Pi-nets: a graphical form of π-calculus [1]

Robin Milner

Laboratory for Foundations of Computer Science
Computer Science Department, University of Edinburgh
The King's Buildings, Edinburgh EH9 3JZ, UK

Abstract An action calculus which closely corresponds to the π-calculus is presented in graphical form, as so-called *π-nets*. First an elementary form of π-net, with no sequential control, is presented. Then, using a construction by Honda and Tokoro, it is shown informally that by adding a single control construction **box** to elementary π-nets, the sequential control present in the π-calculus can be recovered. (Another construction, **rep**, provides replication.) The graphical presentation suggests a few interesting variants of this control regime, which are studied briefly. The main purpose of the paper is to explore informally the power and utility of graphical forms of the π-calculus, in the context of action calculi. It also suggests that graphical forms of other action calculi should be explored.

1 Introduction

Action structures [4] were defined as a framework for studying various notions of concurrent interactive behaviour, in the hope of yielding some taxonomy for these notions, and some uniformity in their presentation. The prime ingredients of an action structure are its *actions*. Each action a has a *source arity* m and a *target arity* n, and we write $a : m \rightarrow n$. These arities m, n are elements of a monoid, which for this paper may be taken to be the natural numbers under addition. Roughly, $a : m \rightarrow n$ is a (perhaps complex) process into which m data are fed and from which n data may be extracted. We say more later about the algebraic properties of action structures.

In a later paper [6][2] an action structure called PIC was defined in the spirit of the π-calculus [8]. Roughly, PIC is what remains of the π-calculus when we remove *replication* (i.e. the power of infinite computation) and also *guarding* – represented by the dot in the process $x(y).P$ for example. An action $a : m \rightarrow n$ of PIC is essentially a collection of π-calculus particles, each being either an input particle $x(y)$, an output particle $\overline{x}\langle z\rangle$ or a restriction particle $\boldsymbol{\nu} x$; it also *imports* m names and *exports* n names. PIC has a simple and illuminating graphical presentation; we call the graph which represents an action a *π-net*.

PIC is not the whole π-calculus, since it cannot simulate the power of guarding or replication; but it was later shown [7] that PIC can be freely extended within the framework of action calculi (a subclass of action structures) by the addition of two so-called *control* contructions: **box** for guarding, and **rep** for replication. (Henceforth we

[1] This work was done with the support of a Senior Fellowship from the Science and Engineering Research Council, UK.

[2] A revised version of the two cited papers [4, 6] will appear as Parts I and II of *Action structures and the π-calculus*, in the Proceedings of the NATO Advanced Study Institute on **Proof and Computation** held at Marktoberdorf in 1993.

shall use the term “boxing” to mean some variant or other of guarding.) The resulting action calculus $\mathsf{PIC}(\mathbf{box}, \mathbf{rep})$ has expressive power which is –informally speaking– comparable with that of the original π-calculus, and also has the extra structure conferred by the algebraic theory of action calculi.

The purpose of this paper is to explore and develop the *graphical* presentation of $\mathsf{PIC}(\mathbf{box}, \mathbf{rep})$. We shall see that these extended π-nets throw light on the structure of interaction. In particular, the graphical presentation suggests several alternative ways in which boxing may constrain reaction.

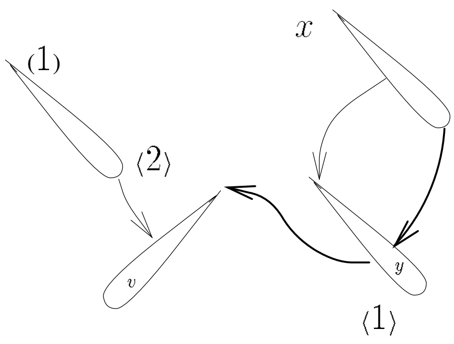
The paper can be read on its own, though the reader may find that the papers [4, 6] provide helpful background. There are similarities between π-nets and Parrow's interaction diagrams [9], though the work was done independently. The differences mainly arise from the fact that the π-nets form an action structure.

Outline: In Section 2 we define the π-nets of PIC informally, by means of an example, and explain how they *react* (their dynamics). In Section 3 we review the notion of action structure, and define the algebraic operations upon π-nets which make PIC an action structure. In Section 4 we enrich π-nets by adding boxing and replication; this yields a version of $\mathsf{PIC}(\mathbf{box}, \mathbf{rep})$ whose control regime does not permit reaction within a box. This regime is quite close to that of the original π-calculus. In Section 5 we illustrate it by performing the elegant construction by Honda and Tokoro [3], which shows that a monadic π-calculus with only input (not output) guards has the full power of polyadic π-calculus [5]. In Section 6 we relax the regime to allow reaction within a box; this is in the spirit of the *solutions* in the Chemical Abstract Machine [2], and still supports the Honda-Tokoro construction. Somewhat surprisingly, we find several variants of this relaxed regime. This is taken to be justification of the use of graphical representation; some variants are suggested by using π-nets which are not otherwise obvious. We end in Section 7 with brief reflections and suggestions for further work.

2 π-nets defined by example

The π-nets of PIC are simple graphical objects. In this section we define them informally by means of an example; we also show how reaction works in π-nets.

Below is an action $a : 1 \to 2$ in PIC, together with its formal expression:

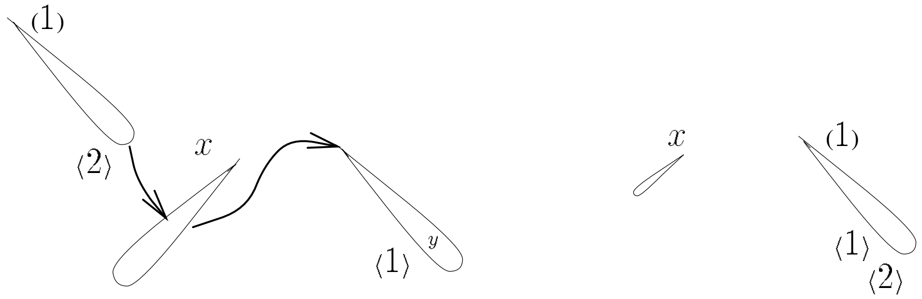


$$a \;=\; (z)\,[\, x(y)\, \underline{\overline{y}\langle x\rangle \; y(v)}\; \overline{v}\langle z\rangle \,]\,\langle yz\rangle$$

Each name (free or bound) in the expression for a corresponds to a node (a torpedo) in the π-net, so there are four nodes. But the only free name in a is x, so x is the only name which occurs properly in the π-net; it appears as a *name tag* at the *tail* (the sharp end) of its node. The labels v and y are present just to aid the eye; an arc or arrow leading to the tail of a node indicates that it corresponds to a bound name. An *imported* name such as z here is indicated by an *import tag*, e.g. (1); an *exported* name such as y or z here is indicated by an *export tag*, e.g. $\langle 1 \rangle$. Thus the number of such tags is determined by the source and target arities respectively.

Each arc in the π-net corresponds to an i/o particle of a. For an input particle like $y(v)$ the arc goes from the *waist* of a node to the tail of a node, while for an output particle like $\overline{y}\langle x \rangle$ it goes from the *head* (the blunt end) of a node to a waist. A pair of arcs such as these which have the same *port* node (the one whose waist they impinge upon) form a *redex*; the redex is shown by underlining in the expression and by thicker lines in the net.

To *reduce* a redex, we simple remove the two arcs and coalesce the *source* and *target* nodes (x and v in this case). This forms a single node which carries all the items (arcs, tags) of both. We write $a \searrow^1 a'$ for such a single reduction, or *reaction*. The net a' which results in this case is shown in the next picture; it too has a redex, so we have another single reaction $a' \searrow^1 a''$. We show a'' too.



$$a' \;=\; (z)\,[\,\underline{x(y)\,\overline{x}\langle z\rangle}\,]\,\langle yz\rangle \qquad\qquad a'' \;=\; (z)\,\langle zz\rangle$$

Note that the named node x in a'' is still present. Such named nodes, bearing no arcs or other tags, make no difference and can be present or absent.

We have almost fully defined π-nets, by giving the above example. A more formal definition appears in [6], and we shall not need it here; we merely emphasize the following points:

- Arcs always join a waist to a head or tail of a node.
- A node tail bears at most one of an arc, a name tag or an import tag. If it bears none of these it corresponds to a restriction particle $\boldsymbol{\nu} x$ in a PIC expression.
- In a π-net of arity $m \to n$, each import tag i ($1 \le i \le m$) or export tag j ($1 \le j \le n$) occurs exactly once.

Thus the waist or head of a node may bear many arcs, and the head may bear many export tags.

To appreciate the difference between π-nets and the processes of the π-calculus [8], let us compare the net $a \in \mathsf{PIC}$ with the process P of the π-calculus in its standard notation:

$$P \;=\; x(y).(\overline{y}\langle x\rangle \mid y(v).\overline{v}\langle z\rangle)\;.$$

There are several points:

1. Note that a is abstracted (upon z), but P is not. Abstraction of port-names was not considered in the original π-calculus of [8]. Abstractions such as $(x)P$ were introduced for the π-calculus in [5], but they were not allowed to react (as $a \in \mathsf{PIC}$ reacts).

2. Further, a exports results (the name-vector $\langle yz\rangle$), since actions are the arrows of an action structure and so possess a target arity. A notion of *sorting* for processes was introduced in [5] –thus $(x)P$ would have sort (1)– but this corresponds only to source arity (the number of imports), not target arity (the number of exports).

3. The input prefix $x(y)$ in P must react (with something else) before a reaction in the body of P can occur. This guarding is at once a strength and a weakness of π-calculus; it allows sequence to be imposed upon actions, but this sequential control is present for *all* i/o particles. The action structure PIC on the other hand imposes no sequencing, except that a reaction may induce an identification of names which enables another reaction. Thus the fact that the redex of a has a bound port (y) does not prevent its reaction; on the other hand the redex of a' only exists because of the first reaction.

Note that a does not include any polyadic particles, like $x(y_1 y_2)$ or $\overline{x}\langle y_1 y_2\rangle$. Such particles were not present in the original π-calculus [8], but were introduced in [5]. To represent them in π-nets we require a kind of multi-arc. We shall have more to say later about whether multi-arcs are necessary, or whether their power can be gained by other means. For the moment we ignore them; thus we are discussing *monadic* PIC.

3 The algebra PIC and its generators

Let us briefly review the algebraic theory and dynamics of action structures [4]. Let X be a set (of names). The algebraic operations of an action structure A over X are the *identities* $\mathbf{id}_m$, the binary operations *product* $\otimes$ and *composition* $\cdot$, and the indexed family $\{\mathbf{ab}_x \mid x \in X\}$ of unary operations called *abstractors*. They obey the following rules of arity, assuming that the arities are natural numbers:

$$\mathbf{id}_m : m \to m \qquad\qquad \frac{a : k \to m \qquad b : m \to p}{a \cdot b : k \to p}$$

$$\frac{a : k \to n \qquad b : \ell \to p}{a \otimes b : k \otimes \ell \to n \otimes p} \qquad\qquad \frac{a : m \to n}{\mathbf{ab}_x a : 1{+}m \to 1{+}n}\;.$$

The following basic axioms hold:

$$
\begin{array}{ll}
a \cdot \mathbf{id} = a = \mathbf{id} \cdot a & a \cdot (b \cdot c) = (a \cdot b) \cdot c \\
a \otimes \mathbf{id}_\epsilon = a = \mathbf{id}_\epsilon \otimes a & a \otimes (b \otimes c) = (a \otimes b) \otimes c \\
\mathbf{id} \otimes \mathbf{id} = \mathbf{id} & (a \cdot b) \otimes (c \cdot d) = (a \otimes c) \cdot (b \otimes d) \\
\mathbf{ab}_x \mathbf{id} = \mathbf{id} & \mathbf{ab}_x (a \cdot b) = (\mathbf{ab}_x a) \cdot (\mathbf{ab}_x b) \; .
\end{array}
$$

The first six axioms make A a strict monoidal category, and the last two assert that the abstractors $\mathbf{ab}_x$ are endofunctors.

For the dynamics there is a preorder $\searrow$ called the *reaction relation* which is preserved by product, composition and abstraction and such that if $a \searrow a'$ then a and a' have the same source and target arities, and if $\mathbf{id} \searrow a$ then $a = \mathbf{id}$.

We have already described the dynamics of PIC. It is straightforward to show that PIC becomes an action structure, when we define the algebraic operations as follows:

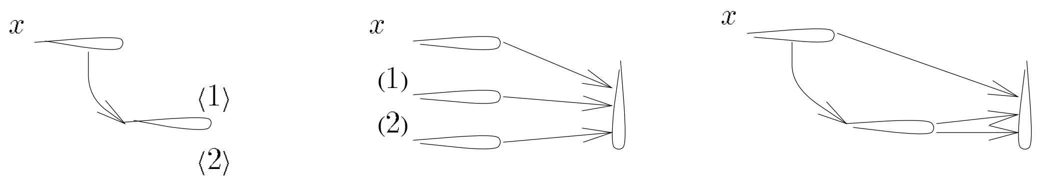
- The identity $\mathbf{id}_m : m \to m$ consists of just m nodes, the i^{th} of which bears input tag (i) and export tag $\langle i \rangle$.

- To form $a \otimes b$, where $a : k \to \ell$, increment b's import tags by k and export tags by ℓ, and coalesce nodes with equal name tags.

- To form $a \cdot b$, where $a : k \to \ell$ and $b : \ell \to m$, coalesce a's node having export tag i with b's node having import tag i (for $1 \le i \le \ell$), remove those tags, and coalesce nodes with equal name tags.

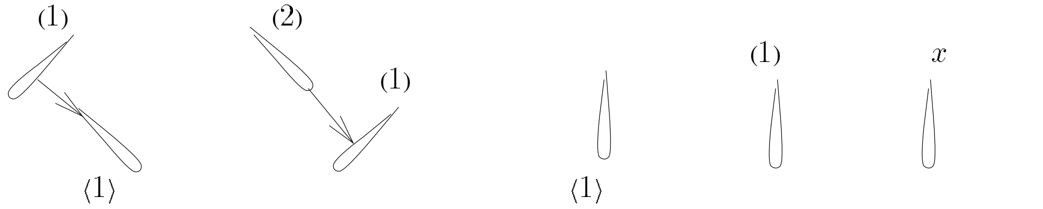
- To form $\mathbf{ab}_x a$, increment a's import and export tags by 1, give the tags (1) and $\langle 1 \rangle$ to the node of a which has name tag x, and remove that name tag.

Here is an example of composition, showing both kinds of coalescence:



a $\qquad$ b $\qquad$ $a \cdot b$

In fact, PIC is not only an action structure but also an *action calculus* [7], a special kind of action structure. We need not be concerned with all the details of action calculi here, but only with a key property –namely that each action calculus is generated by two special kinds of action, $\boldsymbol{\omega} : 1 \to 0$ (*discard*) and $\langle x \rangle : 0 \to 1$ (*datum*, $x \in X$), which are common to all action calculi over X, and a set of actions called *controls* which are specific to each action calculus. Monadic PIC has three controls: $\mathbf{in} : 1 \to 1$ (*input*), $\mathbf{out} : 2 \to 0$ (*output*) and $\boldsymbol{\nu} : 0 \to 1$ (*restriction*). Thus all monadic π-nets in PIC can be constructed by the action structure operations from the following generators:



in : $1 \to 1$ **out** : $2 \to 0$ $\boldsymbol{\nu} : 0 \to 1$ $\boldsymbol{\omega} : 1 \to 0$ $\langle x \rangle : 0 \to 1$

In [6] it was pointed out that not *all* monadic π-nets can be so generated; no net with a *source-cycle* is generated.[3] (A source cycle is a cycle in which each arc leads to a tail – i.e. corresponds to an input particle.) The missing nets can be generated by adding a so-called *reflexion* operator [6]; hence we refer to the action structure of all π-nets as RPIC, or reflexive PIC. Here we shall ignore these matters and stick to PIC, but all our points are relevant to PIC and RPIC alike.

In terms of the algebra we can now define reaction very succinctly. First we define

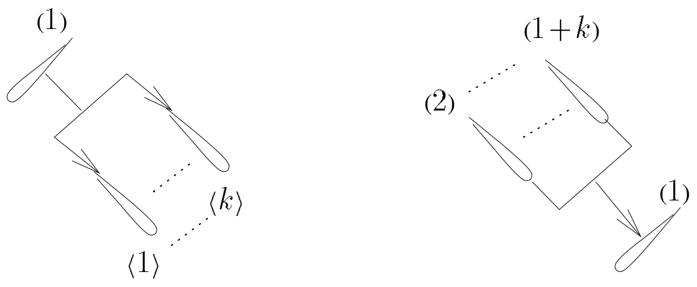
$$\begin{aligned} \mathbf{in}_u : 0 \to 1 &\stackrel{\text{def}}{=} \langle u \rangle \cdot \mathbf{in} \\ \mathbf{out}_u : 1 \to 0 &\stackrel{\text{def}}{=} (\langle u \rangle \otimes \mathbf{id}_1) \cdot \mathbf{out} \ , \end{aligned}$$

which gives both ports the name tag u; then the rule for single reaction is

$$\mathbf{out}_u \otimes \mathbf{in}_u \ \searrow^1 \ \mathbf{id}_1 \ .$$

The one-step reaction relation $\searrow^1$ over π-nets is the smallest which satisfies this rule and is preserved by product, composition and abstraction. This corresponds exactly to our graphical description of reaction in Section 2.

Let us briefly return to the notion of multi-arcs, i.e. polyadic π-nets. These are obtained by generalizing the **in** and **out** generators to



$\mathbf{in}^{(k)} : 1 \to k$ $\mathbf{out}^{(k)} : 1+k \to 0$

[3] This has nothing to do with achieving infinite reaction sequences. Every π-nets as defined in Section 2 is reduced in size by a reaction, so infinite reaction is impossible. It will become possible when we introduce replication.

Then the reaction rule becomes

$$\mathbf{out}_u^{(k)} \otimes \mathbf{in}_u^{(k)} \searrow^1 \mathbf{id}_k \ ,$$

which has the effect of simultaneously coalescing k pairs of source and target nodes. Part of the power of the boxes which we define below is that this polyadic reaction can be simulated by them, so (as was the case with original π-calculus) we lose nothing by taking the monadic form as basic.

Notation We shall freely use the following abbreviations:

$$\begin{array}{rll} ab & \text{for} & a \cdot b \\ \langle xy \rangle & \text{for} & \langle x \rangle \otimes \langle y \rangle \\ (x)a & \text{for} & \mathbf{ab}_x a \cdot (\boldsymbol{\omega} \otimes \mathbf{id}_n) \qquad (a : m \to n) \ . \end{array}$$

For example, we have

$$\begin{array}{rcl} (x)\langle ux \rangle \mathbf{out} & = & \mathbf{out}_u \\ \langle ux \rangle \mathbf{out} & = & \langle x \rangle \cdot \mathbf{out}_u \ . \end{array}$$

4 Adding boxes and replication

Recall the example we gave in Section 2 of a π-calculus process in original notation:

$$P \ = \ x(y).(\overline{y}\langle x \rangle \mid y(v).\overline{v}\langle z \rangle) \ ;$$

because the input prefix $x(y)$ guards what follows, the internal reaction at y cannot occur until an external reaction at x occurs. We cannot match this in PIC; a redex which consists of a pair of particles at the same port, without further instantiation of names, can *always* be reduced.

Recall also, from [5], how the polyadic π-calculus can be derived from the monadic form. This derivation requires that we define the polyadic prefix constructions $u(\vec{x}).P$ and $\overline{u}\langle \vec{x} \rangle.P$, where $\vec{x}$ is any sequence of names. This is done as follows for sequences of length two, thus enabling a pair of names to be transmitted in a single reaction:

$$\begin{array}{rcl} u(x_1 x_2).P & \stackrel{\text{def}}{=} & u(w).w(x_1).w(x_2).P \\ \overline{u}\langle y_1 y_2 \rangle.Q & \stackrel{\text{def}}{=} & (\boldsymbol{\nu}\, w)\, \overline{u}\langle w \rangle.\overline{w}\langle y_1 \rangle.\overline{w}\langle y_2 \rangle.Q \ . \end{array}$$

Note that this uses both input and output monadic guards. Honda and Tokoro [3] made the remarkable discovery that in fact the input monadic guards are enough; with them we can get the power of output guards, and hence also of polyadic communication. Here, in essence, is their construction:

$$\begin{array}{rcl} u(x_1 x_2).P & \stackrel{\text{def}}{=} & u(w).(\boldsymbol{\nu} v_1)(\overline{w}\langle v_1 \rangle \mid v_1(x_1).(\boldsymbol{\nu} v_2)(\overline{w}\langle v_2 \rangle \mid v_2(x_2).P)) \\ \overline{u}\langle y_1 y_2 \rangle.Q & \stackrel{\text{def}}{=} & (\boldsymbol{\nu} w) \Big(\overline{u}\langle w \rangle \mid w(v_1).(\overline{v_1}\langle y_1 \rangle \mid w(v_2).(\overline{v_2}\langle y_2 \rangle \mid Q)) \Big) \ . \end{array}$$

This has all the beauty of a zip-fastener; note how a parallel bar on one side is matched by a prefix on the other, and vice versa.

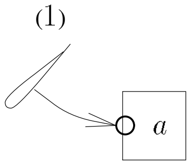
In action structures for π-calculus, it will be product ($\otimes$) and a construction called *boxing* which play this rôle. The construction of Honda and Tokoro justifies the **box** control construction of [7] as a sufficient extension of PIC for imposing sequence upon reactions. The remainder of this section is devoted to defining an extended form of π-net with boxing and replication. This is essentially the action calculus PIC(**box**, **rep**) which was defined in [7]. The new ingredient here is the graphical presentation; we shall use this in Section 5 to present the Honda-Tokoro construction.

We define the control operation **box**, which takes a single action as parameter, with the following arity rule:

$$\frac{a\ :1\rightarrow n}{\textbf{box}\,a\ :1\rightarrow n}\ .$$

The correspondence with the π-calculus is as follows: if a corresponds to the process abstraction $(x)P$ then $\langle u\rangle\cdot\textbf{box}\,a$ corresponds to the input prefix construction $u(x).P$. (Of course, the target arity n of a has no correspondent in the π-calculus.)

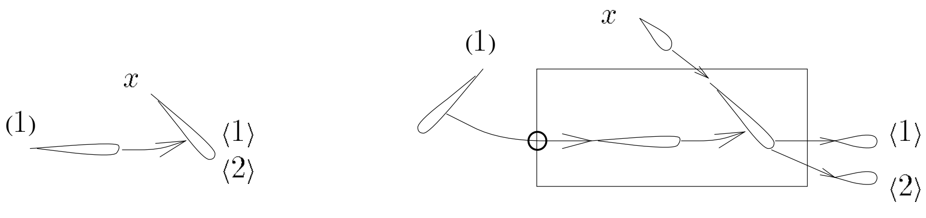
We indicate this construction in π-nets informally as follows:



$\textbf{box}\,a : 1\rightarrow n \qquad (a : 1\rightarrow n)$

The ring indicates that the arc, which we shall call a *box arc*, impinges on a's single import node, which loses its import tag.

In our formal constructions we require a bit more detail, to maintain the convention that each tag in a net occurs uniquely. Let us define a *region* of a net to be a part which is not crossed by a box boundary. We shall ensure that each tag occurs only in the outermost (unboxed) region of a net, as follows. We introduce a new kind of arc, called a *link*, drawn from a node head to a node tail. In forming **box** a, if any name tag or export tag occurs on a node of a then this node is linked to a new distinct external node, which receives the tag instead. Here is an example:

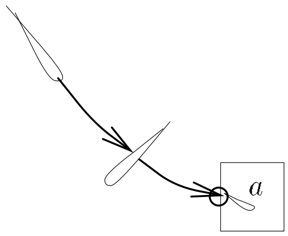


$a \qquad\qquad \textbf{box}\,a$

When boxes are nested, the linking for a name x forms a tree, making x everywhere accessible. Intuitively, we can see that boxes exist to confine *arcs* (which are the ingredients of redexes), not nodes; so the links are just a graphical device to liberate nodes again.

In practice we need not bother always to add the new nodes and the links, since they can be uniquely supplied when missing. But with this formal treatment the operations of product, composition and abstraction remain exactly as they were described earlier. Note in particular that they will not coalesce any node in a box, and that they add no further links.

For boxes we have a new kind of redex, consisting of an arc entering and a box arc leaving the same node waist, thus:



To reduce such a redex we coalesce its source node with the import node of a, remove both arcs and the box surrounding a, and then coalesce any pair of linked nodes in the enlarged outer region, removing the link arcs.

Now, just as we did for **in**, we define

$$\mathbf{box}_u a : 0 \to n \quad \stackrel{\text{def}}{=} \quad \langle u \rangle \mathbf{box}\ a$$

and add a further reaction rule

$$\mathbf{out}_u \otimes \mathbf{box}_u a \ \searrow^1 \ a\ .$$

In passing, note that **in** becomes redundant in the presence of **box**, since it behaves exactly as **box id**. But the main point of concern here is as follows: if $\searrow^1$ is taken to be the smallest relation which satisfies this rule and is preserved by the action structure operations, the effect in terms of π-nets is that reaction cannot occur inside a box.

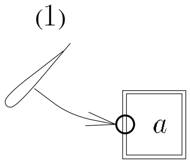
It is a routine matter to check that these operations and the reaction relation yield an action structure. It is very close to the action calculus PIC(**box**) defined in [7], but not identical; in Section 6 we look at the slight adjustment needed, and also some alternative reaction relations.

Let us now look briefly at replication. In [8] a form of replication, written $\pi * P$, was defined in which a copy of the process P is generated every time the prefix π is activated. In [5] a more general form $!P$ was defined, simply by imposing the axiom $!P = P \mid\ !P$; this amounts to declaring that an unbounded number of copies of P exist side-by-side without any need for activation.

Here we adopt the former approach, since it is closer to boxing. In fact, in the same spirit, we require the activating prefix to be an input action. This is the control operation **rep** which was defined in [7]. Its arity rule is

$$\frac{a\ : 1 \to 0}{\mathbf{rep}\ a\ : 1 \to 0}\ ,$$

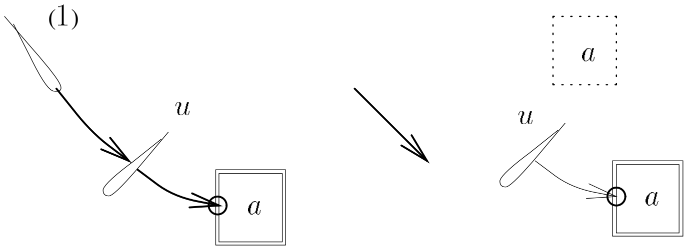
and we indicate the construction graphically as follows:



$$\textbf{rep}\, a : 1 \to 0 \qquad (a : 1 \to 0)$$

The reaction rule for **rep** is just as for **box**, except that the box is retained alongside the copy (which is why the target arity must be 0):

$$\begin{array}{lcl} \textbf{rep}_u\, a : 0 \to 0 & \stackrel{\text{def}}{=} & \langle u \rangle \textbf{rep}\, a \\ \textbf{out}_u \otimes \textbf{rep}_u\, a & \searrow^{1} & a \otimes \textbf{rep}_u\, a\ . \end{array}$$

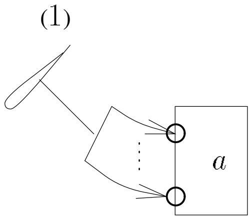


With this rule, we come close to the action calculus PIC(**box**, **rep**) defined in [7]; a similar small adjustment is needed as for **box**. For the purposes of this paper we need say no more about **rep**, since it poses no more problems than **box** for graphical presentation.

5 Using boxes: the Honda-Tokoro construction

We shall now use the construction of Honda and Tokoro to simulate multi-arc reduction, using boxes. (We do not need replication.) We add nothing to their idea, but merely present it in the setting of π-nets.

Let us first generalise **box**, as we generalised **in**. We define the constructor $\textbf{box}^{(k)}$ thus (so that $\textbf{box} = \textbf{box}^{(1)}$):

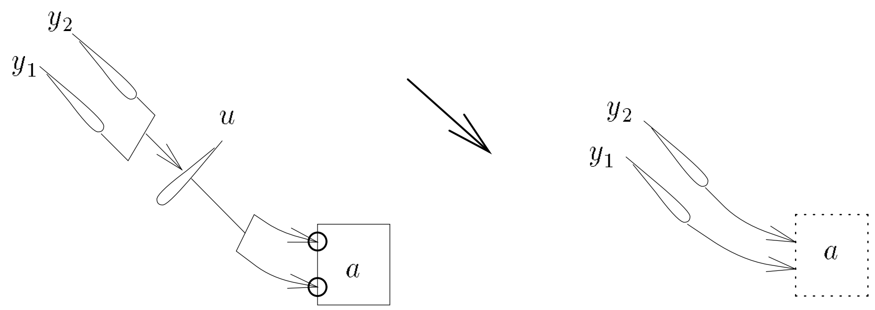


$$\textbf{box}^{(k)} a : 1 \to n \qquad (a : k \to n)$$

Correspondingly we generalise the reaction rule for **box**, for arbitrary $a : k \to n$:

$$\begin{aligned} \mathbf{box}^{(k)}_u a : 0 \to n &\stackrel{\text{def}}{=} \langle u \rangle \mathbf{box}^{(k)} a \\ \mathbf{out}^{(k)}_u \otimes \mathbf{box}^{(k)}_u a &\searrow^1 a\ . \end{aligned}$$

For $k = 2$, if we tag the source nodes with y_1 and y_2 this reaction is just



Our first task then is to encode $\langle y_1 y_2 \rangle \mathbf{out}^{(2)}_u$ and $\mathbf{box}^{(2)}_u a$. We adapt the Honda-Tokoro expressions given at the beginning of this section, as follows (using square brackets for boxes, and staggered to show the structure):

$$\begin{aligned} \langle y_1 y_2 \rangle \mathbf{out}^{(2)}_u \stackrel{\text{def}}{=}\ & \boldsymbol{\nu}(w)\Big(\langle w \rangle \mathbf{out}_u \otimes \\ & \quad \mathbf{box}_w \big[(v_1) \\ & \qquad (\langle y_1 \rangle \mathbf{out}_{v_1} \otimes \\ & \qquad\quad \mathbf{box}_w \big[(v_2) \\ & \qquad\qquad \langle y_2 \rangle \mathbf{out}_{v_2}\big])\big]\Big) \\ \mathbf{box}^{(2)}_u a \stackrel{\text{def}}{=}\ & \mathbf{box}_u \Big[(w) \\ & \quad \boldsymbol{\nu}(v_1)(\langle v_1 \rangle \mathbf{out}_w \otimes \\ & \qquad \mathbf{box}_{v_1}\big[(x_1) \\ & \qquad\quad \boldsymbol{\nu}(v_2)(\langle v_2 \rangle \mathbf{out}_w \otimes \\ & \qquad\qquad \mathbf{box}_{v_2}\big[(x_2)(\langle x_1 x_2 \rangle a\big])\big])\Big]\ . \end{aligned}$$

Now take the product

$$b_0 = \langle y_1 y_2 \rangle \mathbf{out}^{(2)}_u \otimes \mathbf{box}^{(2)}_u a\ ;$$

we must show that $b_0 \searrow \langle y_1 y_2 \rangle a$. We show b_0 as a net in Figure 1. Some annotation has been added to help comparison with the expressions; formally, the only alphabetic symbols in the net are y_1, y_2 and u (apart from the schematic variable a).

The reduction sequence

$$b_0 \searrow^1 b_1 \searrow^1 \cdots \searrow^1 b_5 \sim \langle y_1 y_2 \rangle a$$

is shown in Figure 2. At each stage one box is exploded. Note that there are two links; they are the arcs which cross a box boundary, *other than* box arcs which are

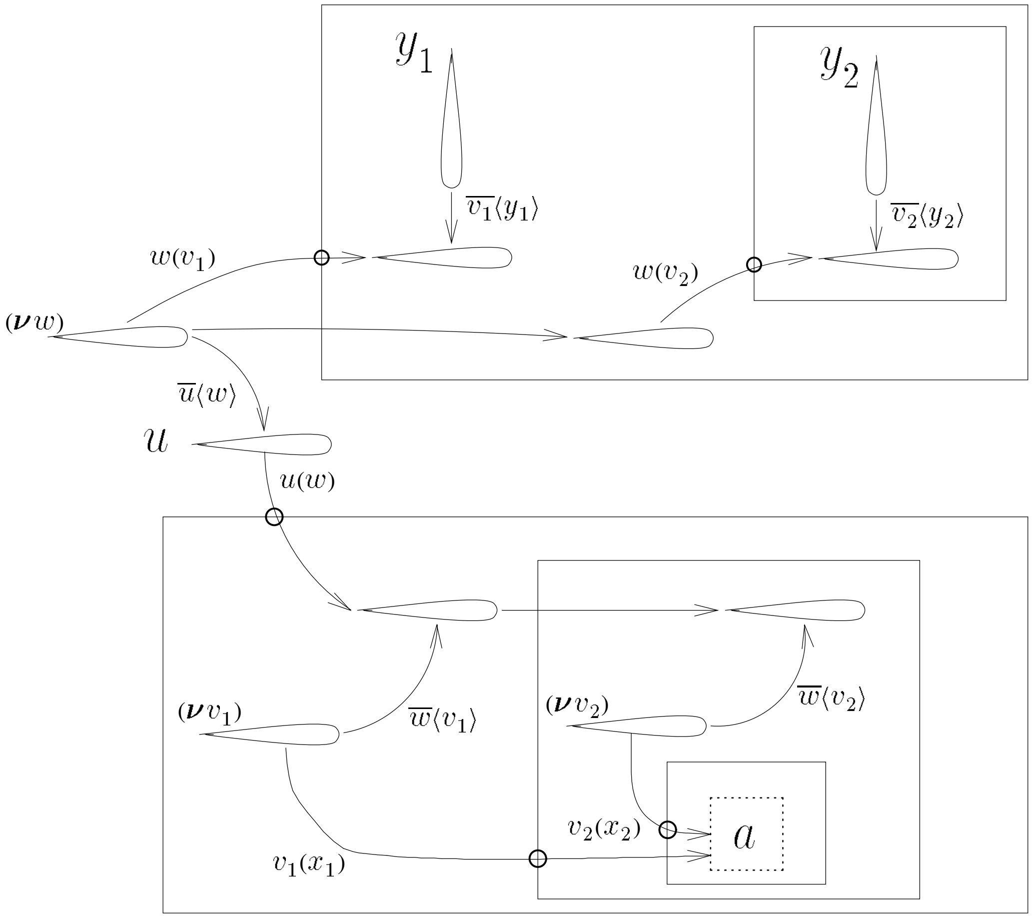


Figure 1: $b_0 \ = \ \langle y_1 y_2\rangle \mathbf{out}_u^{(2)} \otimes \mathbf{box}_u^{(2)} a$

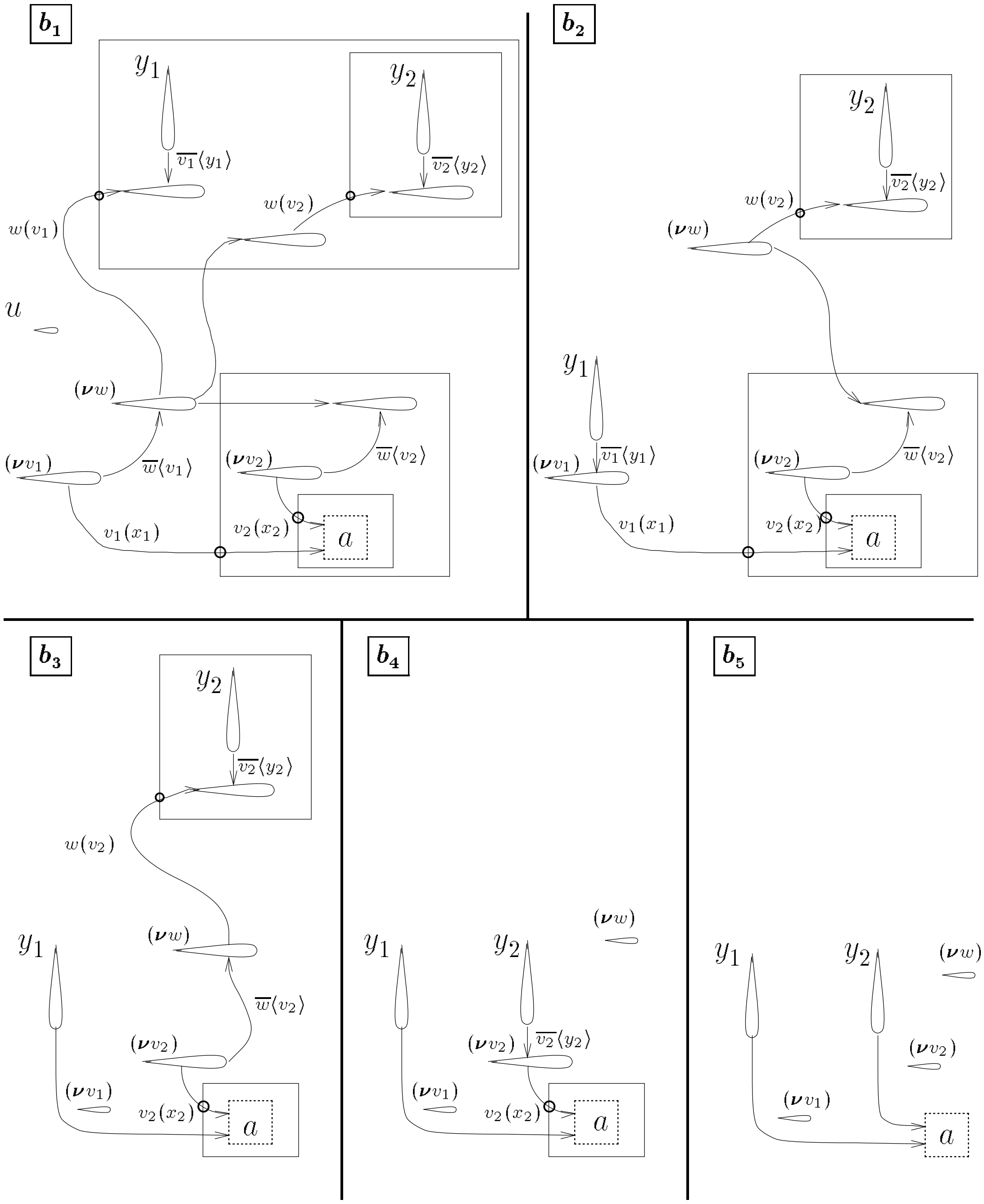


Figure 2: Reduction of b_0: $b_0 \searrow^1 b_1 \searrow^1 b_2 \searrow^1 b_3 \searrow^1 b_4 \searrow^1 b_5 \sim \langle y_1 y_2 \rangle a$

distinguished by a ring. It is exactly the control of these links which imposes the proper sequence on reduction to ensure that y_1 is bound to x_1 and y_2 to x_2.

Essentially we have $b_5 = \langle y_1 y_2 \rangle a$ as required; the only difference is that b_5 has three arcless, tagless nodes as garbage. This garbage is highly inert, and persists harmlessly through all operations and reductions. (An algebraic aside: this garbage is neatly removed by imposing the identity $\boldsymbol{\nu} \cdot \boldsymbol{\omega} = \mathbf{id}_0$ upon the action structure.)

Does this reduction sequence entitle us to claim that we have encoded $\langle y_1 y_2 \rangle \mathbf{out}_u^{(2)}$ properly? Not fully; for its existence still allows that some b_i may have other possible reductions, or may take part in them when placed in a suitable context. If that were so, the encoding would be invalid. However, it is not so. One can convince oneself by inspecting the nets that every arc shown could never take part in any other reduction. To justify this claim rigorously requires an adaptation of the notions of *reachability* (of arcs) and *incident*, discussed in [6].

6 Variations

The reaction relation we have chosen for boxes is probably the smallest which is reasonable. But there is a series of variations which are weaker, in the sense that they allow earlier coalescence of nodes and may therefore permit certain reactions to occur earlier. The Honda-Tokoro construction works in all of them; this is some evidence that each of them imposes "sufficient" sequencing. This appears to be due to the common feature of all the variations: they all require the two arcs of a redex to lie within the same region.

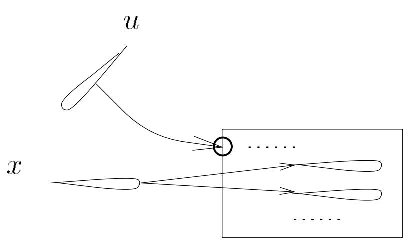
Link immigration We begin by pointing out how the action structure defined above differs from the action calculus $\mathsf{PIC}(\mathbf{box})$ given in [7]. Consider

$$a \;=\; \langle xx \rangle \cdot (yz)\mathbf{box}_u[\ldots y \ldots z \ldots] \;;$$

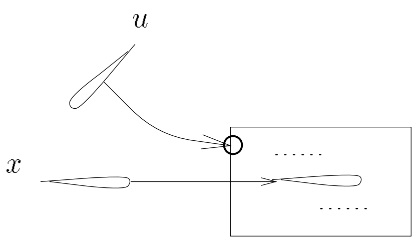
if y, z do not occur elsewhere in the box, in $\mathsf{PIC}(\mathbf{box})$ this is equal to

$$b \;=\; \mathbf{box}_u[\ldots x \ldots x \ldots] \,,$$

i.e. the instantiation of y and z to x may permeate the box. But a and b correspond to different nets:



a



b

Let us call a link *inward* if it points into a box. Then we achieve the effect of instantiation permeating a box if we equate each pair of nets like a and b, i.e. whenever two inward links have the same source node then we may coalesce their target nodes. This induces an equivalence relation upon π-nets (actually a congruence with respect to all the constructions); when we divide by this congruence, I conjecture that the action structure becomes isomorphic to $\mathsf{PIC}(\mathbf{box})$.

If this coalescence is not made, the situation amounts to the enforced delay of a substitution. This strongly suggests the notion of *explicit substitution*, studied by Abadi et al [1] for the λ-calculus. Indeed, by extending PIC to allow substitution particles like $({}^{x}\!/y)$ (pronounced "x for y", where x is free and y bound) within the body of an action, we would hope to model delayed substitutions in an action calculus.

Reaction within a box In the Honda-Tokoro construction the essential use of boxing was to prevent an arc outside a box from reacting with one inside. This purpose is still perfectly achieved even if we allow reaction within a box, or more exactly within any region. This is expressed not by changing the reaction rules

$$\begin{array}{rcl} \mathbf{out}_u \otimes \mathbf{in}_u & \searrow^1 & \mathbf{id}_1 \\ \mathbf{out}_u \otimes \mathbf{box}_u\, a & \searrow^1 & a \\ \mathbf{out}_u \otimes \mathbf{rep}_u\, a & \searrow^1 & a \otimes \mathbf{rep}_u\, a \end{array}$$

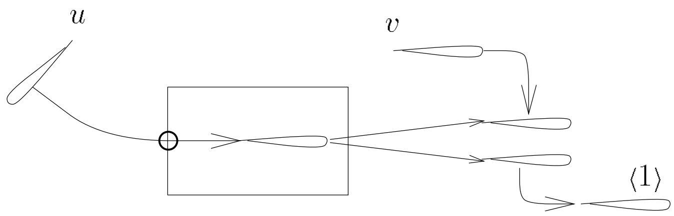
but simply by requiring that they may be applied in any context – i.e. that the reaction relation be preserved by the box and replication constructions as well as by the action structure operations. Of course one may choose to allow reaction within a box but forbid it within a replication (perhaps on the grounds that such reactions count for more, since their effect will be multiplied indefinitely).

In the context of this more liberal regime, link immigration also takes on a greater significance because it will enable earlier reaction within a box.

Link emigration There is a phenomenon dual to the immigration of links. Consider

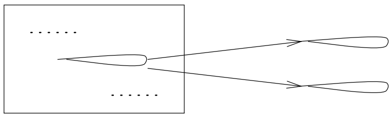
$$a \;=\; \mathbf{box}_u[(x)\langle x\,x\rangle] \cdot (y\,z)(\langle v\rangle \mathbf{out}_y \otimes \mathbf{in}_z)\;;$$

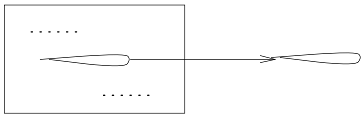
is there a reaction? Apparently not; if we look at the net we can see that the reaction will wait until the box is exploded.



This is in agreement with $\mathsf{PIC}(\mathbf{box})$ as defined in [7] (in contrast with the case for link immigration). There is no obvious reason why such an implication for coalescing two nodes should not be allowed to permeate from the inside to the outside of a box. If reaction within boxes is also allowed (as discussed above), then this is a way to allow any coalescences which arise from internal reaction to be felt externally; the box still prevents reaction *between* its interior and exterior.

How can this outward permeation be achieved? In π-nets we can proceed dually to the immigration case. Let us call a link *outward* if it points out of a box. Then we get what we want by equating each pair of nets like the following:





i.e. whenever two outward links have the same source node then we may coalesce their target nodes. Again, this induces a congruence relation upon π-nets.

How can we match this congruence algebraically? We would hope to match it by imposing an equational axiom in addition to those of [7]. At first sight, there is an intriguing possibility which may be correct; it is to choose an appropriate *effect structure* E (see [4]) for π-nets, and to impose the axiom

$$\mathbf{box}(a \cdot e) = \mathbf{box}\, a \cdot e \qquad (e \in E)\ .$$

Certainly $(x)\langle xx\rangle$ is a member of the natural effect structure, so we would have for the above example

$$\begin{aligned} a &= \mathbf{box}_u[\mathbf{id}_1] \cdot (x)\langle xx\rangle \cdot (yz)(\langle v\rangle \mathbf{out}_y \otimes \mathbf{in}_z) \\ &= \mathbf{box}_u[\mathbf{id}_1] \cdot (x)(\langle v\rangle \mathbf{out}_x \otimes \mathbf{in}_x) \end{aligned}$$

which has a reaction. The ramifications of this idea are interesting; the richer the effect structure, the greater is the influence which reaction within a box may exert upon the environment – without losing the purpose of the box. One should also recall that since effects are inert, to export them from the box cannot reduce the possibility of internal reactions.

This idea remains largely unexplored, and deserves further research.

7 Conclusion

The aim of this informal paper has been to explore the graphical presentation of the various action structures which enrich the π-calculus.

By exploring π-nets with boxes we have illustrated the power of boxing; we have also found a surprising degree of freedom in its precise meaning. Of course the Honda-Tokoro construction was known, and some of the various of the boxing regimes discussed in the previous section were vaguely familiar; so π-nets were not strictly necessary for some of the observations which we have made. But I had no idea about the phenomenon of *link emigration* before looking at π-nets and trying to formalise them. The nets therefore seem to be at least a good auxiliary tool of investigation, provided that they are not allowed to prevent a more abstract treatment when it becomes appropriate. We have also seen that there is good hope for algebraic characterisation of the various boxing regimes.

Little has been said here about the family of *action calculi*, of which PIC and PIC(**box**, **rep**) and its variants are members. But the graphical presentation is by no means restricted to these members of the family. It may be fruitful to identify a class of action calculi which yield naturally to graphical representation; this may be of value not only theoretically, but also for the implementation of such an action calculus considered as a programming language.

References

[1] Abadi, M., Cardelli, L., Curien, P–L. and Lévy, J–J., *Explicit substitutions*, Journal of Functional Programming, Vol 1, 4, pp375–416, 1991.

[2] Berry, G. and Boudol, G., *The chemical abstract machine*, Journal of Theoretical Computer Science, Vol 96, pp217–248, 1992.

[3] Honda, K. and Tokoro, M., *An object calculus for asynchronous communication*, Proc. European Conference on object-oriented programming, Lecture Notes in Computer Science, Vol 512, Springer, pp133–147, 1991.

[4] Milner, R., *Action structures*, Research Report LFCS–92–249, Laboratory for Foundations of Computer Science, Computer Science Department, Edinburgh University, 1992.

[5] Milner, R., *The polyadic π-calculus: a tutorial*, in **Logic and Algebra of Specification**, ed. F.L. Bauer, W. Brauer and H. Schwichtenberg, Springer Verlag, 1993, pp203–246.

[6] Milner, R., *Action structures for the π-calculus*, Research Report ECS–LFCS–93–264, Laboratory for Foundations of Computer Science, Computer Science Department, Edinburgh University, 1992.

[7] Milner, R., *Action calculi I: axioms and applications*, Computer Science Department, Edinburgh University, 1993. Also appeared as *Action calculi, or syntactic action structures*, Proceedings of MFCS '93, Lecture Notes in Computer Science, Springer, 1993.

[8] Milner, R., Parrow, J. and Walker D., *A calculus of mobile processes, Parts I and II*, Journal of Information and Computation, Vol 100, pp1–40 and pp41–77, 1992.

[9] Parrow, J., *Interaction diagrams*, SICS Research Report R93:06, 1993. To appear in proceedings of REX'93 Workshop, Lecture Notes in Computer Science, Springer Verlag.