GRAPH TYPES FOR MONADIC MOBILE PROCESSES

NOBUKO YOSHIDA

Department of Computer Science
University of Edinburgh
The King's Buildings
Mayfield Road
Edinburgh, EH9 3JZ, UK
e-mail: `ny@dcs.ed.ac.uk`

abstract

While types for name passing calculi have been studied extensively in the context of sorting of polyadic π-calculus [26, 7, 43, 11, 36, 41, 24, 42, 14, 22], the type abstraction on the corresponding level is not possible in the monadic setting, which was left as an open issue by Milner [26]. We solve this problem with an extension of sorting which captures dynamic aspects of process behaviour in a simple way. Equationally this results in the full abstraction of the standard encoding of polyadic π-calculus into the monadic one: the sorted polyadic π-terms are equated by the basic behavioural equality in the polyadic calculus if and only if their encodings are equated in the basic typed behavioural equality in the monadic calculus. This is the first result of this kind we know of in the context of the encoding of polyadic name passing, which is a typical example of translation of high-level communication structures into π-calculus. The construction is general enough to be extendable to encodings of calculi with more complex operational structures.

1. Introduction

The monadic π-calculus [28, 25] is a powerful formalism in which we can construct complex structures of concurrent computing by combining simple monadic name passing. The construction of significant computational structures in this calculus is done by passing and using private names between interacting parties to control the sharing of interaction points. For example, the following process expresses communication of a sequence of names (below $ax.P$ is input and $\overline{a}v.P$ is output, $(a)P$ denotes scope restriction, and c, z are fresh).

$$az.zx_1.zx_2.zx_3.P \mid (c)\ \overline{a}c.\overline{c}v_1.\overline{c}v_2.\overline{c}v_3.Q \ \longrightarrow\!\!\!\!\rightarrow\ P\{v_1v_2v_3/x_1x_2x_3\} \mid Q \tag{1.1}$$

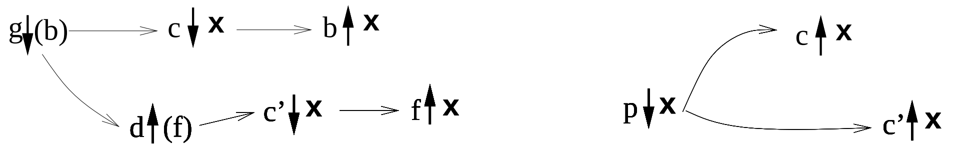
In this example, coming from [28], the private channel c is used during interaction, so that, after the initial step, the communication of v_1, v_2 and v_3 is done deterministically without interference from the third party. This example also shows that we can represent polyadic (multiple) name passing from monadic name passing. Another example with a more complex communication structure follows.

$$az.zx_1.\overline{z}v_1.zx_2.\overline{z}v_2.P \mid (c)\ \overline{a}c.\overline{c}w_1.cy_1.\overline{c}w_2.cy_2.Q \ \longrightarrow\!\!\!\!\rightarrow\ P\{w_1w_2/x_1x_2\} \mid Q\{v_1v_2/y_1y_2\} \tag{1.2}$$

Note input and output are mixed together. As in the previous example, once two parties start interaction, a sequence of communication is done following a prescribed protocol using a private channel. The same scheme can be easily generalised to more complex communication structures, including those with parallelism and complex information flow.

As a means to study rich computational structures representable in name passing processes, a notion of types called *sorting* was introduced by Milner [26] and has been studied extensively since then [7, 43, 11, 36, 41, 24, 42, 14, 22]. Sorting shows how a name carries another name. For example, if v has a type, say, nat, and we have a term $\overline{a}v.\mathbf{0}$, then a should have a type (nat), a type which carries nat. This idea and its ramifications have been used to analyse significant semantic properties of calculi with polyadic name passing. However, as was already noticed in [26], the sorting in the monadic π-calculus is not powerful enough to type-abstract known encodings of polyadic name passing like (1.1) above: Indeed, (1.1) becomes ill-sorted if v_1, v_2, v_3 have different sorts. As far as we know, the situation remains the same with the refinements of sorting proposed in [36, 22, 14]. This means, among other things, the sorting restricted to the monadic terms does not give as rich semantic analysis as in the polyadic case, while behaviourally the above encoding does mimic polyadic name passing. In general, this is because the sorting does not capture the dynamic structure of interaction, especially those using the transmission of private names as in (1.1) and (1.2), which is omnipresent even in the polyadic setting.

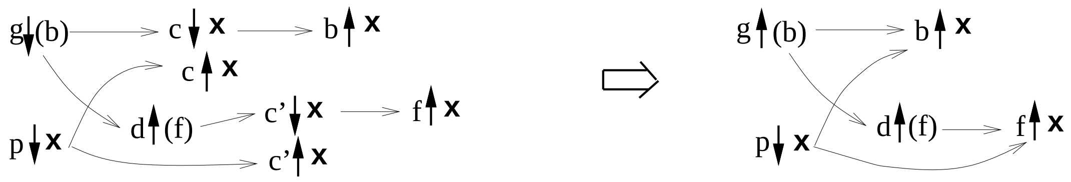
The present paper develops a syntactic theory of types for monadic π-calculus which extends the sorting in a simple way and by which we can extract the abstract operational structures of π-terms including those of (1.1), (1.2) as well as more complex ones (cf.2.5). The key technical idea is to represent a class of dynamic communication behaviour of mobile processes using simple combinatorial expressions, i.e. graphs. In a graph type, nodes denote atomic actions and edges denote the activation ordering between them, representing as a whole a deterministic protocol a process will be engaged in. As a simple example, suppose we have two terms $gb.(cx.\overline{b}x \mid (f)\overline{d}f.c'x.\overline{f}x)$ and $px.(\overline{c}x \mid \overline{c'}x)$. Then they may be given types: Here on the left side, the action



$g \downarrow (b)$ (the input of fresh name b) should take place directly preceding $c \downarrow \mathbf{x}$ and $d \uparrow (f)$ (and indirectly all the rest), while there is no activation ordering between, for example, $c \downarrow \mathbf{x}$ and $f \uparrow \mathbf{x}$ in the same graph. Now if two protocols are compatible, we can compose them via "cuts," i.e. pairs of complementary atomic actions, to yield more complex types. Think of the following composition of the above two terms:

$$(cc'f)\ (\ \overline{g}b.(cx.\overline{b}x \mid \overline{d}f.c'x.\overline{f}x) \mid px.(\overline{c}x \mid \overline{c'}x)\ ) \tag{1.3}$$

The process of giving a type to this composed term is illustrated in the following.



The picture shows how the term (1.3) is given a type on the right hand side after the procedure of "cut elimination." Notice how the original activation ordering relations are merged to generate a new relation, so that a proper graph structure arises even if the original types are trees.[1] In the above example, the existence of two arrows going to $b \uparrow \mathbf{x}$ shows that the action should take place after its two parent actions take place. It also shows how the graph-based representation allows a refined expression of the flow of control in comparison with the syntactic construct like prefix, while still keeping a simple combinatorial structure. Compared with sorting and its refinements, the graph type tries to capture dynamic interactive behaviour of name passing processes including the new name passing (as $d \uparrow (f)$ above) based on a simple graphical expression.

This departure from the type abstraction of static usage of names to that of dynamic process behaviour makes it possible to type-abstract many non-trivial computational structures representable in π-calculus. Indeed it allows us to solve Milner's open

[1]We also note that such protocol composition enables the distribution and merging of information in communication, and the increase of parallelism, so can have a practical significance. It also plays a key role in theory of combinators for mobile processes [18, 19, 46].

issue mentioned above in a sharper form. We show that not only the encoding of sorted polyadic π-terms into the monadic terms is typable preserving the original type structure, but also it results in *equational full abstraction*: let $=_{\mathrm{p}}$ and $=_\pi$ be suitably defined behavioural equalities over sorted polyadic π-terms and monadic π-terms, respectively. Then we get:

$$P =_{\mathrm{p}} Q \quad \Leftrightarrow \quad [\![P]\!] =_\pi [\![Q]\!]$$

where $[\![\cdot]\!]$ is the standard encoding from polyadic π-terms to monadic π-terms. In the untyped setting, we can only get "$\Leftarrow$" (adequacy) direction in terms of usual weak behavioural equivalences. This is due to possible violation of "protocols" by environment processes, and relates to the lack of precise type abstraction of the communication structure of various encodings such as (1.1) and (1.2) in preceding type disciplines, even though the effect of process types on behavioural equalities has been studied since the work of Pierce and Sangiorgi [36, 22]. In this context, our result (which seems the first of this kind with respect to the encoding of polyadic name passing) shows how precisely our type discipline captures essential behavioural information these protocols carry. What may be surprising is that this can be done using a simple idea of graph-based types and a small typing system. Moreover the results can be easily extended to encodings of calculi with more complex communication structures, cf. 7.17/7.18 later. We also notice that the polyadic name passing is a typical example of encoding of high-level communication structures into π-calculus, cf. [25, 21, 44], so that the presented construction and its extensions will hopefully become a basis for using typed equality over π-terms to verify semantic equality of various target languages and calculi in a uniform framework.

We now give the structure of the rest of the paper. Section 2 reviews the basic definition of the untyped monadic π-calculus. Section 3 introduces graph types and their composition. Section 4 gives the typing system for monadic π-terms and proves Subject Reduction property of typed terms. Section 5 introduces reduction-based equality over typed π-terms and clarifies the behavioural properties which the types given to π-terms ensure. Section 6 studies basic properties of reduction-based equality. Section 7 then uses the preceding constructions to show that the standard encoding of polyadic π-calculus into monadic π-calculus preserves both type structures and equality of the original world. We also refer to extensions of the result to encodings of more complex structures at the end. Section 8 concludes the paper with comparison with related work and discussions on further works.

2. Preliminaries on Untyped Monadic π-calculus

2.1. **Terms.** We mainly work with the following syntax [28, 25], where $a, b, c, ..$ or $x, y, z, ..$ range over a countable set $\mathcal{N}$ of names (fixed throughout the paper).

$$P \quad ::= \quad ax.P \;\mid\; \overline{a}x.P \;\mid\; P|Q \;\mid\; (a)P \;\mid\; !P \;\mid\; \mathbf{0}$$

$P, Q, R, \ldots$ range over the set of terms generated by the above grammar. "$\overline{a}v.P$" denotes an agent which sends a value v to a port a and becomes P. "$ax.P$" denotes an agent which receives a name and instantiates it in free x's in P. In $ax.P$, the name x binds free occurrences of x in P (like x in $\lambda x.M$). "$(a)P$" is a name hiding of a in P. The initial a binds its free occurrences in P. "$P|Q$" is a *parallel composition* of P and Q. "$!P$" is a *replicator* which represents the copy of P. "$\mathbf{0}$" is a syntactic convention to denote inaction. The set of *free (resp. bound) names* in P is denoted by $\mathcal{FN}(P)$ (resp. $\mathcal{BN}(P)$). The usual notions of simultaneous substitutions and α-convertibility are assumed. Let us define the set of *active names* as: $a \in \mathcal{AN}_+(P)$ iff $P \equiv (\tilde{c})(\overline{a}v.Q \mid R)$ with $a \notin \{\tilde{c}\}$, and $a \in \mathcal{AN}_-(P)$ iff $P \equiv (\tilde{c})(ax.Q \mid R)$ with $a \notin \{\tilde{c}\}$. Such a occurs *actively* in P.

A term *obeys the binding condition* if bound names are disjoint from free names and binding names are pairwise distinct. We often write $(ab)P$ to denote $(a)(b)P$. $\overline{a}(x).P$ denotes $(x)\overline{a}x.P$. We let "$|$" to be the weakest in association, e.g. $(a)P \mid Q \overset{\text{def}}{=} ((a)P) \mid Q$, and associates to the left. We often cut off the trailing inaction, i.e. $\overline{a}v$ denotes $\overline{a}v.\mathbf{0}$.

Henceforth, terms are often considered modulo the structural congruence, following [25, 3].

2.2. **Structure Rule.** $\equiv$ is the smallest congruence relation over π-terms generated by the following rules.

(i) $P \equiv Q$ if $P \equiv_\alpha Q$

(ii) $P, Q \equiv Q, P \qquad (P, Q), R \equiv P, (Q, R) \qquad P, \mathbf{0} \equiv P \qquad !P \equiv P \mid !P$

(iii) $(aa)P \equiv (a)P \quad (ab)P \equiv (ba)P \quad (a)\mathbf{0} \equiv \mathbf{0} \quad (a)P, Q \equiv (a)(P, Q)$ if $a \notin \mathcal{FN}(Q)$

The reduction relation which provides the notion of computing is given in the following.

2.3. **Reduction.** The one-step reduction relation $\longrightarrow$ over π-terms is generated by the following rule.

$$\begin{array}{ll}
\text{(COM)} & ax.P \mid \overline{a}v.Q \longrightarrow P\{v/x\} \mid Q \\
\text{(PAR)} & P \longrightarrow Q \;\Rightarrow\; P \mid R \longrightarrow Q \mid R. \\
\text{(RES)} & P \longrightarrow Q \;\Rightarrow\; (a)P \longrightarrow (a)Q. \\
\text{(STR)} & P \equiv P' \;\; P' \longrightarrow Q' \;\; Q \equiv Q' \;\Rightarrow\; P \longrightarrow Q.
\end{array}$$

The multi-step reduction relation, $\longrightarrow\!\!\!\!\!\rightarrow$, is defined by $\longrightarrow\!\!\!\!\!\rightarrow \overset{\text{def}}{=} \longrightarrow^* \cup \equiv$.

We shall also use the standard early labeled transition relation, written $P \xrightarrow{l} P'$, as given in e.g. [28].

Several π-terms with significant operational structures are listed in the following. These examples will often be used later.

2.4. **Example.** (i) (polyadic name passing) Let us assume c and y are fresh.

$$a^* : (x_1..x_n).P \stackrel{\text{def}}{=} ay.yx_1.yx_2...yx_n.P$$
$$\overline{a}^* : \langle v_1..v_n\rangle.P \stackrel{\text{def}}{=} \overline{a}(c).\overline{c}v_1.\overline{c}v_2...\overline{c}v_n.P$$

Then we have: $a^* : (x_1x_2).P \mid \overline{a}^* : \langle v_1v_2\rangle.Q \longrightarrow\!\!\!\!\rightarrow P\{v_1v_2/x_1x_2\} \mid Q$. Moreover, with $=_\pi$ being the maximum sound equality [20] on π-terms, we have:

$$a^* : (x_1x_2).P \mid \overline{a}^* : \langle v_1v_2\rangle.Q \longrightarrow =_\pi P\{v_1v_2/x_1x_2\} \mid Q$$

which shows the mutual communication steps after the initial interaction are deterministic, and do not meet interference from outside since they use private names.

(ii) (mixed input/output structure) The above encoding can be easily extended to mixed input and output. We denote $(\tilde{x}_i)[\tilde{x}_{i+1}]_{i\leq 2n} \stackrel{\text{def}}{=} (\tilde{x}_1)[\tilde{x}_2]...(\tilde{x}_{2n-1})[\tilde{x}_{2n}]$ where $\tilde{x}_i = x_{i1}...x_{ik_i}$ with $k_i, n \geq 0$. Similarly for $[\tilde{x}_i](\tilde{x}_{i+1})_{i\leq 2n}$. Then define:

$$a^* : (\tilde{x}_i)[\tilde{v}_{i+1}]_{i\leq 2n}.P \stackrel{\text{def}}{=} az.zx_{11}..zx_{1k_1}.\overline{z}v_{21}..\overline{z}v_{2k_2}...\overline{z}v_{2n1}..\overline{z}v_{2nk_{2n}}.P$$
$$\overline{a}^* : [\tilde{x}_i](\tilde{v}_{i+1})_{i\leq 2n}.P \stackrel{\text{def}}{=} \overline{a}(c).\overline{c}v_{11}..\overline{c}v_{1k_1}.cx_{21}..cx_{2k_2}...cx_{2n1}..cx_{2nk_{2n}}.P$$

with z, c fresh and $x_{ij} \neq x_{kl}$ and $x_{ij} \neq v_{kl}$ for all $i \neq k$ and $j \neq l$. Then we have

$$a^* : (x_1x_2)[v](x_3).P \mid a^* : [w_1w_2](y)[w_3].Q \longrightarrow\!\!\!\!\rightarrow P\{v/y\} \mid Q\{w_1w_2/x_1x_2\}$$

as well as the safety property as in (1).

(iii) (parallel name passing and merging) A more complicated but significant structure follows. We use the notation in (i), with $\overline{a}^* : (\tilde{c}).P$ standing for $(\tilde{c})(\overline{a^*\langle\tilde{c}\rangle.P})$ and $\prod_{i\leq n} P_i$ for $P_1 \mid P_2 \mid ... \mid P_n$. Let us define:

$$a^* : \otimes_n(\tilde{x}_i).P \stackrel{\text{def}}{=} a^* : (\tilde{w}).(\tilde{c})(\textstyle\prod_{i\leq n} w_i^* : (\tilde{z}_i).\overline{c_i}^* : \langle\tilde{z}_i\rangle \mid c_1^* : (\tilde{x}_1)...c_n^* : (\tilde{x}_n).P)$$
$$\overline{a^*} : \otimes_n\langle\tilde{v}_i\rangle.P \stackrel{\text{def}}{=} \overline{a}^* : (\tilde{e}).(\textstyle\prod_{i\leq n} \overline{e_i}^* : \langle\tilde{v}_i\rangle \mid Q)$$

where $\tilde{c} = c_1...c_n$ and $\tilde{e} = c_1...c_n$ are fresh and distinct and z_i and z'_i are all fresh with $n \leq 0$. Now we have:

$$a^* : \otimes_n(\tilde{x}_i).P \mid \overline{a^*} : \otimes_n\langle\tilde{v}_i\rangle.Q \longrightarrow\!\!\!\!\rightarrow P\{\tilde{v}_1/\tilde{x}_1\}..\{\tilde{v}_n/\tilde{x}_n\} \mid Q$$

with $|\tilde{x}_i| = |\tilde{v}_i|$, together with the safety property after the first step as in (i). Here $\overline{a^*} : \otimes_n\langle\tilde{v}_i\rangle.Q$ sends n sequences of data $v_{i1}..v_{ik_i}$ in parallel, which are received and instantiated individually by $a^* : \otimes_n(\tilde{x}_i).P$ but part of n sequences are merged to get instantiated in another agent. See the following remark.

2.5. **Remark.** (parallel name passing) In the third example, there is no order between c_i and c_k in the above encoding, so that we can safely permute the order of prefixes from $c_i^* : (\tilde{x}_i)..c_k^* : (\tilde{x}_k)..P$ to $c_k^* : (\tilde{x}_k)..c_i^* : (\tilde{x}_i)..P$ preserving the semantics, e.g. weak bisimilarity.

3. Graph Types

3.1. Given a finite poset $\langle X, \leq \rangle$, its *covering relation* $\to$ is defined: $x \to y$ iff (1) $x \lneq y$ and (2) for no z, $x \lneq z \lneq y$ where $x, y, z \in X$ (cf. [5], Chap.2). Then $\langle X, \to \rangle$ gives an acyclic directed simple graph (equivalent to the familiar Hasse diagram), from which $\leq$ is recovered as $\to^*$ (the reflexive transitive closure) and $\lneq$ as $\to^+$ (the transitive closure). In this paper the term "graph" always mean this kind of a directed graph, presented by the covering relation denoted $\to$. Note $x \to^* y$ says y is bigger. We write $G, G', \ldots$ for such graphs, $N(G)$ for the sets of nodes of G, and $G_1 \uplus G_2$ for the (disjoint) graph union of G_1 and G_2.

3.2. **Definition.** (graph types) A *graph type* is a graph in the above sense whose nodes are occurrences of the following *atomic types*[2]:

$$\mathbf{n} \quad ::= \quad a\!\downarrow\!(b) \mid a\!\uparrow\!(b) \mid a\!\downarrow\!\mathbf{x} \mid a\!\uparrow\!\mathbf{x}$$

where $a, b, c, \ldots$ range over $\mathcal{N}$ (cf. 2.1) and $\mathbf{x}, \mathbf{x}', \ldots$ range over the set $\mathcal{V}$ of *base types*. In $a \updownarrow (b)$ or $a \updownarrow \mathbf{x}$, a is the *subject* and $b, \mathbf{x}$ are *objects*. Over atomic types we define a self-inverse function $\overline{(\cdot\ )}$ such that: $\overline{a\downarrow(b)} = a\uparrow(b)$ and $\overline{a\downarrow\mathbf{x}} = a\uparrow\mathbf{x}$, which is extended to graph types so that $\overline{G}$ (the *dual* of G) is the result of replacing all occurrences of atomic types in G by their duals. *Heads* of a graph type are those occurrences of atomic types which are minimal w.r.t. the order $\to^*$. We often use atomic types to denote their occurrences in a graph type, writing, for example, $\mathbf{n}_1 \to \mathbf{n}_2$, if no confusion arises.

Intuitively, nodes in a graph type denote atomic actions of a process (which are best understood as (abstraction of) labels in transition relation), while edges denote their synchronisation ordering. If a graph type represents a structure of interaction from one party's viewpoint, then its dual represents the same interaction from another party's viewpoint.

3.3. **Binding.** In atomic types, the only bound occurrence of a name is b in $a \updownarrow (b)$, others occurring free. $\mathsf{bn}(\mathbf{n})$ and $\mathsf{fn}(\mathbf{n})$ are the sets of a bound/free name in $\mathbf{n}$, respectively. If $\mathbf{n}$ occurs in G and $\mathsf{bn}(\mathbf{n}) = \{b\}$, it is a *b-binder*, and its *scope* is the set of nodes strictly bigger than $\mathbf{n}$. Then a b-binder *binds* a free occurrence of b in $\mathbf{n}'$, denoted $\mathbf{n} \rightarrowtail\!\!\!\rightarrow_b \mathbf{n}'$ or simply $\mathbf{n} \rightarrowtail\!\!\!\rightarrow \mathbf{n}'$, if $\mathbf{n}'$ is in $\mathbf{n}$'s scope but is not in the scope of any b-binder strictly bigger than $\mathbf{n}$.

A graph type is *normal* if, in any of its bindings, no two binders bind the same name occurrence (see Example below). *Hereafter a "graph type" always means a normal graph type.* For (normal) graph types, the standard idea of the sets of free/bound names and the α-equality applies, which we denote $\mathsf{bn}(G)$, $\mathsf{fn}(G)$ and $\equiv_\alpha$, respectively.

[2]Equivalently nodes of a graph type are labeled by atomic types. Graph types are considered modulo the graph isomorphism preserving these labels and direction of edges.

W.l.o.g. we always assume that a graph type obeys the usual *binding condition*, i.e. no two binding occurrences are of the same name and $\mathsf{bn}(G)$ and $\mathsf{fn}(G)$ are disjoint.

3.4. **Examples.** Some simple graph types can be written syntactically, for example, $b\downarrow\mathbf{x} \rightarrow a\uparrow\mathbf{x}'$ and $b\uparrow(c) \rightarrow a\uparrow\mathbf{x}$. Further examples are given in Figure 1. Notice the bindings in (2) and (3) of Figure 1 are normal, while that of (1) is not (so (1) is not considered as a graph type). We can also see (2) and (3) α-convertible to each other and both obey the binding condition.

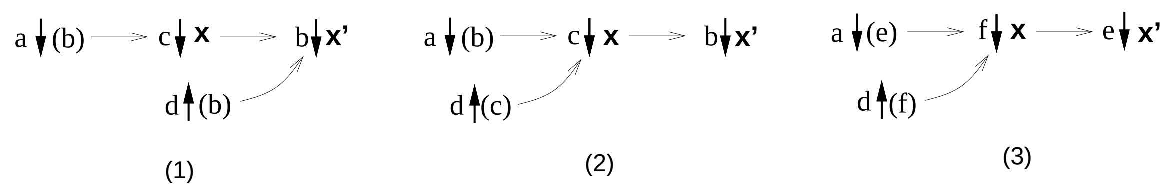


Figure 1. Graph Types

The following two kinds of graph types are important for our theory.

3.5. **Definition.**

(i) (safety) A graph type G is *safe*, if, for any $\mathbf{n}_1, \mathbf{n}_2 \in N(G)$, $\mathsf{fn}(\mathbf{n}_1) = \mathsf{fn}(\mathbf{n}_2)$ implies either $\mathbf{n}_1 \rightarrow^+ \mathbf{n}_2$ or $\mathbf{n}_2 \rightarrow^+ \mathbf{n}_1$.

(ii) (pointedness) A graph type G is *pointed*, if: (1) There is a unique head in G, whose name is free in G. (2) All name occurrences except the above are bound, and (3) G is safe. We write $G_{\langle a\rangle}$ if G is pointed with a free.

In a safe graph type, two nodes with the same subject are strictly ordered so that, while the same subject may be used several times in interaction, at any one time, only one input and one output are active, so that interaction becomes deterministic. In Figure 2, (1) is not safe because there is no ordering between $c\uparrow\mathbf{x}$ and $c\uparrow(e)$.

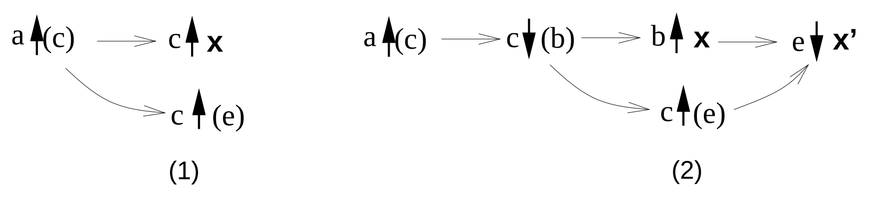


Figure 2. Unsafe and Safe Types

Safe graph types can be shared if they are pointed. The pointedness says that an interaction starts at a single interacting point, and that, after the initial interaction, the communications are always done using private channels, so that they are free from the interference of the third party. An example of a pointed graph type is shown in (2) of Figure 2.

3.6. **Proposition.**

(i) *If G is safe (resp. pointed) then $\overline{G}$ is safe (resp. pointed).*

(ii) *In a pointed graph type G, if $\mathbf{n} \in N(G)$ is not the head of G, there is a sequence $\mathbf{n}_0 \succ\!\!\succ \mathbf{n}_1 \succ\!\!\succ .. \succ\!\!\succ \mathbf{n}_m \stackrel{\text{def}}{=} \mathbf{n}$ where $\mathbf{n}_0$ is the head of G and the subject of $\mathbf{n}_{i+1}$ is bound by $\mathbf{n}_i$.*

P r o o f. Easy by induction on the number of nodes in G. □

3.7. **Cut Function.** We next discuss composition of graph types. Write $\mathsf{dom}(\Psi)$ and $\mathsf{ran}(\Psi)$ for the domain and the range of a partial function Ψ. Given G and G', assume $\mathbf{n}, \mathbf{n}_i \in N(G)$ and $\mathbf{n}', \mathbf{n}'_i \in N(G')$. Then a *cut function* Ψ from G to G' is a partial injection $\Psi : N(G) \rightharpoonup N(G')$ such that:

(i) (duality) If $\mathbf{n} \in \mathsf{dom}(\Psi)$, then $\Phi(\mathbf{n}) = \overline{\mathbf{n}}$.

(ii) (acyclicity) $\mathbf{n}_1 \to^+ \mathbf{n}_2$ and $\Psi(\mathbf{n}_2) \to^+ \Psi(\mathbf{n}_1)$ cannot hold simultaneously for any $\mathbf{n}_1, \mathbf{n}_2 \in \mathsf{dom}(\Psi)$.

(iii) (covering) (1) if $a \in \mathsf{fn}(G) \cap \mathsf{fn}(G')$, then $a \in \mathsf{fn}(\mathbf{n})$ implies $\mathbf{n} \in \mathsf{dom}(\Phi)$ (resp. $a \in \mathsf{fn}(\mathbf{n}')$ implies $\mathbf{n} \in \mathsf{ran}(\Phi)$). (2) if there is any binding chain $\mathbf{n}_1 \succ\!\!\succ \mathbf{n}_2 \succ\!\!\succ \cdots \succ\!\!\succ \mathbf{n}_n$ in G and $\mathbf{n}_i \in \mathsf{dom}(\Phi)$ for some $i \le n$, then for all $1 \le j \le n$, $\mathbf{n}_j \in \mathsf{dom}(\Phi)$, and then $\Phi(\mathbf{n}_1) \succ\!\!\succ \Phi(\mathbf{n}_2) \succ\!\!\succ \cdots \succ\!\!\succ \Phi(\mathbf{n}_n)$. Similarly for any binding chain in G' and Φ^{-1}.

Immediately we have:

3.8. **Proposition.** *Suppose G and G' are safe, then there can be at most one cut function from G to G'.*

P r o o f. For nodes with a common free subject the injection is determined uniquely by safety and acyclicity; then we use the covering to inductively apply the same argument. □

We write $G_1 \asymp G_2$ if, for some $G'_1 \equiv_\alpha G_1$ and $G'_2 \equiv_\alpha G_2$, there is a cut function from G'_1 to G'_2. For convenience we assume, when we write $G_1 \asymp G_2$, they are already suitably α-converted so that the cut function relates their nodes directly. We now introduce two operations on graph types.

3.9. **Operations on Graph Types.**

(i) (prefix) Given a safe graph type G, "$\mathbf{n} \to G$" denotes the result of adding a new node (occurrence) of $\mathbf{n}$ to G together with edges from the new node to all heads of G.

(ii) (parallel composition) Given safe graph types G_1 and G_2 s.t. $G_1 \asymp G_2$ together with the corresponding cut function Φ, we define the *composition* of G_1 and G_2, denoted by $G_1 \odot G_2$, as a graph type obtained by: first identifying Φ-related nodes of $G_1 \uplus G_2$, getting a graph, say, G', and then by eliminating every identified node, preserving and reflecting $\to^*$ of G', deleting all redundant edges if any.

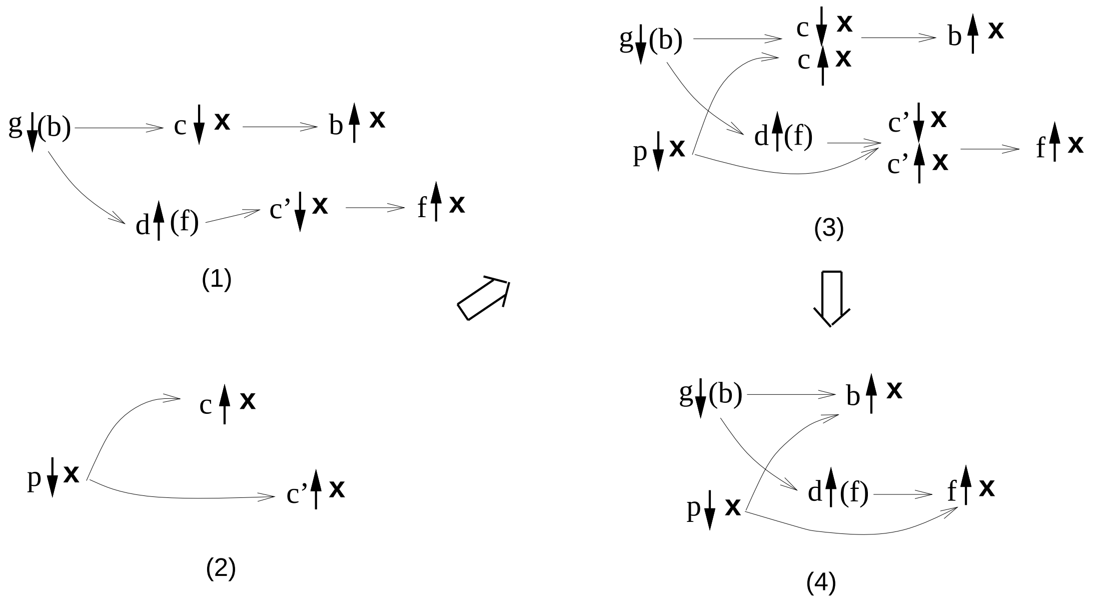


Figure 3. Cut Elimination

Figure 3 shows a simple example of parallel composition. Given two safe graphs in (1) and (2), first we identify the same nodes as in $\langle c\downarrow\mathbf{x},\ c\uparrow\mathbf{x}\rangle$ and $\langle c'\downarrow\mathbf{x},\ c'\uparrow\mathbf{x}\rangle$ in (3), next eliminate one by one (the order does not matter). Then we have the final graph (4). In this way the original dependency relations are merged to generate a more complex one.

A few basic properties of the operations follow.

3.10. **Proposition.**

(i) *If G is safe then (1) $\mathbf{n}_1 \mathrel{\succ\!\!\succ}_b \mathbf{n}_2$ in $\mathbf{n} \to G$ implies* either $\mathbf{n}_1 \mathrel{\succ\!\!\succ}_b \mathbf{n}_2$ *in G* or $\mathbf{n}_1 = \mathbf{n}$ *and b of $\mathbf{n}_2$ occurs free in G, and (2) $\mathbf{n} \to G$ is safe.*

(ii) *If G_1, G_2 are safe and $G_1 \asymp G_2$, then (1) for $\mathbf{n}, \mathbf{n'} \in N(G_1 \odot G_2)$, we have $\mathbf{n} \mathrel{\succ\!\!\succ} \mathbf{n'}$ in $G_1 \odot G_2$ iff $\mathbf{n} \mathrel{\succ\!\!\succ} \mathbf{n'}$ in G_i for $i = 1$ or $i = 2$, and (2) $G_1 \odot G_2$ is again safe. In particular always $G_1 \asymp \overline{G_1}$ and $G_1 \odot \overline{G_1}$ is the empty graph.*

(iii) *Any graph type can be constructed by the operations in 3.9, starting from atomic graph types, i.e. graph types with a unique node.*

Proof. (i) and (ii) are easy (notice the item (1) in each case implies that all bindings in the resulting graph are normal). (iii) uses the fact any connected graph type G consists of a head and subgraph G' together with edges from the head to some nodes of G', some to heads of G' and the rest to its non-heads, and argue by induction on the sum of the number of nodes and that of edges. If G with the sum $n+1$ has a unique head the result is immediate (use (prefix) in 3.9). If not, we take off a head and extend G' with additional fresh action types (a) on the top of each connected head and (b) in the middle of an edge going to each connected non-head. The sum becomes n so the induction applies. Now take the disjoint union of the duals of the newly introduced action types and prefix it with the eliminated node. When two new graphs are composed (which is well-defined) we have the original G, as required. $\square$

An important property of parallel composition follows. For the second clause, notice $(G_1 \odot G_2) \odot G_3 \equiv_\alpha G_1 \odot (G_2 \odot G_3)$ does not always hold.

3.11. **Proposition.** *Suppose G_i is safe. Then:*

(i) $G_1 \asymp G_2$ *implies* $G_1 \odot G_2 \equiv_\alpha G_2 \odot G_1$.

(ii) *Suppose* $\mathsf{fn}(G_1) \cap \mathsf{fn}(G_2) \cap \mathsf{fn}(G_3) = \emptyset$. *If* $((G_1 \odot G_2) \odot G_3)$ *is defined, then* $((G_1 \odot G_2) \odot G_3) \equiv_\alpha (G_1 \odot (G_2 \odot G_3))$.

Proof. (i) is easy. For (ii), we first note: if $((G_1 \odot G_2) \odot G_3)$ is defined, then $G_1 \asymp G_2$ and $(G_1 \odot G_2) \asymp G_3$. By assumption, $a \in \mathsf{fn}(G_1 \odot G_2) \cap \mathsf{fn}(G_3)$ implies either $a \in \mathsf{fn}(G_1)$ or $a \in \mathsf{fn}(G_2)$, hence $G_1 \asymp G_3$ and $G_2 \asymp G_3$, and then $G_1 \asymp (G_2 \odot G_3)$ by Proposition 3.10 (ii-1). Hence $((G_1 \odot G_2) \odot G_3) \equiv_\alpha (G_1 \odot (G_2 \odot G_3))$ by Proposition 3.8. $\square$

The final proposition is essential to prove the typing system introduced in the next section is closed under the structural rule.

4. The Typing System and Its Basic Properties

4.1. **General Idea.** This section introduces a typing system based on graph-types for monadic π-terms and establishes its basic syntactic properties, including the subject reduction property. For simplicity of presentation we do not treat recursive types, whose incorporation is straightforward, cf. 8.2. The typing judgment for a π-term P has a form:

$$\Gamma \vdash P \triangleright_A G$$

where Γ is an *environment* assigning protocol types (which we introduce below) to some of free names in P, P is a π-term, A is a *hidden name set*, and G is a graph type. This judgment tells us: "P has a linear behaviour specified in G, and otherwise obeys the protocols specified in Γ." The separation between linear behaviour and "shared" interaction is essential for tractability of type inference. A is an auxiliary name set to record the linear usage of names, cf.[22]. The sequent is derived roughly in the following three stages:

(1) Starting from a subterm, we construct a graph type G, which should obey a safe condition.

(2) When a term is given as a parallel composition of two subterms, we check whether two corresponding graph types, say G_1 and G_2, are compatible or not, i.e. $G_1 \asymp G_2$ or not, and if so we merge them to get $G_1 \odot G_2$, inducing a more complex structure in general. We record the set of names lost in cut elimination in the hidden name set.

(3) If a subgraph of G is a pointed graph, we may move it from the “linear space” to the “classical space” (shared environments), i.e. from G to Γ. This allows an interaction point to be shared by multiple subterms while ensuring any interaction starting from that point to obey the fixed deterministic protocols. In this way, types which are moved to the non-linear space give *generalisation of sorting.*

The typing system extracts certain significant behavioural properties of a π-term as its type. The relationship between assigned types and terms behavioural properties will be clarified in the next section. We start from introducing the idea of *protocol types*, which are essentially pairs of pointed types, and which generalise the notion of sorting.

4.2. **Protocol Types.** Write “$\lambda a.G_{\langle a\rangle}$” for the *name abstraction* of a in a pointed graph $G_{\langle a\rangle}$, with the usual notion of binding. (We only need this restricted abstraction in this paper, cf. 6.2). Then a *protocol type* of $\lambda a.G_{\langle a\rangle}$ is the set $\{\lambda a.G_{\langle a\rangle},\ \lambda a.\overline{G}_{\langle a\rangle}\}$.

Using this notion, we fix the set of types we shall use in our typing system, given any base $\mathcal{V}'$, we get the set of atomic types, from which we get the set of graph types, and then that of protocol types. The corresponding set functions are written $\psi_A(\mathcal{V}')$, $\psi_G(\mathcal{V}')$ and $\psi_P(\mathcal{V}')$, respectively. Then we define, given an initial base $\mathcal{V}$ (say $\{\mathsf{nat}\}$), $\mathbf{A}_{\mathcal{V}}$, $\mathbf{G}_{\mathcal{V}}$ and $\mathbf{P}_{\mathcal{V}}$ as the smallest sets closed under

$$\mathbf{P}_{\mathcal{V}} \ \supseteq\ \mathcal{V} \cup \psi_P(\mathbf{P}_{\mathcal{V}}), \quad \mathbf{A}_{\mathcal{V}} \ \supseteq\ \psi_A(\mathbf{P}_{\mathcal{V}}) \quad \text{and} \quad \mathbf{G}_{\mathcal{V}} \ \supseteq\ \psi_G(\mathbf{P}_{\mathcal{V}}).$$

$\alpha, \beta, \ldots$ range over $\mathbf{P}_{\mathcal{V}}$. We shall use these sets for our typing system. We also use the following notations.

- “$\updownarrow(x) \to G$” denotes abstraction $\lambda a.(a \updownarrow(x) \to G)$.
- For $\alpha \in \mathbf{P}_{\mathcal{V}}$, we write (α) for $\{\lambda a.a\downarrow\alpha,\ \lambda a.a\uparrow\alpha\}$.

A protocol type (similar to a pointed graph (2) in Figure 2) is given in Figure 4.

$$\left\{ \begin{tikzcd} \uparrow(\mathsf{c}) \arrow[r] & \mathsf{c}\downarrow(\mathsf{b}) \arrow[r] \arrow[dr] & \mathsf{b}\uparrow\mathbf{nat} \arrow[r] & \mathsf{e}\downarrow(\mathbf{nat}) \\ & & \mathsf{c}\uparrow(\mathsf{e}) \arrow[ur] & \end{tikzcd} , \begin{tikzcd} \downarrow(\mathsf{c}) \arrow[r] & \mathsf{c}\uparrow(\mathsf{b}) \arrow[r] \arrow[dr] & \mathsf{b}\downarrow\mathbf{nat} \arrow[r] & \mathsf{e}\uparrow(\mathbf{nat}) \\ & & \mathsf{c}\downarrow(\mathsf{e}) \arrow[ur] & \end{tikzcd} \right\}$$

Figure 4. Protocol Type

4.3. **Remark.** Note the expression corresponding to the monadic sorting, (α), is the simplest kind of protocol types. The notation is intentional (cf. [24, 14]), since it does function as such. This also shows how we generalise the notion of sorting. As in the sorting, a name can carry another type, admitting the nested structures. However carried information can be a complex protocol, by which we can represent (and guarantee) more refined communication behaviour of mobile processes.

(I) Prefix Rules.

$$(\mathrm{in}_1)\quad \frac{\Gamma \vdash P \triangleright_A G}{\Gamma \vdash ax.P \triangleright_A a{\downarrow}(x) \to G}\ (1,2) \qquad (\mathrm{out}_1)\quad \frac{\Gamma \vdash P \triangleright_A G}{\Gamma \vdash \overline{a}x.P \triangleright_{A\cup\{x\}} a{\uparrow}(x) \to G}\ (1,2)$$

$$(\mathrm{in}_2)\quad \frac{x\colon \alpha, \Gamma \vdash P \triangleright_A G}{\Gamma \vdash ax.P \triangleright_A a{\downarrow}\alpha \to G}\ (1) \qquad (\mathrm{out}_2)\quad \frac{x\colon \alpha, \Gamma \vdash P \triangleright_A G}{x\colon \alpha, \Gamma \vdash \overline{a}x.P \triangleright_A a{\uparrow}\alpha \to G}\ (1)$$

$$(\mathrm{in}_3)\quad \frac{a\colon \beta, \Gamma \vdash P \triangleright_A G}{a\colon \beta, \Gamma \vdash ax.P \triangleright_A}\ (2,3) \qquad (\mathrm{out}_3)\quad \frac{a\colon \beta,\, \Gamma \vdash P \triangleright_A \overline{G}}{a\colon \beta,\, \Gamma \vdash \overline{a}x.P \triangleright_{A\cup\{x\}}}\ (2,3)$$

$$(\mathrm{in}_4)\quad \frac{a\colon (\alpha),\, x\colon \alpha,\, \Gamma \vdash P \triangleright_A}{a\colon (\alpha),\, \Gamma \vdash ax.P \triangleright_A} \qquad (\mathrm{out}_4)\quad \frac{a\colon (\alpha),\, x\colon \alpha, \Gamma \vdash P \triangleright_A}{a\colon (\alpha),\, x\colon \alpha, \Gamma \vdash \overline{a}x.P \triangleright_A}$$

(II) Other Rules.

$$(\mathrm{par})\quad \frac{\Gamma \vdash P \triangleright_{A_1} G_1 \quad \Gamma \vdash Q \triangleright_{A_2} G_2}{\Gamma \vdash P \mid Q \triangleright_{A_1\cup A_2\cup A} G_1 \odot G_2}\ (4) \qquad (\mathrm{res})\quad \frac{\Gamma \vdash P \triangleright_A G}{\Gamma\backslash a \vdash (a)P \triangleright_{A\backslash\{a\}} G}\ (5)$$

$$(\mathrm{rep})\quad \frac{\Gamma \vdash P}{\Gamma \vdash !P} \qquad (\mathrm{weak})\quad \frac{\Gamma \vdash P \triangleright_A G}{a\colon \alpha,\, \Gamma \vdash P \triangleright_{A\cup\{b\}} G}\ (6)$$

$$(\mathrm{nil})\quad \vdash \mathbf{0} \qquad (\mathrm{alpha})\quad \frac{\Gamma \vdash P \triangleright_A G \quad G \equiv_\alpha G' \quad \Gamma \equiv_\alpha \Gamma'}{\Gamma' \vdash P \triangleright_A G'}$$

(1) $a \notin \mathsf{fn}(\Gamma)\cup A$. (2) $x \notin \mathsf{fn}(\Gamma)\cup A$. (3) $\beta = \{{\downarrow}(x) \to G, {\uparrow}(x) \to \overline{G}\}$. (4) $\langle G_1, A_1\rangle \asymp \langle G_2, A_2\rangle$, $A = \mathsf{fn}(G_1) \cap \mathsf{fn}(G_2)$. (5) $a \notin \mathsf{fn}(G)$. (6) $a \notin A \cup \mathsf{fn}(G)$, $b \notin \mathsf{fn}(\Gamma) \cup \mathsf{fn}(G)$.

Figure 5. Typing System π_{G}

4.4. **Typing Function.** A *typing function* is a function from a finite subset of $\mathcal{N}$ to a set of protocol types. The set of typing functions are ranged over by $\Gamma, \Delta, \ldots$. We often regard Γ as a finite set of elements of form: $a\colon \alpha$. $\mathsf{fn}(\Gamma)$ denotes $\{a \mid a\colon \alpha \in \Gamma\}$. Let $A, B, C, \ldots$ denote a finite subset of $\mathcal{N}$. "$\Gamma\backslash A$" denotes $\{a\colon \alpha \in \Gamma \mid a \notin A\}$, while "$\Gamma\lceil A$" denotes $\{a\colon \alpha \in \Gamma \mid a \in A\}$. "$a : \alpha, \Gamma$" means $\Gamma \cup \{a\colon \alpha\}$ together with the assumption $a \notin \mathsf{fn}(\Gamma)$. "$\Gamma \asymp \Delta$" represents, if $x : \alpha \in \Gamma$ and $x : \beta \in \Delta$, we have $\alpha = \beta$. If we get Γ by α-conversion of some protocol graph in Δ, then we write $\Gamma \equiv_\alpha \Delta$.

4.5. **Sequent and Typing System.** The sequent has a form $\Gamma \vdash P \triangleright_A G$, which we read: "$P$ has a type G under Γ with hidden names A". Γ is called the *basis* of the sequent, P is its *subject*, A is the *hidden name set*, and G is its *type*. We write $\Gamma \vdash P \triangleright G$ if $A = \emptyset$ (read "P has a type G under Γ"), $\Gamma \vdash P \triangleright_A$ if $G = \emptyset$, and $\Gamma \vdash P$ if both are empty. The typing system, denoted π_{G}, is given in **Figure 5**, where "$\langle G_1, A_1\rangle \asymp \langle G_2, A_2\rangle$" denotes both $G_1 \asymp G_2$ and $(\mathsf{fn}(G_i) \cup A_i) \cap A_j = \emptyset$ for $i \neq j$. We hereafter write $\Gamma \vdash P \triangleright_A G$ if the sequent is derivable in π_{G}.

4.6. **Comments on Typing Rules (1): Prefix Rules.** There are four pairs of (in) and (out) rules, corresponding to the kind of types to be introduced.

(in$_1$, out$_1$): In (in$_1$), "$a\downarrow(x) \to G$" represents a process gets a new name x and acts with behaviour G (corresponding to $ax.P \xrightarrow{a(x)} P$ in the late transition [28]). Similarly (out$_1$), represents a "bound output" $(x)\overline{a}x.P \xrightarrow{\overline{a}(x)} P$. In the latter x is recorded as a hidden name because it should be later restricted by (res) rule.

(in$_2$, out$_2$): Each rule represents that name a gets/emits a name with a protocol type α, and then acts like G.

(in$_3$, out$_3$): These show that if we have successfully construct protocol type β by finally putting $a\updownarrow(x)$ to G, then we may remove the whole of the pointed graph type to the shared environment. In (out$_3$), we again memorize "x" as a hidden name.

(in$_4$, out$_4$): These rules correspond to monadic sorting.

4.7. **Remark.**

(i) Using (in$_2$) and (out$_2$) instead of (in$_4$) and (out$_4$), we can keep $a\updownarrow(x) \to G$ in the linear realm to construct more complex protocol type. This also shows the usual notion of principal types does not exist in π_{G}.

(ii) By (in$_{3,4}$) and (out$_{3,4}$), we make it possible to infer types of a term which contains a shared name at which multiple agents with consistent protocols may interact.

4.8. **Comments on Typing Rules (2): Other Rules.**

(par): This is the key rule which allows us to extract versatile forms of interaction structures from π-terms. Notice that, even if G_1 and G_2 are sequential (i.e. $\to^*$ is a total order), we may get a non-sequential graph (but still representing a linear behaviour, see 5.12/5.14). Cuts and their elimination also play the central role in our main results of this paper, Theorems 4.13 and 7.16 (cf. Remark 4.10). Notice also we memorize the set of names lost by cut-elimination: this prevents further connection to these names so that reduction at these points is deterministic.

(res): The combination of this rule and other rules that manipulate the hidden name set such as (out$_{1,3}$) and (par) enables us to record the linear usage of names during the type inference before the actual occurrence of the scope restriction.

(rep, nil, weak, alpha): These are standard. By (alpha), we hereafter safely consider graph types and basis modulo their respective α-equality.

4.9. **Example.** Some examples of terms and their graph types follow. Note that all the final graph types in the following are a-pointed, so that we could have moved them using (in$_3$) to the l.h.s. as protocol types.

(i) (polyadic name passing) Define $P \overset{\mathrm{def}}{=} a^*(x_1x_2).\overline{x_2}x_1$. Then

$$\vdash P \rhd a\downarrow(z) \to z\downarrow\mathsf{nat} \to z\downarrow(x_2) \to x_2\uparrow\mathsf{nat}$$

as well as

$$\vdash P \rhd a\downarrow(z) \to z\downarrow\mathsf{nat} \to z\downarrow(\mathsf{nat})$$

(note $(\mathsf{nat}) \overset{\mathrm{def}}{=} \{\downarrow\mathsf{nat}, \uparrow\mathsf{nat}\}$ in 4.2).

(ii) (input/output structure) Assume given $P \overset{\text{def}}{=} a^*(x)[w].w^*[vv]$. Then

$$v\colon \mathsf{nat}, w\colon \beta \vdash P \triangleright a\!:\!\downarrow\!(z) \to z\downarrow\!\mathsf{nat} \to z\uparrow\!\beta$$

as well as

$$v\colon \mathsf{nat} \vdash P \triangleright a\downarrow\!(z) \to z\downarrow\!\mathsf{nat} \to z\uparrow\! w \to w\downarrow\!(x) \to x\uparrow\!\mathsf{nat} \to x\uparrow\!\mathsf{nat}$$

where $\beta \overset{\text{def}}{=} \{\downarrow\!(x) \to x\downarrow\!\mathsf{nat} \to x\downarrow\!\mathsf{nat},\ \uparrow\!(x) \to x\uparrow\!\mathsf{nat} \to x\uparrow\!\mathsf{nat}\}$. Note input and output types indeed appear in turn.

(iii) (protocol with parallelism) Let $P_1 \overset{\text{def}}{=} z_1y.cx.\overline{z_1}c$, $P_2 \overset{\text{def}}{=} z_2x.\overline{c}w$ and $P \overset{\text{def}}{=} az_1.z_1z_2.(c)\,(P_1 \mid P_2)$. Then we have:

$$w\colon \mathsf{nat} \ \vdash\ P_1\ \triangleright\ z_1\downarrow\!\mathsf{nat} \to c\downarrow\!\mathsf{nat} \to z_1\uparrow\!(\mathsf{nat}) \quad \text{and} \quad w\colon \mathsf{nat}\ \vdash\ P_2\ \triangleright\ z_2\downarrow\!\mathsf{nat} \to c\uparrow\!\mathsf{nat}$$

Note z_1 carries two different types. By (par) rule a proper graph structure arises.

$$\begin{tikzcd} w : \mathsf{nat}\ \vdash\ (P_1 \mid P_2)\ \triangleright_{\{c\}}\ z_1\downarrow\!\mathsf{nat} \arrow[r] & z_1\uparrow\!(\mathsf{nat}). \\ z_2\downarrow\!\mathsf{nat} \arrow[ur] & \end{tikzcd}$$

After restricting and prefixing, we have the final state.

$$\begin{tikzcd} v : \mathsf{nat}, w : \mathsf{nat}\ \vdash\ P\ \triangleright\ a\downarrow\!(z_1) \arrow[r] & z_1\downarrow\!(z_2) \arrow[r] \arrow[dr] & z_1\downarrow\!\mathsf{nat} \arrow[r] & z_1\uparrow\!(\mathsf{nat}). \\ & & z_2\downarrow\!\mathsf{nat} \arrow[ur] & \end{tikzcd}$$

4.10. **Remark.** Assume we use (par0) rule below instead of (par) in Figure 5:

$$\text{(par0)}\quad \frac{\Gamma \vdash P \triangleright_{A_1} \quad \Gamma \vdash Q \triangleright_{A_2}}{\Gamma \vdash P \mid Q \triangleright_{A_1 \cup A_2}} \qquad (A_1 \cap A_2 = \emptyset)$$

Let us consider following terms

$$P \equiv \overline{c}v_1.\overline{c}v_2 \quad Q \equiv xy_1.xy_2 \quad \overline{a}(c).P \mid ax.Q \longrightarrow (c)(P \mid Q\{c/x\})$$

where v_1 has type nat and v_2 has a different type (nat). With (par0) rule, $(\overline{a}(c).P|ax.Q)$ is typable, while $(c)(P \mid Q\{c/x\})$ is not typable, even with (weak) rule, which means the subject reduction doesn't hold. Thus the incorporation of cuts plays a crucial role in the consistency of the present type system. We also note graph structures naturally arise, once some form of cuts is introduced, since without graphs we should artificially restrict the form of composition of types.

One basic result concerning the derivable sequent follows. The second half is notable in that it shows all graph types can be realisable by π-terms.

4.11. **Proposition.** *If $\Gamma \vdash P \triangleright_A G$, then G is safe graph type. Conversely, for any safe graph type G, there is a term P such that, for some Γ, we have $\Gamma \vdash P \triangleright G$.*

Proof. By Proposition 3.10 (the first half uses the clauses (i)(ii) while the second half uses the third clause). $\square$

We list several basic syntactic properties of π_G.

4.12. **Lemma.**

(i) (properties of names) *Suppose* $\Gamma \vdash P \triangleright_A G$. *Then:* $\mathsf{fn}(\Gamma) \cap \mathsf{fn}(G) = A \cap \mathsf{fn}(G) = A \cap \mathsf{fn}(\Gamma) = \emptyset$, $\mathsf{fn}(P) \subset \mathsf{fn}(G) \cup A \cup \mathsf{fn}(\Gamma)$, $\mathsf{fn}(G) \subset \mathsf{fn}(P)$, and $\Gamma \lceil \mathsf{fn}(P) \vdash P \triangleright_{A \cap \mathsf{fn}(P)} G$.

(ii) (renaming) *Suppose* $\Gamma \vdash P \triangleright_A G$ *and* $b \notin \mathsf{fn}(\Gamma) \cup A \cup \mathsf{fn}(G)$. *Then we have* $\Gamma\{b/a\} \vdash P\{b/a\} \triangleright_{A\{b/a\}} G\{b/a\}$.

(iii) (subterm property) *Suppose* $\Gamma \vdash P \triangleright_A G$. *Then any term occurring as the subject of some sequent in the proof is a subterm of* P *and, conversely, any subterm of* P *occurs as the subject of some sequent in the proof.*

(iv) ($\equiv$) *If* $\Gamma \vdash P \triangleright_A G$ *and* $P \equiv Q$, *then* $\Gamma \vdash Q \triangleright_A G$.

(v) (combination) *Suppose there is the following derivation. Then* $i = j$.

$$\dfrac{\dfrac{\Gamma_1 \vdash P \triangleright_{A_1} G_1'}{\Gamma \vdash ax.P \triangleright_{A_1} G_1}\,(\mathrm{in}_i) \qquad \dfrac{\Gamma \vdash Q \triangleright_{A_2'} G_2'}{\Gamma \vdash \overline{a}v.Q \triangleright_{A_2} G_2}\,(\mathrm{out}_j)}{\Gamma \vdash ax.P \mid \overline{a}v.Q \triangleright_{A_1 \cup \mathcal{A}_2 \cup A'} G_1 \odot G_2}\,(\mathrm{par})$$

(vi) (substitution) *If* $a\colon \beta, b\colon \beta, \Gamma \vdash P \triangleright_A G$, *then* $b\colon \beta, \Gamma \vdash P\{b/a\} \triangleright_A G$.

Proof. (i) is easy by checking each rule, staring from (nil). (ii) uses (i), verifying each rule considering in which of Γ, A, G a occurs. (iii) is immediate from the form of each rule. In (iv), we mention three cases. For the closure under α-convertibility we use (ii) above. The commutativity follows from Proposition 3.11 (i). $\Gamma \vdash (P_1 \mid P_2) \mid P_3 \triangleright_A G \;\Leftrightarrow\; \Gamma \vdash P_1 \mid (P_2 \mid P_3) \triangleright_A G$ is most non-trivial. But we can easily check if which one of above sequent is derived from $\Gamma_i \vdash P_i \triangleright_{A_i} G_i$, then $\mathsf{fn}(G_1) \cap \mathsf{fn}(G_2) \cap \mathsf{fn}(G_3) = \emptyset$ (because we record the lost names). Then we can use Proposition 3.11 (ii). (v) is easy by checking the forms of graph type or protocol type in each rule. (vi) is by induction on the structure of P using the above results. Appendix A shows the details of (iv)(v) and (vi). $\square$

Notice, by (i,ii) we safely assume binding occurrences in types and a term of $\Gamma \vdash P \triangleright_A G$ are all distinct and disjoint from free occurrences.

The main result about the typing system follows, establishing the subject reduction property. It is notable that not only the environment Γ (which corresponds to sorting) but also the graph type of a term is invariant under reduction. This is because all possible reduction concerning the (names of the) graph type is already consumed when (par) rule is applied: and reduction concerning the basis only creates dual graphs which eliminate each other to leave the graph type unchanged.

4.13. **Theorem.** (Subject Reduction).

If $\Gamma \vdash P \triangleright_A G$ *and* $P \longrightarrow\!\!\!\!\!\rightarrow P'$, *then we have* $\Gamma \vdash P' \triangleright_A G$.

Proof. By induction on the derivation of the one-step reduction. By Lemma 4.12 (iii,iv), we can show that it is enough to prove the following property.

$$\Gamma \vdash ax.P \mid \overline{a}v.Q \triangleright_A G \;\Rightarrow\; \Gamma \vdash P\{v/x\} \mid Q \triangleright_A G$$

Without loss of generality, we assume $ax.P \mid \overline{a}v.Q$ obeys the binding condition. Then we can show its type is derived directly from the following antecedents:

$$(\mathrm{in}_i)\ \frac{\Gamma_1 \vdash P \triangleright_{A_1} G_1'}{\Gamma \vdash ax.P \triangleright_{A_1} G_1} \quad (\mathrm{out}_j)\ \frac{\Gamma \vdash Q \triangleright_{A_2'} G_2'}{\Gamma \vdash \overline{a}v.Q \triangleright_{A_2} G_2} \tag{4.1}$$

The existence of multiple (in)(out) rules seems to make the reasoning complex, but by Combination Lemma (Lemma 4.12 (v)), we know $i = j$ in (4.1). Thus the case analysis becomes simpler and leads to a tractable proof. In the cases of $(\mathrm{in}_{1,2})$ and $(\mathrm{out}_{1,2})$, note G_1' and G_2' are the results of eliminating unique and mutually dual head nodes from G_1 and G_2, respectively. Since $(G_1 \odot G_2) \equiv_\alpha ((G_1 \backslash \mathbf{n}) \odot (G_2 \backslash \overline{\mathbf{n}}))$ we obtain the result. The case of (in_3) and (out_3) is similar, considering the cut elimination corresponding protocol types. In (in_4) and (out_4), we use Substitution Lemma (Lemma 4.12 (vi)). The full proof of the theorem is given in Appendix B. □

The subject reduction property immediately implies there is no type error if we assume communication of constant values. Also, in such a setting, the set of typable terms is extended non-trivially even under the recursive sorting (with the corresponding extension of protocol types, cf. Section 8), as can be seen from e.g. Example 4.9 (ii).

5. Behavioural Equality over Typed π-terms

This section introduces the basic constructions of typed behavioural equality over monadic π-terms following the untyped construction in [17, 20]. A derived equality is then used to clarify the behavioural properties of a typed term ensured by the assigned type.

5.1. **Typed Terms and Typed Relation.**

A typed term is a tuple $\langle \Gamma,\ P,\ A,\ G\rangle$ such that $\Gamma \vdash P \triangleright_A G$. We simply write $\Gamma \vdash P \triangleright_A G$ for a typed term $\langle \Gamma,\ P,\ A,\ G\rangle$. A relation over typed terms, ranged over $\mathcal{R}, \mathcal{R}'\ldots$, is called a *typed relation*, if:

(a) $\Gamma \vdash P \triangleright_A G \;\mathcal{R}\; \Delta \vdash Q \triangleright_B G' \;\Rightarrow\; A = B,\ G = G'$, and $\Gamma = \Delta$, and

(b) $\Gamma \vdash P \triangleright_A G \;\mathcal{R}\; \Gamma \vdash Q \triangleright_A G \;\Rightarrow\; (a\colon \alpha, \Gamma \vdash P \triangleright_{A \cup \{b\}} G) \;\mathcal{R}\; (a\colon \alpha, \Gamma \vdash P \triangleright_{A \cup \{b\}} G)$ with (6) in Fig.5.

We often write $\Gamma \vdash P \;\mathcal{R}\; Q \triangleright_A G$ for a tuple of a typed relation, or just $P \;\mathcal{R}\; Q$ if the type assignment is obvious from the context.

Notice $\equiv$ and $\longrightarrow$ restricted to typed terms are typed relations (by Lemma 4.12 (iv) and Theorem 4.13 respectively). Several important classes of typed relations are:

(i) A typed relation is *reflexive* if it contains the structural equality over typed π-terms. The *transitive/symmetric relations* are defined in a similar way. If a typed relation is reflexive, transitive and symmetric, then it is an *equivalence*.

(ii) The relation $\mathcal{R}$ is *substitution closed* if it is closed under the following rules.
  (1) $\Gamma \vdash P \;\mathcal{R}\; Q \triangleright_A G$ and $b \notin \mathsf{fn}(\Gamma) \cup A \cup \mathsf{fn}(G)$, then $\Gamma\{b/a\} \vdash P\{b/a\} \;\mathcal{R}\; Q\{b/a\} \triangleright_{A\{b/a\}} G\{b/a\}$.
  (2) $a\colon \beta, b\colon \beta, \Gamma \vdash P \;\mathcal{R}\; Q \triangleright_A G$, then $b\colon \beta, \Gamma \vdash P\{b/a\} \;\mathcal{R}\; Q\{b/a\} \triangleright_A G$.

(iii) A typed relation $\mathcal{R}$ is a *congruence* if $\mathcal{R}$ is an equivalence and moreover is closed under the following rule, for each inference rule (and in (par) for each selection of one side of the antecedents) in Figure 5: $\Gamma \vdash P_1 \; \mathcal{R} \; P_2 \triangleright_A G \;\Rightarrow\; \Gamma' \vdash C[P_1] \; \mathcal{R} \; C[P_2] \triangleright_{A'} G'$ if, in an instance of the inference rule, $\Gamma' \vdash C[P_i] \triangleright_{A'} G'$ is the conclusion with an antecedent $\Gamma \vdash P_i \triangleright_A G$, for $i = 1, 2$ respectively. We write $\cong, \cong', ...$ for congruence relations.

Hereafter binary relations over typed terms are always considered to be typed, unless otherwise stated.

5.2. **Definition.** (sound congruence)

(1) A congruence $\cong$ over typed π-terms is *reduction-closed* (r.c) iff, whenever $\Gamma \vdash P \cong Q \triangleright_A G$, $P \longrightarrow\!\!\!\!\rightarrow P'$ implies, for some Q', $Q \longrightarrow\!\!\!\!\rightarrow Q'$ and $\Gamma \vdash P' \cong Q' \triangleright_A G$.

(2) With θ ranging over $+$ and $-$, we define a family of *action predicates*: $P \Downarrow_{a^\theta} \overset{\text{def}}{\Leftrightarrow} \exists P'.\, P \longrightarrow\!\!\!\!\rightarrow P' \;\wedge\; a \in \mathcal{AN}_\theta(P')$. Then $\Gamma \vdash P \Downarrow_{a^\theta} \; \triangleright_A G \overset{\text{def}}{\Leftrightarrow} (P \Downarrow_{a^\theta} \;\wedge\; a \notin A)$. A typed relation $\mathcal{R}$ *respects action predicates* iff $\Gamma \vdash P \mathcal{R} Q \triangleright_A G$ and $\Gamma \vdash P \Downarrow_{a^\theta} \triangleright_A G$ implies $\Gamma \vdash Q \Downarrow_{a^\theta} \; \triangleright_A G$.

A congruence $\cong$ is $\Downarrow_{a^\theta}$*-sound*, or simply *sound*, if it is reduction closed and respects action predicates.

Note that a sound congruence is automatically non-trivial (i.e. neither universal nor empty). Moreover we can easily verify that the congruence closure of a family of sound congruences is again sound. Then, by taking the congruence closure of the whole family of sound congruences, we immediately know:

5.3. **Proposition.** *There is the maximum sound congruence within the family of all sound congruences. The sound maximum equality is denoted* $=_\pi$.

In the following we use $=_\pi$ as the representative notion of equality over typed π-terms.

5.4. **Remark.** (semantics based on reduction relation)
Semantics based on reduction relation are now used in many studies both in untyped setting [17, 20, 18, 19, 29, 40, 33] and the typed setting [36, 22]. The above definition is based on the semantic framework developed in [17, 18, 19, 12, 46, 20]. There are two notable points. One is the use of reduction closure. A contrasting method is *barbed congruence* [39]. We notice that if we combine the usual notion of equality in algebra with reduction relation, reduction-closure would be more natural since it guarantees that a process in some congruent class moves to the same congruent class again.

The second point is the choice of action predicates. We discuss this point below.

5.5. **Definition.** We define two kinds of weaker action predicates over untyped π-terms w.r.t [29] and [39], respectively.

$$P \Downarrow_a \;\overset{\text{def}}{\Leftrightarrow}\; P \Downarrow_{a^+} \text{ or } P \Downarrow_{a^-}. \quad \text{and} \quad P \Downarrow \;\overset{\text{def}}{\Leftrightarrow}\; \exists a.\, P \Downarrow_a$$

We then say $\Gamma \vdash P \triangleright_A G$ is *insensitive* if $\neg\Gamma \vdash P \Downarrow\ \triangleright_A G$ (cf. Definition 3.11 and 4.12 in [20]).

5.6. **Proposition.** *Let $\cong$ be a non-trivial r.c. congruence in which we have: $\Gamma \vdash P_i \triangleright_A G$ is insensitive $(i = 1, 2)$ $\quad\Rightarrow\quad$ $\Gamma \vdash P_1 \cong P_2 \triangleright_A G$. Then $\cong$ is $\Downarrow_{a^\theta}$-sound.*

Proof. By the similar reasoning as in Proposition 4.13 of [20], using the type-correct contexts. □

The following result shows we can use any of these criteria to get the unique behavioural equality (as far as I know, it has not been proved yet whether the result similar to the following proposition holds or not in weak barbed congruence, cf. p.49 in [39]).

5.7. **Proposition.** *Let us define $=_1$ and $=_2$ are maximum sound congruences based on $\Downarrow$ and $\Downarrow_a$, respectively. Let $=_3$ be any maximal and non-trivial r.c.congruence extending $=_2$. Then $=_3$ is uniquely determined and $=_\pi\ =\ =_1\ =\ =_2\ =\ =_3$.*

Proof. Note that $=_\pi\ \subseteq\ =_1\ \subseteq\ =_2\ \subseteq\ =_3\ \subseteq\ \cong$, hence the result by the above proposition. □

This may suggest we may use $=_\pi$ as the basic (weak) behavioural equality over typed π-terms. The proposition also implies that we can use any of these predicates which is most convenient to a given situation, in order to check if two terms are equated or not in $=_\pi$ (we however believe that the same full abstraction result can be gained with the barbed congruence).

We next clarify basic behavioural properties types ensure for π-terms. We start from the reduction corresponding to cut elimination.

5.8. **Definition.** (β-reduction) The one-step β-reduction, $\rightarrow_\beta$, is the smallest relation defined by

$$
\begin{array}{lll}
(\beta_1) & \Gamma \vdash\ ax.P \mid \overline{a}v.Q \rightarrow_\beta\ P\{v/x\} \mid Q\ \triangleright_A G & (a \in A) \\
(\beta_2) & \Gamma \vdash\ (a)(ax.P \mid \overline{a}v.Q) \rightarrow_\beta\ (a)(P\{v/x\} \mid Q)\ \triangleright_A G & \\
(\mathrm{Par}) & \Gamma \vdash P \rightarrow_\beta P' \triangleright_{A_1} G_1\ \wedge\ \Gamma \vdash Q \triangleright_{A_2} G_2 & \\
 & \qquad\qquad \Rightarrow\ \ \Gamma \vdash P \mid Q \rightarrow_\beta P' \mid Q \triangleright_{A_1 \cup A_2 \cup A} G_1 \odot G_2 & (4) \\
(\mathrm{Res}) & \Gamma \vdash P \rightarrow_\beta P'\ \triangleright_A G\ \ \Rightarrow\ \ \Gamma\backslash a \vdash (a)P \rightarrow_\beta (a)P'\ \triangleright_{A\backslash\{a\}} G & (5) \\
(\mathrm{Str}) & P\ \equiv\ P' \quad \Gamma \vdash P'\ \rightarrow_\beta\ Q' \triangleright_A G \quad Q\ \equiv\ Q' \quad \Rightarrow\ \ \Gamma \vdash P\ \rightarrow_\beta\ Q \triangleright_A G &
\end{array}
$$

where (4) and (5) come from Figure 5. $\twoheadrightarrow_\beta$ is reflexive transitive closure of $\rightarrow_\beta$, while $=_\beta$ is the congruent closure of $\rightarrow_\beta$.

We note $\rightarrow_\beta$, which is clearly a subset of $\longrightarrow$, is more general than *linear reduction* studied in [23, 14, 22] in the sense that it allows the case when one name occurs many times on a process but at any one time only one input/output pair occurs actively (cf. Figure 3) and similar to the one studied in [18]. As in all these "safe" reductions, we have he following *non-interference* property for $\rightarrow_\beta$.

5.9. **Proposition.** (non-interference)
Suppose $P \twoheadrightarrow_\beta Q_1$ *and* $P \longrightarrow\!\!\!\!\!\rightarrow Q_2$. *Then there exists* Q' *s.t.* $Q_1 \longrightarrow\!\!\!\!\!\rightarrow Q'$ *and* $Q_2 \twoheadrightarrow_\beta Q'$.

P r o o f. See Appendix C. $\square$

A notable fact is $=_\beta$ is behaviourally sound.

5.10. **Proposition.** $=_\beta$ *is a sound congruence, hence* $P =_\beta Q$ *implies* $P =_\pi Q$. *Also* $=_\beta$ *is substitution closed.*

P r o o f. The reduction closure property is obtained by the non-interference property (Proposition 5.9) as proved in Proposition 3.7 in [18]. For action predicate, we firstly note $\Gamma \vdash P \rightarrow_\beta Q \triangleright_A G \;\Rightarrow\; \mathcal{AN}_\theta(P)\backslash A \subset \mathcal{AN}_\theta(Q)\backslash A$. Then we get the result with the same reasoning as in Proposition 3.7 in [20] and in Proposition 2.3.9 in [12]. The substitution closure property is mechanical by induction on derivation of $\rightarrow_\beta$ with renaming and substitution lemma in Lemma 4.12 (ii,iv). $\square$

We also notice, using the above Proposition:

5.11. **Proposition.** $=_\pi$ *is closed under substitution.*

P r o o f. The condition (1) in 5.1 is obvious. For the condition (2), let $a\colon \beta, b\colon \beta, \Gamma \vdash P \;=_\pi\; Q \triangleright_A G$ and c fresh. Then $b\colon \beta, \Gamma \vdash ca.P \mid \overline{c}b \;=_\pi\; ca.Q \mid \overline{c}b \;\triangleright_{A\cup\{c\}} G$, where both sides β-reduce to $P\{b/a\}$ and $Q\{b/a\}$, hence by $=_\beta\, \subset\, =_\pi$, we have the result. $\square$

Using $\rightarrow_\beta$, which is behaviourally "neutral" by Proposition 5.10, we can clarify how types of typed terms ensure their basic behavioural properties. We first give the case for "linear" behaviour, i.e. the behaviour corresponding to a given graph type. Essentially the following two propositions say that the term does act as its graph type specifies, following the activation ordering in the graph. Hereafter $\xrightarrow{l}$ denotes the standard early transition relation, cf.[28].

5.12. **Proposition.** (behaviour of typed terms: (1) linear case) *Write* $\mathsf{Hd}(G)$ *for the set of head nodes in* G *and* $G\backslash \mathbf{n}$ *for the elimination of the head with a label* $\mathbf{n}$ *from* G. *Then* $\Gamma \vdash P \triangleright_A G$ *implies:*

(i) $a\!\downarrow\!(x) \in \mathsf{Hd}(G) \quad\Rightarrow\quad \forall x'.\;$ *x' fresh in* Γ, A, G.

$$P \twoheadrightarrow_\beta \xrightarrow{ax'} P' \text{ and } \Gamma \vdash P' \triangleright_A (G\backslash a\!\downarrow\!(x))\{x'/x\}.$$

(ii) $a\uparrow(x) \in \mathsf{Hd}(G) \;\Rightarrow$

$$\begin{cases} P \twoheadrightarrow_\beta \xrightarrow{\overline{a}(x')} P' \quad \textit{and} \quad \Gamma \vdash P' \triangleright_A (G \backslash a\uparrow(x))\{x'/x\} \quad \textit{with } x' \notin A, \quad \textit{or}: \\ P \twoheadrightarrow_\beta \xrightarrow{\overline{a}x'} P' \quad \textit{and} \quad \Gamma \vdash P' \triangleright_{A\backslash\{x'\}} (G \backslash a\uparrow(x))\{x'/x\} \quad \textit{with } x' \in A \end{cases}$$

(iii) $a\downarrow\alpha \in \mathsf{Hd}(G) \;\Rightarrow\; \exists x.\, \{x\colon \alpha\} \asymp \Gamma \;\wedge\; x \notin (\mathsf{fn}(G) \cup A) \;\wedge$

$$P \twoheadrightarrow_\beta \xrightarrow{ax} P' \quad \textit{and} \quad \{x\colon\alpha\} \cup \Gamma \vdash P' \triangleright_A G\backslash a\downarrow\alpha.$$

(iv) $a\uparrow\alpha \in \mathsf{Hd}(G) \;\Rightarrow\; \exists x.\, x \notin \mathsf{fn}(G) \cup A \;\wedge$

$$\begin{cases} P \twoheadrightarrow_\beta \xrightarrow{\overline{a}x} P' \quad \textit{and} \quad \Gamma \vdash P' \triangleright_A G\backslash a\uparrow\alpha \quad \textit{with } x\colon\alpha \in \Gamma, \;\textit{or}: \\ P \twoheadrightarrow_\beta \xrightarrow{\overline{a}(x)} P' \quad \textit{and} \quad \{x\colon\alpha\} \cup \Gamma \vdash P' \triangleright_A G\backslash a\uparrow\alpha \quad \textit{if else} \end{cases}$$

(v) $\mathsf{fn}(\mathsf{Hd}(G)) = \{a_1, .., a_n\} \;\Rightarrow\; P \twoheadrightarrow_\beta P'$ *s.t. there is a unique free active occurrence of each* a_i *in* P'.

(vi) $a \in \mathsf{fn}(G) \;\wedge\; a \notin \mathsf{fn}(\mathsf{Hd}(G)) \cup A \;\Rightarrow\; \neg\exists l.\; P \xRightarrow{l}$ *with* $a \in \mathsf{fn}(l)$.

Proof. (i–iv) are by induction on P. The most difficult case is the parallel composition. We leave the detailed proof to Appendix D. (v) and (vi) are easy by checking the inference rules in Figure 5. □

5.13. **Definition.** If $\Gamma \vdash P \triangleright_A G$ and $\mathbf{n} \in \mathsf{Hd}(G)$, we say "$P$ has a transition $P \twoheadrightarrow_\beta \xrightarrow{l} P'$ corresponding to $\mathbf{n}$" if the transition is the one given in (i)..(iv) of Proposition 5.12 according to the form of $\mathbf{n}$.

The following is also notable.

5.14. **Corollary.** *Suppose* $\Gamma \vdash P \triangleright_A G$ *and* $\mathbf{n}_1, \mathbf{n}_2 \in \mathsf{Hd}(G)$ *which are distinct. Suppose also if* $P \twoheadrightarrow_\beta \xrightarrow{l_1} P_1$ *and* $P \twoheadrightarrow_\beta \xrightarrow{l_2} P_2$ *corresponding to* $\mathbf{n}_1$ *and* $\mathbf{n}_2$*, respectively. Then there exists* Q *such that* $P_1 \twoheadrightarrow_\beta \xrightarrow{l_2} Q$ *and* $P_2 \twoheadrightarrow_\beta \xrightarrow{l_1} Q$.

Proof. By the analysis of the case for (par) in the proof of Proposition 5.12. □

Next we present how protocol types constrain the behaviour of a typed term. Together with the preceding Propositions 5.12 and 5.9, the following says that a typed term does communicate obeying the specified protocol, up to β-equality.

5.15. **Proposition.** (behaviour of typed terms: (2) protocol types) *Suppose* $\Gamma \vdash P \triangleright_A G$ *and* $\{a\colon\beta\} \in \Gamma$.

(i) *Assume* $\beta = \{\downarrow(x) \to G',\; \uparrow(x) \to \overline{G}'\}$*. Then:*
(1) $a \in \mathcal{AN}_-(P) \;\Rightarrow\; \forall x'.\; x'$ *fresh in* Γ, A, G.

$$P \xrightarrow{ax'} P' \textit{ and } \Gamma \vdash P' \triangleright_A (G \uplus G'\{x'/x\}).$$

(2) $a \in \mathcal{AN}_+(P) \;\; \Rightarrow$

$$\begin{cases} P \xrightarrow{\overline{a}(x')} P' \quad \textit{and} \quad \Gamma \vdash P' \rhd_A (G \uplus \overline{G}'\{x'/x\}) & \textit{with } x' \notin A, \; \textit{or}: \\ P \xrightarrow{\overline{a}x'} P' \quad \textit{and} \quad \Gamma \vdash P' \rhd_{A \setminus \{x'\}} (G \uplus \overline{G}'\{x'/x\}) & \textit{with } x' \in A. \end{cases}$$

(ii) *Assume* $\beta = (\alpha)$. *Then:*

(1) $a \in \mathcal{AN}_-(P) \;\; \Rightarrow \;\; \forall b.\;\; \{b\colon \alpha\} \asymp \Gamma \;\wedge\; b \notin \mathsf{fn}(G) \cup A \;\wedge$
$$P \xrightarrow{ab} P' \quad \textit{and} \quad \{b\colon \alpha\} \cup \Gamma \vdash P' \rhd_A G.$$

(2) $a \in \mathcal{AN}_+(P) \;\; \Rightarrow \;\; \exists b.\, b \notin \mathsf{fn}(G) \cup A \;\wedge$

$$\begin{cases} P \xrightarrow{\overline{a}b} P' \quad \textit{and} \quad \Gamma \vdash P' \rhd_A G \quad \textit{with } b\colon \alpha \in \Gamma, \; \textit{or}: \\ P \xrightarrow{\overline{a}(b)} P' \quad \textit{and} \quad \{b\colon \alpha\} \cup \Gamma \vdash P' \rhd_A G \quad \textit{if else}. \end{cases}$$

Proof. Easily derived by case analysis on the form of the protocol types and binding condition between Γ and G. □

We use Propositions 5.12 and 5.15 for the proof of the main theorem in Section 7.

6. Properties of Reduction-Closed Equalities

In the following, we list basic properties of reduction-closed equalities, introducing a general proof method suggested by [40]. The schemes from Definition 6.1 to Proposition 6.7 can often be adopted to other calculi, see [48] for details. We use the final proposition (Proposition 6.7) for the proof of the main theorem. Those who are not interested in general proof methods may safely skip this section without losing consistency.

First we define several closure operation over typed relations.

6.1. **Definition.** (closure operation) Let Φ, Φ',... range over functions between relations on typed π-terms. We write $\Phi_p(\mathcal{R})$ for the result of closing $\mathcal{R}$ under parallel composition and structure rules. $\Phi_{pr}(\mathcal{R})$ denotes the result of closing $\mathcal{R}$ under under parallel composition, restriction and structure rules. Then we say $\mathcal{R}$ is *p-closed* (resp. *pres-closed*) if we have $\Phi_p(\mathcal{R}) = \mathcal{R}$ (resp. $\Phi_{pr}(\mathcal{R}) = \mathcal{R}$). $\Phi_c()$ denotes a congruence closure operation. We also define *up-to closure of* $\mathcal{R}$ *under* $\sim$, as: $\Phi_\sim(\mathcal{R}) \overset{\mathrm{def}}{=} \sim \mathcal{R} \sim$.

First let us note the following fact on substitution closure.

6.2. **Proposition.** *Suppose* $\mathcal{R}$ *is substitution closed. Then* $\Phi_p(\mathcal{R})$*,* $\Phi_{pr}(\mathcal{R})$ *and* $\Phi_c(\mathcal{R})$ *are substitution closed.*

Proof. Mechanical by induction on derivation using Lemma 4.12 (ii) and (vi). □

In the weak setting, we introduce the following definitions corresponding to those in [40] without using label transition relation.

6.3. **Definition.** (r.c.progression, r.c.respectable)

(1) *Given two reflexive relations $\mathcal{R}$ and $\mathcal{S}$, we say that $\mathcal{R}$* r.c.progresses *to $\mathcal{S}$, written $\mathcal{R} \ \mapsto\ \mathcal{S}$, if P and Q implies whenever $P \longrightarrow\!\!\!\!\rightarrow P'$, there exists Q' s.t. $Q \longrightarrow\!\!\!\!\rightarrow Q'$ with $P'\ \mathcal{S}\ Q'$, and the symmetric case.*

(2) *A function Φ is* r.c. respectable *if Φ is a closure operator over typed relations, and $\mathcal{R} \mapsto \mathcal{S}$ implies $\Phi(\mathcal{R}) \mapsto \Phi(\mathcal{S})$.*

Given a typed relation $\mathcal{R}$, the notions of reduction closed property and respectness $\Downarrow_{a^\theta}$ are easily adopted to (any) typed relations in general: $\mathcal{R}$ is reduction-closed iff $\mathcal{R} \mapsto \mathcal{R}$, and we say $\mathcal{R}$ respects $\Downarrow_{a^\theta}$ if $\Gamma \vdash P \mathcal{R} Q \triangleright_A G$ and $\Gamma \vdash P \Downarrow_{a^\theta} \ \triangleright_A G$ then $\Gamma \vdash Q \Downarrow_{a^\theta} \ \triangleright_A G$, and the symmetric case.

We list the following three technical lemmas, whose detailed proof is given in Appendix E. Lemma 6.4 says r.c.respectable functions are compositional under a mild condition and Lemma 6.5 tells us that we only have to consider p-closed context and closure under substitution, while Lemma 6.6 states that on reduction closed theories, we can easily apply "up to technique" in the weak setting.

6.4. **Lemma.**

(i) *Suppose Φ is r.c.respectable. Then $\mathcal{R} \mapsto \Phi(\mathcal{R})$ implies $\Phi(\mathcal{R})$ is reduction closed.*

(ii) *Suppose Φ and Φ' are r.c.respectable and $\Phi \circ \Phi' \ \supset\ \Phi' \circ \Phi$. Then $\Phi \circ \Phi'$ is r.c.respectable.*

6.5. **Lemma.** *Assume $\mathcal{R}$ is substitution closed.*

(i) *$\Phi_p(\mathcal{R})$ is reduction closed $\quad\Rightarrow\quad$ $\Phi_c(\mathcal{R})$ is reduction closed.*

(ii) *$\Phi_p(\mathcal{R})$ respects $\Downarrow_{a^\theta}$ $\quad\Rightarrow\quad$ $\Phi_c(\mathcal{R})$ respects $\Downarrow_{a^\theta}$.*

6.6. **Lemma.** (up to reduction-closed congruence) *Suppose $\cong$ is a congruence relation, $\mathcal{R}$ is a p-closed relation, and both are substitution and reduction closed. Then:*

(i) *$\Phi_{\cong}$ is r.c.respectable and $\Phi_c(\mathcal{R}\cup \cong)$ is reduction closed.*

(ii) *$\Phi_{pr}(\mathcal{R})$ is reduction closed.*

(iii) *$\mathcal{R} \ \mapsto\ \Phi_{\cong}(\Phi_{pr}(\mathcal{R}))$ implies $\Phi_c(\mathcal{R}\cup \cong)$ is reduction and substitution closed.*

We use these lemmas to prove:

6.7. **Proposition.** *Suppose $=_s$ is sound and substitution closed and $\mathcal{R}$ is a substitution and p-closed relation, and moreover satisfies the following properties:*

(1) *$\mathcal{R} \ \mapsto\ \Phi_{=_s}(\Phi_{pr}(\mathcal{R}))$.*

(2) *$\mathcal{R}$ respects the action predicate.*

Then $\Phi_c(=_s \cup \mathcal{R})$ is sound and substitution closed.

P r o o f. First the substitution closure is given by Lemma 6.6 (iii). Assume $\Gamma \vdash C_r[\ ]_{\Delta,A_0,G_0} \triangleright_{A'} G'$ (often simply $C_r[\ ]_{\Delta,A_0,G_0}$) denotes a typed reduction context, i.e. a context whose hole is not under prefix nor replication, with the type of the hole specified. C_r, C'_r, C''_r... range over reduction context. We firstly note that (1) and (2) are equivalent to the following property by Lemma 6.6 (iii).

(1) whenever $\Gamma \vdash C_r[P]_{\Delta,A_0,G_0} \ \mathcal{R} \ C_r[Q]_{\Delta,A_0,G_0} \rhd_A G$, $C_r[P] \longrightarrow\!\!\!\!\!\rightarrow P'$ implies, for some Q', $C_r[Q] \longrightarrow\!\!\!\!\!\rightarrow Q'$, $P' =_s C'_r[P'_0]_{\Delta,A_0,G_0}$, $\Delta \vdash P'_0 \ \mathcal{R} \ Q'_0 \rhd_{A_0} G_0$, and $C'_r[Q'_0]_{\Delta,A_0,G_0} =_s Q'$, and its symmetric case.

(2) $\mathcal{R}$ respects the action predicate.

The reduction closure property is by Lemma 6.6 (iii). For the action predicate, first we prove if $\mathcal{R}$ as above respects $\Downarrow_{a^\theta}$, then $\Phi_{pr}(\mathcal{R})$ respects $\Downarrow_{a^\theta}$. If these are proved, then we know, if $\Phi_{pr}(\mathcal{R})$ respects $\Downarrow_{a^\theta}$, the transitive closure of $(=_s \Phi_{pr}(R) =_s)$ respects $\Downarrow_{a^\theta}$, which means the p-closed relation respects $\Downarrow_{a^\theta}$, hence the result by Lemma 6.5. $\square$

7. Fully Abstract Encoding for Polyadic π-calculus

This section establishes the full abstraction result for the standard encoding of polyadic π-calculus into monadic π-calculus in terms of a basic behavioural equality in each setting, as a non-trivial application of our type discipline. We first give a brief review of Polyadic π-calculus and its sorting, including behavioural equality. Then, after introducing the encodings, we make it clear what is the essential problems for such an equational embedding. Then we go into the main technical discussions, establishing the full abstraction result.

As discussed at the end of the section, the result is easily extended to encodings of calculi with more general and complex operational structures realizable in monadic π-calculus. We start from a review of polyadic π-calculus and its sorting.

7.1. **Polyadic π-calculus.** The syntax of the polyadic π-calculus is given below [26, 39].

$$P \ ::= \ a(x_1..x_n).P \ \mid \ \overline{a}\langle v_1..v_n\rangle.P \ \mid \ P\,|\,Q \ \mid \ (a)P \ \mid \ !P \ \mid \ \mathbf{0}$$

$a(x_1..x_n).P$ and $\overline{a}\langle v_1..v_n\rangle.P$ are input and output processes respectively, with n being the arity of the input (output) prefix. The set of *sorting* $\mathcal{S}$ (S_1, S_2, ...) is given by the following grammar:

$$S ::= \mathbf{x} \mid (S_1 S_2 ... S_n)$$

with $n \geq 0$. (We omit the recursive sorting for simplicity of presentation, whose incorporation does not affect the reasoning and the result in the following, cf. 8.2). $\Gamma, \Gamma', \ldots$ denote typing functions, i.e. functions from a finite subset of $\mathcal{N}$ to the above set.

7.2. **Reduction and Equality over Sorted Terms.** The reduction relation in polyadic π-calculus is given by replacing (COM) rule in 2.3 with the following (COM$_\mathrm{p}$) as in the standard way.

$$(\mathrm{COM_p}) \quad a(x_1..x_n).P \,|\, \overline{a}\langle v_1..v_n\rangle.Q \ \longrightarrow \ P\{v_1v_2...v_n/x_1x_2...x_n\} \,|\, Q$$

We use the standard typing system and the early labeled transition system for polyadic π-terms as in [43] and [39], respectively, which we list in Appendix F for easy reference. We write $\Gamma \vdash P$ in the following, where Γ is a map from a finite set of names to the set of sortings. The following subject reduction property is known concerning the sorted terms, cf. [26, 43].

If $\Gamma \vdash P$ *and* $P \longrightarrow\!\!\!\!\rightarrow P'$, *then we have* $\Gamma \vdash P'$.

To avoid ambiguity, here and henceforth we often write, $\Gamma_{\mathsf{p}}, \Delta_{\mathsf{p}}, ...$ for the set of sorting, $\rightarrow_{\mathsf{p}}$ for the reduction relation and $\xrightarrow{l}_{\mathsf{p}}$ for the labeled transition relation. Typed relations and various classes of relations over polyadic π-terms are defined in the same way as in 5.1. Notice the subject reduction property tells us $\rightarrow_{\mathsf{p}}$ is a typed relation in this setting. We define a sound congruence as in Definition 5.2 based on action predicates. Then we have:

7.3. **Proposition.** *There is the maximum sound equality over sorted polyadic* π*-terms which we denote by* $=_{\mathsf{p}}$ *.*

By the similar reasoning to Proposition 5.11, $=_{\mathsf{p}}$ is substitution closed. We use $=_{\mathsf{p}}$, which can again be characterised in various ways like $=_{\pi}$, as the representative notion of behavioural equality over polyadic π-terms from now on.

The following mapping of polyadic π-terms into monadic π-terms is standard, except we present the mapping at the level of *types* too (which, as we shall see later, plays the central role for our main result).

7.4. **Definition.** (translation from polyadic to monadic π-calculus, [28, 26])

- Mapping for terms (with c, z fresh).

$$[\![\overline{a}\langle v_1..v_n\rangle.P]\!] \stackrel{\text{def}}{=} \overline{a}^*\langle v_1..v_n\rangle.[\![P]\!] \qquad [\![P_1 \mid P_2]\!] \stackrel{\text{def}}{=} [\![P_1]\!] \mid [\![P_2]\!] \qquad [\![\mathbf{0}]\!] \stackrel{\text{def}}{=} \mathbf{0}$$

$$[\![a(x_1..x_n).P]\!] \stackrel{\text{def}}{=} a^*(x_1..x_n).[\![P]\!] \qquad [\![(a)P]\!] \stackrel{\text{def}}{=} (a)[\![P]\!] \qquad [\![!P]\!] \stackrel{\text{def}}{=} ![\![P]\!]$$

- Mapping for types (with x fresh).

$$[\![\mathbf{x}]\!] \stackrel{\text{def}}{=} \mathbf{x} \quad [\![(S_1...S_n)]\!] \stackrel{\text{def}}{=} \{\downarrow(x) \rightarrow x\downarrow[\![S_1]\!] \cdots \rightarrow x\downarrow[\![S_n]\!],\ \uparrow(x) \rightarrow x\uparrow[\![S_1]\!] \cdots \rightarrow x\uparrow[\![S_n]\!]\}$$

Notice a sorting is mapped to a protocol type which consists only of consecutive inputs or outputs. The syntactic correspondence follows.

7.5. **Lemma.**

(i) $\mathsf{fn}(P) = \mathsf{fn}([\![P]\!])$, $\mathsf{fn}(\Gamma_{\mathsf{p}}) = \mathsf{fn}([\![\Gamma_{\mathsf{p}}]\!])$, $\mathcal{AN}_\theta(P) = \mathcal{AN}_\theta([\![P]\!])$.

(ii) *For any substitution* σ, $[\![P\sigma]\!] \equiv_\alpha [\![P]\!]\sigma$ *and* $[\![\Gamma_{\mathsf{p}}\sigma]\!] \equiv_\alpha [\![\Gamma_{\mathsf{p}}]\!]\sigma$.

(iii) $P \equiv Q \;\Leftrightarrow\; [\![P]\!] \equiv [\![Q]\!]$.

Proof. Easy by induction. □

Now we show the first basic result concerning Milner's open issue [26]: a polyadic π-term has a sort if and only if its mapping in monadic π-term has the corresponding graph (protocol) type.

7.6. **Proposition.** $\Gamma_{\mathsf{p}} \vdash P \quad \Leftrightarrow \quad [\![\Gamma_{\mathsf{p}}]\!] \vdash [\![P]\!]$.

Proof. By the structural induction on a polyadic term P. For ($\Rightarrow$) direction, the cases $P \equiv \mathbf{0}$, $P \equiv !P'$ and $P \equiv (a)P'$ are trivial. For the case $P \equiv a(x_1...x_n).P'$, we first use (in$_2$) rules n times.

$$[\![\Gamma]\!] \vdash zx_1....zx_n.[\![P]\!] \triangleright z\downarrow[\![S_2]\!] \rightarrow \cdots \rightarrow z\downarrow[\![S_n]\!]$$

Then we apply (weak) rules as several times as we need and (in$_3$) rule once to get the result. The case of $P \equiv \overline{a}\langle v_1...v_n\rangle.P'$ is just similar. The case of parallel composition is obvious by taking $A_1 = A_2 = \emptyset$ and $G_1 = G_2 = \emptyset$ in (par) rule. ($\Leftarrow$) direction is also easily obtained with the same reasoning by translating the protocol type into the sorting. □

While the result may seem straightforward, we note that none of the preceding refinements of sorting [36, 35, 14, 22] may give this result, at least with this generality. Notice also that the above result allows us to regard the mapping as the one from explicitly sorted terms to typed monadic terms. Thus we have embedded the sorted terms to the universe of typed monadic π-calculus: our interest is how this embedding is faithful at the semantic (observational) level, i.e. at the level of $=_\pi$ and $=_{\mathsf{p}}$. Before going into technical development, we recall the following fact in the untyped setting.

7.7. **Fact.** *Let $=^u_{\mathsf{p}}$ and $=^u_\pi$ be the maximum sound equalities for untyped polyadic and monadic calculi, defined as in 5.2. Then:*

$$[\![P]\!] =^u_\pi [\![Q]\!] \quad \Rightarrow \quad P =^u_{\mathsf{p}} Q, \quad \text{but} \quad P =^u_{\mathsf{p}} Q \quad \not\Rightarrow \quad [\![P]\!] =^u_\pi [\![Q]\!].$$

Proof. We know the first clause holds as seen in [26]. For the second clause, we present the following counter example.

$$P \equiv \overline{a}\langle v\rangle.\overline{a}\langle v\rangle.\mathbf{0} \qquad Q \equiv \overline{a}\langle v\rangle.\mathbf{0} \mid \overline{a}\langle v\rangle.\mathbf{0} \tag{7.1}$$

Obviously $P =^u_{\mathsf{p}} Q$. The translations are given as:

$$[\![P]\!] \equiv \overline{a}(c).\overline{c}v.\overline{a}(c).\overline{c}v.\mathbf{0} \qquad [\![Q]\!] \equiv \overline{a}(c).\overline{c}v.\mathbf{0} \mid \overline{a}(c).\overline{c}v.\mathbf{0} \tag{7.2}$$

Then take the context $(a)([\ ] \mid ax.ax.\overline{e}.\mathbf{0})$ where e is fresh. Then $C[[\![Q]\!]]$ gives an observable at e while $[\![P]\!]$ does not. □

The same results hold taking any of weak/strong versions of late/early/barbed bisimilarities and congruences [28, 29, 39], and other kinds of examples also exist. The failure of full abstraction is because the atomicity of the original polyadic name passing is not preserved: one action is decomposed into many finer communication, and intermediate steps can be interfered by another party. In the following, we show that atomicity is semantically preserved for the same encoding, if we use the theory of types we have developed so far. We start from the adequacy result. This is easily proved by the following operational correspondence.

7.8. **Proposition.** (operational correspondence)

(i) *If* $\Gamma_{\mathrm{p}} \vdash P \rightarrow_{\mathrm{p}} Q$*, then* $[\![\Gamma_{\mathrm{p}}]\!] \vdash [\![P]\!] \longrightarrow\!\!\!\twoheadrightarrow_{\beta} [\![Q]\!]$.

(ii) *If* $[\![\Gamma_{\mathrm{p}}]\!] \vdash [\![P]\!] \longrightarrow P'$*, then for some* Q*, we have* $[\![\Gamma_{\mathrm{p}}]\!] \vdash P' \twoheadrightarrow_{\beta} [\![Q]\!]$ *with* $\Gamma_{\mathrm{p}} \vdash P \rightarrow_{\mathrm{p}} Q$.

Proof. By structural induction of P. We use, as well as syntactic checking, the subject reduction theorems of both of polyadic π-calculus (Theorem 2.7 in [43]) and monadic one (Theorem 4.13). Suppose $P \equiv a(x_1x_2...x_n).Q \mid \overline{a}\langle v_1v_2...v_n\rangle.R$ and $\Gamma_{\mathrm{p}} \vdash P \rightarrow_{\mathrm{p}} Q\{\tilde{v}/\tilde{x}\} \mid R$. By Proposition 7.6, we have

$$[\![\Gamma_{\mathrm{p}}]\!] \vdash az.zx_1.zx_2...zx_n.[\![Q]\!] \mid \overline{a}(c).\bar{c}x_1.\bar{c}x_2...\bar{c}x_n.[\![R]\!]$$

By Theorem 4.13, after one step reduction, we get:

$$[\![\Gamma_{\mathrm{p}}]\!] \vdash (c)(cx_1.cx_2...cx_n.[\![Q]\!] \mid \bar{c}x_1.\bar{c}x_2...\bar{c}x_n.[\![R]\!]) \tag{7.3}$$

In fact, assuming $v_i : \alpha_i \in [\![\Gamma_{\mathrm{p}}]\!]$, this sequent can be derived from the following deductions.

$$\dfrac{\dfrac{[\![\Gamma_{\mathrm{p}}]\!] \vdash cx_1.cx_2...cx_n.[\![Q]\!] \triangleright c\downarrow\alpha_1 \to \cdots c\downarrow\alpha_n \quad [\![\Gamma_{\mathrm{p}}]\!] \vdash \bar{c}v_1.\bar{c}v_2...\bar{c}v_n.[\![R]\!] \triangleright c\uparrow\alpha_1 \to \cdots c\uparrow\alpha_n}{[\![\Gamma_{\mathrm{p}}]\!] \vdash (cx_1.cx_2...cx_n.[\![Q]\!] \mid \bar{c}x_1.\bar{c}x_2...\bar{c}x_n.[\![R]\!]) \triangleright_{\{c\}}}\text{(par)}}{[\![\Gamma_{\mathrm{p}}]\!] \vdash (c)(cx_1.cx_2...cx_n.[\![Q]\!] \mid \bar{c}v_1.\bar{c}v_2...\bar{c}v_n.[\![R]\!])}\text{(res)}$$

Now we know

$$[\![\Gamma_{\mathrm{p}}]\!] \vdash (cx_1.cx_2...cx_n.[\![Q]\!] \mid \bar{c}x_1.\bar{c}x_2...\bar{c}x_n.[\![R]\!]) \triangleright_{\{c\}} \twoheadrightarrow_{\beta} \ [\![\Gamma_{\mathrm{p}}]\!] \vdash [\![Q]\!]\{\tilde{v}/\tilde{x}\} \mid [\![R]\!] \triangleright_{\{c\}}$$

Since $\rightarrow_{\beta}$ is closed under name restriction by definition, we get the following required result from 7.3.

$$[\![\Gamma_{\mathrm{p}}]\!] \vdash (c)(cx_1.cx_2...cx_n.[\![Q]\!] \mid \bar{c}x_1.\bar{c}x_2...\bar{c}x_n.[\![R]\!]) \twoheadrightarrow_{\beta} \ [\![\Gamma_{\mathrm{p}}]\!] \vdash [\![Q]\!]\{\tilde{v}/\tilde{x}\} \mid [\![R]\!] \quad \Box$$

From this Proposition, together with Lemma 7.5 and Proposition 5.10, we obtain the correspondence in terms of the *action predicate.*

7.9. **Proposition.** (action predicate) $\quad \Gamma_{\mathrm{p}} \vdash P \Downarrow_{a^{\theta}} \quad \Leftrightarrow \quad [\![\Gamma_{\mathrm{p}}]\!] \vdash [\![P]\!] \Downarrow_{a^{\theta}}$.

Proof. Firstly note that by Lemma 7.5 (i), $a \in \mathcal{AN}_{\theta}(P) \ \Leftrightarrow \ a \in \mathcal{AN}_{\theta}([\![P]\!])$. ($\Rightarrow$) direction is obvious because $\Gamma \vdash P \rightarrow_{\mathrm{p}} Q \ \Rightarrow \ [\![\Gamma]\!] \vdash [\![P]\!] \longrightarrow\!\!\!\twoheadrightarrow [\![Q]\!]$ by Proposition 7.8 (i), and this implies $\Gamma \vdash P \Downarrow_{a^{\theta}} \ \Rightarrow \ [\![\Gamma]\!] \vdash [\![P]\!] \Downarrow_{a^{\theta}}$. ($\Leftarrow$) direction is also easy because $=_{\beta}$ is sound equality. $\Box$

Now we can show that the encoding is equationally "adequate."

7.10. **Theorem. (Adequacy)**

$$[\![\Gamma_{\mathrm{p}}]\!] \vdash [\![P]\!] =_{\pi} [\![Q]\!] \quad \Rightarrow \quad \Gamma_{\mathrm{p}} \vdash P =_{\mathrm{p}} Q.$$

Proof. We first construct a relation $\cong$ as follows.

$$[\![\Gamma_{\mathrm{p}}]\!] \vdash [\![P]\!] =_{\pi} [\![Q]\!] \ \Rightarrow \ \Gamma_{\mathrm{p}} \vdash P \cong Q$$

Then we show that $\cong$ is (a) substitution-closed congruence, (b) reduction-closed, and (c) respects the action predicate $\Downarrow_{a^{\theta}}$. (a) is obvious by substitution closure of the mapping, congruence of $=_\pi$, and Lemma 7.5 (iii). For (b), let us assume $[\![P]\!] =_\pi [\![Q]\!]$. By Proposition 3.2 (ii) in [20], it only has to be proved the one step reduction from P. Then

$$\begin{array}{lcll}
\Gamma_{\mathsf{p}} \vdash\ P \to_{\mathsf{p}} P' & \Rightarrow & [\![\Gamma_{\mathsf{p}}]\!] \vdash [\![P]\!] \longrightarrow^{+} [\![P']\!] & \text{(Prop.5.9, 7.8(i))} \\
 & \Rightarrow & [\![\Gamma_{\mathsf{p}}]\!] \vdash [\![Q]\!] \longrightarrow\!\!\!\!\rightarrow Q'' \ \wedge\ Q'' =_\pi [\![P']\!] & \text{(reduction closure of } =_\pi) \\
 & \Rightarrow & Q'' \twoheadrightarrow_\beta [\![Q']\!] \ \wedge\ Q \twoheadrightarrow_{\mathsf{p}} Q' & \text{(Prop 5.10, 7.8(ii))} \\
 & \Rightarrow & [\![P']\!] =_\pi Q'' =_\pi [\![Q']\!] & \text{(soundness of } =_\beta) \\
 & \Rightarrow & \Gamma_{\mathsf{p}} \vdash P' \cong Q'. & \text{(transitivity of } =_\pi)
\end{array}$$

(c) is obvious because $P_{\mathsf{p}} \Downarrow_{a^\theta} \ \Leftrightarrow\ [\![P_{\mathsf{p}}]\!] \Downarrow_{a^\theta} \ \Leftrightarrow\ [\![Q_{\mathsf{p}}]\!] \Downarrow_{a^\theta} \ \Leftrightarrow\ Q_{\mathsf{p}} \Downarrow_{a^\theta}$ by Proposition 7.9. □

Now we turn to the difficult direction, i.e. completeness. The completeness part crucially relies on two essential roles of the types in the present setting, i.e. as information to control the composition of terms and as a guarantee of terms' behaviour. We need to establish a few basic behavioural properties of the encoding, using the relationship between types and behaviour studied in Section 5. We start from an easy but useful corollary of Proposition 5.12. Below $\leftrightarrow$ is defined as a symmetric irreflexive relation over non-τ labels for the early transition relation generated from $\overline{a}c \leftrightarrow ac$ and $\overline{a}(c) \leftrightarrow ac$.

7.11. **Lemma.** *Suppose there is a derivation:*

$$\frac{\Gamma \vdash P_1 \rhd_{A_1} G_1 \qquad \Gamma \vdash P_2 \rhd_{A_2} G_2}{\Gamma \vdash P_1 \mid P_2 \rhd_{A_1 \cup A_2 \cup A} G_1 \odot G_2}\ (\text{par})$$

Then if $\mathbf{n} \in \mathsf{Hd}(G_1)$ *and* $\overline{\mathbf{n}} \in \mathsf{Hd}(G_2)$, *for some* R *we have* $\Gamma \vdash P_1 \mid P_2 \twoheadrightarrow_\beta R \rhd_{A_1 \cup A_2 \cup A} G_1 \odot G_2$ *such that* $P_1 \twoheadrightarrow_\beta \xrightarrow{l_1} P_1'$ *corresponding to* $\mathbf{n}$, $P_2 \twoheadrightarrow_\beta \xrightarrow{l_2} P_2'$ *corresponding to* $\overline{\mathbf{n}}$, $l_1 \leftrightarrow l_2$, *and (1)* $\mathcal{BN}(l_1) \cup \mathcal{BN}(l_2) = \emptyset$ *then* $R \equiv P_1' | P_2'$, *(2)* $\mathcal{BN}(l_1) \cup \mathcal{BN}(l_2) = \{c\}$ *then* $R \equiv (c)(P_1' | P_2')$.

Proof. Easy syntactic analysis. □

The next result shows the effect of the observable action initiating the encoded protocol for a polyadic name passing.

7.12. **Proposition.** *Suppose* $[\![\Gamma_0]\!] \vdash [\![P]\!]$, $a : [\![(S_1..S_n)]\!] \in [\![\Gamma_0]\!]$ *and* $[\![\Gamma_0]\!] \vdash [\![P]\!] \longrightarrow\!\!\!\!\rightarrow P'$. *Then:*

(i) *If* $a \in \mathcal{AN}_{-}(P')$, *then for all* x'. x' *fresh in* Γ_0, $P' \xrightarrow{ax'} P''$ *and* $[\![\Gamma_0]\!] \vdash P'' \rhd x' \downarrow [\![S_1]\!] \to \cdots \to x' \downarrow [\![S_n]\!]$.

(ii) *If* $a \in \mathcal{AN}_{-}(P')$, *then for any name* c, $P' \xrightarrow{\overline{a}c}$ *is impossible, and for some* P'' *we have* $P' \xrightarrow{\overline{a}(x')} P''$ *such that* $[\![\Gamma_0]\!] \vdash P' \rhd x' \uparrow [\![S_1]\!] \to \ldots \to x' \uparrow [\![S_n]\!]$.

Proof. Obvious by Proposition 5.15 (i-1) and the first clause in (i-2), respectively, considering the form of the mapping of sorting. □

For stating further results we need the mappings at the level of labeled transition relation, following the mapping in Definition 7.4.

$$\begin{array}{rcll} [\![\tau]\!] & \overset{\text{def}}{=} & \tau & \\ [\![a\langle x_1...x_n\rangle]\!] & \overset{\text{def}}{=} & az \cdot zx_1 \cdots zx_n & (z \text{ fresh}) \\ [\![(\tilde{b})\overline{a}\langle v_1...v_n\rangle]\!] & \overset{\text{def}}{=} & \overline{a}(c) \cdot (\tilde{b}_1)cv_1 \cdots (\tilde{b}_n)cv_n & (c \text{ fresh}) \end{array}$$

where $\tilde{b}_i \overset{\text{def}}{=} \tilde{b}\backslash v_1..v_{i-1}$ (the result of taking off each $v_1..v_{i-1}$ from $\tilde{b}$), and $(\tilde{b}_i)cv_i \overset{\text{def}}{=} c(v_i)$ if $v_i \in \{\tilde{b}_i\}$. Then the decomposition at the level of labeled transition relation becomes:

$$\begin{array}{rcl} \xrightarrow{[\![l]\!]} & \overset{\text{def}}{\Leftrightarrow} & \xrightarrow{l_1}\xrightarrow{l_2}\cdots\xrightarrow{l_n} \\ \overset{[\![l]\!]}{\Longrightarrow} & \overset{\text{def}}{\Leftrightarrow} & \longrightarrow\!\!\!\!\!\rightarrow\xrightarrow{l_1}\longrightarrow\!\!\!\!\!\rightarrow\xrightarrow{l_2}\longrightarrow\!\!\!\!\!\rightarrow\cdots\longrightarrow\!\!\!\!\!\rightarrow\xrightarrow{l_n}\longrightarrow\!\!\!\!\!\rightarrow \end{array}$$

where $[\![l]\!] = l_1 \cdot l_2 \cdots l_n$. In particular, if all $\longrightarrow\!\!\!\!\!\rightarrow$ is $\twoheadrightarrow_\beta$ in the second line, we write $\overset{[\![l]\!]}{\Longrightarrow}_\beta$.

We notice the following property of the mapped transition.

7.13. **Proposition.** *For $l \neq \tau$, $P \xrightarrow{l}_{\mathrm{p}} P' \;\Leftrightarrow\; [\![P]\!] \xrightarrow{[\![l]\!]} [\![P']\!]$.*

Proof. Easy syntactic analysis, observing, for $\Leftarrow$ direction, that the fresh name used for the initial action of the encoding occurs only in the part which encodes the concerned polyadic name passing. □

7.14. **Remark.** While Proposition 7.13 together with Proposition 7.8 gives the strong correspondence between behaviour of the original polyadic terms and their encodings, the analysis of semantic properties of the encodings cannot rely on such correspondence only. This is because we should think of interaction between the encoding and monadic terms which *may not be the encodings themselves.* The following monadic term makes us understood the reason why we should consider such a situation.

$$R \equiv (bb_1b_2)(\overline{a}(c).(\overline{b}v \mid \overline{c}v_1.bz.\overline{c}v_2) \mid \overline{e}b_1.\overline{f}b_2)$$

R is clearly not in the form of the encoding, but R is typable in the way $[\![a\colon(\mathbf{x}_1\mathbf{x}_2)]\!] \vdash R \triangleright e\uparrow(b_1) \rightarrow f\uparrow(b_2)$, and therefore can be composed to, say, $[\![a\colon(\mathbf{x}_1\mathbf{x}_2)]\!] \vdash [\![a(x_1x_2).\mathbf{0}]\!]$. This example shows that what we need to establish is:

> A sequence of transitions $\xrightarrow{[\![l]\!]}$ by $[\![\Gamma]\!] \vdash [\![P]\!]$ cannot be interfered if the other party is a monadic typed term R which is composable to $[\![\Gamma]\!] \vdash [\![P]\!]$.

This is the content of the following key lemma.

7.15. **Lemma.** (Composition Lemma) *First assume* $\Gamma_{\mathsf{p}} \vdash P$. *Next suppose, with* $\Gamma_0 \supset [\![\Gamma_{\mathsf{p}}]\!]$, *we have a derivation:*

$$\frac{\Gamma_0 \vdash [\![P]\!] \qquad \Gamma_0 \vdash R \triangleright_A G}{\Gamma_0 \vdash [\![P]\!] \mid R \triangleright_A G}\ (par)$$

Then if $[\![P]\!] \mid R \longrightarrow P_0$, *one of the following is satisfied.*

(1) $P_0 \equiv [\![P]\!] \mid R'$ *with* $R \longrightarrow R'$.

(2) $P_0 \twoheadrightarrow_\beta [\![P']\!] \mid R$ *with* $P \rightarrow_{\mathsf{p}} P'$.

(3) $P_0 \twoheadrightarrow_\beta (\tilde{c})([\![P']\!] \mid R')$ *with* $[\![P]\!] \xrightarrow{[\![l]\!]} [\![P']\!]$ *and* $P \xrightarrow{l}_{\mathsf{p}} P'$, *such that either:*

(a) $l = a\langle v_1 v_2 ... v_n\rangle$ *and* $R \overset{[\![(\tilde{c})\overline{a}\langle v_1 v_2 ... v_n\rangle]\!]}{\Longrightarrow_\beta} R'$, *or*

(b) $l = (\tilde{c})\overline{a}\langle v_1 v_2 ... v_n\rangle$ *and* $R \overset{[\![a\langle v_1 v_2 ... v_n\rangle]\!]}{\Longrightarrow_\beta} R'$.

Proof. (1) is obvious. (2) is from Proposition 7.8 (ii). Suppose neither (1) nor (2) holds. We note if $[\![P]\!] \xrightarrow{l} P_1$ where a free subject of l is a, then a type mapped from a sorting, say $a\colon [\![(S_1..S_n)]\!]$ should be in Γ_0. This implies that there is no free output from R or $[\![P]\!]$ by Proposition 7.12, so there are two cases for (3): by Lemma 7.11, $P \xrightarrow{l} P_1$ and $P_0 \equiv (c)(P_1 \mid R'')$ where, with $c \notin \mathsf{fn}(G) \cup A$ and $a\colon \{\downarrow(c') \to G', \uparrow(c') \to \overline{G}'\} \in \Gamma_0$,

(a) $[\![P]\!] \xrightarrow{ac} P_1$ *with* $\Gamma \vdash P_1 \triangleright G'\{c/c'\}$, *and* $R \xrightarrow{\overline{a}(c)} R''$ *with* $\Gamma \vdash R'' \triangleright_A \overline{G}'\{c/c'\} \uplus G$.

(b) $[\![P]\!] \xrightarrow{\overline{a}(c)} P_1$ *with* $\Gamma \vdash P_1 \triangleright \overline{G}'\{c/c'\}$, *and* $R \xrightarrow{ac} R''$ *with* $\Gamma \vdash R'' \triangleright_A G'\{c/c'\} \uplus G$.

For (a), by Proposition 7.12 (ii), we have

$$\Gamma_0 \vdash P_1 \triangleright\ c{\downarrow}[\![S_1]\!] \to ... \to c{\downarrow}[\![S_n]\!] \quad \text{and} \quad \Gamma_0 \vdash R'' \triangleright\ c{\uparrow}[\![S_1]\!] \to ... \to c{\uparrow}[\![S_n]\!] \uplus G$$

By repeatedly applying Lemma 7.11, we get the behaviour of R and P_1 (hence $[\![P]\!]$). The latter is then translated back to the corresponding single transition of P by Proposition 7.13. The next case (b) is similar starting from Proposition 7.12 (i). □

Now we are ready to prove the main theorem of this paper.

7.16. **Theorem.** (**Main Theorem, Full Abstraction**)

$$\Gamma_{\mathsf{p}} \vdash P =_{\mathsf{p}} Q \;\Leftrightarrow\; [\![\Gamma_{\mathsf{p}}]\!] \vdash [\![P]\!] =_\pi [\![Q]\!].$$

Proof. By Theorem 7.10, we have only to prove ($\Rightarrow$) direction. Let us first construct the relation $\cong^-$ as the smallest typed relation including the following base equations and closed under parallel composition.

$$[\![\Gamma_{\mathsf{p}}]\!] \vdash [\![P]\!] \cong^- [\![Q]\!] \quad \text{if } \Gamma_{\mathsf{p}} \vdash P =_{\mathsf{p}} Q$$

Clearly $\cong^-$ is neither congruent nor reduction-closed. However if $\cong^-$ satisfies the following Claim, by Proposition 6.7, taking $=_s \,=\, =_\beta$ the congruence relation which is generated by $\cong^-$ is sound, which leads to the required result. Note: $\Gamma_0 \vdash P_1 \cong^- Q_2 \triangleright_A G$ implies $P_1 \equiv [\![P]\!] \mid R$ and $Q_1 \equiv [\![Q]\!] \mid R$ with $\Gamma_{\mathsf{p}} \vdash P =_{\mathsf{p}} Q$ and $[\![\Gamma_{\mathsf{p}}]\!] \subset \Gamma_0$, by induction of the derivations of $\cong^-$ (hereafter we often write such a situation just

$[\![\Gamma_{\mathrm{p}}]\!] \subset \Gamma_0 \vdash [\![P]\!]|R \cong^{-} [\![Q]\!]|R \rhd_A G$, understanding conditions on terms and types as just noted).

Claim:

(i) $\cong^{-}$ *is substitution closed.*

(ii) *whenever* $\Gamma_{\mathrm{p}} \vdash [\![P]\!] \mid R \;\; \cong^{-} \;\; [\![Q]\!] \mid R \rhd_A G$, $P \longrightarrow\!\!\!\!\to P_0$ *implies, for some* Q_0, $Q \longrightarrow\!\!\!\!\to Q_0$, $P_0 =_\beta C_r[[\![P']\!]]_{[\![\Gamma_{\mathrm{p}}]\!]}$, $\Gamma_{\mathrm{p}} \vdash P' =_{\mathrm{p}} Q'$ *and* $C_r[[\![Q']\!]]_{[\![\Gamma_{\mathrm{p}}]\!]} =_\beta Q_0$, *and the symmetric case.*

(iii) $\cong^{-}$ *respects the action predicate.*

(i) is easy by $\cong^{-} \supset \twoheadrightarrow_\beta$ and reasoning similarly as in Proposition 5.11.

Now we show $\cong^{-}$ satisfies the other two conditions in the above Claim. For reduction closure, it is enough to prove the case when $P_1 \equiv [\![P]\!]|R$ reduces by one step. Assume:

$$\Gamma_0 \vdash [\![P]\!] \mid R \longrightarrow P_0 \rhd_A G$$

Then there are three cases by Lemma 7.15.

(1) $P_0 \equiv [\![P]\!] \mid R'$ with $R \longrightarrow R'$.

(2) $P_0 \twoheadrightarrow_\beta [\![P']\!] \mid R$ with $P \rightarrow_{\mathrm{p}} P'$.

(3) $P_0 \twoheadrightarrow_\beta (\tilde{b})([\![P']\!] \mid R')$ with $[\![P]\!] \xrightarrow{[\![l]\!]} [\![P']\!]$ and $P \xrightarrow{l}_{\mathrm{p}} P'$, such that either:

(a) $l = a\langle v_1 v_2 ... v_n\rangle$ and $R \overset{[\![(\tilde{b})\overline{a}\langle v_1 v_2 ... v_n\rangle]\!]}{\Longrightarrow_\beta} R'$, or

(b) $l = (\tilde{b})\overline{a}\langle v_1 v_2 ... v_n\rangle$ and $R \overset{[\![a\langle v_1 v_2 ... v_n\rangle]\!]}{\Longrightarrow_\beta} R'$.

Case (1): Straightforward by taking $([\![Q]\!] \mid R) \longrightarrow ([\![Q]\!] \mid R')$.

Case (2): $\Gamma_{\mathrm{p}} \vdash \; P \rightarrow_{\mathrm{p}} P'$

$$\begin{array}{lll}
\Rightarrow & \exists\, Q'.\, \Gamma_{\mathrm{p}} \vdash \; Q \twoheadrightarrow_{\mathrm{p}} Q' \;\wedge\; \Gamma_{\mathrm{p}} \vdash \; P' =_{\mathrm{p}} Q' & (\text{soundness of } =_{\mathrm{p}}) \\
\Rightarrow & \Gamma_0 \vdash [\![Q]\!] \mid R \longrightarrow\!\!\!\!\to [\![Q']\!] \mid R & (\text{Prop.7.8(i)}) \\
\Rightarrow & P_0 \twoheadrightarrow_\beta C_r[[\![P']\!]] \;\wedge Q_0 \equiv C_r[[\![Q']\!]] & \text{with } C_r[\cdot] \overset{\mathrm{def}}{=} [\cdot] \mid R
\end{array}$$

For case (3-a), the following property of $=_{\mathrm{p}}$ is used: it says that the input observability is derived from sound theories (cf. Propositions 4.13 in [20]).

Claim (A): *Suppose* $\Gamma^0 \;\vdash\; P =_{\mathrm{p}} Q$. *Then, with some* $\Gamma \;\supset\; \Gamma^0$,

$$\Gamma \vdash P \overset{a\langle v_1 v_2 ... v_n\rangle}{\longrightarrow_{\mathrm{p}}} P' \;\;\Rightarrow\;\; \Gamma \vdash Q \;\; \overset{a\langle v_1 v_2 ... v_n\rangle}{\Longrightarrow_{\mathrm{p}}} \;\; Q' \textit{ for some } Q' \;\; \textit{with} \;\; \Gamma \vdash P' =_{\mathrm{p}} Q'. \quad (7.4)$$

Proof of Claim (A): Let us take the context $C[\cdot] \;\overset{\mathrm{def}}{=}\; [\cdot] \mid \overline{a}\langle v_1..v_n\rangle.f \mid \overline{f}$ with f fresh. First $\Gamma \vdash C[P] \longrightarrow^2 P'$ and $\neg P' \Downarrow_{f^\theta}$. Thus by definition of sound congruence, $\Gamma \vdash C[Q] \longrightarrow\!\!\!\!\to Q''$ and $Q'' =_{\mathrm{p}} P'$ with $\neg Q'' \Downarrow_{f^\theta}$. But if $\neg Q'' \Downarrow_{f^\theta}$, there should be a transition from Q such that $\Gamma \vdash Q \;\; \longrightarrow\!\!\!\!\to \overset{a\langle v_1 v_2 ... v_n\rangle}{\longrightarrow_{\mathrm{p}}} \longrightarrow\!\!\!\!\to \;\; Q'$. The closure property is proved with the same reasoning found in Theorem 3.19 in [20] using the incompatible pairs $\overline{f} \# \mathbf{0}$ and $(f \mid \overline{f}) \# \mathbf{0}$. (end of the proof of $\mathsf{Claim\ (A)}$). $\square$

With this claim the input case is easily obtained as follows.

Case (3-a): $l \ = \ a\langle v_1 v_2 ... v_n \rangle$.

$$\Gamma_{\mathsf{p}} \vdash \ P \overset{a\langle v_1v_2...v_n\rangle}{\longrightarrow_{\mathsf{p}}} P' \ \wedge \ R \overset{[\![(\tilde{b})\overline{a}\langle v_1v_2...v_n\rangle]\!]}{\Longrightarrow_{\beta}} R'$$

$$\begin{array}{lll} \Rightarrow & \exists\, Q'.\Gamma_{\mathsf{p}} \vdash \ Q \overset{a\langle v_1v_2...v_n\rangle}{\Longrightarrow_{\mathsf{p}}} Q' \ \wedge \ \Gamma_{\mathsf{p}} \vdash \ P' =_{\mathsf{p}} Q' & (\mathsf{Claim\ (A)}) \\ \Rightarrow & \Gamma_0 \vdash [\![Q]\!] \mid R \longrightarrow\!\!\!\!\to (\tilde{b})([\![Q']\!] \mid R') & \text{(Prop.7.8(i) and Lem.7.15(3-a))} \\ \Rightarrow & P_0 \longrightarrow\!\!\!\!\to_{\beta} C_r[[\![P']\!]]_{[\![\Gamma_{\mathsf{p}}]\!]} \ \wedge Q_0 \equiv C_r[[\![Q']\!]]_{[\![\Gamma_{\mathsf{p}}]\!]} & \text{with } \ C_r[\cdot] \ \overset{\text{def}}{=} \ (\tilde{b})([\cdot] \mid R') \end{array}$$

The final output case is the most non-trivial since we can not observe the value of the output label (cf. [20]). We also note that we will be often using *late* transition below and for the next claim, which is defined in the standard way, see e.g. [28].

Case (3-b): $l \ = \ (\tilde{b})\overline{a}\langle v_1 v_2 ... v_n\rangle$.

Let $C[\ ] \ \overset{\text{def}}{=} \ [\ ] \mid a(x_1...x_n).(\overline{c}\langle x_1...x_n\rangle \mid \overline{f}) \mid f$ with f, c fresh. W.l.o.g we can assume $\Gamma_{\mathsf{p}} \vdash \ C[P]_{\Gamma_{\mathsf{p}}} =_{\mathsf{p}} C[Q]_{\Gamma_{\mathsf{p}}}$. Then we have:

$$\Gamma_{\mathsf{p}} \vdash \ P \overset{(\tilde{b})\overline{a}\langle v_1v_2...v_n\rangle}{\longrightarrow_{\mathsf{p}}} P'$$

$$\Rightarrow \ \Gamma_{\mathsf{p}} \vdash \ C[P]_{\Gamma_{\mathsf{p}}} \longrightarrow^2 (\tilde{b})(P' \mid \overline{c}\langle v_1 v_2 ... v_n\rangle) \ \overset{\text{def}}{=} \ P'' \tag{7.5}$$

$$\Rightarrow \ \Gamma_{\mathsf{p}} \vdash \ C[Q]_{\Gamma_{\mathsf{p}}} \longrightarrow\!\!\!\!\to (\tilde{b'})(Q' \mid \overline{c}\langle v'_1 v'_2 ... v'_n\rangle) \ \overset{\text{def}}{=} \ Q'' =_{\mathsf{p}} P'' \tag{7.6}$$

with $\Gamma_{\mathsf{p}} \vdash \ Q \overset{(\tilde{b'})\overline{a}\langle v'_1v'_2...v'_n\rangle}{\Longrightarrow_{\mathsf{p}}} Q'$ because $\neg Q'' \Downarrow_{f^\theta}$ and $Q'' \Downarrow_{c^-}$. Then from (7.5) we have:

$$\Gamma_0 \vdash [\![P]\!] \mid R \longrightarrow P_0 \longrightarrow\!\!\!\!\to_{\beta} (\tilde{b})([\![P']\!] \mid R''\{\tilde{v}/\tilde{x}\}) \rhd_A G \text{ for some } R \overset{[\![a\langle v_1...v_n\rangle]\!]}{\Longrightarrow_{\beta}} R''\{\tilde{v}/\tilde{x}\}$$

by again Lemma 7.15 (3-b) where $R \overset{[\![a(x_1...x_n)]\!]}{\Longrightarrow_{\beta}} R''$ in the late transition relation [28]. Suppose $v_i : \alpha_i \in \Gamma_{\mathsf{p}}$. Then there exists a transition $R \overset{[\![a\langle v'_1...v'_n\rangle]\!]}{\longrightarrow} R''\{\tilde{v}'/\tilde{x}\}$ with $v'_i\!: \alpha_i \in \Gamma_{\mathsf{p}}$. Thus from (7.6), we have

$$\Gamma_0 \vdash [\![Q]\!] \mid R \longrightarrow\!\!\!\!\to (\tilde{b'})([\![Q']\!] \mid R''\{\tilde{v}'/\tilde{x}\}) \rhd_A G \text{ with } R \overset{[\![a\langle v'_1...v'_n\rangle]\!]}{\Longrightarrow_{\beta}} R''\{\tilde{v}'/\tilde{x}\}$$

by Lemma 7.15 (3-b) and Proposition 7.8 (i). Then, by noticing c occurs as only one positive and one negative active name, we have the following sequence of relations:

$$\begin{array}{lllll} [\![P]\!] \mid R & \longrightarrow & P_0 & \longrightarrow\!\!\!\!\to_{\beta} & (\tilde{b})([\![P']\!] \mid R''\{\tilde{v}/\tilde{x}\}) \quad {}_{\beta}\!\leftarrow\!\!\!\!\leftarrow \quad (c)([\![P'']\!] \mid c^*(x_1x_2...x_n).R'') \\ [\![Q]\!] \mid R & \longrightarrow\!\!\!\!\to & Q_0 & \equiv & (\tilde{b'})([\![Q']\!] \mid R''\{\tilde{v}'/\tilde{x}\}) \quad {}_{\beta}\!\leftarrow\!\!\!\!\leftarrow \quad (c)([\![Q'']\!] \mid c^*(x_1x_2...x_n).R'') \end{array}$$

Taking $C_r[\cdot]_{[\![\Gamma_{\mathsf{p}}]\!]} \overset{\text{def}}{=} (c)([\cdot] \mid c^*(x_1x_2...x_n).R'')$, we are done with $[\![\Gamma_{\mathsf{p}}]\!] \vdash \ [\![P'']\!] \cong^- [\![Q'']\!]$ by $\Gamma_{\mathsf{p}} \vdash \ P'' =_{\mathsf{p}} Q''$ in (7.6).

For the action predicate, assume

$$[\![\Gamma_{\mathsf{p}}]\!] \subset \Gamma_0 \vdash ([\![P]\!]|R) \cong^{-} ([\![Q]\!]|R) \triangleright_A G \tag{7.7}$$

with $\Gamma_{\mathsf{p}} \vdash P =_{\mathsf{p}} Q$. Notice, if $\Gamma_0 \vdash ([\![P]\!]|R) \triangleright_A G \Downarrow_{a^\theta}$ with $a \in \mathsf{fn}(G) \setminus A$, then by Proposition 5.12 we immediately know $\Gamma_0 \vdash ([\![Q]\!]|R) \triangleright_A G \Downarrow_{a^\theta}$. Hence we are only concerned at the observable at Γ_0. For the purpose, we establish the following claim.

Claim (B): *Assume (7.7). There is a well-sorted polyadic term $\Gamma_{\mathsf{p}} \subset \Gamma'_0 \vdash R_0$ with $\mathsf{fn}(\Gamma'_0) = \mathsf{fn}(\Gamma)$ such that, for any $\Gamma_{\mathsf{p}} \vdash P'$, we have:*

(i) *If $\Gamma_0 \vdash ([\![P']\!]|R) \Downarrow^m_{a^\theta} \triangleright_A G \ \wedge\ a \in \mathsf{fn}(\Gamma_0)$ then $\Gamma'_0 \vdash P'|R_0 \Downarrow_{a^\theta}$ where $\Downarrow^m_{a^\theta}$ means, after m reduction steps, one reaches an immediate observable a with polarity θ.*

(ii) *If $\Gamma'_0 \vdash P'|R_0 \Downarrow_{a^\theta}$ then $\Gamma_0 \vdash P'|R_0 \Downarrow_{a^\theta}$.*

Proof of Claim (B): We construct such R_0 indexed by some n, called an *n-simulator*, which simulates R in the above way up to the n transition sequences R may be engaged in, by the following procedure. We use a sequence of prefixes, like $a(x_1..x_n)(v_iv_j)\overline{b}\langle v_1..v_n\rangle$ (v_iv_j shows the bound output), ranged over $s, s_1, \ldots$. Each prefix itself is written $p, p', \ldots$. The null sequence is denoted ε. Notice $s.\mathbf{0}$ gives a polyadic π-term.

(1) Unfold R_0's replications n-times, and let the term obey the binding condition. By this note any τ-transition from R_0 does not need α-conversion until n-steps.

(2) We use three sets, Ξ_0, Ξ_1, Ξ_2. An element of the first one is essentially a tuple of prefix sequence together with an intermediate term which arises as a result of the transitions corresponding to the prefix sequence. An element of the second one is a triple of a prefix sequence, an intermediate term, and a name z which is being used by R for communication via a name with a type of encoded sorting. The third contains completed prefix sequences.

(3) We initially set $\Xi_0 = \{\langle \varepsilon,\ \Gamma_0 \vdash R \triangleright_A G\rangle\}$, and $\Xi_1 = \Xi_2 = \emptyset$. This is the state for $n = 0$.

(4) Let $n = m$ and Ξ_0, Ξ_1, Ξ_2 are given. We use auxiliary sets Ξ'_0, Ξ'_1, Ξ'_2 which are initially given by: $\Xi'_0 = \Xi'_1 = \emptyset$ and $\Xi'_2 = \Xi_2$.

(5) For each $\langle s,\ \Gamma' \vdash R' \triangleright_{A'} G'\rangle \in \Xi_0$, we do the following.

(1) For each free active occurrence of a in R' where $a \in \mathsf{fn}(\Gamma')$ and which, in addition, has the type of encoded sorting, we first let R' make a *late* transition corresponding to that active occurrence, say $R' \xrightarrow{l}_l R''$, with $\Gamma' \vdash R'' \triangleright_{A'} G''$ (note G'' is the graph union of G' and the tail of the graph type for the sorting). Now we get the continuation for polyadic name passing (z in $\updownarrow(z) \to G$), and track the binding structure (for input) or the vector of output names (for output) analysing the term structure of R'' (which is possible since cuts are always finite), thus obtaining the corresponding polyadic prefix, say p, and then let $\langle s \cdot p,\ z,\ \Gamma' \vdash R'' \triangleright_{A'} G'\rangle$ be added to Ξ'_1.

(2) For each free active occurrence of a in R' such that $a \in \mathsf{fn}(\Gamma_0)$ (note we go back to Γ_0 here) and which does not have the type of encoded sorting, if it is input we add $s \cdot a(\varepsilon)$ into Ξ'_2, if it is output we add $s \cdot \overline{a}\langle\varepsilon\rangle$ into Ξ'_2.

(3) For each redex in R', let $\Gamma' \vdash R' \longrightarrow R'' \triangleright_{A'} G'$ be the corresponding reduction, then we put the new term into Ξ'_0. If neither of (1)(2)(3) is possible, then we put s into Ξ'_2.

(6) For each $\langle s,\ z,\ \Gamma' \vdash R' \triangleright_{A'} G' \rangle \in \Xi_1$, we do the following.
(1) If there is a free active z in R', then there is a corresponding late transition, say $R' \xrightarrow{l}_l R''$, and correspondingly a new name with type of the encoded sorting may be added to Γ', resulting in Γ'' (if not $\Gamma'' = \Gamma'$), while one head of G' will be gone, resulting in G''. If z still occurs in R' (showing it has not finished that part of interaction) then $\langle s,\ z,\ \Gamma'' \vdash R'' \triangleright_{A'} G'' \rangle$ is added to Ξ'_1. If z does not occur, then $\langle s,\ \Gamma'' \vdash R'' \triangleright_{A'} G'' \rangle$ is added to Ξ'_0.
(2) If there is no free active z in R', there should be some β-reduction. For each β-reduction $\Gamma' \vdash R' \longrightarrow R'' \triangleright_{A'} G'$, we put $\langle s,\ z,\ \Gamma' \vdash R'' \triangleright_{A'} G' \rangle$ to Ξ'_1.

(7) Finally we set $\Xi_i = \Xi'_i$ for $i = 0, 1, 2$ and n becomes $n + 1$.

Now let Ξ_0, Ξ_1, Ξ_2 are the above sets with $n = m + 1$. Then an *m-simulator* is given by extracting all prefix sequences from elements of these sets, and postfix $\mathbf{0}$ to each to get polyadic terms (say $P_0, ..P_k$), and define:

$$R_0 \overset{\text{def}}{=} \bigoplus_{0 \le i \le k} P_k$$

where the right hand side is given by $(c)(\prod_{i \le k} c(\varepsilon).P_k \mid \overline{c}\langle\varepsilon\rangle)$ with c fresh. We then sort the term as:

$$\Gamma'_0 \vdash R_0$$

where $\mathsf{fn}(\Gamma'_0) = \mathsf{fn}(\Gamma_0)$ and (1) for a such that $\Gamma_0(a)$ is the encoding of sorting, we have $\Gamma'_0(a) = \Gamma_0(a)$, while (2) for a such that $\Gamma_0(a)$ is not the encoding of sorting, then $\Gamma'_0(a) = (\varepsilon)$. This is clearly well-sorted. Now using Lemma 7.15 repeatedly, we can check, by induction on the construction of R_0, at least up to m-steps of (monadic) transitions of R, $[\![R_0]\!]$ can simulate all transitions of R as far as the interaction with $[\![P_0]\!]$ and observability at Γ_0 go. But this means $[\![P]\!]|R$ has more than m steps of monadic transitions, hence we have the clause (i) of Claim (B). Moreover clearly the transition traces of $[\![R_0]\!]$ is smaller than R (by induction on n we can verify, for each visible transition of $[\![R_0]\!]$, there is the corresponding transition of R). This gives the clause (ii) of Claim (B). (end of the proof of **Claim (B)**). □

We now prove that $\cong^-$ respects the action predicates. Suppose $\Gamma_0 \vdash [\![P]\!]|R \Downarrow_{a^\theta} \triangleright_A G$ with $a \in \Gamma_0$, taking n reduction steps to reach the immediate observable. Take the n'-simulator of $\Gamma'_0 \vdash R$ with any $n' \gneq n$ (actually $n' = n$ suffices). Then by the clause (i) of Claim (B) we know $\Gamma'_0 \vdash P'|R_0 \Downarrow_{a^\theta}$. But because $\Gamma'_0 \vdash P' =_{\mathsf{p}} Q'$, we know $\Gamma'_0 \vdash Q'|R_0 \Downarrow_{a^\theta}$. By Proposition 7.8 (iii) this implies $[\![\Gamma'_0]\!] \vdash [\![Q']\!]|[\![R_0]\!] \Downarrow_{a^\theta}$. Hence by the clause (ii) of Claim (B) we know $\Gamma_0 \vdash [\![Q']\!]|R \Downarrow_{a^\theta}$. This shows that two conditions of Lemma 6.7 are satisfied by $\cong^-$, hence it is in $=_\pi$, concluding the proof. □

Now we will briefly touch upon some extensions. Our type framework is general enough to apply to more complex operational structures, cf. Example 2.4 (ii) and (iii).

7.17. **Mixed Input/Output Interaction.** The abstract syntax of the calculus (which is a subset of [11]) is given by replacing input and output agents with the following terms (for notation, see 2.4 (ii)):

$$a:(\tilde{x}_i)[\tilde{v}_{i+1}]_{i\le 2n}.P \qquad\qquad \overline{a}:[\tilde{x}_i](\tilde{v}_{i+1})_{i\le 2n}.P$$

where $\tilde{x}_i = x_{i1}...x_{ik_i}$ with $k_i, n \ge 0$ and we assume all binding names are pairwise distinct and disjoint from free names. In IO interaction types, $\downarrow S$ (resp. $\uparrow S$) represents a type which receives (resp. emits) S, while $\updownarrow S; \updownarrow S'$ denotes the sequential composition of $\updownarrow S$ and $\updownarrow S'$. We use abbreviation $\downarrow\tilde{S}_i \stackrel{\text{def}}{=} \downarrow S_{i1};...;\downarrow S_{ik_i}$ and similarly for $\uparrow\tilde{S}_i$ with $k_i \ge 0$, where we set $\emptyset;\updownarrow S = \updownarrow S;\emptyset = \updownarrow S$. Then the set of *IO interaction types* $\mathbf{I}$ $(S, S', \dots)$, is given by following grammar:

$$S ::= \mathbf{x} \;\;|\;\; \{\downarrow\tilde{S}_1;\uparrow\tilde{S}_2;...;\downarrow\tilde{S}_{2n-1};\uparrow\tilde{S}_{2n}\;,\;\uparrow\tilde{S}_1;\downarrow\tilde{S}_2;...;\uparrow\tilde{S}_{2n-1};\downarrow\tilde{S}_{2n}\}$$

with $n \le 0$. Note the sorting is the special mixed interaction type such that: $\{\downarrow \tilde{S}_1,\ \uparrow\tilde{S}_1\}$. The reduction relation is given by replacing (COM) rule in 2.3 with the following rule, where we assume the whole term obeys the binding condition.

$$\begin{aligned} &a:(\tilde{x}_i)[\tilde{v}_{i+1}]_{i\le 2n}.P \mid \overline{a}:[\tilde{w}_i](\tilde{y}_{i+1})_{i\le 2n}.Q \\ &\qquad\qquad\longrightarrow (P \mid Q)\{\tilde{v}_{2n}/\tilde{y}_{2n}\}\{\tilde{w}_{2n-1}/\tilde{x}_{2n-1}\}...\{\tilde{v}_2/\tilde{y}_2\}\{\tilde{w}_1/\tilde{x}_1\}\end{aligned}$$

with $|\tilde{w}_i| = |\tilde{x}_i|$ and $|\tilde{v}_i| = |\tilde{y}_i|$. Note the sequence of substitution is reversed to represent receiving and sending the same name at one time, e.g. $a:(x_1)[x_1].P$. The typing system and labeled transition relation are easily defined by "mixing" input/output rules (cf. 4.2 in [11]). This system also has the subject reduction property (cf. Theorem 4.3 in [11]). We use the encoding in 2.4 (ii) for the mapping of terms. Based on this, the mapping of type $[\![\ ]\!]_{\rm i}$ is given as:

$$\{\downarrow(x) \to x\downarrow[\![\tilde{S}_1]\!]_{\rm i} \to x\uparrow[\![\tilde{S}_2]\!]_{\rm i}\cdots \to x\uparrow[\![\tilde{S}_{2n}]\!]_{\rm i},\ \uparrow(x) \to x\uparrow[\![\tilde{S}_1]\!]_{\rm i} \to x\downarrow[\![\tilde{S}_2]\!]_{\rm i}\cdots \to x\downarrow[\![\tilde{S}_{2n}]\!]_{\rm i}\}$$

where $x\updownarrow[\![\tilde{S}_i]\!]_{\rm i} \stackrel{\text{def}}{=} x\updownarrow[\![S_{i1}]\!]_{\rm i} \to x\updownarrow[\![S_{i2}]\!]_{\rm i}\cdots \to x\updownarrow[\![S_{ik_i}]\!]_{\rm i}$. Now defining $=_{\rm i}$ as the maximum sound congruence (cf. Definition 5.2), we have:

$$\Gamma_{\rm i} \vdash P =_{\rm i} Q \;\;\Leftrightarrow\;\; [\![\Gamma_{\rm i}]\!]_{\rm i} \vdash [\![P]\!]_{\rm i} =_\pi [\![Q]\!]_{\rm i}.$$

By the same operational correspondence as Proposition 7.8, $\Rightarrow$ is immediately obtained. For the $\Leftarrow$ direction, by Proposition 6.7, we prove the same claim together with Lemma 7.15 using the label and its mappings. But by Proposition 5.15, we only have to replace one-way name passing with IO interaction in order to obtain the same previous propositions and lemmas: e.g. in Proposition 7.12, we replace $x'\downarrow[\![S_1]\!]\cdots \to x'\downarrow[\![S_{n-1}]\!] \to x'\downarrow[\![S_n]\!]$ with

$$\Gamma_0 \vdash P'' \rhd x'\downarrow[\![\tilde{S}_1]\!]_{\rm i} \to x'\uparrow[\![\tilde{S}_2]\!]_{\rm i}\cdots \to x'\downarrow[\![\tilde{S}_{2n-1}]\!]_{\rm i} \to x'\uparrow[\![\tilde{S}_{2n}]\!]_{\rm i}$$

7.18. **Parallel Name Passing.** Now we show graph types make it possible to type-abstract the following *non-sequential protocols*. The abstract syntax of the calculus is given by replacing input and output agents with the following terms (for notation, see 2.4 (iii)):

$$a\!:\!\otimes_n(\tilde{x}_i).P \qquad\qquad \overline{a}\!:\!\otimes_n\langle\tilde{x}_i\rangle.P$$

with $n \geq 0$ and we assume all binding names are pairwise distinct and disjoint from free names. Then the set of *parallel types* $\mathbf{P}$ $(S, S', \dots)$ is simply given as $S ::= \mathbf{x} \mid \otimes_n\tilde{S}_i$ with $n \leq 0$. The sorting is the case $n = 1$. The reduction relation is given as:

$$(\mathrm{COM}_\otimes) \quad a\!:\!\otimes_n(\tilde{x}_i).P \mid \overline{a}\!:\!\otimes_n\langle\tilde{v}_i\rangle.Q \;\longrightarrow\; P\tilde{\sigma} \mid Q$$

where $\tilde{\sigma}$ is a sequence of substitutions $\sigma_i = \{\tilde{v}_i/\tilde{x}_i\}$ with $|\tilde{x}_i| = |\tilde{v}_i|$ and $1 \leq i \leq m$ (actually the ordering does not matter). Then the typing system/labeled transition relation are defined just by replacing $a : \tilde{S}$ with $a : \otimes_n\tilde{S}$. We use Example 2.4 (iii) for the mapping of terms. Now we translate its types $[\![\otimes_n \tilde{S}_i]\!]_\otimes$ to "graph" types $\{\lambda a.G,\ \lambda a.\overline{G}\}$ where

$$\begin{tikzcd}
G \overset{\mathrm{def}}{=} a\!\downarrow\!(c) \arrow[r] & c\!\downarrow\!(\tilde{c_i}) \arrow[r] \arrow[ddr] & c_1\!\downarrow\![\![S_{11}]\!]_\otimes & \cdots \arrow[r] & c_1\!\downarrow\![\![S_{1k_i}]\!]_\otimes \\
 & & & \vdots & \\
 & & c_n\!\downarrow\![\![S'_{n1}]\!]_\otimes & \cdots \arrow[r] & c_n\!\downarrow\![\![S'_{nk_n}]\!]_\otimes
\end{tikzcd}$$

with $\tilde{c} \overset{\mathrm{def}}{=} c_1...c_n$ fresh and distinct. Note the above mapping to graph types shows that it does not matter to change the order of prefixes of c_i and c_j in the mapping $a\!:\!\otimes_n(\tilde{x}_i).P$, as mentioned in Remark 2.5. The difference from the previous encodings of calculi with sequential types is n-redex pairs appear at once during β-reduction between $[\![P]\!]_\otimes$ and another typed agent (i.e. R in Lemma 7.15). By defining the mapping $[\![l]\!]$ by a function from a label l to a set of sequences of labels $\{\tilde{l}_1, \tilde{l}_2, .., \tilde{l}_m\}$ in which we preserve the ordering between the same subject names, we can use the same proof technique developed in this section because the basic properties studied in Section 5 (Prop.5.12,5.15 and Col.5.14) do not depend on sequential graph types. Finally we can establish the full abstraction result.

We can combine the mixed IO interaction and parallel (multi) name passing, and show the fully abstraction results, though we omit the formulation of the superset calculus and its mappings.

8. Discussion

8.1. **Related Works.** While there have been many studies on types for mobile processes based on sorting [26, 7, 36, 11, 43, 41, 24, 22, 14, 42], as far as we know, the present work is the first one which captures the structure of protocol construction as types and establishes the full abstraction results for the basic encoding such as polyadic name passing.

(types with sequencing): A study to give a type-abstraction in the monadic asynchronous π-calculus with the same aim had been tried by Honda in 1992 [10], whose idea is reflected to the present construction to some extent. In that moment, however, his formulation (especially binding construction) was very complex. Independently, Pierce discovered the way to type-check a class of encodings of polyadic name passing in (a variant of) monadic π-calculus in which two local names are used only for input and output, respectively [35]. The differences are (1) a name can only carry one kind of types so typable terms are restricted, and (2) however the length of sequence can be indefinite. In both studies, the basic results as we established here, such as the subject reduction theorem and the full abstraction, are not presented.

(graphs): Representation of processes in the graphical (or geometric) framework gives us another grasp of name passing from the different viewpoint [13, 27, 31, 34, 46]. Our aim is to capture process behaviour as "types" and to obtain full abstraction, which is different from these works.

(linearity in processes): The notion of linearity is studied by Kobayashi and others [22] in Pict language and by [14] in an algebraic framework, though dynamic communication structures are not captured. In the former, the idea of "linearizability," based on the above mentioned Pierce's work, was proposed as an extension of their linear channel types. However structures of polyadic name passing or of (1.2) may not be typable when a name carries different types. Another difference is our explicit treatment of name transmission in a type, which we believe to be important for the kind of results we have obtained here.

(linear logic): Some ideas of Linear Logic [9] and Interaction Categories [1, 2, 8] are related (e.g. "multi-cuts" in [8]). Prasad [37] studies a term assignment to a generalisation of linear logic, where the distinction between "linear" and "classical" realms exists, which is also related.

(true concurrency): The causality of communication in process calculi has been studied in the setting of true concurrency, cf. [45]. The order relation in our safe graph is a concise way to encapsulate linear interaction behaviour rather than to describe general dependency among communication events. To capture more complex causality structure (such as labeled event structure in [45]) may be one possibility, though we may lose a simple combinatorial expression as we have in the present paper.

8.2. **Extensions and Further Issues.** There are several possible extensions and further issues of the present typing system, which we outline below.

(recursive type): For simplicity of presentation we do not form recursive protocol types. Its incorporation is standard by introducing the expressions $\mu\mathbf{x}.\alpha$ and consider them modulo their tree unfoldings, cf.[43]. Immediately all results in the paper hold with this extension, with the corresponding set of polyadic terms extended with recursive sortings.

(full abstraction): There are many significant and useful encodings in (polyadic) π-calculus in which full abstraction results have not been known [25, 21, 44, 32]. This framework with extension [47] will be able to analyse and justify such non-trivial problems on translations [21, 25].

(asynchronous π and its combinators): In this paper we have exclusively dealt with graph types for synchronous monadic π-calculus. It is however possible to extend the present type discipline so that we can type asynchronous monadic mobile processes and their combinators [4, 12, 17, 18] (cf. [33, 38]), which is smaller than the present system but as expressive. Because there is no direct sequencing in output, the same notion of "safety condition" cannot be used. However, an extension of the notion of graph types and the safety condition can be done so that we can form a typing system for the asynchronous calculus along the same line as in Section 5. This will be treated in the sequel to present paper [47].

(other typing constructs): The construction of present types can be combined with various refinements on sorting studied by e.g. in [36, 11, 41, 24, 22, 14, 42] in either monadic or polyadic setting, which will be an interesting subject of study.

(algorithm): The present system does not induce the typing algorithm because a name can be used in several ways in linear space and classical space (cf. [22]). In this context, finding some analogue of the principal type schemes would be an interesting subject of study.

(general protocol types): It may be interesting to consider extensions such as a general name abstraction of graph types, pointed types with two or more heads (cf. [6]), and with a non-dual pairs (cf. [24]).

(semantics): Apart from practical viewpoints, one of the most essential issues for further study is to seek the canonical notion of semantics of types in process calculi, like "arrow types" well studied in λ-calculi and combinators [30]. Since our framework suggests one way to build up a significant computational causal chains in the basic π-calculus like functional types, we hope this work becomes a stepping stone in this line of research.

Acknowledgments. The author is indebted to Kohei Honda for his invaluable suggestion and continuous discussions on this work. Naoki Kobayashi provided helpful comments on the early version. She thanks Samson Abramsky, Cliff Jones, Benjamin Pierce, Kenjiro Taura and the members of Interaction Club at Edinburgh for their discussions, and Mario Tokoro for his encouragements. She is partially supported by JSPS Fellowship for Japanese Junior Scientists and Ogata-jyosei grant of Keio University.

References

[1] Abramsky, S., Computational interpretations of linear logic. TCS, 111(1-2):3-57, 1993.

[2] Abramsky, S., Gay, S. and Nagarajan, R., Interaction Categories and Foundations of Typed Concurrent Computing. *Deductive Program Design*, Springer-Verlag, 1995.

[3] Berry, G. and Boudol, G., The Chemical Abstract Machine. TCS, vol 96, pp. 217–248, 1992.

[4] Boudol, G., *Asynchrony and π-calculus*. INRIA Report 1702, INRIA, Sophia Antipolis, 1992.

[5] Davey, B.A. and Priestley, H.A., *Introduction to Lattices and Order*, CUP, 1990.

[6] Fournet, C. and Gonthier, G., The reflexive CHAM and the join-calculus, *POPL'96*, pp.372–385, ACM Press, 1996.

[7] Gay, S., A Sort Inference Algorithm for the Polyadic π-Calculus. *POPL'93*, ACM Press, 1993.

[8] Gay, S. and Nagarajan, R., A Typed Calculus of Synchronous Processes. *LICS'95*, IEEE, pp.210–220, 1995.

[9] Girard, J.-Y., Linear Logic, *TCS*, Vol. 50, pp.1–102, North-Holland, 1987.

[10] Honda, K., Pre-types in mobile processes, *a manuscript*, 1992.

[11] Honda, K., Types for Dyadic Interaction. *CONCUR'93*, LNCS 715, pp.509–523, Springer-Verlag, 1993.

[12] Honda, K., *A Study of ν-calculus and its Combinatory Representation*, Phd Thesis in the Department of Computer Science, October, 1994.

[13] Honda, K., *Notes on P-Algebra (1): Process Structure. Proc. TPPP'94*, LNCS 907, pp.25–44, Springer-Verlag, 1995.

[14] Honda, K., Composing Processes, *POPL'96*, pp.344-357, ACM Press, 1996.

[15] Honda, K., An Object Calculus for Asynchronous Communication. To appear as LFCS report, 1996.

[16] Honda, K. and Tokoro, M., An Object Calculus for Asynchronous Communication. *ECOOP'91*, LNCS 512, pp.133–147, Springer-Verlag 1991.

[17] Honda, K. and Yoshida, N., On Reduction-Based Process Semantics. *FSTTCS'13*, LNCS 761, pp. 373–387, Springer-Verlag, December 1993.

[18] Honda, K. and Yoshida, N., Combinatory Representation of Mobile Processes. *POPL'94*, pp.348–360, ACM Press, 1994.

[19] Honda, K. and Yoshida, N., Replication in Concurrent Combinators, *TACS'94*, LNCS 789, pp.786–805, Springer, 1994.

[20] Honda, K. and Yoshida, N., On Reduction-Based Process Semantics. Full version of [17], *TCS*, pp.437–486, No.151, North-Holland, December, 1995.

[21] Jones, C.B., *Process-Algebraic Foundations for an Object-Based Design Notation.* UMCS-93-10-1, Computer Science Department, Manchester University, 1993.

[22] Kobayashi, N., Pierce, B., and Turner, D., Linear Types and π-calculus, *POPL'96*, pp.358–371, ACM Press, 1996.

[23] Lafont, Y., Interaction Nets, *POPL'90*, pp. 95–108, ACM press, 1990.

[24] Liu, X. and Walker, D., A polymorphic type system for the polyadic pi-calculus, CONCUR'95, LNCS, Springer-Verlag, 1995.

[25] Milner, R., Functions as Processes. *Mathematical Structure in Computer Science*, 2(2), pp.119–146, 1992.

[26] Milner, R., Polyadic π-Calculus: a tutorial. *Logic and Algebra of Specification*, Springer-Verlag, 1992.

[27] Milner, R., Action structure for the π-calculus. Research Report ECS-LFCS-93-264, Department of Computer Science, University of Edinburgh 1993.

[28] Milner, R., Parrow, J.G. and Walker, D.J., A Calculus of Mobile Processes, *Information and Computation* 100(1), pp.1–77, 1992.

[29] Milner, R. and Sangiorgi, D., Barbed Bisimulation. *Proc. of ICALP'92*, LNCS 623, pp.685–695, Springer-Verlag, 1992.

[30] Mitchell, J., Type Systems for Programming Languages. *Handbook of Theoretical Computer Science* B, pp.367–458, MIT Press, 1990.

[31] Montanari, U. and Pistore, M., Concurrent Semantics for π-calculus, *MFCS'95* and ENTCS, Vol. 1. Elsevier, 1995.

[32] Odersky, M., Applying π: Towards a basis for concurrent imperative programming, *2nd. SIPL*, pp.95–108, 1995.
[33] Odersky, M., Polized Name Passing. *FSTTCS'15*, LNCS, Springer-Verlag, 1995.
[34] Parrow, J.G., Interaction Diagrams, *REX'93.*, LNCS, Springer-Verlag, 1993.
[35] Pierce, B.C., Linearized Types for the π-calculus, *a type script*, December, 1994.
[36] Pierce, B.C. and Sangiorgi. D, Typing and subtyping for mobile processes. *LICS'93*, pp.187–215, 1993.
[37] Prasad, S., *Towards a Formulae-as-Types View of Communicating Applicative Processes*, Technical report ECRC-94-32, ECRC, 1994.
[38] Raja, N. and Shyamasundar, R.K., Combinatory Formulations of Concurrent Languages, pp. 156–170, ACSC'95, LNCS 1023, 1995.
[39] Sangiorgi, D., *Expressing Mobility in Process Algebras: First Order and Higher Order Paradigms*. Ph.D. Thesis, University of Edinburgh, 1992.
[40] Sangiorgi, D., On the Bisimulation Proof Method, LFCS report, ECS-LFCS-94-299, University of Edinburgh, 1994.
[41] Takeuchi, K., Honda, K. and Kubo, M., An Interaction-based Language and its Typing System. *PARLE'94*, LNCS 817, pp.398–413, Springer-Verlag, 1994.
[42] Turner, D., The π-calculus: Types, polymorphism and implementation, Phd Thesis, University of Edinburgh, 1996.
[43] Vasconcelos, V. and Honda, K., Principal Typing Scheme for Polyadic π-Calculus. *CONCUR'93*, LNCS 715, pp.524-538, Springer-Verlag, 1993.
[44] Walker, D., Objects in the π-calculus. *Information and Computation*, Vol. 116, pp.253–271, 1995.
[45] Winskel, G., An Introduction to Event Structures. *In Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, LNCS 354, pp. 364–397,
[46] Yoshida, N., Graph Notation for Concurrent Combinators, *TPPP'94*, LNCS 907, pp.393–412, Springer-Verlag, 1995.
[47] Yoshida, N., Graph Types for Mobile Process Calculi II and III. To appear as CS technical reports, Keio University.
[48] Yoshida, N., A Study of Behavioural Semantics for Concurrent Calculi, Forthcoming PhD Thesis, Department of Computer Science, Keio University, October, 1996.

Appendix A. Proof of Lemma 4.12

We start from two lemmas: the first clause of the next one is needed for the proof of Lemma A.2 (ii), while the second one is for Lemma A.4. In the following, we use a notation: let us define the set of *free subjects* of P denoted by $\mathsf{fs}(P)$ inductively as follows.

$$\begin{array}{lll} \mathsf{fs}(\mathbf{0}) = \emptyset & \mathsf{fs}(\overline{a}v.P) = \{a\} \cup \mathsf{fs}(P) & \mathsf{fs}(ax.P) = \{a\} \cup (\mathsf{fs}(P) \backslash \{x\}) \\ \mathsf{fs}((a)P) = \mathsf{fs}(P) \backslash \{a\} & \mathsf{fs}(P \mid Q) = \mathsf{fs}(P) \cup \mathsf{fs}(Q) & \mathsf{fs}(!P) = \mathsf{fs}(P) \end{array}$$

A.1. **Lemma.** *Suppose* $\Gamma' \vdash P_1 \mid P_2 \triangleright_{A'} G$ *is derived by applying a sequence of zero or more (weak) rules and (par) rule like the following.*

$$\dfrac{\begin{array}{cc} \Gamma_1 \vdash P_1 \triangleright_{A_1} G_1 & \Gamma_2 \vdash P_2 \triangleright_{A_2} G_2 \\ \vdots\ (\text{weak}) \times m & \vdots\ (\text{weak}) \times n \\ \Gamma \vdash P_1 \triangleright_{A'_1} G_1 & \Gamma \vdash P_2 \triangleright_{A'_2} G_2 \end{array}}{\begin{array}{c} \Gamma \vdash P_1 \mid P_2 \triangleright_{A'_1 \cup A'_2 \cup A} G_1 \odot G_2 \\ \vdots\ (\text{weak}) \times k \\ \Gamma' \vdash P_1 \mid P_2 \triangleright_{A'} G_1 \odot G_2 \end{array}} \ (\text{par})$$

Then we have:

(i) $a \in \mathsf{fs}(P_1) \cap \mathsf{fs}(P_2)$ *implies either* $a \in A'$ *or* $a \in \mathsf{fn}(\Gamma')$*. Moreover, if* $a \in A'$*, there is only one prime term* $a \in \mathsf{fs}(P_{1i})$ *in* P_1 *and also only one prime term* $a \in \mathsf{fs}(P_{2i})$ *in* P_2*.*

(ii) $a \in \mathsf{fn}(\Gamma')$ *implies* $a \notin \mathsf{fn}(G_1) \cup \mathsf{fn}(G_2) \cup A_1 \cup A_2$*.*

Proof. For the first clause in (i), we show $a \in \mathsf{fs}(P_1) \cap \mathsf{fs}(P_2) \ \Rightarrow \ a \notin A_1 \cup A_2 \cup \mathsf{fn}(G_1 \odot G_2)$. $a \notin \mathsf{fn}(G_1 \odot G_2)$ is obvious by the definition of cut function. Then we know $a \in A_1 \ \Rightarrow \ a \notin A_2 \wedge a \notin \mathsf{fn}(G_2) \wedge a \notin \mathsf{fn}(\Gamma)$. But $\mathsf{fn}(P_2) \subset \mathsf{fn}(\Gamma) \cup A_2 \cup \mathsf{fn}(G_2)$. Therefore $a \notin \mathsf{fn}(P_2)$, a contradiction. Similarly for the case $a \in A_2$, hence done.

(ii) is easy because if $a \in \mathsf{fn}(\Gamma)$, then $a \notin A'$ nor $a \notin \mathsf{fn}(G_1 \odot G_2) = (\mathsf{fn}(G_1) \cup \mathsf{fn}(G_2)) \backslash (\mathsf{fn}(G_1) \cap \mathsf{fn}(G_2))$, which implies $a \notin A_1 \cup A_2$ because $A' \supset A_1 \cup A_2$. Moreover, $a \notin (\mathsf{fn}(G_1) \cap \mathsf{fn}(G_2)) = A$ because $A' \supset A$, hence the result. □

The following lemma is necessary for the proof of (iv) of Lemma 4.12.

A.2. **Lemma.** *Suppose* $\Gamma \vdash (P|Q)|R \triangleright_A G$ *is derived from* $\Gamma \vdash P \triangleright_{A_1} G_1$, $\Gamma \vdash Q \triangleright_{A_2} G_2$ *and* $\Gamma \vdash R \triangleright_{A_3} G_3$ *with* $G \equiv_\alpha ((G_1 \odot G_2) \odot G_3)$*. Then we have:*

(i) $\mathsf{fn}(G_1) \cap \mathsf{fn}(G_2) \cap \mathsf{fn}(G_3) = \emptyset$.

(ii) $\Gamma \vdash P \mid (Q \mid R) \triangleright_A G$ *is proved.*

Proof. For (i), suppose $\Gamma \vdash (P \mid Q) \triangleright_{A_1 \cup A_2 \cup A'} G'$ is derived from $\Gamma \vdash P \triangleright_{A_1} G_1$ and $\Gamma \vdash Q \triangleright_{A_2} G_2$. Then by the definition of the cut function, we know $a \in \mathsf{fn}(G_1) \cap \mathsf{fn}(G_2) \ \Rightarrow \ a \in A'$, which means $a \notin \mathsf{fn}(G_3)$. (ii) is easy from Proposition 3.11 (ii), Lemma A.1 (i) and the above. □

Now we prove the typing system is closed under structural rules.

Proof of Lemma 4.12 (iv) (structural rules)

For α-conversion, the case $ax.P \equiv_\alpha ay.P\{y/x\}$ (y fresh) is easy by checking (in$_i$) rules. For $(a)P \equiv_\alpha (b)P\{b/a\}$ (b fresh), assume

$$\dfrac{\Gamma \vdash P \triangleright_A G}{\Gamma \backslash a \vdash (a)P \triangleright_{A \backslash \{a\}} G} \quad (\text{res})$$

then with noting $a \notin \mathsf{fn}(G)$ and assuming b fresh, we have:

$$\dfrac{\dfrac{\Gamma \vdash P \triangleright_A G}{\Gamma\{b/a\} \vdash P\{b/a\} \triangleright_{A\{b/a\}} G\{b/a\}}\ \text{(renaming lemma)}}{\Gamma\{b/a\}\backslash b \vdash (b)P\{b/a\} \triangleright_{A\{b/a\}\backslash\{b\}} G}\ \text{(res)}$$

The most careful rule is $(P \mid Q)R \equiv P \mid (Q \mid R)$, but it is proved with Lemma A.2 (ii). For the case of "scope open", i.e. $(a)P \mid Q \equiv (a)(P \mid Q)$ $(a \notin \mathcal{FN}(Q))$, assume

$$\dfrac{\dfrac{\Gamma \vdash P \triangleright_{A_1} G_1}{\Gamma\backslash a \vdash (a)P \triangleright_{A_1\backslash\{a\}} G_1}\ \text{(res)} \qquad \Gamma\backslash a \vdash Q \triangleright_{A_2} G_2}{\Gamma\backslash a \vdash (a)P \mid Q \triangleright_{A_1\backslash\{a\}\cup A_2 \cup A'} G_1 \odot G_2}\ \text{(par)}$$

W.l.o.g. we can assume $a \notin A_2 \cup \mathsf{fn}(G_2)$ because typing system is closed under α-conversion and renaming lemma. Then by (weak) rule, $\Gamma \vdash Q \triangleright_{A_2} G_2$ is derived from $\Gamma\backslash a \vdash Q \triangleright_{A_2} G_2$. Hence $\Gamma\backslash a \vdash (a)(P \mid Q) \triangleright_{(A_1 \cup A_2 \cup A')\backslash\{a\}} G_1 \odot G_2$ is proved. The other direction is similar. Other rules are obvious. □

Proof of Lemma 4.12 (v): (combination)

Suppose (in$_1$) is used. Then by Lemma 4.12 (i), a cannot be in Γ, therefore it should again be in G_1 and G_2. So $j = 3, 4$ are impossible. But the combination with (out$_2$) is also forbidden because there is no complete type like "(y)". Similarly for other cases. □

For the next lemma, we define "inversion". Hereafter we assume π-terms always obey the binding condition. Also, by 4.13 (iv), we shall safely consider typed π-terms modulo their α-equality, so that α-conversion of terms may be done implicitly during deduction.

A.3. **Definition.** (inversion) We say an ordered tuple of rules written (rule A) $\rightarrow$ (rule B) is *invertible* when, if any proof ends with a successive application of those two rules, there is a proof with the same conclusion in which the final two deductions are by those two rules in the reversed order.

A.4. **Lemma.** (inversions) (any rule in π_{G}) $\rightarrow$ (weak) *is invertible.*

Proof. For (par) rule, we use Lemma A.2 (ii). Others are mechanical. □

Now we can prove the fundamental lemma using Lemma 5.8 (i–v).

Proof of Lemma 4.12 (vi): (substitution lemma)

By the last clause of Lemma 4.12 (i) we assume names in $\mathsf{fn}(\Gamma) \cup \{a, b\}$ and A are restricted to free names of P. When either a or b is not in $\mathsf{fn}(P)$, or when we have

$a = b$, the statement is immediate using (Renaming Lemma when $b \notin \mathsf{fn}(P)$). We thus assume $a, b \in \mathsf{fn}(P)$ and $a \neq b$ in the following. This means that we do not have to consider (weak) rule by Lemma 4.12 (i).

So assume $a\colon \beta, b\colon \beta, \Gamma \vdash P \triangleright_A G$ is derived. We use induction on the structure of P.

(i) The cases $P \overset{\text{def}}{=} \mathbf{0}$ is trivial using Lemma 4.12 (i).

(ii) The cases $P \overset{\text{def}}{=} !P'$ is obvious by Lemma 4.12 (vi) with inductive hypothesis of P'.

(iii) For the case $P \overset{\text{def}}{=} (c)P'$, by Lemma A.4, we safely assume the last deduction is:

$$\text{(res)}\quad \frac{a\colon \beta, b\colon \beta, \Gamma \vdash P' \triangleright_{A\cup\{c\}} G}{a\colon \beta, b\colon \beta, \Gamma \vdash (c)P' \triangleright_A G}$$

Note, by assumption, we have $a, b \in \mathsf{fn}(P)$ hence $a \neq c$ and $b \neq c$. Thus $b : \beta, \Gamma \vdash P'\{b/a\} \triangleright_{A\cup\{c\}} G$ by induction, and then by using (res), we get the required sequent. The case $c \in \mathsf{fn}(\Gamma)$ is similar.

(iv) For the case $P \overset{\text{def}}{=} P_1 \,|\, P_2$, by Lemma A.4, we can safely assume it is derived from (par). Thus we have:

$$\text{(par)}\quad \frac{a\colon \beta, b\colon \beta, \Gamma \vdash P_1 \triangleright_{A_1} G_1 \qquad a\colon \beta, b\colon \beta, \Gamma \vdash P_2 \triangleright_{A_2} G_2}{a\colon \beta, b\colon \beta, \Gamma \vdash P_1 \mid P_2 \triangleright_{A_1\cup A_2\cup A} (G_1 \odot G_2)}$$

By applying inductive hypothesis, we have $b\colon \beta, \Gamma \vdash P_i\{b/a\} \triangleright_{A_i} G_i$ $(i = 1, 2)$. Then using (par) rule again, we are done.

(v) If $P \overset{\text{def}}{=} cx.P'$, we have $a\colon \beta, b\colon \beta, \Gamma \vdash cx.P \triangleright_A G$. We safely assume, by Lemma A.4, the sequent is derived using (in) directly and $x \notin \{a, b\}$. If $c \notin \{a, b\}$, the inductive hypothesis can be easily applied. So assume $a = c$. We first notice the last rule cannot be (in$_i$) for $1 \leq i \leq 2$. So there are two cases.

(1) Suppose the sequent is derived with the last rule (in$_3$), i.e.

$$\frac{a\colon \beta, b\colon \beta, \Gamma \vdash P \triangleright_A G \quad \beta = \{\downarrow(x) \to G, \uparrow(x) \to \overline{G}\}}{a\colon \beta, b\colon \beta, \Gamma \vdash ax.P \triangleright_A}$$

Note a cannot occur in G, hence by induction we have: $b\colon \beta, \Gamma \vdash P\{b/a\} \triangleright_A G$. By applying (in$_3$) rule again, we get $b\colon \beta, \Gamma \vdash bx.P\{b/a\} \triangleright_A$, as required.

(2) Suppose the sequent is derived with the last rule (in$_4$), i.e.

$$\text{(in}_4\text{)}\quad \frac{a\colon (\alpha),\ b\colon (\alpha),\ x\colon \alpha, \Gamma \vdash P \triangleright_A}{a\colon (\alpha),\ b\colon (\alpha),\ \Gamma \vdash ax.P \triangleright_A}$$

By induction we have $b\colon (\alpha),\ x\colon \alpha, \Gamma \vdash P\{b/a\} \triangleright_A$. Then, using (in$_4$) again, we get $b\colon (\alpha),\ \Gamma \vdash bx.P\{b/a\} \triangleright_A$, as required.

(vi) The case for $P \overset{\text{def}}{=} \overline{c}x.P'$ is essentially the same as above, with some care in the manipulation of the auxiliary name set when (out$_3$).

The case $b = c$ is similar so we omit it. □

Appendix B. Proof of Subject Reduction Theorem

First we state the property on cut-elimination.

B.1. **Lemma.** *Suppose: (1) $G_1 \asymp G_2$ (2) G_1 is a-pointed whose head is $\mathbf{n}$ and G_2 is a-pointed whose whose head is $\overline{\mathbf{n}}$. Then:*

(i) $(G_1 \backslash \mathbf{n}) \asymp (G_2 \backslash \overline{\mathbf{n}})$.

(ii) $(G_1 \odot G_2) \;\cong\; ((G_1 \backslash \mathbf{n}) \odot (G_2 \backslash \overline{\mathbf{n}}))$.

Proof. Easy by Propositions 3.8 and 3.10. □

To prove subject reduction, we use the following characterisation of one-step reduction and the form of π-terms.

B.2. **Lemma.** *Define $\longrightarrow'$ by the least relation generated by:*

$$\begin{array}{ll}(\text{COM'}) & (\tilde{c})(P_1 \,|\, ... \,|\, (ax.P \,|\, \overline{a}v.Q) \,|\, ... \,|\, P_n) \;\longrightarrow'\; (\tilde{c})(P_1 \,|\, ... \,|\, (P\{v/x\} \,|\, Q) \,|\, ... \,|\, P_n) \\ (\text{STR}) & P \equiv P' \;\; P' \longrightarrow' Q' \;\; Q \equiv Q' \;\Rightarrow\; P \longrightarrow' Q.\end{array}$$

where we assume P_i in (COM') are all input, output agents, or replicator. Then we have $\longrightarrow' = \longrightarrow$.

Proof. By checking each rule for $\longrightarrow'$ is derivable from those for $\longrightarrow$ and vice versa. □

Now we are ready to prove the subject reduction theorem.

Proof of the Theorem 4.13: (Subject Reduction Theorem)

By induction on the length of derivation of $\longrightarrow'$. We assume $\longrightarrow'$ is derived by the rules in Lemma B.1. If the derivation ends with the application of (STR), then the result is immediate from Lemma 4.12 (iv). So suppose the derivation ends with (COM') rule. Then we have a sequent: $\Delta \vdash (\tilde{c})(P_1|P_2|...|(ax.P|\overline{a}v.Q)|...|P_n) \triangleright_{A_0} G_0$. By Lemma 4.12 (iii), we can assume there is a deduction:

$$\Gamma \vdash ax.P \,|\, \overline{a}v.Q \triangleright_A G. \tag{B.1}$$

Now we prove

$$\Gamma \vdash P\{v/x\} \,|\, Q \triangleright_A G \tag{B.2}$$

Once (B.2) is proved, by replacing the deduction for (B.1), we can get the sequent $\Delta \vdash (\tilde{c})(P_1 \,|\, P_2 \,|\, ... \,|\, P\{v/x\} \,|\, Q \,|\, ... \,|\, P_n) \triangleright_{A_0} G_0$, hence the result.

Note, by Lemma A.4, we can assume that the application of (weak) occurs only at the first step after each axiom. So we can safely assume that (B.1) is derived by applying (par) rule at the last step, with antecedents $\Gamma \vdash ax.P \triangleright_{A_1} G_1$ and $\Gamma \vdash \overline{a}v.Q \triangleright_{A_2} G_2$, where, by Lemma 4.12 (i), we have

$$\mathsf{fn}(\Gamma) \cap A = \mathsf{fn}(\Gamma) \cap \mathsf{fn}(G) = A \cap \mathsf{fn}(G) = \emptyset. \tag{B.3}$$

Now we can assume that these antecedents are derived first introducing these prime terms by:

$$(\mathrm{in}_i)\ \frac{\Gamma_1 \vdash P \triangleright_{A_1} G_1'}{\Gamma \vdash ax.P \triangleright_{A_1} G_1} \quad (\mathrm{out}_j)\ \frac{\Gamma \vdash Q \triangleright_{A_2'} G_2'}{\Gamma \vdash \overline{a}v.Q \triangleright_{A_2} G_2} \tag{B.4}$$

with $A = A_1 \cup A_2 \cup A'$ ($A' = \mathsf{fn}(G_1) \cap \mathsf{fn}(G_2)$) and $G = G_1 \odot G_2$.

By Combination Lemma, the step (B.4) above cannot be done for arbitrary combination of i and j. So there are following four cases.

Case (in$_1$)–(out$_1$): There are derivations such that

$$\frac{\Gamma \vdash P \triangleright_{A_1} G_1'}{\Gamma \vdash ax.P \triangleright_{A_1} a\!\downarrow\!(x) \to G_1'} \qquad \frac{\Gamma \vdash Q \triangleright_{A_2 \setminus \{v\}} G_2'}{\Gamma \vdash \overline{a}v.Q \triangleright_{A_2} a\!\uparrow\!(v) \to G_2'} \tag{B.5}$$

We immediately know $v \notin \mathsf{fn}(G_1')$, hence, by Renaming Lemma, that

$$\Gamma \vdash P\{v/x\} \triangleright_{A_1} G_1'\{v/x\} \qquad \Gamma \vdash Q \triangleright_{A_2} G_2'. \tag{B.6}$$

Now we note that, by the condition of (par) rule,

$$(a\!\downarrow\!(x) \to G_1') \asymp (a\!\uparrow\!(v) \to G_2') \tag{B.7}$$

By α-conversion (noting $v \notin \mathsf{fn}(G_1')$), (B.7) becomes:

$$(a\!\downarrow\!(v) \to G_1'\{v/x\}) \asymp (a\!\uparrow\!(v) \to G_2') \tag{B.8}$$

We then safely assume $(G_1'\{v/x\} \odot G_2') \equiv_\alpha G$ by Lemma B.1. This shows, if we apply (par) to (B.6), that:

$$\Gamma \vdash P\{v/x\} \mid Q \triangleright_{A_1 \cup A_2 \cup A''} G$$

where $A'' \cup \{a, v\} = A'$ (note $v, a \notin \mathsf{fn}(\Gamma) \cup \mathsf{fn}(G)$ by $v \in A_2$ and (B.5)). Then applying (weak) rule, we get the required result.

Case (in$_2$)–(out$_2$): There are derivations such that

$$\frac{x : \alpha, \Gamma \vdash P \triangleright_{A_1} G_1'}{\Gamma \vdash ax.P \triangleright_{A_1} a\!\downarrow\!\alpha \to G_1'} \qquad \frac{\Gamma \vdash Q \triangleright_{A_2 \setminus \{v\}} G_2'}{\Gamma \vdash \overline{a}v.Q \triangleright_{A_2} a\!\uparrow\!\alpha \to G_2'} \tag{B.9}$$

with $v \neq x$, $x, v \notin \mathsf{fn}(G_1) \cup A_1$ and $v : \alpha \subset \Gamma$. By Substitution Lemma, we have $\Gamma \vdash P\{v/x\} \triangleright_{A_1} G_1'$. Then, using Lemma B.1, we have the required result.

Case (in$_3$)–(out$_3$): There are derivations such that:

$$\frac{a : \beta,\, \Gamma' \vdash P \triangleright_{A_1} G_1'}{a : \beta,\, \Gamma' \vdash ax.P \triangleright_{A_1}} \qquad \frac{a : \beta,\, \Gamma' \vdash Q \triangleright_{A_2 \setminus \{v\}} \overline{G_1'}\{v/x\}}{a : \beta,\, \Gamma' \vdash \overline{a}v.Q \triangleright_{A_2}} \tag{B.10}$$

with $a : \beta, \Gamma' = \Gamma$, $\beta = \{\downarrow(x) \to G_1', \uparrow(x) \to \overline{G_1'}\}$. By Proposition 3.10 (ii) ($G_1 \odot \overline{G}_1 = \emptyset$) and Lemma B.1, α-conversion as in the case of (in$_1$)–(out$_1$), we have the required result.

Case (in$_4$)–(out$_4$) Easy by Substitution Lemma (cf. the case (in$_2$)–(out$_2$)). $\square$

Appendix C. Proof for Lemma 5.9

.

Suppose $\Gamma \vdash P \rhd_A G$. Then we can write P as a normal form: $P \equiv (\tilde{b})P' \equiv (\tilde{b})(P_1 \mid P_2 \mid ... \mid P_n)$ where P_i is an input, output agents, or replicator. We prove the strong statement: $P \rightarrow_\beta Q_1$ and $P \longrightarrow Q_2$ with $Q_1 \not\equiv Q_2$ then there exists Q' s.t. $Q_1 \longrightarrow Q'$ and $Q_2 \rightarrow_\beta Q'$.

If $P \rightarrow_\beta Q_1$ is derived starting from the rule (β_2), it is obvious (cf. [18, 19, 22]). So suppose $P \rightarrow_\beta Q_1$ is derived starting from the rule (β_1). Then there should be a name c s.t. $c \in A \cap \mathsf{fs}(P')$, which is obtained by cut eliminations in (par) rule. So by Lemma 4.12 (iv) and Lemma A.4, there is a derivation

$$\frac{\Gamma \vdash P_i \rhd_{A_i} G_i \quad \Gamma \vdash P_j \rhd_{A_j} G_j}{\Gamma \vdash P_i \mid P_j \rhd_{A'} G_i \odot G_j}\ (\text{par})$$

for some $1 \leq i \neq j \leq n$ with $A' \subset A$. By Lemma A.1 (i), if $P_i \equiv cx.P_i'$, there is no other receptor whose subject is c in P, and there is only one output agent whose subject is c by definition of a cut function. So we set $P_j \equiv \overline{c}v.P_j'$.

Suppose $P \equiv (\tilde{b})(cx.P_i' \mid \overline{c}v.P_j' \mid R) \rightarrow_\beta (\tilde{b})(P_i'\{v/x\} \mid P_j' \mid R) \equiv Q_1$ and $P \longrightarrow Q_2$ with $Q_1 \not\equiv Q_2$. Then Q_2 should have a form: $Q_2 \equiv (\tilde{b})(cx.P_i' \mid \overline{c}v.P_j' \mid R')$ with $R \longrightarrow R'$ because of $c \notin \mathsf{fs}(R')$. Thus two redex pairs are never overlapped. Hence $Q_1 \longrightarrow (\tilde{b})(P_i'\{v/x\} \mid P_j' \mid R') \overset{\text{def}}{=} Q'$ and $Q_2 \rightarrow_\beta Q'$, as required. $\square$

Appendix D. Proof of Proposition 5.12

By induction on P. Suppose $\Gamma \vdash P \rhd_A G$.

There is no case for $P \equiv \mathbf{0}$ and $P \equiv !Q$ because $G \neq \emptyset$.

Case $P \equiv ax.Q$. By rules in Figure 5, if $\mathbf{n}$ is head in G, then $\mathbf{n} = a\downarrow(x)$ or $\mathbf{n} = a\downarrow\alpha$, and $G \equiv_\alpha \mathbf{n} \rightarrow G_0$. So we only consider (in$_1$) and (in$_2$) neglecting (weak) rule by Lemma A.4.

If $\mathbf{n} = a\downarrow(x)$, there is a derivation

$$(\text{in}_1)\quad \frac{\Gamma \vdash Q \rhd_A G_0}{\Gamma \vdash ax.Q \rhd_A a\downarrow(x) \rightarrow G_0} \tag{D.1}$$

with $x \notin \mathsf{fn}(\Gamma) \cup A$. Thus by selecting $x' \notin \mathsf{fn}(Q) \cup \mathsf{fn}(G) \cup \{a\}$ and then renaming x with x', we have $\Gamma \vdash Q\{x'/x\} \rhd_A G_0\{x'/x\}$ with $ax.Q \xrightarrow{ax'} Q\{x'/x\}$, which satisfies the (i), as required.

If $\mathbf{n} = a\downarrow\alpha$, there is a derivation

$$(\text{in}_2)\quad \frac{x\colon \alpha, \Gamma \vdash Q \rhd_A G_0}{\Gamma \vdash ax.Q \rhd_A a\downarrow\alpha \rightarrow G_0} \tag{D.2}$$

with $x \notin \mathsf{fn}(G) \cup A$. Hence $x\colon \alpha, \Gamma \vdash Q \triangleright_A G$ with $ax.Q \xrightarrow{ax} Q$, which satisfies (iii).

Case $P \equiv \overline{a}x.Q$. By the same reasoning, we only have to consider (out_1) and (out_2).

If $\mathbf{n} = a\uparrow(x)$, there is a derivation

$$(\text{out}_1)\quad \frac{\Gamma \vdash Q \triangleright_{A_0} G_0}{\Gamma \vdash \overline{a}x.Q \triangleright_{A_0 \cup \{x\}} a\uparrow(x) \to G_0} \tag{D.3}$$

with $A_0 \cup \{x\} = A$. Thus $\overline{a}x.Q \xrightarrow{\overline{a}x} Q$, which is just the second clause in (ii).

If $\mathbf{n} = a\uparrow\alpha$, there is a derivation

$$(\text{out}_2)\quad \frac{x\colon \alpha, \Gamma_0 \vdash Q \triangleright_A G_0}{x\colon \alpha, \Gamma_0 \vdash \overline{a}x.Q \triangleright_A a\uparrow\alpha \to G_0} \tag{D.4}$$

with $\Gamma = x\colon \alpha, \Gamma_0$. Thus $\overline{a}x.Q \xrightarrow{\overline{a}x} Q$, which is just the second clause in (ii).

Case $P \equiv (b)Q$. Suppose there is a derivation

$$(\text{res})\quad \frac{\Gamma_0 \vdash Q \triangleright_{A_0} G}{\Gamma_0 \backslash b \vdash (b)Q \triangleright_{A_0 \backslash \{b\}} G} \quad (b \notin \mathsf{fn}(G)) \tag{D.5}$$

and $Q \twoheadrightarrow_\beta Q'' \xrightarrow{l} Q'$ with either $l = ac$, $\overline{a}c$ or $a(c)$. First we note $b \neq a$ by the condition in (res) rule. If $c \neq b$, it is obvious by the induction on Q. Suppose $l = \overline{a}b$ (when $l = a(b), ab$, we can use α-convertibility on $(b)Q$). Then it relates with either the second clause in (ii) or the second one in (iv). In the second clause in (ii),

$$Q'' \xrightarrow{\overline{a}b} Q' \quad \text{and} \quad \Gamma \vdash Q' \triangleright_A (G \backslash a\uparrow(x))\{b/x\} \quad \text{with} \quad A_0 = A \cup \{b\}$$

By scope open rule in the labeled transition system, we know $(b)Q'' \xrightarrow{\overline{a}(b)} Q'$. By (D.5) together with the subject reduction theorem, now $\Gamma \vdash (b)Q \twoheadrightarrow_\beta (b)Q'' \triangleright_A G$ fits the first clause in (ii), as required. The second case in (iv) is similar with considering the elimination of b from Γ_0 in (D.5).

Case $P \equiv P_1 \mid P_2$. This is the most non-trivial case. Suppose $a\downarrow(x) \in \mathsf{Hd}(G)$.

There exists the following deduction:

$$\frac{\Gamma \vdash P_1 \triangleright_{A_1} G_1 \quad \Gamma \vdash P_2 \triangleright_{A_2} G_2}{\Gamma \vdash P_1 \mid P_2 \triangleright_{A_1 \cup A_2 \cup A'} G_1 \odot G_2}\ (\text{par})$$

with $A = A_1 \cup A_2 \cup A'$ and $G_1 \odot G_2 \equiv_\alpha G$. By inductive hypothesis, $\Gamma \vdash P_i \triangleright_{A_i} G_i$ satisfies one of the cases from (i) to (iv) in Proposition 5.12.

Then there are two cases.

(1) $a\downarrow(x) \in \mathsf{Hd}(G_i)$.

Then $a \notin \mathsf{fn}(G_j)$ with $i \neq j$ because $A' = \mathsf{fn}(G_1) \cap \mathsf{fn}(G_2)$ by definition of cut function.

(2) $a\downarrow(x) \notin \mathsf{Hd}(G_1) \cup \mathsf{Hd}(G_2)$, i.e. a head $a\downarrow(x)$ is newly appeared after multi-cuts.

Case (1) is easy by induction hypothesis. In case (2), w.o.l.g. we assume newly appeared head was contained in G_1. Then this implies, in (par), we did multi-cuts of all the nodes of $\mathbf{n}_i$ such that $\mathbf{n}_1 \to \mathbf{n}_2 \cdots \to \mathbf{n}_n \to \mathbf{n}$ in G_1.

Assume b is a subject in $\mathbf{n}_i$. If b is bound name in G_1, there is an atomic node $\mathbf{n}_j$ whose label is $c{\uparrow}(b)$ or $c{\downarrow}(b)$ with $j < i$. Note by covering condition (ii-2) of the cut function, such a node $\mathbf{n}_j$ is in domain preserving the ordering $\mathbf{n}_j \not\succ \mathbf{n}_i$. If $\mathbf{n}_j = c{\uparrow}(b)$, then $b \in A_1$, or if $\mathbf{n}_j = c{\downarrow}(b)$, then $b \in A_2$ by definition of (out$_1$) in Figure 5. On the other hand, if $b = \mathsf{fn}(\mathbf{n}_i)$ is free in G_1, then clearly $b \in A'$ again by definition of cut function. So we have $b \in \mathsf{fn}(\mathbf{n}_i) \cup \mathsf{fn}(\mathsf{Hd}(G_1 \odot G_2)) \;\Rightarrow\; b \in A$.

Now noting ordering $\to$ is preserved by multi-cuts by Proposition 3.10, we know if $\mathbf{n}_1$ is a head of G_1 and in the domain of the cut function, then $\overline{\mathbf{n}}_1$ is also a head of G_2. Suppose $e \in \mathsf{fn}(\mathbf{n}_1)$. Then, by the previous argument $e \in A'$ and, by inductive hypothesis, there is a β-reduction $P_1 \mid P_2 \twoheadrightarrow_\beta P_1' \mid P_2'$ with $P_1 \twoheadrightarrow_\beta P_1'$ and $P_2 \twoheadrightarrow_\beta P_2'$ with $e \in \mathcal{AN}_-(P_1') \cap \mathcal{AN}_+(P_2')$. Therefore, $P_1' \mid P_2' \equiv (\tilde{c})(ex.R_1 \mid \overline{e}v.R_2 \mid R_3) \to_\beta (\tilde{c})(R_1\{v/x\} \mid R_2 \mid R_3)$ in which prefixes "ex" and "$\overline{e}v$" corresponding to $\mathbf{n}_1$ and $\overline{\mathbf{n}}_1$ are eliminated. By applying this to all the nodes $\mathbf{n}_i$ one by one, noting $\mathsf{fn}(\mathbf{n}_i) \in A$, from i to n, we have $P_1 \mid P_2 \twoheadrightarrow_\beta P'$ with $\Gamma \vdash P' \triangleright_A G_1 \odot G_2$ and $a \in \mathcal{AN}_-(P')$, hence done with (in) case. Other cases are just the same. $\square$

Appendix E. Basic Properties in Reduction Closed Theories

This appendix collects the proofs of basic properties on reduction closed theories. More development on other calculi is left to [48].

E.1. **Proposition.** *Assume $\mathcal{R}$ and $\mathcal{R}_0$ are reflexive. Then we have:*

(i) $\Phi_p(\mathcal{R}) \;\subset\; \Phi_{pr}(\mathcal{R}) \;\subset\; \Phi_c(\mathcal{R})$.

(ii) $\Phi_{pr}(\Phi_{\Phi_c(\mathcal{R}_0)}(\mathcal{R})) \;\subset\; \Phi_{\Phi_c(\mathcal{R}_0)}(\Phi_{pr}(\mathcal{R}))$.

Proof. (i) is obvious. For (ii), let $\Phi_c(\mathcal{R}_0) \overset{\text{def}}{=} \cong$, then

$$\begin{array}{lll} P\ \Phi_{pr}(\cong\ \mathcal{R}\ \cong)\ Q & \Rightarrow & P \equiv C_r[P_0] \ \wedge\ P_0 \cong P_1\,\mathcal{R}\,Q_1 \cong Q_0 \wedge\ C_r[Q_0] \equiv Q \\ & \Rightarrow & P \equiv C_r[P_0] \cong C_r[P_1] \ \wedge\ P_1\,\mathcal{R}\,Q_1 \ \wedge\ C_r[Q_0] \equiv Q \\ & \Rightarrow & P \equiv C_r[P_0] \cong C_r[P_1]\,\Phi_{pr}(\mathcal{R})\,C_r[Q_0] \equiv Q \end{array}$$ $\square$

Proof of Lemma 6.4

For (i), set $\mathcal{S} \overset{\text{def}}{=} \Phi(\mathcal{R})$. Then obviously $\Phi(\mathcal{R}) \mapsto \Phi(\mathcal{R})$ with using monotonicity of Φ and identity of Φ. For (ii), firstly for monotonicity, we note: $\mathcal{R} \mapsto \mathcal{S} \;\Rightarrow\; \Phi'(\mathcal{R}) \mapsto \Phi'(\mathcal{S}) \;\Rightarrow\; \Phi \circ \Phi'(\mathcal{R}) \mapsto \Phi \circ \Phi'(\mathcal{S})$ by monotonicity of Φ and Φ'. For identity of $\Phi \circ \Phi'$, the inclusion $\Phi \circ \Phi' \circ \Phi \circ \Phi' \;\supset\; \Phi \circ \Phi'$ is obvious with the above result, while the other inclusion $\Phi \circ \underline{\Phi' \circ \Phi} \circ \Phi' \;\subset\; \Phi \circ \Phi'$ is proved by applying the condition $\Phi' \circ \Phi \;\subset\; \Phi \circ \Phi'$ to the underlined part. $\square$

Proof of Lemma 6.5

First we note $\Phi_p(\mathcal{R})$ and $\Phi_c(\mathcal{R})$ are substitution closed by Proposition 6.2. Then the rest is also by induction on the derivation in Figure 5, using the same reasoning as in the proof in Lemma 3.6 in [20]. Let $\cong \overset{\text{def}}{=} \Phi_c(\mathcal{R})$. We establish, for a pair P and Q such that $P \cong Q$, the following stronger statement (note $\cong$ contains $\equiv$ below), we only have to check: for any R, $P \,|\, R \longrightarrow P'$ implies, for some Q', we have $Q \,|\, R \longrightarrow\!\!\!\!\rightarrow Q'$ with $P' \cong Q'$, assuming $P \cong Q$, at each step of derivation. As the proof in Lemma 3.6 in [20], the only interesting case is that $ax.P \cong ax.Q$ is derived by (in_i) at the last step. The case $(P \,|\, R) \longrightarrow (P \,|\, R')$ is trivial. So assume P and R are interacting together. Then for some $\tilde{c}$, $\overline{a}v.R_0$ and R_1, we can assume

$$(P \,|\, R) \equiv (\tilde{c})(ax.P \,|\, \overline{a}v.R_0 \,|\, R_1) \longrightarrow (\tilde{c})(P\{x/v\} \,|\, R_0 \,|\, R_1)$$

By Lemma 4.12 (iii,vi), we have the following deduction for each $i = 1, 2, 3, 4$.

$$(\text{in}_i)\ \frac{\Gamma_1 \vdash P \triangleright_{A_1} G_1'}{\Gamma \vdash ax.P \triangleright_{A_1} G_1} \quad (\text{out}_i)\ \frac{\Gamma \vdash R_0 \triangleright_{A_2'} G_2'}{\Gamma \vdash \overline{a}v.R_0 \triangleright_{A_2} G_2} \tag{E.1}$$

Note the step E.1 above cannot be done for arbitrary combination of (in_i) and (out_j) by Combination Lemma (Lemma 4.12(v)). Now we want to prove, if $\Gamma \vdash ax.P \cong ax.Q \triangleright_{A_1} G_1$, then, $\Gamma \vdash P\{v/x\} \cong Q\{v/x\} \triangleright_{A_1} G_1'\{v/x\}$ with $G_1'\{v/x\} \odot G_2' = G_1 \odot G_2$. By Lemma B.1, as seen in the proof of Subject Reduction Theorem, we know there exists, the following deduction: $\Gamma\{v/x\} \vdash P\{v/x\} \triangleright_{A_1} G_1'\{v/x\}$ which satisfies the above graph condition. So we only have to prove $P\{v/x\} \cong Q\{v/x\}$ can be derived, which leads us to the required result: for some Δ, A, G we have the equation $\Delta \vdash (\tilde{c})(P\{v/x\} \,|\, R_0 \,|\, R_1) \cong (\tilde{c})(Q\{x/v\} \,|\, R_0 \,|\, R_1) \triangleright_{A'} G'$.

We will prove following four cases.

Case $i = 1$**:** There are derivations such that

$$\frac{\Gamma \vdash P \triangleright_{A_1} G_1'}{\Gamma \vdash ax.P \triangleright_{A_1} a\!\downarrow\!(x) \to G_1'} \qquad \frac{\Gamma \vdash R_0 \triangleright_{A_2 \setminus \{v\}} G_2'}{\Gamma \vdash \overline{a}v.R_0 \triangleright_{A_2} a\!\uparrow\!(v) \to G_2'} \tag{E.2}$$

To compose $ax.Q$ with $\overline{a}v.R_0$, there should be a deduction

$$\frac{\Gamma \vdash Q \triangleright_{A_1} G_1'}{\Gamma \vdash ax.Q \triangleright_{A_1} a\!\downarrow\!(x) \to G_1'} \tag{E.3}$$

with $\Gamma \vdash ax.P \cong ax.Q \triangleright_{A_1} a\!\downarrow\!(x) \to G_1'$. We immediately know $v \notin \mathsf{fn}(G_1')$, hence, by Lemma 4.12 (iii) (renaming lemma), from E.2 and E.3, that

$$\Gamma \vdash P\{v/x\} \triangleright_{A_1} G_1'\{v/x\} \quad \text{and} \quad \Gamma \vdash Q\{v/x\} \triangleright_{A_1} G_1'\{v/x\} \tag{E.4}$$

This implies, by substitution closure of $\cong$, $\Gamma \vdash P\{v/x\} \cong Q\{v/x\} \triangleright_{A_1} G_1'\{v/x\}$, as required.

Case $i = 2$**:** We can set $\Gamma = v\colon \alpha, \Gamma_0$. Then $v\colon \alpha, \Gamma_0 \vdash ax.P \cong ax.Q \triangleright_{A_1} a\!\downarrow\!\alpha \to G_1'$ comes from $v\colon \alpha, x\colon \alpha, \Gamma \vdash P \cong Q \triangleright_{A_1} G_1$, then we use substitution closure.

Case $i = 3$**:** Similar with the case $i = 1$.

Case $i = 4$**:** Similar with the case $i = 2$.

(ii) is similarly established with the same reasoning: for each derivation $P \cong Q$, we prove $(P \mid R) \Downarrow_{a^\theta} \;\Rightarrow\; (Q \mid R) \Downarrow_{a^\theta}$ for all R, noting in particular, $ax.P \Downarrow_{a^-}$

$$\text{Rcpt}\ \frac{a\colon (S_1,..,S_n), x_i\colon S_i, \Gamma \vdash P}{a\colon (S_1,..,S_n), \Gamma \vdash a(x_1...x_n).P} \qquad \text{Emt}\ \frac{a\colon (S_1,..,S_n), x_i\colon S_i, \Gamma \vdash P}{a\colon (S_1,..,S_n), x_i\colon S_i, \Gamma \vdash \overline{a}\langle x_1..x_n\rangle.P}$$

$$\text{Comp}\ \frac{\Gamma \vdash P \quad \Gamma \vdash Q}{\Gamma \vdash P \mid Q} \qquad \text{Scop}\ \frac{\Gamma \vdash P}{\Gamma\backslash\{x\} \vdash (x)P}$$

$$\text{Repl}\ \frac{\Gamma \vdash P}{\Gamma \vdash !P} \qquad \text{Weak}\ \frac{\Gamma \vdash P}{a\colon (S_1,..,S_n), \Gamma \vdash P} \qquad \text{Nil}\ \ \emptyset \vdash \mathbf{0}$$

Figure 6. Typing System for Polyadic π-calculus.

$\Rightarrow\ \ ax.Q \Downarrow_{a^-}$. □

Proof of Lemma 6.6

By Subject Reduction Theorem, we safely omit the type annotation. For (i), first we show that $\Phi_{\cong}$ is r.c. respectable. Suppose $\mathcal{R} \mapsto \mathcal{S}$. We will establish $\Phi_{\cong}(\mathcal{R}) \mapsto \Phi_{\cong}(\mathcal{S})$. Assume $P \cong P_0\, \mathcal{R}\, Q_0 \cong Q$. Then whenever $P \longrightarrow\!\!\!\!\rightarrow P'$, there exists $P_0 \longrightarrow\!\!\!\!\rightarrow P_0'$ and $P_0 \cong P_0'$. Then by assumption $\mathcal{R} \mapsto \mathcal{S}$, we have $Q_0 \longrightarrow\!\!\!\!\rightarrow Q_0'$ with $P_0' \cong \mathcal{S} \cong Q_0'$. By also applying reduction closure property between Q_0 and Q, we have $P' \cong P_0' \cong \mathcal{S} \cong Q_0' \cong Q'$, which implies $P'\, \Phi_{\cong}(\mathcal{S})\, Q'$ by symmetry of $\cong$. Identity of $\Phi_{\cong}(\mathcal{S})$ is mechanically calculated by the property $\cong\cong\, = \,\cong$. The monotonicity is obvious. For the second case, since the transitive closure of $\cong \mathcal{R} \cong$ is a reducition-closed p-relation, $\Phi_c(\cong \cup \mathcal{R})$ is reduction-closed congruence by Lemma 6.5 together with $\Phi_c(\cong \cup \mathcal{R}) = \Phi_c(\cong \mathcal{R} \cong)$.

(ii) is also mechanical.

(iii) First we note by Proposition 6.2, $\Phi_{\cong}(\mathcal{R})$ and $\Phi_{pr}(\mathcal{R})$ are substitution closed, and then $\Phi_c(\mathcal{R} \cup \cong)$ is. Now we use Lemma 6.4 (ii). Let us set $\Phi_{\cong}$ to Φ and Φ_{pr} to Φ' of Lemma 6.4, and then check (1) $\Phi_{\cong}$ and Φ_{pr} are r.c. respectable and (2) $\Phi_{\cong} \circ \Phi_{pr} \supseteq \Phi_{pr} \circ \Phi_{\cong}$. First r.c. respectability of $\Phi_{\cong}$ and Φ_{pr} are given by (i) and (ii). The second clause $\Phi_{\cong} \circ \Phi_{pr} \supseteq \Phi_{pr} \circ \Phi_{\cong}$ is proved by Lemma E.1 (ii). □

Appendix F. Typing System and Labeled Transition Relation in Polyadic π-calculus

First we re-introduce the typing system in [43] in Figure 6, essentially equivalent to the system in [26], give the sequent of form $\vdash P \succ \Gamma$,which we write $\Gamma \vdash P$ where Γ is a map from a finite set of names to the set of sortings.

Next we introduce the labels which are same as 2.2.2 in [39]. The set of labels common to two labeled transition relations, ranged over by $l, l', ..$, is given by:

$$l \quad = \quad \tau \mid a\langle x_1...x_n\rangle \mid (\tilde{y})\, \overline{a}\langle v_1...v_n\rangle$$

where in output $\{\tilde{y}\} \subseteq \{\tilde{v}\}\backslash\{a\}$ *bounds* the values $\tilde{v}$. We write $\mathsf{n}(l)$, $\mathsf{bn}(l)$ and $\mathsf{fn}(l)$ for the sets of names, bound and free names in l. Then the transition system is presented as follows.

F.1. **Definition.** (labeled transition system in polyadic π-calculus)
In the following, we sometimes write $\Gamma \vdash P \xrightarrow{l}_{\mathsf{p}} P'$ as $\Gamma \vdash P \xrightarrow{l}_{\mathsf{p}} \Gamma \vdash P'$.

$$\texttt{inp} \quad a\colon (S_1, .., S_n), v_i\colon S_i, \Gamma \vdash a(x_1...x_2).P \xrightarrow{a\langle v_1...v_n\rangle}_{\mathsf{p}} P\{v_1...v_n/x_1...x_n\}$$

$$\texttt{out} \quad a\colon (S_1, .., S_n), v_i\colon S_i, \Gamma \vdash \overline{a}\langle v_1...v_2\rangle.P \xrightarrow{\overline{a}\langle v_1...v_n\rangle}_{\mathsf{p}} P$$

$$\texttt{par} \qquad \frac{\Gamma \vdash P \xrightarrow{l}_{\mathsf{p}} P' \quad \Gamma \vdash Q}{\Gamma \vdash P\,|\,Q \xrightarrow{l}_{\mathsf{p}} P'\,|\,Q} \quad (\mathsf{fn}(l) \cap \mathsf{fn}(Q)) = \emptyset)$$

$$\texttt{com} \qquad \frac{\Gamma \vdash P \xrightarrow{(\tilde{b})\,\overline{a}\langle v_1..v_n\rangle}_{\mathsf{p}} P' \quad \Gamma \vdash Q \xrightarrow{a\langle v_1..v_n\rangle}_{\mathsf{p}} Q'}{\Gamma \vdash P\,|\,Q \xrightarrow{\tau}_{\mathsf{p}} (\tilde{b})(P'\,|\,Q')} \quad (\{\tilde{b}\} \cap \mathsf{fn}(Q) = \emptyset)$$

$$\texttt{res} \qquad \frac{\Gamma \vdash P \xrightarrow{l}_{\mathsf{p}} P'}{\Gamma \backslash a \vdash (a)P \xrightarrow{l}_{\mathsf{p}} (a)P'} \quad (a \notin \mathsf{n}(l))$$

$$\texttt{open} \qquad \frac{\Gamma \vdash P \xrightarrow{(\tilde{b})\,\overline{a}\langle v_1..v_n\rangle}_{\mathsf{p}} P'}{\Gamma \backslash c \vdash (c)P \xrightarrow{(c\tilde{b})\overline{a}\langle v_1..v_n\rangle} \Gamma \vdash P'} \quad (c \neq a,\ c \in (\tilde{v} \backslash \tilde{b}))$$

$$\texttt{weak} \quad \frac{\Gamma_1 \vdash P \xrightarrow{l}_{\mathsf{p}} \Gamma_2 \vdash P' \ \wedge\ P \equiv_\alpha Q \ \wedge\ \Gamma_i \asymp \Delta_i \ \wedge\ \Gamma_1 \vdash Q \ \wedge\ \Gamma_2 \vdash P'}{\Delta_1 \vdash Q \xrightarrow{l}_{\mathsf{p}} \Delta_2 \vdash P'}$$

where we omit the symmetric version of `com` and `par`.