GRAPH NOTATION FOR CONCURRENT COMBINATORS

NOBUKO YOSHIDA

yoshida@mt.cs.keio.ac.jp

Department of Computer Science, Keio University

3-14-1 Hiyoshi, Kohoku-ku, Yokohama, 223, Japan

Abstract. We introduce graph notation for concurrent processes which does not use the notion of port names for its formulation. The operators in the algebra of graphs proposed in this paper are quite different from those in the original term representation, making such notions as connection and correspondence of communication ports explicit. We show how basic elements of process calculi such as agents, reduction, and behavioural equivalences are soundly formulated in the new setting. The work is based on the authors' study on concurrent combinators [9, 10], and can be considered as offering another mathematical representation of the formal notion studied therein.w

1. Introduction

The aim of the present paper is to introduce process graphs and their algebra, a formalism of concurrent processes which does not use the notion of port names for its formulation. The intention is to gain a deeper understanding about the basic elements of theory of processes by formulating them with a quite different orientation from the usual term-based one. The resulting construction may also serve as a theory of processes in its own right, where explicit manipulation of ports connection and correspondence is a fundamental element. We show how basic elements of process calculi such as agents, reduction, and behavioural equality are formulated in the new setting, shedding a fresh light on those familiar notions. Some applications of the theory to other graphical formalisms are also given along the way. The work is based on the authors' study on concurrent combinators [9, 10], offering another presentation of the formal notion studied therein.

Systems of concurrent combinators introduced in [9, 10] were born as a "combinatory" decomposition of calculi of mobile processes, where finite atoms and their combination give the same expressive power as e.g. π-calculus in [17] and ν-calculus in [8, 9]. That is, without using inductively defined constructs for prefix and replication, a system with finite atoms and fixed interaction rules among them is nevertheless capable of representing the same functionalities. Just with the same motivation, now we try to "analyse away" port names from the name-based process formalism.

We will begin by considering how we represent concurrent combinators in the nameless world of graphs. Let us suppose parallel composition of two messages $\mathcal{M}(ab)$, $\mathcal{M}(ac)$ and a duplicator $\mathcal{D}(ade)$[1] in the original term-based concurrent combinators [9, 10]. As

[1] In π-calculus [17], $\mathcal{M}(ab)$ and $\mathcal{D}(abc)$ mean $\bar{a}b.0$ and $a(x).(\bar{b}x.0 \mid \bar{c}x.0)$, respectively.

the fundamental elements of concurrent computation, the name "a" functions as an interaction point for multiple processes, so we can naturally represent this as a vertex $\bullet$ which is shared by three agents as seen in Figure 1 (a).

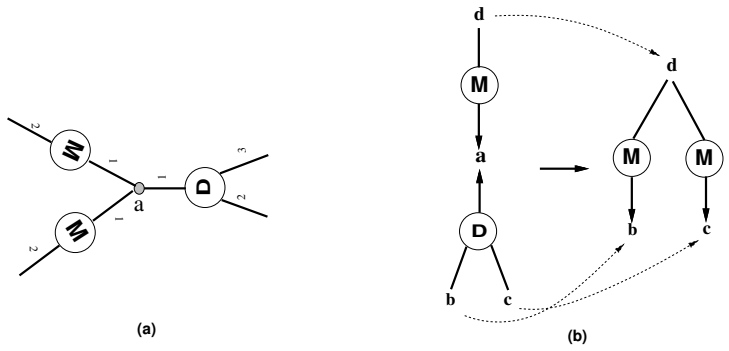

FIGURE 1. Atoms and Reduction with Names

In the next step, we consider how to formalise the process dynamics without using names. Let us turn to the notion of reduction relation in the world of processes which have names. In Figure 1 (b), an example of reduction: $\mathcal{D}(abc), \mathcal{M}(ad) \longrightarrow \mathcal{M}(bd),\ \mathcal{M}(cd)$[2] is shown. Thus the common interaction port invokes reduction, but what we notice is that the correspondence of free communication ports between two configurations is implicitly maintained by free names. Note also (1) no two distinct names in the left hand side (l.h.s for short) are collapsed into the same name in the right hand side (r.h.s for short), (2) it is possible that names (here a) will be lost after the reduction, and (3) all free names in r.h.s come from l.h.s. Now take off names from the processes: then clearly this correspondence can only be maintained in the nameless graphs by introducing *surjective partial injection between free handles of two graphs*, say δ. Thus one-step reduction from a graph G to a graph G' is defined by a triple $\langle G,\ \delta,\ G' \rangle$. The idea (possibly without surjectivity) is pertinent to defining any binary relations over processes such as behavioural semantics, as we shall see.

This paper presents the basic constructs of algebra of graphs and their elementary properties with illustrative examples. Section 2 defines the basic entities called *process graphs* together with operators for their algebra. Section 3 formulates several technical ideas needed to deal with graphs and operations over them with precision. Then they are immediately used in isomorphism laws (which may correspond to some equality laws in process algebra). Section 4 defines reduction with the above ideas and gives some examples. Section 5 presents the construction of behavioural equalities over process graphs with examples and studies their elementary properties. Section 6 summarises other results on process graphs and discusses related work and further issues.

2. Process Graphs (1): Basic Definition

Process graphs (cf. [12]) are nameless representation of processes, using the notion of graphs and their composition. Any graph is composed from a finite number of atomic agents and their connection. The results in [9, 10] already showed that the full expressive power of interacting processes can be obtained using a finite number of atoms and their interaction. Our process graphs are developed on this simplest possible basis, though its adaptation to other (possibly more complex) formalisms is possible (cf. 6.2).

[2]In [17], this reduction means: $a(x).(\bar{b}x.0 \mid \bar{c}x.0) \mid \bar{a}d.0 \longrightarrow \bar{b}d.0 \mid \bar{c}d.0$.

2.1. Atomic Graphs. Let $\mathbb{A}$ be a finite set of *atoms*, ranged over by $\mathcal{C}, \mathcal{C}', ...$, with two associating functions $ar : \mathbb{A} \to \mathbb{N}^+$ and $pol : \mathbb{A} \to \{+, -\}$. $ar(\mathcal{C})$ and $pol(\mathcal{C})$ are called the *arity* and the *polarity* of $\mathcal{C}$, respectively. $\theta, \theta', ..$ range over the set $\{+, -\}$. Essentially speaking, atoms denote the *units of behaviour* of processes: arities are the number of connections they can have with the outside, while polarities are the basic "type" of such behaviour (a positive atom and a negative atom will interact). Then the *atomic graph* of $\mathcal{C}^{(n)}$, written $\mathbf{g}(\mathcal{C})$, is a graph with only one *agent vertex* labeled with $\mathcal{C}$, n *free vertices* represented by $\bullet$, and n *edges* numbered by $\{1, 2, ..., n\}$ such as:

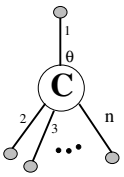

FIGURE 2. Atomic Graph

The handles are nameless representation of communication ports. A free handle corresponds to a free port name in usual process calculi, where "free" means "connectable to other handles". We call an edge numbered by 1 a *principal edge*, and a handle connected with such an edge a *principal handle*. A principal handle is a place where interaction takes place, like the initial prefix in CCS [16].[3] We also say the second, (third, ...) handle of an agent, if it is connected to the edges numbered accordingly.

2.2. Example. We take **cc** in [9], which was born in the course of analysis of prefix. With base $\mathbf{A_{cc}} \stackrel{\text{def}}{=} \{\mathcal{M}^{(2)+}, \mathcal{D}^{(3)-}, \mathcal{FW}^{(2)-}, \mathcal{K}^{(1)-}, \mathcal{B}_l^{(2)-}, \mathcal{B}_r^{(2)-}, \mathcal{S}^{(3)-}\}$, seven atomic graphs are given in Figure 3. Each agent vertex is labeled by the name of the corresponding atom. As shown, we sometimes write polarities of atoms in their principal edges.

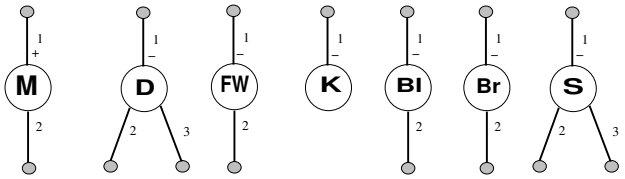

FIGURE 3. Atomic Graphs from $\mathbf{A_{cc}}$

2.3. Process Graphs. Hereafter we fix $\mathbb{A}$ as given. Let $\mathbf{E}$ be a set of *edges*, ranged over by $e, e_1, e_2, ..$, $\mathbf{V}_{\mathbb{A}}$ be a set of *agent vertices*, ranged over by $a, a_1, a_2, ..$ and $\mathbf{V}_{\mathrm{F}}$ be a set of *free handles*, ranged over by $h, h_1, h_2, ...$ Then the set of *process graphs* with base $\mathbb{A}$, ranged over by $G, G', ...$, is defined inductively in the following.

- **(empty graph)** The empty graph, denoted by $\mathbf{1}$, is a process graph.
- **(atomic graph)** An atomic graph is a process graph.

[3] A process which has more than one interaction point can be represented by some combination of the original atomic graphs with dyadic interaction rules defined later: accordingly we treat this type of a simple graph as an atom.

- **(juxtaposition)** If two graphs, say G_1 and G_2, are process graphs, then their *juxtaposition*, which is the (graph) union of G_1 and G_2, written $G_1 \,\|\, G_2$, is a process graph (Figure 4 (a)).
- **(connection)** Suppose that Σ is a partition on a set of free handles of a process graph G. Then the *connection of G by Σ*, written $G\langle\Sigma\rangle$, which is a graph G' such that handles grouped together by Σ are adjoined in G, is a process graph (Figure 4 (b)).
- **(hiding)** Suppose that H is a subset of free handles in a process graph G. Then the *hiding* of H on G, denoted by $H \blacktriangleright G$, which is a graph replacing each $h_i \in H$ of G with a new (distinct) hidden handle denoted by $\circ$, is a process graph (Figure 4 (c)).

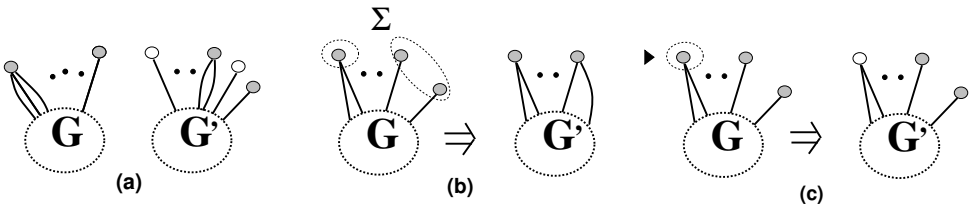

FIGURE 4. Process Graphs

In *juxtaposition*, two processes are composed to run on parallel but without any interaction between them (this corresponds to "$P|Q$" in CCS where the sets of free names in P and Q are disjoint). Interactivity between subsystems increases by *connection* of hitherto unconnected handles (roughly this is like making $(a.0\,|\,\bar{b}.0)$ to $(a.0\,|\,\bar{a}.0)$ by name substitution). Finally the *hiding* makes some handles unconnectable to other handles ($P\backslash L$ in CCS).

2.4. Notations. We let $\mathbf{V}_{\mathrm{R}}$ denote a set of *hidden handles*, ranged by $r, r_1, r_2, \ldots$. Note that $\mathbf{V}_{\mathbb{A}}$, $\mathbf{V}_{\mathrm{F}}$ and $\mathbf{V}_{\mathrm{R}}$ are disjoint. Then $\mathbf{V}_{\mathbb{A}}(G)$, $\mathbf{V}_{\mathrm{F}}(G)$, $\mathbf{V}_{\mathrm{R}}(G)$ and $\mathbf{E}(G)$ denote agent vertices, free handles, hidden handles and edges, in a process graph G, respectively. An *isomorphism* between two process graphs, written as $G_1 \simeq G_2$, is a usual one for graphs except that it should respect two labeling functions as well as distinction between free and hidden handles. We regard two isomorphic graphs as denoting the same object, as in the usual practice in the graph theory. A *pre-atomic graph* of $\mathcal{C}$ is a graph G with one agent vertex labeled with $\mathcal{C}$, such that $ar(\mathcal{C}) \supseteq |\mathbf{V}_{\mathrm{F}}(G)|$ and $\mathbf{V}_{\mathrm{R}}(G) = \emptyset$. If a graph is not pre-atomic, then it is *composite*.

2.5. Example. Assume the base $\mathbf{A_{cc}}$ again. Figure 5 (a) gives a simple example of juxtaposition. The hiding replaces $\bullet$ with $\circ$, as in (b). The connection is also simple in diagram. Suppose, in G of Figure 5 (c)-1, that $\Sigma_1 = \{\{h_1\}, \{h_2, h_3\}\}$ and $\Sigma_2 = \{\{h_1, h_2, h_3\}\}$. Then $G\langle\Sigma_1\rangle$ becomes like (c)-2 and $G\langle\Sigma_2\rangle$ becomes like (c)-3. Figure 5 (d) is an example of a composite graph. (e) shows all the pre-atomic graphs for $\mathcal{D}$ except the atomic graph itself (already given in Figure 3). So we have 5 pre-atomic graphs of $\mathcal{D}$ up to isomorphism.

2.6. Parallel Composition. The operation of *connection* we have introduced has no analogue in the operators in the usual process calculi, while being quite natural from a graph-theoretic point of view. However there is yet another fairly natural — but less general — way of "connecting agents," which is directly related to parallel composition in the term based process algebra.

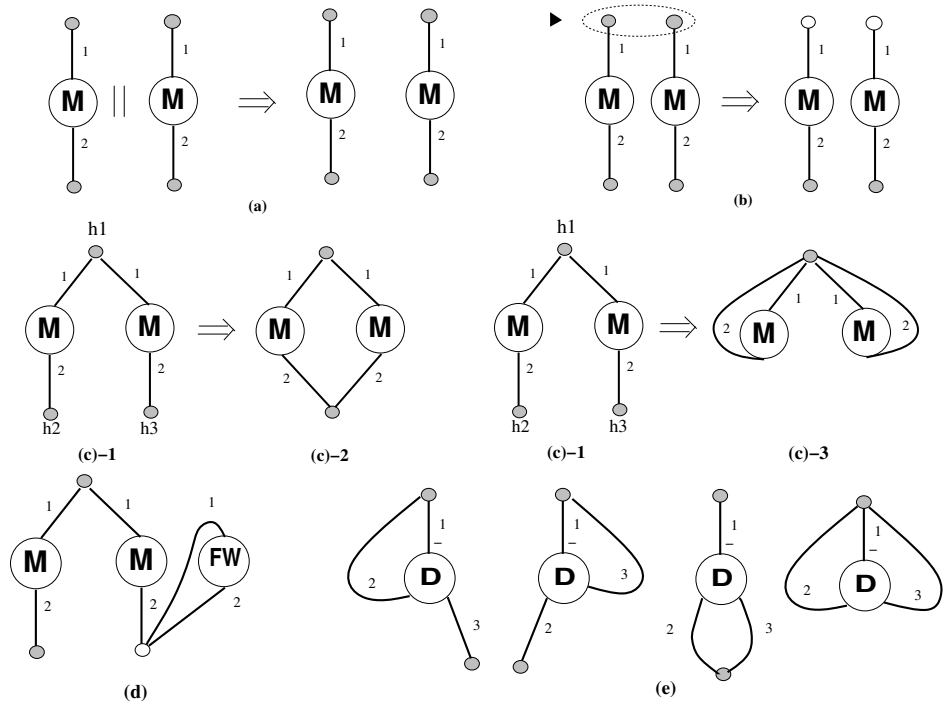

FIGURE 5. Operations and Pre-atomic Graphs

Think of a CCS term "$a.b.0|\bar{a}.0$." This simple example already makes it clear that the usual idea of parallel composition includes *both* juxtaposition *and* connection. But the involved connection has a specific feature, in that a single port of l.h.s agent is connected to a single port of r.h.s agent, and vice versa.

Let δ be a partial injection from $\mathbf{V}_{\mathrm{F}}(G_1)$ to $\mathbf{V}_{\mathrm{F}}(G_2)$. Then *a parallel composition of* G_1 *and* G_2 *by* δ, written $G_1 \parallel_\delta G_2$, is the union of G_1 and G_2 where if $\delta : h \mapsto h'$ we adjoin h into h'.

Note that $G_1 \parallel G_2$ corresponds to the case when the above δ is empty.

2.7. Example. Given two $\mathbf{g}(\mathcal{M})$'s and their correspondence δ as in Figure 6 (a)-1, $\mathbf{g}(\mathcal{M}) \parallel_\delta \mathbf{g}(\mathcal{M})$ is as (a)-2. Note that any two distinct free handles in the source graph must not be connected to the same handle in the target graph; so (b)-2 is not obtained from (b)-1 because the function f is not injection, but from (b)-3. We find that this is precisely what is happening in parallel composition in the usual process algebra.

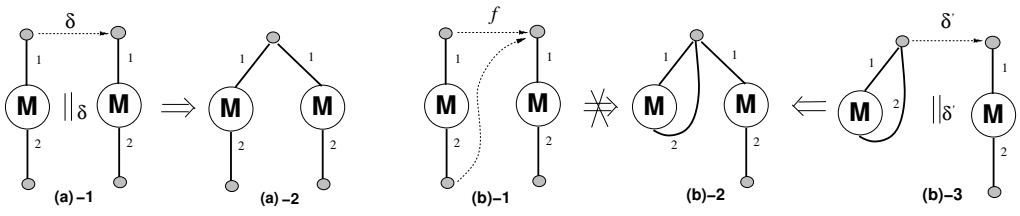

FIGURE 6. Concurrent Composition

More technical discussions on these operators will be developed in the next section.

3. Process Graphs (2): the Basic Algebra

This section continues the study of the graph-based theory of processes. By definition of our process graphs, we know any process graph is constructed by three basic operators. To begin with, we define textual notations which have been used somewhat informally so far but will be used as a rigorous basis for formulating various ideas in the algebra of process graphs. Then the equality laws for isomorphic graphs are developed as an example of such an algebra. Finally we establish that parallel composition which has been regarded as a "primitive" operation in the usual process calculi can be translated in our three basic operations which are more in the spirit of graph theory.

3.1. Textual Notation. We introduce the following formal textual notations which can express any process graph, fixing a base. Assuming the set of atoms and handles as given, the *textual notations*, again ranged over by $G_1, G_2, ...$, are given by:

- $\mathbf{1}$,
- $\mathcal{C}^{(n)}[h_1..h_n]$ ($h_i \neq h_j$ if $i \neq j$) for each atom $\mathcal{C}^{(n)}$. Then $\mathbf{V}_{\mathrm{F}}(\mathcal{C}^{(n)}[h_1..h_n]) = \{h_1, .., h_n\}$, which *occur free* in this notation.
- $G_1 \parallel G_2$, where we assume no common free handles occur in both G_1 and G_2.
- $G\langle\Sigma\rangle^f$, where Σ is a partition on $\mathbf{V}_{\mathrm{F}}(G)$ and f is a total map from $\mathbf{V}_{\mathrm{F}}(G)$ to distinct fresh handles, its kernel inducing the partition Σ. Then the fresh handles become the free handles in the new notation. A notation like $G\langle\{h_1,h_2\}_{h'_1},\{h_3\}_{h'_2}\rangle$ is also used (assuming $\mathbf{V}_{\mathrm{F}}(G) = \{h_1,h_2,h_3\}$): then this notation denotes a new graph with the new set of free handles $\{h'_1,h'_2\}$.
- $H \blacktriangleright G$, when H is a subset of free handles in a notation G. Then the complement of H becomes the free handles of the new notation.
- $G_1 \parallel_\delta G_2$, where no free handles occur common between two G_1 and G_2 and δ is a partial injection between $\mathbf{V}_{\mathrm{F}}(G_1)$ and $\mathbf{V}_{\mathrm{F}}(G_2)$. The free handles in G_1 are assumed to be rewritten by δ.

Also, when writing down composed operations, we assume both of $\parallel$ and $\parallel_\delta$ are the weakest in association and associate to the left, and $\blacktriangleright$ is weaker than $\langle\Sigma\rangle$ in association. Note that, as in graph theory, e.g. $h_1 \blacktriangleright \mathcal{M}[h_1h_2]$ denotes the same process graph as $h_3 \blacktriangleright \mathcal{M}[h_3h_4]$: accordingly we introduce the global injective relabeling of handles from G_1 to G_2, denoted by $G_1 \triangleright G_2$.

3.2. Normal Form. Assume a composite graph in Figure 5 (d) again. Then we find a lot of ways to construct this graph by applying the operations to atomic graphs. However we immediately know that we can generate the same (isomorphic) process graphs with the fixed order of operations.

Proposition. *Any process graph can be generated from atomic graphs by applying firstly* juxtapositions, *secondly* connections, *and finally* hiding.

Proof: First observe that, for any graph G, there exists G' with $\mathbf{V}_{\mathrm{R}}(G') = \emptyset$, for which $G = H \blacktriangleright G'$ for some $H \subset \mathbf{V}_{\mathrm{F}}(G')$. Now it is easy to observe that $(G_1\langle\Sigma_1\rangle \parallel G_2\langle\Sigma_2\rangle)$ can be also composed as $(G_1 \parallel G_2)\langle\Sigma'\rangle$, and that composition of a sequence of connections is again a connection. Applying this fact to G' repeatedly, we know any G' can be written: $(\mathbf{g}(\mathcal{C}_1)\|\mathbf{g}(\mathcal{C}_2)\|...\|\mathbf{g}(\mathcal{C}_n))\langle\Sigma\rangle$. Hence the result. □

Now let us define a *normal form* of a textual notation: we say a textual notation is a normal form if it is formed as: $H \blacktriangleright ((\mathbf{g}(\mathcal{C}_1)\|...\|\mathbf{g}(\mathcal{C}_n))\langle\Sigma\rangle)$. Then in the scheme of Chemical Abstract Machine [1] we present the graph in Figure 5 (d) as a triple of a

multi-set of atomic graphs, a partition, and an indication of a set of hidden handles like: $\langle\{\!|\mathcal{M}[h_1h_2], \mathcal{M}[h_3h_4], \mathcal{FW}[h_5h_6]|\!\}, \{\{h_1, h_3\}, \{h_2\}, \{h_4, h_5, h_6\}_{h_7}\}, \{h_7\}\rangle$. Considering this representation, we say that two textual notations are the same normal form if firstly all atomic graphs are represented as the same multi-set, and secondly partitions and indications of hiding coincide, after using $\triangleright$.

3.3. Isomorphism Laws. We now develop the isomorphism laws for textual notations. Some of them resemble what we find as equality laws in the process algebra [1, 17]. The following proposition says that if and only if G and G' denote the same (isomorphic) graph, we can rewrite G to G' applying the following laws plus global relabeling. We use a notation: $\{H_i\}_{i\leq n}$ stands for $\{H_1, H_2, ..., H_n\}$.

Proposition. (soundness and completeness of the isomorphism laws) *$G \simeq G'$ iff $G \stackrel{\text{iso}}{=} G''$ and $G'' \triangleright G'$ where $\stackrel{\text{iso}}{=}$ is the smallest congruent relation over textual notations derived by the following rules.*

I. $G_1 \parallel G_2 \stackrel{\text{iso}}{=} G_2 \parallel G_1 \quad G \parallel \mathbf{1} \stackrel{\text{iso}}{=} G \quad (G_1 \parallel G_2) \parallel G_3 \stackrel{\text{iso}}{=} G_1 \parallel (G_2 \parallel G_3)$

II. $(G\langle\Sigma_1\rangle^{f_1})\langle\Sigma_2\rangle^{f_2} \stackrel{\text{iso}}{=} G\langle\Sigma\rangle^{f}$ with $f = f_2 \circ f_1$.

III. $(G_1 \parallel G_2\langle\Sigma\rangle^{f}) \stackrel{\text{iso}}{=} (G_1 \parallel G_2)\langle\Sigma'\rangle^{f'}$ where $f' = f \cup \{h_i \mapsto h'_i\}_{i\leq n}$ with $\mathbf{V}_{\mathrm{F}}(G_1) = \{h_1, ..., h_n\}$ and h'_i fresh.

IV. $H_2 \blacktriangleright (H_1 \blacktriangleright G) \stackrel{\text{iso}}{=} (H_1 \cup H_2) \blacktriangleright G$.

V. $(H \blacktriangleright G_1) \parallel G_2 \stackrel{\text{iso}}{=} H \blacktriangleright (G_1 \parallel G_2)$ with $H \cap \mathbf{V}_{\mathrm{F}}(G_2) = \emptyset$.

VI. $(H \blacktriangleright G)\langle\Sigma\rangle^{f} \stackrel{\text{iso}}{=} H' \blacktriangleright (G\langle\Sigma'\rangle^{f'})$ where $f' = f \cup \{h_i \mapsto h'_i\}_{i\leq n}$ with $H = \{h_1, ..., h_n\}$, $H' = \{h'_1, ..., h'_n\}$ and h'_i fresh.

PROOF: First we show soundness, i.e. l.h.s. is isomorphic to r.h.s. in each rule. I, II and V are obvious. For III, we can easily check $H_1 \cap H_2 = \emptyset$ and $H_1 \cup H_2 \subset \mathbf{V}_{\mathrm{F}}(G)$ iff $H_1 \subset \mathbf{V}_{\mathrm{F}}(G)$, $H_2 \subset (\mathbf{V}_{\mathrm{F}}(G) \setminus H_1)$. For VI, with $f_1 \stackrel{\text{def}}{=} \{h_i \mapsto h'_i\}_{i\leq n}$, we know $(G_1 \parallel G_2)\langle\Sigma'\rangle^{f'} \simeq (G_1\langle\Sigma_1\rangle^{f_1} \parallel G_2\langle\Sigma\rangle^{f}) \simeq (G_1 \parallel G_2\langle\Sigma\rangle)$. IV is proved similarly.

For completeness, by Proposition 3.2 and I, we can claim that $G \simeq G'$, then G and G' can be written as the same normal form [since: we can show that two graphs which can not have the same normal form are not isomorphic by induction on the size of a textual notation]. Therefore if we can rewrite any G to some normal form by $\stackrel{\text{iso}}{=}$ (and $\triangleright$), then the result is obtained. But this is mechanically proved by induction on the size of a textual notation applying $\stackrel{\text{iso}}{=}$ from l.h.s. to r.h.s. $\square$

Now we can check mechanically whether two notations denote the same graph or not.

3.4. Parallel Composition (2). As we already noted, a graph given by $G_1 \parallel_\delta G_2$ can also be composed as $(G_1 \parallel G_2)\langle\Sigma\rangle$. From this, we know that adding this operation into our repertoire cannot increase the number of constructible graphs. A more interesting point however is whether it can make the substitute for the connection operation or not. Before answering this question, we show a small lemma.

Lemma. *For all $G_1, G_2, \langle\Sigma\rangle$, $(G_1 \parallel G_2)\langle\Sigma\rangle$ is also written $G_1\langle\Sigma_1\rangle \parallel_\delta G_2\langle\Sigma_2\rangle$ for some δ, $\langle\Sigma_1\rangle$ and $\langle\Sigma_2\rangle$.*

PROOF: We use the argument with colored handles. Given $G_1 \parallel G_2$, put a (distinct)

color to each set of free handles to be adjoined by $\langle\Sigma\rangle$. Since the graph is still disconnected, we can think of its G_1 element and G_2 element separately. Now, in each subgraph, connect free handles with the common colors until each color appears exactly in one free handle. Call these graphs by $G_1\langle\Sigma_1\rangle$ and $G_2\langle\Sigma_2\rangle$. Finally define δ as partial injection which maps a free handle of $G_1\langle\Sigma_1\rangle$ to a free handle of $G_2\langle\Sigma_2\rangle$ with the same color. Clearly the resulting graph is the same as that which l.h.s. denotes. $\square$

Now we give the answer in the following.

Proposition. *Any process graph can be generated from pre-atomic graphs only using operations* $\langle \|_\delta\,,\ \blacktriangleright\rangle$.

PROOF: By Proposition 3.2, we know any graph can be represented in some normal form such that: $H\blacktriangleright((\mathbf{g}(\mathcal{C}_1)\|...\|\mathbf{g}(\mathcal{C}_n))\langle\Sigma\rangle)$. Next note that composition of a sequence of connections is again a connection. Then applying the preceding lemma and the isomorphism laws starting from a graph in a normal form repeatedly, we get: $H\blacktriangleright(\mathbf{g}(\mathcal{C}_1)\langle\Sigma_1\rangle\|_{\delta_1}...\|_{\delta_{n-1}}\mathbf{g}(\mathcal{C}_n)\langle\Sigma_n\rangle)$. Hence the result since $\mathbf{g}(\mathcal{C}_i)\langle\Sigma_i\rangle$ is pre-atomic. $\square$

However if we start from *atomic* graphs, parallel composition and restriction are not enough to generate all the graphs: just take a pre-atomic agent which is not an atomic agent (such an agent can always be constructed if the base has an atom with more than one arity). This is because the connection *within* an atomic graph cannot be done by parallel composition. For instance, if we use parallel composition, 19 bases are needed to represent any process graphs based on $\mathbf{A_{CC}}$, while only 7 are enough in the original operations: we still keep parallel composition due to its usefulness and its correspondence with the term-based counterpart (cf. 6.1).

4. Graph Relation (1): Reduction over Process Graphs

As mentioned in Section 1, one-step reduction from G to G' can be defined by a triple $\langle G,\ \delta,\ G'\rangle$ where δ is a partial injection from $\mathbf{V}_{\mathrm{F}}(G)$ to $\mathbf{V}_{\mathrm{F}}(G')$. Let us now attempt to extend the observation into the idea of general graph relations in the nameless format. Some related notations: $\mathtt{dom}(\delta)$ and $\mathtt{cod}(\delta)$ denote the set of handles in the domain and the codomain of δ. δ^{-1} denotes the inverse of δ. With $\mathtt{dom}(\delta_2)=\mathtt{cod}(\delta_1)$, $\delta_2\circ\delta_1$ denotes the composition of δ_1 and δ_2. Note that both are correspondences again. $\mathbf{id}(G)$ denotes the correspondence induced from identity isomorphism on G.

4.1. Correspondence and Standard Operations. A *correspondence* is any triple $\langle G_1,\ \delta,\ G_2\rangle$ where δ is a partial injection from $\mathbf{V}_{\mathrm{F}}(G_1)$ to $\mathbf{V}_{\mathrm{F}}(G_2)$. By abuse of notation, we let $\delta,\delta',\delta'',...$ range over the set of correspondences. A *g-relation* is any set of correspondences where if δ is in the set then so is $\rho\circ\delta\circ\rho'$ where ρ and ρ' are correspondences induced by isomorphisms at the domain and co-domain. g-relations will be denoted by $\mathfrak{R},\mathfrak{R}',...$ If G_1 and G_2 are related by δ in a g-relation $\mathfrak{R}$, we write this $G_1\,\mathfrak{R}_\delta\,G_2$ (so $\mathfrak{R}\stackrel{\mathrm{def}}{=}\{\mathfrak{R}_{\delta_i}\}_{i\in I}$).

A *sub g-relation* of a g-relation $\mathfrak{R}$ is a subset of $\mathfrak{R}$ which is again a g-relation. Also note that the set of all g-relations is closed under union, intersection, difference, as well as having the top (called the *universal g-relation*) and the bottom (called the *empty g-relation*). The *identity g-relation* denoted by **ID** is the set of correspondences induced by all isomorphisms.

We define two operations on g-relations, reminiscent of those on usual relations. *Composition* of $\mathfrak{R}$ and $\mathfrak{R}'$, written $\mathfrak{R}'\circ\mathfrak{R}$, is defined: $\{\delta'\circ\delta\mid\delta\in\mathfrak{R}\ \wedge\ \delta'\in\mathfrak{R}'\wedge\mathtt{cod}(\delta)=$

$\mathtt{dom}(\delta')\}$. Similarly the *inverse* of $\mathfrak{R}$, written $\mathfrak{R}^{-1}$, is defined as $\{\delta^{-1} \mid \delta \in \mathfrak{R}\}$. Note that **ID** is a (unique) unit for $\circ$.

Some g-relations have closure properties just as in usual relations. We say $\mathfrak{R}$ is *reflexive* if $\mathfrak{R} \supset \mathbf{ID}$, *transitive* if $\mathfrak{R} \circ \mathfrak{R} \subset \mathfrak{R}$, *symmetric* if $\mathfrak{R}^{-1} = \mathfrak{R}$, and *equivalence* if it is reflexive, transitive, and symmetric. Associated closure operations are called, as usual, the *reflexive* (resp. *symmetric, transitive, equivalence*) closure.

4.2. Compatibility. The notion of compatibility is defined with some side conditions concerning correspondences. We say a g-relation $\mathfrak{R}$ is *compatible* if $G_1 \,\mathfrak{R}_\delta\, G_2$ implies:

(JUX)	$(G_1 \parallel G)\,\mathfrak{R}_{\delta'}\,(G_2 \parallel G)$	with	$\delta' = \delta \cup \mathbf{id}(G)$
(CON)	$G_1\langle\Sigma_1\rangle^{f_1}\,\mathfrak{R}_{\delta'}\,G_2\langle\Sigma_2\rangle^{f_2}$	with	$\delta' \circ f_1 \supset f_2 \circ \delta$
(RES)	$H_1 \blacktriangleright G_1\,\mathfrak{R}_{\delta'}\,H_2 \blacktriangleright G_2$	with	(1) $\delta(H_1) \subset H_2$ and $\delta^{-1}(H_2) \subset H_1$ (2) $\delta' \circ (\mathbf{id}(G_1) \setminus H_1) = (\mathbf{id}(G_2) \setminus H_2) \circ \delta$

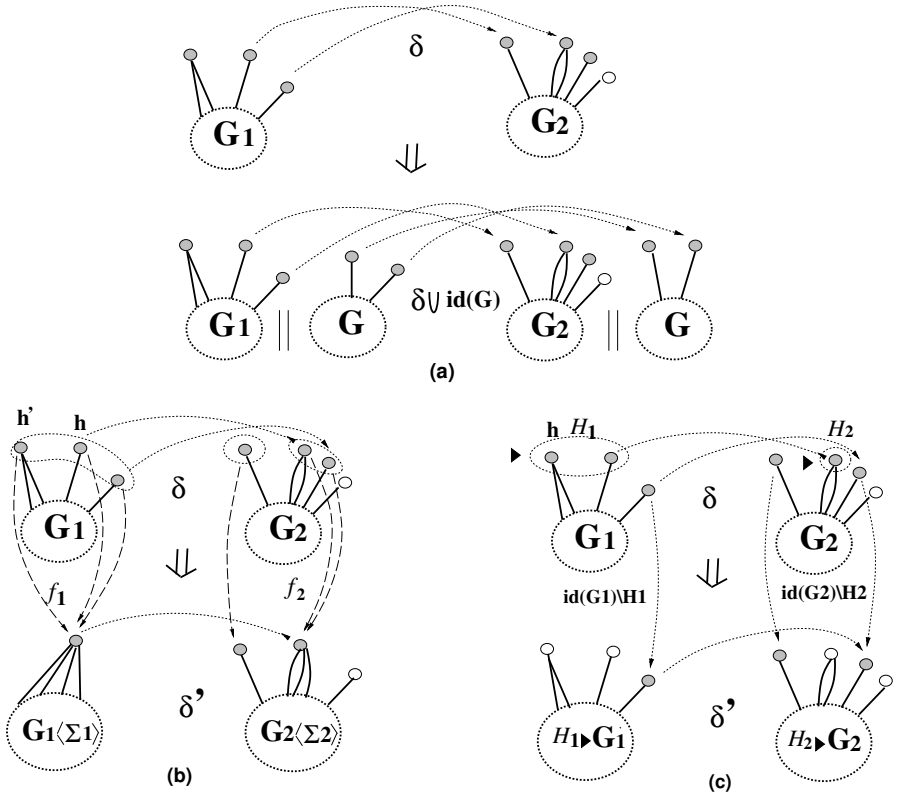

FIGURE 7. Compatibility over g-Relations

We show the idea of this definition in Figure 7. (JUX) illustrated in (a) is easy to understand, while (CON) and (RES) are somewhat cumbersome. (CON) described in (b) says that as far as the part related by δ is concerned, the connections $\langle\Sigma_1\rangle$ and $\langle\Sigma_2\rangle$, and the resulting correspondence δ' should work in a complete accordance with each other, i.e. $\delta'(f_1(h)) = f_2(\delta(h))$ for any $h \in \mathbf{V}_{\mathrm{F}}(G_1)$ s.t. $\delta(h) \in \mathbf{V}_{\mathrm{F}}(G_2)$. The inclusion comes from the observation that we may freely connect the free handles not related by δ, e.g. h' in (b). In (RES) whose picture is in (c), the first condition says that

two hiding operations should function on the part related by δ in the correct way, but unrelated handles (here h) are not considered.[4] The second condition means that the new correspondence δ' is obtained by decreasing hidden handles from the domain of δ.

We say that a g-relation is a *congruence* if it is an equivalence and compatible. Compatible and congruent closures are respective closure operations. Then we can easily check the set of all g-relations is closed under all the operations in this and the previous subsections.

4.3. Reduction. As one of the most significant g-relations in the present formalism, we introduce the process dynamics over graphs, following the original idea as given in [9, 10]: if a pair of atomic graphs with different polarities are connected through their principal handles, the graph is transformed into another graph via a certain correspondence. Each such pair, written by $\mathbf{g}(\mathcal{C}_1^+) \bullet \mathbf{g}(\mathcal{C}_2^-)$, is called *a reduction pair of* $\mathcal{C}_1$ *and* $\mathcal{C}_2$. Then the set of interaction rules with a base $\mathbb{A}$, which is denoted by $\Re$, takes a form, with $\mathcal{C}_1^+$, $\mathcal{C}_2^-$ from $\mathbb{A}$:

$$\mathbf{g}(\mathcal{C}_1^+) \bullet \mathbf{g}(\mathcal{C}_2^-) \; \Re_\delta \; G'$$

where δ should be surjective and, moreover, for each combination of a positive atomic graph and a negative atomic graph, there should be one and only one such rule in $\Re$. As an example, the set of interaction rules for atoms in 2.2 is given in Figure 8.

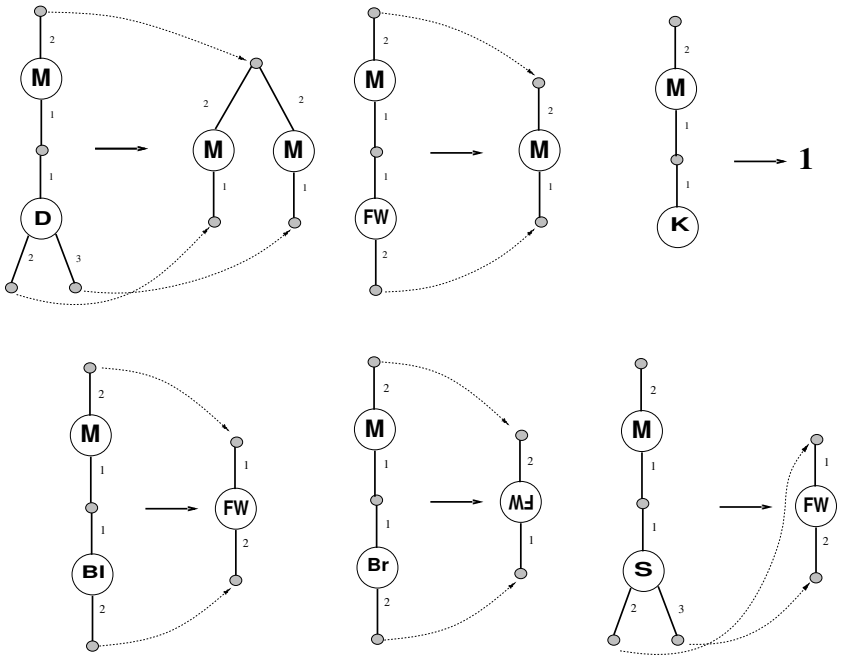

FIGURE 8. Reduction in Concurrent Combinators

Given a certain basis and the interaction rules, say $\Re$, the *one-step reduction* is the compatible closure of $\Re$, denoted by $\longrightarrow$ and the *multi-step reduction*, denoted by $\longrightarrow\!\!\!\!\!\rightarrow$, is the transitive and reflexive closure of $\longrightarrow$.

[4]This is related with the property that the usual binary relations of process calculi are closed under structure rules [1, 17]; especially note $e \blacktriangleright P \equiv P$ with $e \notin \mathcal{FN}(P)$ concerning restriction. Consider the reduction $a \blacktriangleright (\mathcal{M}(ab), \mathcal{D}(acd)) \longrightarrow (\mathcal{M}(cb), \mathcal{M}(db))$ for instance; the unrelated name in r.h.s (here a) can freely be restricted in l.h.s.

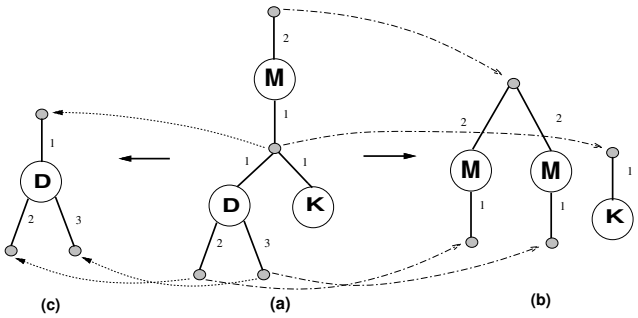

FIGURE 9. Nondeterminism in Process Graphs

Thus we write e.g. $G \to_\delta G'$ which we read: G one-step reduces to G' via the correspondence δ. Similarly for multi-step reduction. We note that a correspondence in $\longrightarrow$ and $-\!\!\twoheadrightarrow$ is always surjective, as is inferred from the definition of compatibility.

4.4. Semi-compatibility. In addition to compatibility, we add the notion of *semi-compatibility*, and, consequently, *semi-congruence*, where connection is replaced by parallel composition. We need some preliminary notations: we call $\{\delta(t) \mid t \in \mathtt{dom}(\delta)\}$ an *image* of δ, written $\mathtt{im}(\delta)$, and $\{t \mid \delta(t) \in \mathtt{cod}(\delta)\}$ a *pre-image* of δ, written $\mathtt{preim}(\delta)$. We write $\delta \asymp \delta'$ when $\delta(h) = \delta'(h)$ for any $h \in \mathtt{preim}(\delta) \cap \mathtt{preim}(\delta')$, and $\delta^{-1}(h) = \delta'^{-1}(h)$ for any $h \in \mathtt{im}(\delta) \cap \mathtt{im}(\delta')$. Note $\asymp$ is equivalence and closed under inverse and composition. Then a g-relation is *semi-compatible* if $G_1 \,\mathfrak{R}_\delta\, G_2$ implies:

$$\text{(PAR)} \quad (G_1 \parallel_{\delta_1} G)\, \mathfrak{R}_{\delta'}\, (G_2 \parallel_{\delta_2} G) \quad \text{where} \quad \delta_2 \circ \delta \asymp \delta_1 \quad \text{and} \quad \delta' = \mathbf{id}(G) \cup (\delta \backslash \mathtt{preim}(\delta_1))$$

together with (RES) in 4.2. We say that it is *semi-congruence* if it is semi-compatible and equivalent. The set of g-relations is also closed under associated closure operations. This is used when we study the correspondence with the arbitrary congruence (possibly without name-substitutivity) in the term-based theory, cf. 6.1.

4.5. Important Remark. The reduction in process graphs has essential features of concurrent processes: (1) computation with sharing (or, equivalently, interference [13]), and, as a consequence, (2) non-determinism. They come from arbitrary connection of free handles using (CON) rule. Figure 9 (a) depicts the situation where the principal handle of a message is connected with those of two negative atomic graphs. The reduction inevitably becomes non-deterministic. (b) is the case when a message and a duplicator interact, while (c) shows a case when a message and a killer interact, resulting in distinct (and semantically different) configurations accordingly.

Now we proceed to some examples comparing our graph representation with Lafont's Interaction Nets [14], which can be regarded as a special form of process graphs without multiple connection of principal handles.

4.6. SK-combinators (1): Interaction Nets. We shall show the "interactive" version of Curry's classical combinators, where an interaction between atoms is always done via a hidden handle, to which only one negative atom and one positive atom are connected. Thus this system is exactly one of Lafont's Interaction Nets.[5] In this system, the original combinators **S** and **K** are decomposed into three or two atoms, respectively. Moreover we need to explicitly represent "Copy" and "Kill" of terms. Following this, all

[5]This encoding is the exactly same one as Simon Gay gave in [2], though it was independently defined and has been published in [6].

nine atomic graphs are defined in Figure 10 (a), and the interaction rules are specified in (b). In this encoding we use a generation of a linkage called a (generalised) forwarder: this is because a direct identification of two ports is prohibited in the present scheme (cf. 6.2). In this system **K** is regarded as an agent which just makes a link to the first argument.

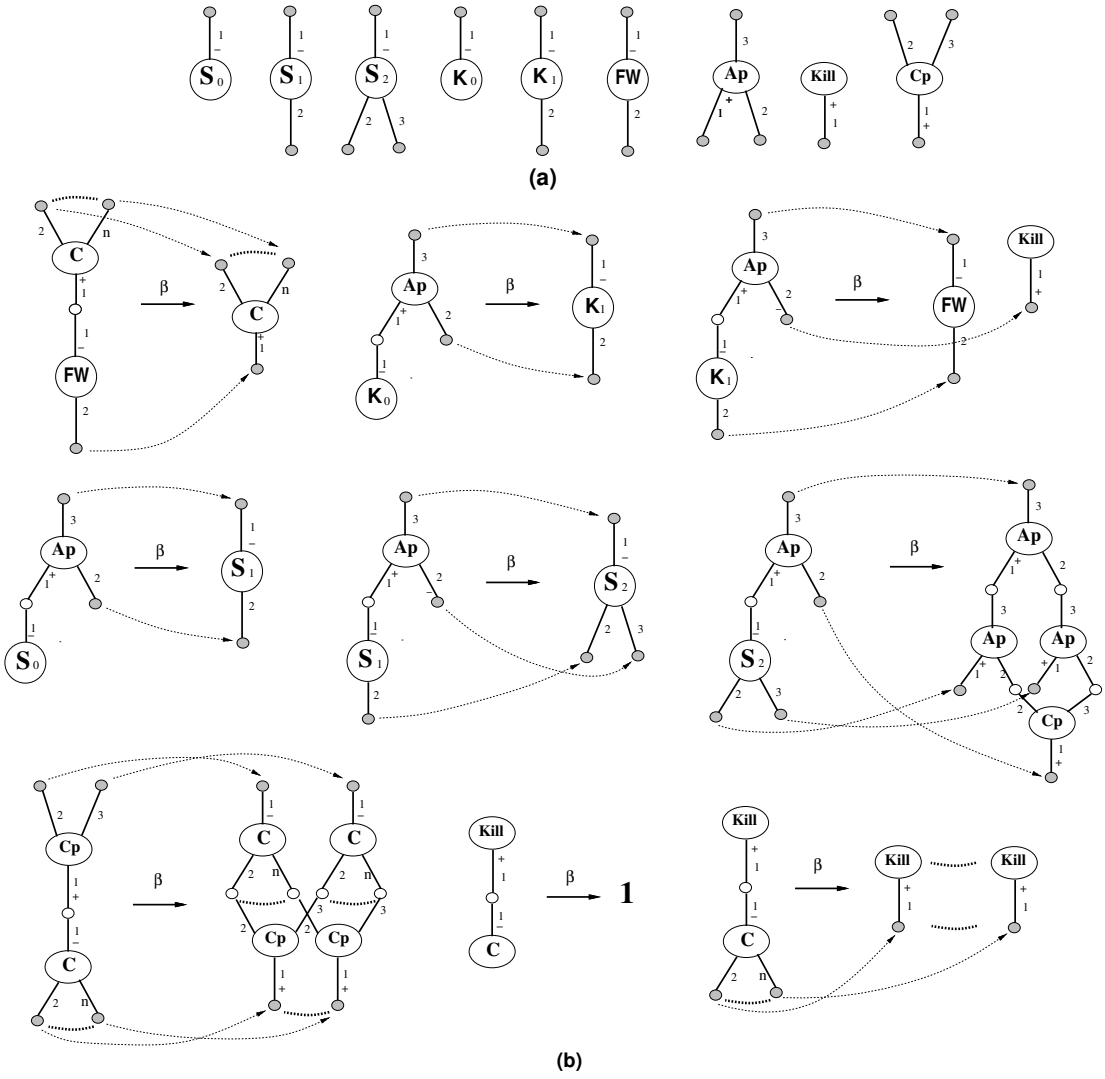

FIGURE 10. Interaction Net (SK-combinators)

Now let us map a sequential combinator M to a process graph inductively; **S** and **K** are mapped to "S_0" and "K_0" respectively, and the application MN to a graph in which the principal handles of two graphs representing M and N are connected with the first handle and the second one of "Ap" respectively, and they are finally hidden.

4.7. SK-combinators (2): Shared Version. In the above example, "S_0" and "K_0" cannot be shared. But by allowing overlapped free handles, we easily define a shared version of SK-combinators in Figure 11. Interestingly we do not need explicit "Copy" and "Kill" agents (so the numbers of atomic graphs and reduction rules decrease), the persistent behaviour of the shared negative atomic graphs is required. In (b), "$!\mathcal{FW}$" is a persistent (generalised) forwarder which links two handles and still remains after interaction.

The mapping of a sequential combinator to a process graph is given in the same way as in 4.6, though, in the new version, we can share "S_0" and "K_0" arbitrarily.[6]

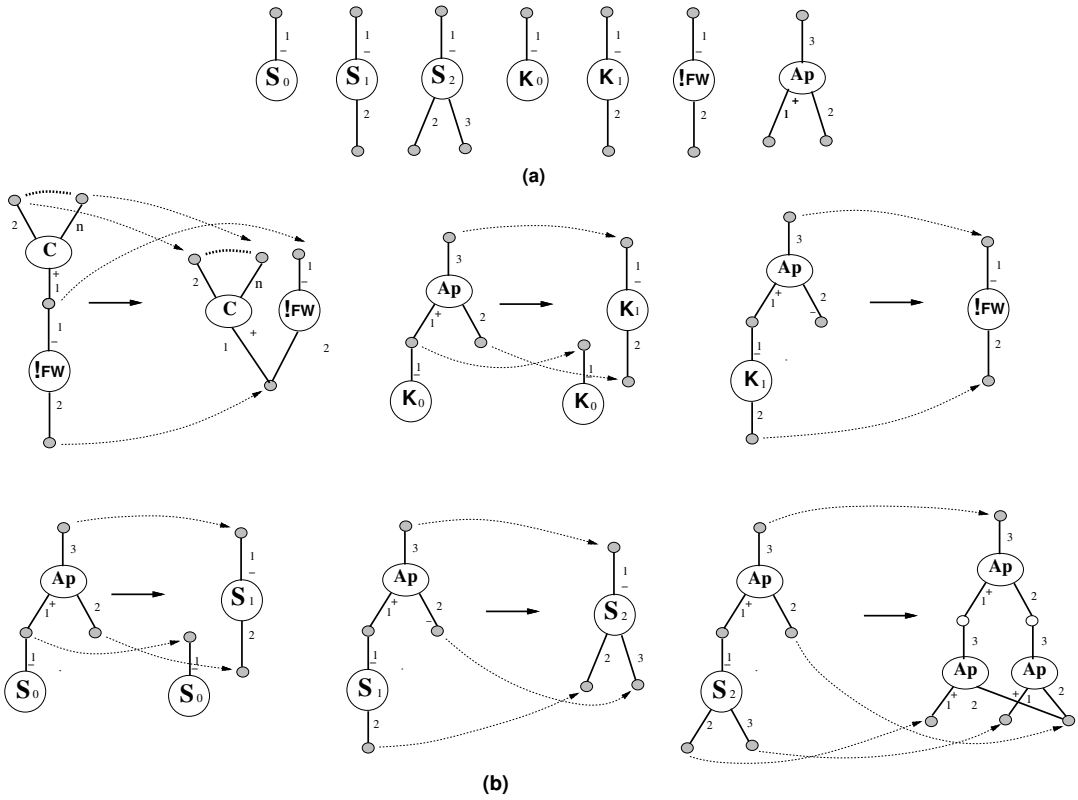

FIGURE 11. Shared SK-combinators

4.8. Boolean Expression. We show an encoding of Parallel Or which may not be represented as Lafont's Interaction Nets, but can be naturally described as our process graphs. First we introduce the following boolean atomic graphs in Figure 12 (a) and their interaction rules in Figure 12 (b). Note that we again use the persistent forwarder. Then Parallel Or is encoded as in Figure 12 (c) by simply connecting two free handles of two "OR" atomic graphs (this is not allowed in Interaction Nets). Thus "false or true" is reduced like (d); intuitively this is equated by "true" because two persistent forwarders

[6]We can easily extend this encoding to share any application (e.g. (**SK**) in ((**SK**)(**SK**))) by defining that S_1, S_2 and K_1 are persistent.

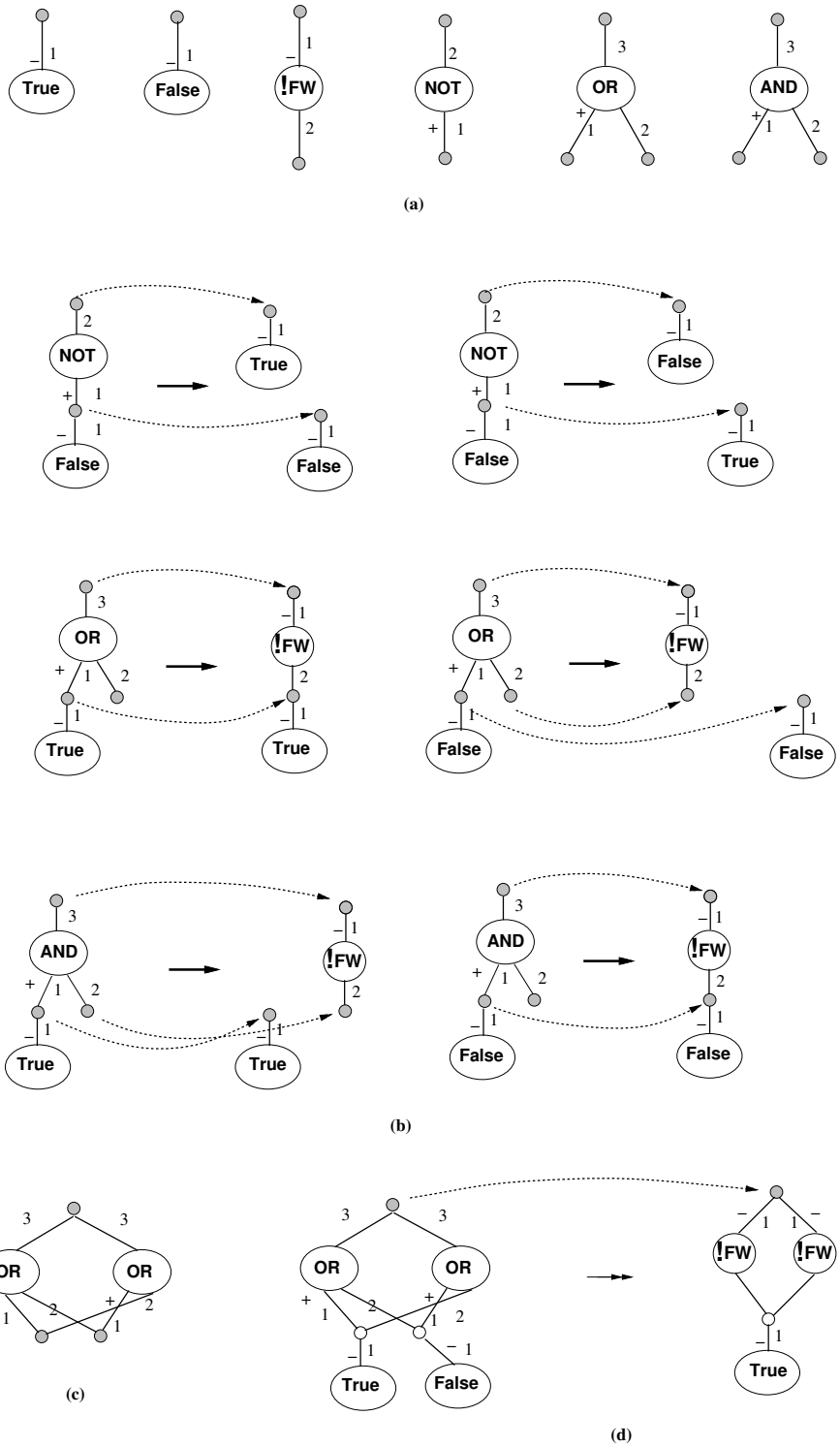

FIGURE 12. Boolean Agents

function as linkers and "false" becomes an inactive agent. The technical development of such an equality will be discussed in the next section.

5. Graph Relation (2): Behavioural Equivalence over Process Graphs

The notion of behavioural equivalence over processes is the most fundamental element in process formalisms, originating from observational equivalence for CCS in [16]. For the process graphs to be a meaningful formalism for concurrency, we need some ways of relating agents based on their behaviour. This is done in the present section, using the notion of congruential g-relation defined in the previous section. Just like some relations over terms in a term algebra such as bisimulations, some g-relations are behaviourally significant. We should notice that unlike reduction, a correspondence in a congruence may *not* be surjective (cf. garbage collection in 5.2). At the end, we present an interesting general property of any congruence class.

5.1. Behavioural Equality for Process Graphs. To introduce the notion of the canonical behavioural equality over graphs based on the idea in [8, 9, 10], we need two conditions: the first condition, the *reduction-closure*, says that the equality is in harmony with the notion of reduction, while the second, the *action predicate*, filters only behaviourally meaningful congruences (cf. [19]).

(1) A congruence $\overset{s}{\cong}$ is *reduction-closed*, if $G_1 \overset{s}{\cong}_{\delta} G_2$ and $G_1 \twoheadrightarrow_{\delta_1} G_1'$ implies, for some G_2', $G_2 \twoheadrightarrow_{\delta_2} G_2'$ and $G_1' \overset{s}{\cong}_{\delta'} G_2'$ with $\delta' = \delta_2 \circ \delta \circ \delta_1^{-1}$.

(2) The *action predicate* is give as: $G \Downarrow_{h^\theta} \overset{\text{def}}{\Leftrightarrow} G \twoheadrightarrow_{\delta} G' \wedge \delta(h) \in \mathbf{Act}_\theta(G')$ (note $h \in \mathbf{V}_{\mathrm{F}}(G)$) where $\mathbf{Act}_\theta(G)$ is a set of free principal handles connected with some $\mathbf{g}(\mathcal{C}^\theta)$. Now let us call a congruence $\overset{s}{\cong}$ *respects the action predicates*, if $G_1 \overset{s}{\cong}_{\delta} G_2$ and $G_1 \Downarrow_{h^\theta}$ implies $G_2 \Downarrow_{\delta(h)^\theta}$.

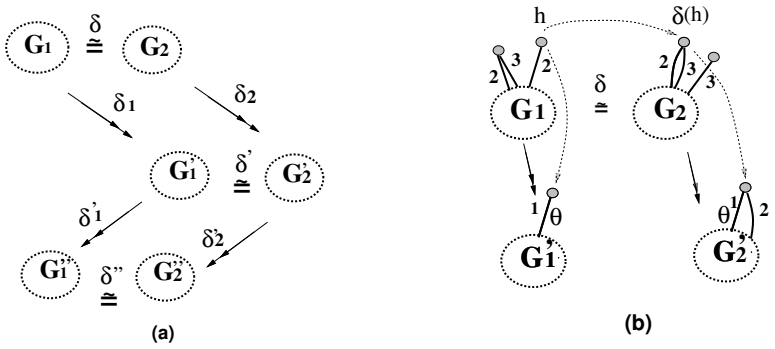

FIGURE 13. Reduction-closure and Action Predicate

(1) makes it clear that the notion of equality should be consistent with the notion of state change induced by reduction: e.g. in Figure 13 (a), if G_1 and G_2 are equated and G_1 reduces to any graph, then G_2 should reduce to an equated graph again. (2) fits the basic idea of actions in the present computational framework. As indicated in (b), if some free handle in G_1, say h, can be used to interact with $\mathbf{g}(\mathcal{C}^{\bar{\theta}})$ after multi-step reduction, then the related handle $\delta(h)$ in G_2 can also play the same role ($\bar{\theta}$ is the inverse of θ).

The basic result of the equality defined with the above notions follows.

Proposition. *We say that a congruence is* $\Downarrow_{h^\theta}$-sound, *or simply* sound, *if it is reduction closed and respects the action predicate. Then there is the maximum* $\Downarrow_{h^\theta}$*-sound congruence.*

PROOF: Note that, in the congruent closure of any family of congruences, a correspondence there can always be factored out by a chain of correspondences in the original congruences (this is essentially *Chain Lemma* in [8]). Using this, an easy reasoning based on commutativity of maps proves that the congruent closure of any family of reduction-closed congruences is again reduction-closed. Similarly for the condition on the action predicate. □

Some concrete sound congruences might be interesting.

5.2. Example. Let us see what is done in Interaction Net using the notion of congruence over g-relations. Suppose we are given two atomic graphs with different polarities, and construct the graph by connecting them only by their principal handles and then hide only that handle only. Now we have a reduction from such a graph, and take the congruent closure of such correspondences in any system of process graphs. We call this congruence a *simple* β*-equality* denoted by $\overset{s\beta}{=}$. Figure 14 (a) depicts one example of an equation by $\overset{s\beta}{=}$ (cf. Example 4.6). With the reasoning in the same way as that of Proposition 3.7 in [9], we get:

Proposition I *In any system of process graphs,* $\overset{s\beta}{=}$ *is sound.*

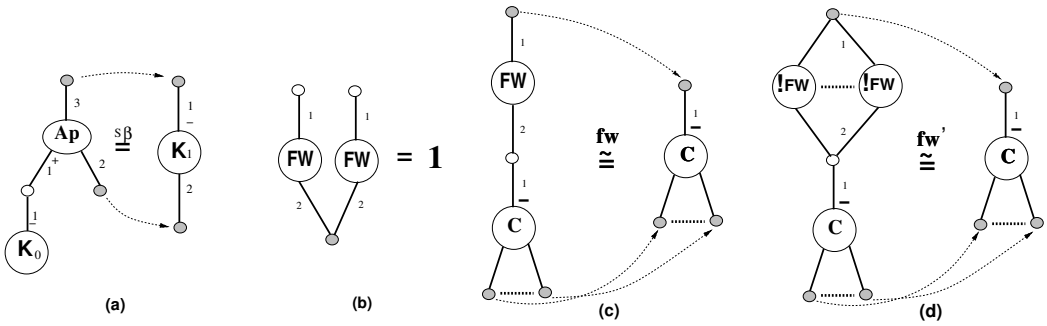

FIGURE 14. Behavioural Equalities

As another example, an idea frequently found in graph-based formalisms is *garbage collection*. To offer a process theoretic foundation to "garbages" in these examples, the following will give a tentative but useful answer.

Proposition II *Given a process graph* G*, if* G *satisfies one of the following conditions, then the congruent closure of the equation* $G =_\emptyset \mathbf{1}$ *is always sound.*

(1) G *does not one-step reduce, and* G *has no free active handles, or*

(2) G *has no free handles.*

Figure 14 (b) shows an example of garbage collection (1). More refined treatment will allow garbage collecting divergent configuration with free handles, which however would let us lose effective judgment of garbages.

Finally due to the existence of forwarders, the full justification of the examples needs the equalities from the two laws in Figure 14 (c) and (d) where $\mathbf{g}(\mathcal{C}^-)$ in (c) is erased after interacting with any positive agent, while $\mathbf{g}(\mathcal{C}^-)$ in (d) is persistent. Then we have:

Proposition III *The congruent closure of the equations given by Figure 14 (c) and (d) is sound.*

Now we recall Examples 4.6, 4.7 and 4.8. The easy inductive reasoning gives us if classical combinators M and N are weak-β equated, then the encodings of M and N defined in 4.6 are also equated by the maximum sound equality using Proposition I and III-(c). Similarly for the encodings of 4.7 by Proposition II and III-(d). Finally in Example 4.8, we can check "false or true", "true or false", or even "true or Ω" is "true" up to the maximum sound equality over boolean agents by Propositions II and III-(d).

We hope that the present construction over g-relations offers a rigorous basis for reasoning about various (static and dynamic) structures of concurrent computation in the graphical framework.

5.3. The Minimum Graph in a Congruence Class. Finally we show one important result in the general congruences: there is the minimum representation in some sense in any congruence class (e.g. l.h.s is simpler than r.h.s in each congruence in Figure 14). Let $\cong$ be a congruence. *A congruence class* from $\cong$ is a set of process graphs which is closed under $\cong$ and in which any two graphs are mutually equated by $\cong$.

Proposition. *Given any $\cong$ and any congruence class from $\cong$, say Φ, there is a graph $G \in \Phi$ such that $G' \in \Phi \;\Rightarrow\; |\mathbf{V}_{\mathrm{F}}(G)| \leqq |\mathbf{V}_{\mathrm{F}}(G')|$. Moreover, if we have $G_1 \cong_{\delta_1} G \cong_{\delta_2} G_2$ for such G, then δ_1 and δ_2^{-1} are always surjective and $\mathtt{im}(\delta_1) = \mathtt{preim}(\delta_2)$.*

PROOF: Take the set of correspondences in $\cong$ which equate the graphs in the congruence class. Out of the set, take any element whose cardinality (i.e. the number of handles in its pre-image) is the minimum among them (any such element is fine). Call it δ. Then, for some G_1 and G_2, we have $G_1 \cong_\delta G_2$. If $|\delta| \neq 0$, using (CON) rule of congruence, we can adjoin all free handles to those in the pre-image of δ in G_1. Let the result be G. Then we have $G \cong_{\delta'} G_2$ with $|\delta'| = |\delta|$. Now suppose there is some G' in the congruence class, for which we have $|\mathbf{V}_{\mathrm{F}}(G')| \lneqq |\mathbf{V}_{\mathrm{F}}(G)|$. Then $G' \cong_{\delta''} G$ for some δ'' but then $|\delta''| \leqq |\mathbf{V}_{\mathrm{F}}(G')| \lneqq |\mathbf{V}_{\mathrm{F}}(G)| = |\delta|$ which contradicts the way of selecting δ. Thus $|\mathbf{V}_{\mathrm{F}}(G)| \leqq |\mathbf{V}_{\mathrm{F}}(G')|$ for any G' in the congruence class. Moreover, by the above discussions, if we have $G_1 \cong_{\delta_1} G \cong_{\delta_2} G_2$, the conditions clearly hold. If $|\delta| = 0$, using (RES) rule of congruence, we hide all the free handles of G_1 and call the result G, for which we still have $G \cong_\delta G_2$. Take this G and it clearly satisfies the conditions. □

Note the proof uses connection and hiding in our graph-based formalism essentially. The result says that if the number of free handles in G is the minimum in a congruence class, all free handles in G are always connected by correspondences in the congruence class. Hence one may define some group of the process graphs as the minimum representation by checking firstly the number of *free handles*, secondly *atomic vertices* and finally *hidden handles*.

6. Discussions

In this section, we first summarise our result on the correspondence between the graph representation and term-based processes, as developed in [11]. Then we give comparisons with some related work and discuss further issues.

6.1. Correspondence with Term-Based Theory. In view of our intention to study the nameless representation, it should be interesting to see how the concurrent computation in this graphical framework commutes with the one in the world which uses names, and also differs from it, so that both enrich our understanding about concurrency.

As the first step, to forget names from the terms, one can observe that there are not any essential "structural" difference between P and $P\sigma$ where σ is *a name permutation*, a one-one map over names. But the essential idea is not only processes and their composition, but *relationship* between (or predicate on) processes. Since the notion of operations in the graph-based theory itself does not conform to the framework of the Σ-algebra (it does not even induce so-called clones of operations), such mappings can *not* be a homomorphism in the usual sense. But what matters is, as in $\mathcal{M}(ab) \approx c\blacktriangleright(\mathcal{FW}(ca), \mathcal{M}(ab))$, *correspondence of names* between two terms. Thus we introduce the idea of the *name-coherent* relations over terms (i.e. relations for which $P\,\mathcal{R}\,Q$ implies $P\sigma\,\mathcal{R}\,Q\sigma$ for any name-permutation σ, and which are also closed under $\equiv$). Notice that the most of standard congruences in process calculi (e.g. bisimilarities in CCS) are name-coherent.

The set of name-coherent relations forms a boolean lattice which is closed under the boolean and relational operations. We note that the set of g-relations also forms a boolean lattice equipped with all the operations in 4.1, 4.2 and 4.4. Let $\mathbb{A}$ be fixed and call the two lattices NCoh and gRel respectively. Then the main result in [11] says: *With the common bases, there are mutually inverse bijections between* NCoh *and* gRel *such that:*

- *There are isomorphisms between two algebras w.r.t. the respective operations with the same names defined in* 4.1.
- *The compatible closure operation in* NCoh *is mapped to the semi-compatible closure in* gRel *and the compatible, name-substitution closure in* NCoh *to the compatible closure in* gRel. *Similarly for the inverse. The similar result is established by replacing (semi-)compatible with (semi-)congruent.*

The isomorphism at the level of the standard operations gives us the correspondence on such important notions as reduction, behavioural equivalences etc. between two domains. Note also that congruence notions in two domains are related, where congruence in graphs corresponds to congruence in terms which is also closed under general name substitution. Note that some "good" process equivalences indeed require that they are name substitutive (e.g. [20, 25]), so that this notion deserves further study. One of the most important remarks is the maximum sound equality in the system of concurrent combinators called $\mathbf{cc}_\varphi$ in [10] is name-substitutive. This result says there is a leaner mathematical representation in which truly finite (at most 27) atoms and fixed interaction rules can operationally encode mobile processes (embedding of related calculi, e.g. Polyadic π-calculus [18] and CHOCS [26] as well as λ-calculus naturally follow) up to the sound equality proposed in 5.1.

6.2. Related Work and Further Issues. From the original exposition on concurrent processes [16], Milner developed his formalism both in the usual syntax and in the corresponding graph-based formalism called flow graphs [15]. The flow graphs use port names and thus their algebra is homomorphic to the term-based one. Flow graphs are used in recent literature as an aid to grasp configuration of processes intuitively,

e.g. [20]. Milner's recent π-*net* [21, 22] is another graphical representation of processes where the algebra and dynamics of the several versions of the π-calculus are graphically represented, which correspond to their formulations in action structure [21], using port names essentially. Parrow's Interaction Diagram [23] also presents the algebra and computation of π-calculus in a graphical format. In both cases analysis of computational structures is done making use of graphical ideas. An apparent difference between the present work and these graphical representations is, while theirs basically aim at faithful representation of the original term-based theory in a graphical format (though with additional insights), the present work tries to establish a formal theory of nameless processes where the graphical objects are formalised in an undiluted form, avoiding maybe sometimes unnecessary proliferation of entities (e.g. the number of atoms). Technically speaking, we can regard graphical representation with names as standing between terms and nameless graph representation, corresponding to terms modulo structural rules introduced by Berry and Boudol [1].

We already gave some discussions on Lafont's Interaction Net [14] in 4.6 and 5.2. In Lafont's work, the idea of named ports is not explicitly present, not because it has been abstracted away but because arbitrary connection among agents is not used. Furthermore Lafont does not develop algebraic structures underlying his construction, so the direct comparison is difficult. Nevertheless it suggests one of the restricted but disciplined ways of connecting agents. We hope that the formal study of various modes of connection in concurrent systems would be encouraged by the conscious formalisation of the notion in the present theory, possibly leading to effective engineering principles (cf. Sections 4 and 5). The work by Cliff Jones [13] (where the notion called *object graphs* is studied) may be suggestive in this respect.

One may also note that, while not oriented towards modeling graphical processes, the basic idea of Chemical Abstract Machine [1] naturally emerges in our graphical representation. In fact, a process graph is formulated as a multi-set of atomic graphs (see 3.2) and the scheme of dyadic interaction fits the idea of *reaction* without using the (somewhat unnatural) rule called *heavy ion* which forces to transfer a prefix. As seen in the isomorphism laws in 3.3, our graphs can successfully represent terms modulo structural rules *and* any name-permutations, though we need to introduce g-relations to reveal the role of names in the nameless word.

For another but related point, there are properties of a graph-based theory which *cannot* be easily translated into a term-based theory. For example, collapsing two *free* handles during reduction seems difficult to treat in the term-based framework (this relates to the use of *forwarder* in 4.6). Also we easily extend the definition of reduction so that correspondence is not surjective. Besides [5] suggests not only the untyped version (where atoms which have the same polarity may interact) but also an infinite number of atoms for Interaction Nets. The theoretical investigation of the generalisation to such graphs is one of interesting topics.

Finally our work can be regarded, in one aspect, as an investigation to find the analogue of SK-combinators in the concurrency setting, possibly sharing intention with several recent work including Raja and Shymasundar [24] and Simon Gay [3]. From this perspective, the semantic basis of the present graph-based theory would be of particular interest. One may refer to [7] for some work in this direction.

Acknowledgments. The author gratefully acknowledges helpful discussion with Kohei Honda on a lot of points on this paper. She is partially supported by JSPS Fellowship for Japanese Junior Scientists.

References

1. Berry, G. and Boudol, G., The Chemical Abstract Machine. *Theoretical Computer Science*, vol 96, pp. 217–248, 1992.
2. Gay, S., Interaction Nets, Diploma in Computer Science in Queens', 1991.
3. Gay, S., Combinators for Interaction Nets, draft, July 29, 1994.
4. Girard, J.-Y., Linear Logic, *Theoretical Computer Science*, Vol. 50, pp.1–102, 1987.
5. Gonthier, G., Abadi, M. and Levy, J.-J., The Geometry of Optimal Lambda Reduction, *POPL'92*, pp.15–26, ACM press, 1992.
6. Honda, K., *A Study of ν-calculus and its Combinatory Representation*, Phd Thesis in the Department of Computer Science, October, 1994.
7. Honda, K., *Notes on P-Algebra 1: Process Structure*, in this volume.
8. Honda, K. and Yoshida, N., On Reduction-Based Process Semantics, *13th. FST/TCS*, LNCS 761, pp.371–387, Springer-Verlag, 1993.
9. Honda, K. and Yoshida, N., Combinatory Representation of Mobile Processes, *POPL'94*, pp.348–360, ACM press, 1994.
10. Honda, K. and Yoshida, N., Replication in Concurrent Combinators, *TACS'94*, LNCS 789, pp.786–805, Springer-Verlag, 1994.
11. Honda, K. and Yoshida, N., *Graphs for Connection and Interaction*, the early version of this paper, May 1994.
12. Honda, K. and Yoshida, N., *Nameless Concurrent Combinators*, under preparation, December 1994, to appear as a Keio-CS-report.
13. Jones, C.B., Process-Algebraic Foundations for an Object-Based Design Notation, *Technical Report UMCS-93-10-1*, Manchester University, 1993.
14. Lafont, Y., Interaction Nets, *POPL'90*, pp. 95–108, ACM press, 1990.
15. Milner, R., Flowgraphs and Flow Algebras, *Journal of ACM*, 26(4), pp.794 – 818, 1979.
16. Milner, R., *A Calculus of Communicating Systems*, LNCS 76, Springer-Verlag, 1980.
17. Milner, R., Functions as Processes. *Mathematical Structure in Computer Science*, 2(2), pp.119–146, 1992.
18. Milner, R., Polyadic π-Calculus. *Logic and Algebra of Specification*, Springer-Verlag, 1992.
19. Milner, R. and Sangiorgi, D., Barbed Bisimulation. *ICALP'92*, LNCS 623, pp.685–695, Springer-Verlag, 1992.
20. Milner, R., Parrow, J.G. and Walker, D.J., A Calculus of Mobile Processes, *Information and Computation* 100(1), pp.1–77, 1992.
21. Milner, R., Action structure for the π-calculus. Research Report ECS-LFCS-93-264, Department of Computer Science, University of Edinburgh 1993.
22. Milner, R., Notes on actions I: π-nets with boxes and replication. Edinburgh, June 1993.
23. Parrow, J.G., Interaction Diagrams, *REX'93.*, LNCS, Springer-Verlag, 1993.
24. Raja, N. and Shymasundar, R.K., *The Next 700 Combinatory Representations of Mobile Processes*, a manuscript, 1994.
25. Sangiorgi, D., A Theory of Bisimulation for the π-calculus. *CONCUR'93*, LNCS 715, pp.127–142, Springer-Verlag, 1992.
26. Thomsen, B., A calculus of higher order communicating systems. *POPL'89*, 1989.