ[P^{\star\circ}] \Downarrow^i_x$. Again by Theorem 1 we have $C^\circ[P] \Downarrow^i_x$. By assumption we have $C^\circ[Q] \Downarrow^i_x$, that is $(C^\circ[Q])^\star \Downarrow^i_x$, from which we know, again by Theorem 1, $C[Q^\star] \Downarrow^i_x$, as required. The converse is trivial since $(\ )^\star$ is essentially identity. The proof of the second half is precisely symmetric.

References

1. Berger, M. Honda, K. and N. Yoshida. *Sequentiality and the π-Calculus.* TLCA'01, 2001.
2. Honda, K. and Yoshida, N., On Reduction-Based Process Semantics. *TCS*, 437–486, Vol. 151, North-Holland, 1995.
3. Yoshida, N., Berger, M. and Honda, K., *Strong Normalisation in the π-Calculus*, To appear as a MCS Technical Report, University of Leicester, 2001.

A The System with Bound Name Output

The following typing system is merely a different presentation of what is given in [3].

(Zero)
$$\frac{-}{\vdash_{\mathrm{b}} \mathbf{0} \triangleright \emptyset}$$

(Par)
$$\frac{\vdash_{\mathrm{b}} P_i \triangleright A_i \quad (i=1,2) \qquad A_1 \asymp A_2}{\vdash_{\mathrm{b}} P_1|P_2 \triangleright A_1 \odot A_2}$$

(Res)
$$\frac{\vdash_{\mathrm{b}} P \triangleright C \qquad C/x^\alpha = A}{\vdash_{\mathrm{b}} (\boldsymbol{\nu}\, x)P \triangleright A}$$

(In$^\downarrow$) $(C/\vec{y}^{\vec{\tau}} = \uparrow A \otimes ?B)$
$$\frac{\vdash_{\mathrm{b}} P \triangleright C^{-x}}{\vdash_{\mathrm{b}} x(\vec{y}).P \triangleright (x^{(\vec{\tau})^\downarrow} \to A) \otimes B}$$

(Out$^\uparrow$) $(C/\vec{y}^{\vec{\tau}} = A \asymp x^{(\vec{\tau})^\uparrow})$
$$\frac{\vdash_{\mathrm{b}} P \triangleright C}{\vdash_{\mathrm{b}} \overline{x}(\vec{y})P \triangleright A \odot x^{(\vec{\tau})^\uparrow}}$$

(Weak-$\perp$)
$$\frac{\vdash_{\mathrm{b}} P \triangleright A^{-x} \qquad \mathsf{md}(\alpha) = \perp}{\vdash_{\mathrm{b}} P \triangleright A \otimes x^\alpha}$$

(In$^!$) $(C/\vec{y}^{\vec{\tau}} = ?A)$
$$\frac{\vdash_{\mathrm{b}} P \triangleright C^{-x}}{\vdash_{\mathrm{b}} !\, x(\vec{y}).P \triangleright x^{(\vec{\tau})^!} \to A}$$

(Out$^?$) $(C/\vec{y}^{\vec{\tau}} = A \asymp x^{(\vec{\tau})^?})$
$$\frac{\vdash_{\mathrm{b}} P \triangleright C}{\vdash_{\mathrm{b}} \overline{x}(\vec{y})P \triangleright A \odot x^{(\vec{\tau})^?}}$$

(Weak-$?$)
$$\frac{\vdash_{\mathrm{b}} P \triangleright A^{-x}}{\vdash_{\mathrm{b}} P \triangleright A \otimes x^{(\vec{\tau})^?}}$$