Action Structures for the π-Calculus

Robin Milner

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

May 1993

Abstract

In a previous paper, action structures were proposed as a variety of algebra to underlie concrete models of concurrent computation and interaction. That work is summarised here, to make the paper self-contained. In particular, the uniform construction of a process calculus upon an arbitrary action structure is reviewed. Another relevant concept from the previous paper is recalled, namely the notion of *incident set*. Its importance is that, in the process calculus uniformly constructed upon any action structure, a bisimulation equivalence which rests upon an incident set is guaranteed to be a congruence for the calculus.

The main purpose of this paper is to give a family of action structures for the π-calculus. Using one of these, the original π-calculus is obtained by the uniform construction. The most substantial technical element here is the construction of an appropriate incident set for this action structure, yielding a bisimulation congruence for the π-calculus.

Another action structure is used to provide (again via the uniform construction) a *synchronous* version of the π-calculus, in the sense that MEIJE or SCCS is a synchronous version of CCS. The natural version of this action structure contains an anomaly which appears to prevent the existence of the appropriate incident set. However, a so-called *reflexive* version of the action structure does indeed yield a natural incident set, and thereby again a bisimulation congruence.

Two innovations are noteworthy. One is the definition of *reflexive substitution*. With this new kind of substitution, *restricting* a port-name x (in the sense of CCS) amounts to "making x denote itself" – i.e. reflexively substituting x for itself. This notion simplifies the action structure for the synchronous π-calculus. The second innovation is that the action structures are presented graphically, giving a new perspective on the π-calculus. Indeed, both definition and proof of the incident set for the reflexive π-calculus are in terms of a simple graph-theoretic notion of *reachability*.

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

1 Introduction

Action structures [8] have been proposed as a variety of algebra to underlie concrete models of concurrent computation and interaction. Over the last decade we have come to appreciate the difficulty of finding an abstract treatment of interactive computation, in the sense that Scott's domain theory is an abstract treatment of non-interactive computation. Progress with concrete models, notably Petri nets and process algebras, has given us a better understanding of the wide range of choice to be made, as to what should or should not be abstracted away in such a treatment; consequently we tread more carefully in proposing abstractions.

Action structures aim not to make too much commitment as to the level of abstraction, but rather to unite the different concrete models. But there is danger in providing a general framework; to achieve generality can be to avoid understanding the specific nature of a topic. Action structures therefore make a commitment – not to a level of abstraction, but to the nature of the basic combinators for building concurrent systems. They take three combinators as basic: composition (of a certain kind), tensor product (representing parallel activity) and a form of parametrization. Added to these is a preorder representing the performance of actions; this performance can be represented either in full detail in a very concrete action structure, or in less detail in a more abstract one. An example was given in [8] of how to relate the two by a homomorphism of action structures.

A process calculus such as the π-calculus, aiming at expressive power which is adequate in some practical sense, requires a larger repertoire of constructions than just those which are basic to action structures. In fact, a strong motive in defining action structures was to understand better the status of the π-calculus as a model for concurrency. By setting it in the action structure framework one can make a better assessment of the constructions which are special to the π-calculus. In particular, the π-calculus models very directly the phenomenon of *mobility* or *changing configuration* in systems of processes, whereas this phenomenon is not part of the action structure framework; by looking at π-calculus within the framework we may find that its specific treatment of mobility suggests a more canonical treatment. One may say that action structures look towards a canonical treatment of process *constructions*, in contrast with Petri nets [13] or event structures [12] which focus upon the notions of *conflict*, *causality* and *independence*.

One aspect of the general theory of [8] is a uniform way of building a process calculus from an action structure. Part of that theory is a treatment of bisimilarity. The main technical content of this paper is a study of several action structures for the π-calculus. By using one of these, we first attain a presentation of π-calculus which comes very close to its original presentation [11]. The notion of strong bisimilarity which arises naturally from the action structure treatment is in fact slightly stronger than the original, and throws new light upon the latter. But the second and third action structures presented here yield –via the general theory– a *synchronous* version of the π-calculus; by this we mean that a transition which is atomic at the process level may be arbitrarily complex in terms of reactions in the underlying action structure. This synchronous π-calculus corresponds to the original π-calculus in exactly the way that synchronous calculi such as MEIJE [1] or SCCS [10] is correspond to CSP [3] or CCS [7].

The paper is organised as follows. First, the basic notions of action structures and effect structures are reviewed in Section 2. This makes the paper technically self-contained, though previous knowledge of action structures from [8] will certainly help. In Section 3 this review continues with the uniform treatment of processes; it ends by considering what an action structure for π-calculus will have to supply in order to recover the original π-calculus via the uniform treatment. Section 4 presents the action structure PIC_τ for the original π-calculus, together with its effect structure; it also presents PIC, which is a candidate to underlie the synchronous π-calculus. Section 4 finishes with an illustration of synchronous activity, in accessing compound data structures. Section 5 returns to the general theory of action structures to review the notion of *incident set*, in terms of which bisimilarity is defined. An incident set for PIC_τ is presented, which gives a satisfactory bisimilarity for the original π-calculus. An incident set is also presented for the *monadic* PIC. But it is shown that the analogous set for the full *polyadic* PIC is *not* an incident set; there is an anomaly.

Section 6 gives a graphical presentation of PIC. This graphical form is a new way of thinking about process mobility, and supplies intuitions which are not seen so clearly in the syntactic form. But it does more; for it suggests a natural way to overcome the anomaly of the previous section. Finally, Section 7 takes up this suggestion; it defines –both graphically and syntactically– the *reflexive* π-calculus, in terms of the action structure RPIC. This rests upon the new technical notion of *reflexive substitution*, which allows that the restriction of a name (in the usual sense of π-calculus or CCS) to be generated by –loosely speaking– *substituting the name for itself*. This action structure RPIC seems the best candidate to underlie the synchronous π-calculus. It is actually simpler than PIC, since it has a purer graphical form; it also yields a natural and elegant incident set, thereby providing a bisimilarity congruence for the synchronous π-calculus. This incident set, and the proof that it is one, rest on a simple graph-theoretic notion of *reachability*.

2 Review of Action Structures and Effects

In this section we review the important basic notions of action structures, up to and including the treatment of *effects* of an action. Thus we are summarising Sections 2 and 4 of [8]. It will help to have read those sections, but technically what follows is self-contained.

The most succinct definition of action structure is in category-theoretic terms. In [8] it was defined algebraically; here it is defined categorically first but then elaborated in algebraic terms. So those unfamiliar with categories should focus upon the discussion following the definition, and should not then find any difficulty.

2.1 Action structures: Definition An action structure A is a strict monoidal category, with two extra items:

- a set X_A referred to as *names*, and for each $x \in X_A$ an endo-functor upon A known as an *abstractor*;
- a preorder $\searrow_A$ over each hom-set of A, called *reaction*, which is preserved by composition, monoidal product and abstraction, and for which the units are minimal. ■

Even more succinctly, an action structure is a preordered strict monoidal category with an indexed set of endo-functors.

2.2 Algebraic Elaboration We shall work with the following algebraic characterization of action structures.

First, an action structure A has a monoid $(M_A, \otimes, \epsilon)$ of objects; these are the objects of the category and we shall call them *arities*. We shall use $k, l, m, n, \ldots$ to range over arities. The arities may be, for example, sequences of sorts (like INT, BOOL) under concatenation; in the special case when there is only one sort, 1, the monoid M_A is just the natural numbers under addition.

Second, for each pair m, n of arities A has a family $A_{m,n}$ of *actions*; they are the morphisms of A as a category. If a is a member of this set we write $a : m \to n$ and call m the *source arity* and n the *target arity* of a. We shall use a, b, c, d to range over actions.

Third, since A is a monoidal category with abstractors, there is a unit action $\mathbf{1}_m : m \to m$ for each arity m, and the operations of composition $\cdot$, tensor or monoidal product $\otimes$, and abstraction $\mathbf{ab}_x$ for each $x \in X_A$. They obey the following arity rules:

$$\mathbf{1}_m : m \to m \qquad\qquad \frac{a : k \to m \qquad b : m \to p}{a \cdot b : k \to p}$$

$$\frac{a : k \to n \qquad b : l \to p}{a \otimes b : k \otimes l \to n \otimes p} \qquad\qquad \frac{a : m \to n}{\mathbf{ab}_x a : \mathbf{ab}_x m \to \mathbf{ab}_x n}$$

Note that composition is written forwards; we write $a \cdot b$ where the standard in category theory is to write $b \circ a$.

Note also that the operation of $\mathbf{ab}_x$ upon arities is not specified. In our intended applications there is usually an arity k_x associated with each names x, and then $\mathbf{ab}_x m$ is just $k_x \otimes m$. For example if arities are sort sequences we may have $k_x = \text{INT}$, which may mean that x is an integer variable; then for $a : \text{BOOL} \otimes \text{BOOL} \to \text{INT}$ we shall have $\mathbf{ab}_x a : \text{INT} \otimes \text{BOOL} \otimes \text{BOOL} \to \text{INT} \otimes \text{INT}$. In such cases $\mathbf{ab}_x$ is close to λ-abstraction; but in general we are not constrained to think of the set X_A as containing variables.

The status of A as a strict monoidal category with functorial abstractors is directly expressed by eight equational axioms, which we now give. Here (and later) we imagine arities to be ascribed in any way which respects the arity rules, and which gives the same arity to each side of an equation:

C1.	$a \cdot \mathbf{1} = a = \mathbf{1} \cdot a$	C2.	$a \cdot (b \cdot c) = (a \cdot b) \cdot c$
P1.	$a \otimes \mathbf{1}_\epsilon = a = \mathbf{1}_\epsilon \otimes a$	P2.	$a \otimes (b \otimes c) = (a \otimes b) \otimes c$
PF1.	$\mathbf{1} \otimes \mathbf{1} = \mathbf{1}$	PF2.	$(a \cdot b) \otimes (c \cdot d) = (a \otimes c) \cdot (b \otimes d)$
AF1.	$\mathbf{ab}_x \mathbf{1} = \mathbf{1}$	AF2.	$\mathbf{ab}_x (a \cdot b) = (\mathbf{ab}_x a) \cdot (\mathbf{ab}_x b)$.

Finally the reaction relation $\searrow_A$, which will often be written just $\searrow$, is a preorder on each $A_{m,n}$; so $a \searrow a'$ implies that a and a' have the same source and target arities. Reaction is also preserved by every operation; that is, $a \searrow a'$ implies $b \otimes a \searrow b \otimes a'$, $\mathbf{ab}_x a \searrow \mathbf{ab}_x a'$, etc. We stipulate that if $\mathbf{1} \searrow a$ then $a = \mathbf{1}$.

2.3 Intuition There are no doubt many action structures which have not much to do with computation. But the notion arose from trying to understand concurrency, so some intuition of the significance of the action structure operations from that viewpoint will be useful.

We think of tensor product ($\otimes$) as parallel composition. Reactions within a and within b can occur independently within $a \otimes b$. But there may be reactions $a \otimes b \searrow c$ which do not arise from a alone or from b alone; these represent communication or interaction between a and b. In particular, a and b may "use" the same name x, and the manner of use may constitute communication via x as a port.

Abstraction $\mathbf{ab}_x$ allows parametrization upon x. In particular, when x is a means of interaction as just described, then abstraction upon x allows one to vary the interface x, i.e. to vary the partners with whom communication via x may take place.

Composition $a \cdot b$ does not represent *sequential* composition. Given $a : m \to n$ and $b : n \to p$, the arity n indicates the extent to which b is parametric; to this extent, information may flow from a to b at any time during the continuing reaction within each of them. (It is sometimes better to think of an action as an *activity*, since it may enjoy an unbounded number of reactions.) This information can affect the further reaction within b; for example it may instantiate a parametric port (name) in b, as described in the previous paragraph, enabling interactions which were previously impossible.

The reader is invited to consider these discursive comments briefly at first, and later more thoroughly after considering the action structure for the π-calculus.

2.4 Effects of actions Consider briefly a transition in CCS [9], in which a value v is transmitted from one agent to another:

$$\overline{a}v.P \mid a(x).Q \xrightarrow{\tau} P \mid \{v/x\}Q.$$

We may think of the substitution $\{v/x\}$ here as the *effect* of the communication via the channel a; this effect influences Q in its further performance. We wish to capture the notion of effect in action structures, but in an abstract way which will allow effects other than substitutions. We shall think of an effect as a kind of degenerate action. Thus, if $a \searrow a' \cdot e$, where e is an effect, we imagine that no further reaction within a' or e, or between them, can withdraw the effect e. The most that can happen is that a' generates a further effect e', by the reaction $a' \searrow a'' \cdot e'$. We may think of this in two ways:

- The effects of a accumulate through the successive reactions

$$a \searrow a' \cdot e \searrow (a'' \cdot e') \cdot e = a'' \cdot (e' \cdot e),$$

  and $e' \cdot e$ is also an effect.

- In the composition $a \cdot b$, each effect of a can influence b. Thus $a \cdot b \searrow a' \cdot e \cdot b$, and a reaction $e \cdot b \searrow b'$ may become possible. Then a further effect e' from a' may enable a further reaction $e' \cdot b' \searrow b''$. In all,

$$a \cdot b \searrow a' \cdot e \cdot b \searrow a' \cdot b' \searrow a'' \cdot e' \cdot b' \searrow a'' \cdot b''.$$

Let us now define effects precisely.

2.5 Inactivity and inertia: Definition An action a is *inactive* if $a \searrow a'$ implies $a' = a$. An action a is *inert* if whenever $b \cdot a \searrow c$ then there exists b' such that $b \searrow b'$ and $c = b' \cdot a$. ■

Note that the units $\mathbf{1}_m$ are, trivially, inert in any action structure. It would not follow that they are inactive; but recall that we *required* them to be inactive in Definition 2.1.

Exercise Using this, prove that *every* inert action is inactive. ■

We are going to demand that effects are inert actions. Intuitively, if e is inert the definition says that e cannot help any action b *precomposed* to it (i.e. in the composition $b \cdot e$) to react; so inertia captures something of the direction of information-flow.

We are going to place two further conditions upon effects, concerning their algebraic status. First, they must constitute a sub-actionstructure; the following definition is adapted from [8]:

2.6 Sub-actionstructure: Definition A *sub-actionstructure* B of A consists of

- a subset X_B of the names X_A;
- a submonoid M_B of the arities M_A;
- for each pair of arities m, n a subset $B_{m,n}$ of the actions $A_{m,n}$, this family of subsets containing all the units and being closed under composition, product and abstraction;
- a reaction relation $\searrow_B$ which is the restriction of $\searrow_A$ to the actions in B. ■

Now for certain sub-actionstructures B we shall be interested in compositions $a \cdot b$ with $a \in A$ and $b \in B$. We call (a', b) a *decomposition of* a *for* B if $a = a' \cdot b$ and $b \in B$. We shall also refer to b as a *postcomponent* of a. We can preorder the decompositions of a for B by defining $(a_1, b_1) \leq (a_2, b_2)$ if some $b \in B$ makes the following diagram commute:

$$\begin{tikzcd}
& \cdot \arrow[dd, "b"] \arrow[dr, "b_2"] & \\
\cdot \arrow[ur, "a_2"] \arrow[dr, "a_1"'] & & \cdot \\
& \cdot \arrow[ur, "b_1"'] &
\end{tikzcd}$$

It is easy to check that this is a preorder since B is a sub-actionstructure.

Now, thinking of the B potentially as the effects which actions may have, we want to find a condition which allows us to speak about *the* effect of an action a, meaning in some sense the accumulation of all postcomponents of a which lie in B. The weakest condition on B which makes sense of this is that B is a postcomponent of A (note that we shall now use this term for sub-actionstructures as well as for actions), according to the following:

2.7 Postcomponent: Definition A sub-actionstructure B of A is a *postcomponent* of A if for every $a \in A$ the decompositions of a for B form a directed set under $\leq$. ■

$$\begin{tikzcd}
& & \cdot \arrow[dr, "b_1"] & \\
\cdot \arrow[urr, "a_1"] \arrow[r, "a'"] \arrow[drr, "a_2"'] & \cdot \arrow[ur, "b'_1"] \arrow[dr, "b'_2"'] & & \cdot \\
& & \cdot \arrow[ur, "b_2"'] &
\end{tikzcd}$$

The picture shows what is meant; given two decompositions (a_1, b_1) and (a_2, b_2) of the same action, there must exist $a' \in A$ and $b'_1, b'_2 \in B$ to make the diagram commute.

The notion of "postcomponent" is purely static. We want our set of effects to constitute a postcomponent, but we also want them to make *dynamic* sense of the notion of accumulation. This means that as a accumulates effects through performing a sequence of reactions

$$a \searrow a_1 \cdot e_1 \searrow a_2 \cdot e_2 \cdot e_1 \quad \cdots$$

the effects should be a kind of precipitation of the performance, in the sense that they can contribute no further to the performance. But this is just what inertia means. So, finally:

2.8 Effect structure: Definition A sub-actionstructure E of A is an *effect structure for* A if E is a postcomponent whose members are all inert. ■

It is important to note that E need not contain all the arities. If not, then actions $a : m \to n$ where $n \notin E$ have no effect at all, i.e. no decomposition for E.

Several examples of effect structures were given in [8]; also several successively stronger conditions were given for a sub-actionstructure E to be an effect structure. It will suffice to recall one of these.

First, let us say that (a, e) is a *maximal* decomposition (for E) if it dominates (under $\leq$) any other decomposition of $a \cdot e$ for E. We say that $a^* \in A$ is *pure for* E if every decomposition (a^*, e), with $e \in E$, is maximal. We also say that (a^*, e) is a *pure decomposition for* E if a^* is pure for E and $e \in E$. The sense of "pure" here is that no further effect can be extracted (as a postcomponent) from a^*. Now it is easy to prove

2.9 Proposition If E is a sub-actionstructure of A, and every $a \in A$ has a pure decomposition for E, then E is a postcomponent of A. ■

In what follows, when an effect structure E is understood we shall consistently use e, f, g, h to range over effects, while a, b, c, d continue to stand for arbitrary actions.

3 Process Structure and the π-calculus

3.1 Processes: grammar and structural congruence In a previous paper [8] a uniform way of extending an arbitrary action structure A to a process algebra was given. Letting a range over A, the grammar of processes (which we shall denote by $P, Q, \ldots$) is[1]

$$\begin{array}{rcl} P & ::= & a \cdot P \;\big|\; P \otimes P \;\big|\; (x)P \\ & & \bullet P \;\big|\; \mathbf{0} \;\big|\; P + P \;\big|\; \boldsymbol{\partial} P \;\big|\; !P \end{array}$$

subject to the following arity rules, which ascribe an arity $m \in M_A$ to each process:

$$\frac{a : m \to n \qquad P : n}{a \cdot P : m} \qquad \frac{P : m \qquad Q : n}{P \otimes Q : m \otimes n} \qquad \frac{P : m}{(x)P : \mathbf{ab}_x m}$$

$$\frac{P : m}{\bullet P : m} \qquad \mathbf{0} : m \qquad \frac{P : m \qquad Q : m}{P + Q : m} \qquad \frac{P : m}{\boldsymbol{\partial} P : m} \qquad \frac{P : \epsilon}{!P : \epsilon}$$

Of these process constructions, the first three lift the action constructions to the level of processes – but only admitting composition between an action and a process, not between two processes. We impose a *structural congruence* upon the grammar, lifting the action structure algebra to processes as far as possible. Structural congruence is the smallest congruence relation $\equiv$ satisfying these four laws:

$$\begin{array}{rcl} P & \equiv & \mathbf{1} \cdot P \\ a \cdot (b \cdot P) & \equiv & (a \cdot b) \cdot P \\ (a \cdot P) \times (b \cdot Q) & \equiv & (a \otimes b) \cdot (P \otimes Q) \\ (x)(a \cdot P) & \equiv & (\mathbf{ab}_x a) \cdot (x)P \;. \end{array}$$

[1] The notation $(x)P$ is used here, where $\mathbf{ab}_x P$ was used in the previous paper [8]. It seems better to reserve the latter for the functorial operation.

From now on we shall elide the composition symbol $\cdot$, writing ab for $a \cdot b$ and aP for $a \cdot P$. We also give (elided) composition highest binding power and all other binary operators lowest binding power, so that $ab \otimes cd$ means $(ab) \otimes (cd)$, $(x)aP$ means $(x)(aP)$, $\mathbf{ab}_x c \otimes d$ means $(\mathbf{ab}_x c) \otimes d$ and $\boldsymbol{\partial} P + Q$ means $(\boldsymbol{\partial} P) + Q$.

3.2 Control operators and dynamics The last five process constructions are called *control terms*, constructed by the *control operators* $\bullet, \mathbf{0}, +, \boldsymbol{\partial}$ and $\mathbf{!}$. They are called respectively *committal*, *zero*, *sum*, *delay* and *replication*. Their meaning is give not by equations but by reduction rules; these provide the dynamics of processes.

We say that an occurrence in P of an action a or process term Q is *ready* if it is not inside any control term. A *context* $\mathcal{C}$ is a process term with a single hole, denoted by [ ], to be filled by an action or process term. We write $\mathcal{C}[P]$ for result of placing P in the context $\mathcal{C}$. If the hole occurs readily then $\mathcal{C}$ is a *ready* context.

We now define a one-step reduction relation reduction relation $\searrow^1$ over processes; it is the union of (one-step) *action reduction* $\searrow^1_{\mathbf{a}}$ and (one-step) *control reduction* $\searrow^1_{\mathbf{c}}$. There is a single rule for action reduction:

$$\text{if } a \searrow a' \text{ and } \mathcal{C} \text{ is ready then } \mathcal{C}[a] \searrow^1_{\mathbf{a}} \mathcal{C}[a'].$$

For control reduction there are two rules each for sum, delay and replication:

$$\begin{array}{lll} P + Q \searrow^1_{\mathbf{c}} P & \boldsymbol{\partial} P \searrow^1_{\mathbf{c}} P & \mathbf{!}P \searrow^1_{\mathbf{c}} P \otimes \mathbf{!}P \\ P + Q \searrow^1_{\mathbf{c}} Q & \boldsymbol{\partial} P \searrow^1_{\mathbf{c}} \bullet \boldsymbol{\partial} P & \mathbf{!}P \searrow^1_{\mathbf{c}} \bullet \mathbf{!}P \end{array}$$

and two rules for committal:

$$\begin{array}{rcl} \bullet P \otimes \bullet Q & \searrow^1_{\mathbf{c}} & \bullet (P \otimes Q) \\ (x) \bullet P & \searrow^1_{\mathbf{c}} & \bullet (x)P \end{array}$$

together with the context rule

$$\text{if } P \searrow^1_{\mathbf{c}} P' \text{ and } \mathcal{C} \text{ is ready then } \mathcal{C}[P] \searrow^1_{\mathbf{c}} \mathcal{C}[P'].$$

We also define $\searrow$, $\searrow_{\mathbf{a}}$ and $\searrow_{\mathbf{c}}$ as the transitive reflexive closures of $\searrow^1$, $\searrow^1_{\mathbf{a}}$ and $\searrow^1_{\mathbf{c}}$. We take all these relations to be closed under structural congruence.

The purpose of these rules is to recover, in a general framework, the familiar notion of transition in existing process calculi. We shall shortly define labelled transitions $P \xrightarrow{a} P'$ in terms of reductions of the special form

$$P \searrow a \bullet P'.$$

We shall call such a reduction a *commitment*; it reduces P to the composition of an action with a committal. Consider for example the following transition in SCCS:

$$a \bullet P \times \boldsymbol{\partial} \overline{a} \bullet Q \xrightarrow{\mathbf{1}} P \times Q;$$

in this framework it will be inferred as follows:

$$\begin{array}{rcll} a \bullet P \otimes \partial\overline{a} \bullet Q & \searrow^{1}_{c} & a \bullet P \otimes \overline{a} \bullet Q & \text{(first } \partial \text{ rule)} \\ & \equiv & (a \otimes \overline{a})(\bullet P \otimes \bullet Q) & \\ & \searrow^{1}_{c} & (a \otimes \overline{a}) \bullet (P \otimes Q) & \text{(first committal rule)} \\ & \searrow^{1}_{c} & \mathbf{1} \bullet (P \otimes Q) & \text{(action reduction rule).} \end{array}$$

Another commitment can also be inferred for this process, namely

$$a \bullet P \otimes \partial\overline{a} \bullet Q \quad \searrow \quad a \bullet (P \otimes \partial\overline{a} \bullet Q).$$

We are now ready to define labelled transitions over processes. As we said above, they will be defined in terms of commitments, $P \searrow a \bullet P'$. We wish to allow any effect of a in such a commitment to act upon P', which is achieved as follows:

3.3 Process transition: Definition Given an action structure A, the *transition relations* $\{\xrightarrow{a} \mid a \in A\}$ are to be the smallest binary relations on processes over A which satisfy the rule

$$\frac{P \searrow ae \bullet P'}{P \xrightarrow{a} eP'}.$$

■

See also [8], Section 6. These relations are the basis for a uniform definition of bisimilarity over processes.

3.4 The π-calculus We shall now see what is needed in an action structure for the π-calculus, to recover the version of polyadic π-calculus defined in [9]. (We shall not recover the *concretions* of that paper; but they do not add expressive power as far as processes are concerned.)

We find that there are three process constructions in the polyadic π-calculus which are not among those in Paragraph 3.1; they are

$P \mid Q$	parallel composition
$(\boldsymbol{\nu} x)P$	restriction
$\pi.P$	prefixing

where π may be one of the following:

$x(\vec{y})$	(input $\vec{y}$ through port x)
$\overline{x}\langle\vec{y}\rangle$	(output $\vec{y}$ through port x)
τ	(silent action)

Here $\vec{y}$ is a vector of names, assumed distinct in the case of input since they are binding occurrences.

Parallel composition $\mid$ is familiar from CCS; here we define it in terms of other operations, as was done in [10] to derive CCS from SCCS, as follows:

$$P \mid Q \stackrel{\text{def}}{=} (P \otimes \boldsymbol{\partial} Q) + (\boldsymbol{\partial} P \otimes Q).$$

For the rest, we need only require that the π-calculus action structure [2] contain the following five elementary kinds of action:

$\tau : 0 \to 0$	the silent action
$\mathbf{out}_x : m \to 0$	output m names through port x
$\mathbf{in}_x : 0 \to m$	input m names through port x
$\langle x \rangle : 0 \to 1$	the name x as a datum
$\boldsymbol{\nu} : 0 \to 1$	a new name as a datum.

Two abbreviations which we shall use immediately are

$$\begin{aligned} \langle \vec{y} \rangle &\stackrel{\text{def}}{=} \langle y_1 \rangle \otimes \cdots \otimes \langle y_m \rangle \\ (\vec{y})(-) &\stackrel{\text{def}}{=} (y_1) \cdots (y_m)(-) \end{aligned}$$

where m is the length of $\vec{y}$.

We can now define prefixing and restriction upon processes as follows, adopting the device of [10] that prefixing incorporates a delay ($\boldsymbol{\partial}$):

$$\begin{aligned} \tau.P &\stackrel{\text{def}}{=} \tau \cdot \bullet \boldsymbol{\partial} P \\ \overline{x}\langle \vec{y} \rangle.P &\stackrel{\text{def}}{=} \langle \vec{y} \rangle \cdot \mathbf{out}_x \cdot \bullet \boldsymbol{\partial} P \\ x(\vec{y}).P &\stackrel{\text{def}}{=} \mathbf{in}_x \cdot (\vec{y}) \bullet \boldsymbol{\partial} P \\ (\boldsymbol{\nu} x)P &\stackrel{\text{def}}{=} \boldsymbol{\nu} \cdot (x)P \end{aligned}$$

where for clarity, just for now, we have written composition $\cdot$ explicitly. What is more, the reaction relation $\searrow$ will be the least for which

$$\mathbf{out}_x \otimes \mathbf{in}_x \searrow \tau \otimes \mathbf{1}_m .$$

We have now seen informally what should be in an action structure for the π-calculus; in the next section we define two such action structures.

4 Action Structures for the π-calculus

In the last section we found that the prefixing and restriction operations of the polyadic π-calculus can be defined in terms of a few more elementary kinds of action. We are now read to examine action structures generated by these primitives (and one other).

[2] We assume now that the arity monoid in this action structure is the natural numbers under addition, and that each name has arity 1. The action of abstraction upon arities is therefore given by $\mathbf{ab}_x m = 1 + m$. There is an easy generalization if we wish the names to be sorted, as discussed near the beginning of Section 2.2 above.

The most natural action structure is actually one which underlies a synchronous version of π-calculus, analogous to SCCS rather than CCS. ("Synchronous" here means that arbitrarily many interactions *may* be synchronized; it does not imply that synchronization is *necessary*.) This action structure, which we shall call PIC, needs no silent τ action. We shall also define PIC_τ, differing from PIC only in the presence of τ and a corresponding adjustment to the reaction rule; by this means the interleaving treatment of actions in the original π-calculus is faithfully recovered.

4.1 The action structures PIC and PIC_τ In [8], Section 3, the action structure SCCS is defined. The body of each action is a multiset over $Z \cup \overline{Z}$, where Z is a set known as the (positive) particles, and $\overline{Z} = \{\overline{\pi} \mid \pi \in Z\}$ is a set known as the negative particles, disjoint from Z. Reaction consists simply in cancelling a complementary pair $\{\pi, \overline{\pi}\}$. SCCS also allows some particles in an action to be restricted.

To get PIC from SCCS, all we need to do is to allow information, i.e. names, to flow between complementary particles in a reaction. It is also convenient to think of a restriction $\boldsymbol{\nu}x$ as a particle. So particles π of PIC are now given by

$$\pi \ ::= \ x(\vec{y}) \ \big| \ \overline{x}\langle\vec{y}\rangle \ \big| \ \boldsymbol{\nu}x .$$

The first two kinds are *input* and *output* (collectively *i/o*) particles, of positive and negative sign respectively. In the input particle $x(\vec{y})$ the vector $\vec{y}$ consists of distinct names, which are binding occurrences. The name in a restriction particle is also binding.

In PIC_τ, τ is admitted as a fourth kind of particle.

Because of binding, we cannot take the body of an action to be simply a multiset. We wish to allow an action to contain a sequence such as

$$\ldots \ x(y)\,\overline{y}\langle z\rangle \ \ldots$$

representing the receipt of a name y followed by its use as a port. At the process level we may think of the successive reactions of these particles as simultaneous; but we must still admit that the reaction of the second depends upon the receipt of the datum y via the reaction of the first.

We therefore declare the body S of an action to be a *partial sequence*

$$[\,\pi_1 \ \ldots \ \pi_n\,]$$

of particles; that is, a sequence in which we allow the commutation $\pi\pi' = \pi'\pi$ of any neighbouring pair neither of which binds a name occurring in the other. The general form of an action $a : m \to n$ is then

$$a \ = \ (\vec{x})S\langle\vec{y}\rangle,$$

where

- $\vec{x}$ is an m-vector of distinct names, the *imported* names;

- S is a partial sequence of particles, the *body*;

- $\vec{y}$ is an n-vector of names, not necessarily distinct, the *exported* names.

The imported names are bound. The scope of each bound name extends to the right of its binding occurrence, and includes the exported names. We identify actions which only differ by alpha-conversion (change of bound names). We tacitly assume in what follows that by suitable alpha-conversion, a name bound in a action is distinct from any other name in that action or in any other action under discussion.

As examples, we immediately define the kinds of action (five for PIC, six for PIC_τ) in terms of which all others may be expressed. They are all listed in Paragraph 3.4, except $\boldsymbol{\omega}$.

$$\begin{array}{rrcll} & \mathbf{out}_x : m \to 0 & \stackrel{\text{def}}{=} & (\vec{y})[\,\overline{x}\langle\vec{y}\rangle\,]\langle\,\rangle & (|\vec{y}| = m) \\ & \mathbf{in}_x : 0 \to m & \stackrel{\text{def}}{=} & (\,)[\,x(\vec{y})\,]\langle\vec{y}\rangle & (|\vec{y}| = m) \\ & \langle x\rangle : 0 \to 1 & \stackrel{\text{def}}{=} & (\,)[\quad]\langle x\rangle & \\ & \boldsymbol{\nu} : 0 \to 1 & \stackrel{\text{def}}{=} & (\,)[\,\boldsymbol{\nu}x\,]\langle x\rangle & \\ & \boldsymbol{\omega} : 1 \to 0 & \stackrel{\text{def}}{=} & (x)[\quad]\langle\,\rangle & \\ (\mathsf{PIC}_\tau \text{ only}) & \tau : 0 \to 0 & \stackrel{\text{def}}{=} & (\,)[\,\tau\,]\langle\,\rangle. & \end{array}$$

We shall allow ourselves to omit any empty component of an action; so we may write $[\,\tau\,]$ for $(\,)[\,\tau\,]\langle\,\rangle$, and $(\vec{y})[\,\overline{x}\langle\vec{y}\rangle\,]$ for $(\vec{y})[\,\overline{x}\langle\vec{y}\rangle\,]\langle\,\rangle$. We shall also write $\boldsymbol{\nu}\vec{x}$ for a sequence $\boldsymbol{\nu}x_1\,\boldsymbol{\nu}x_2\,\cdots$ of particles.

4.2 Operations The action structure operations are defined identically for PIC and PIC_τ. First, the identities are given by

$$\mathbf{1}_m \stackrel{\text{def}}{=} (\vec{x})\langle\vec{x}\rangle \qquad (|\vec{x}| = m).$$

To define the other operations we use substitutions like $\sigma = \{\vec{u}/\vec{v}\}$, where $\vec{v}$ is a vector of distinct names; $\sigma(-)$ is the result of simultaneously replacing each free occurrence of u_i in $(-)$ by the corresponding v_i, first alpha-converting $(-)$ to change any bound uses of $\vec{v}$. We shall also use $\frown$ to concatenate partial sequences. Suppose $a = (\vec{u})S\langle\vec{v}\rangle$ and $b = (\vec{x})T\langle\vec{y}\rangle$; then composition, product and abstraction are given as follows:

$$\begin{array}{rcll} a \cdot b & \stackrel{\text{def}}{=} & (\vec{u})S \frown \sigma T\langle\sigma\vec{y}\rangle & (\sigma = \{\vec{v}/\vec{x}\}) \\ a \otimes b & \stackrel{\text{def}}{=} & (\vec{u}\vec{x})S \frown T\langle\vec{v}\vec{y}\rangle & \\ \mathbf{ab}_x a & \stackrel{\text{def}}{=} & (x\vec{u})S\langle x\vec{v}\rangle. & \end{array}$$

Recall our convention that a bound name differs from all names used outside its scope. Thus $a \otimes b$ is simply the juxtaposition of the two actions; no order is dictated by the concatenation, since the convention ensures that $S \frown T = T \frown S$ in this case. But in $a \cdot b$ the substitution σ may replace some names free in T by names bound in S, and then $S \frown \sigma T \neq \sigma T \frown S$.

It is a routine matter to verify the action structure axioms. Note in particular that the export of x in $\mathbf{ab}_x a$ ensures that $\mathbf{ab}_x$ is functorial.

Exercise Show that ω cannot be expressed in terms of the other basic actions and the above-defined operations. ■

For future convenience we define the following:

$$
\begin{aligned}
(x)a &\stackrel{\text{def}}{=} (x\vec{u})S\langle\vec{v}\rangle \qquad (\text{where } a = (\vec{u})S\langle\vec{v}\rangle,\ x \notin \vec{u}) \\
(\boldsymbol{\nu}x)a &\stackrel{\text{def}}{=} \boldsymbol{\nu}\cdot(x)a
\end{aligned}
$$

4.3 Dynamics The atomic reaction relation $\searrow^{1}$ of PIC is defined as follows. Whenever a takes the form

$$(\vec{x})\, S\,{}^\frown[\,\overline{u}\langle\vec{v}\rangle\ u(\vec{w})\,]{}^\frown T\,\langle\vec{y}\rangle$$

(after suitable commutations), then

$$a \searrow^{1} (\vec{x})\, S\,{}^\frown\sigma T\,\langle\sigma\vec{y}\rangle,$$

where σ is the substitution $\{\vec{v}/\vec{w}\}$. We call $\overline{u}\langle\vec{v}\rangle\ u(\vec{w})$ a *redex* of a.

Example Let $a = (x)[\,\overline{u}\langle x\rangle\ u(w)\,\overline{w}\langle y\rangle\ x(z)\,]\langle wz\rangle$. Then

$$
\begin{aligned}
a &\searrow^{1} (x)[\,\overline{x}\langle y\rangle\ x(z)\,]\langle xz\rangle \\
&\searrow^{1} (x)\langle xy\rangle.
\end{aligned}
$$

■

Note that the port of a redex (in the example, first u then x) may be free or bound.

The only difference in PIC_τ is the the redex is not just deleted, but replaced by τ. So in PIC_τ the example becomes

$$a \searrow^{1} \cdots \searrow^{1} (x)[\,\tau\ \tau\,]\langle xy\rangle.$$

It is important to check that $\searrow^{1}$ is indeed preserved by the operations; for example $a \searrow^{1} a'$ implies $b\cdot a \searrow^{1} b\cdot a'$.

The reaction relation $\searrow$ is defined to be $(\searrow^{1})^*$, the transitive reflexive closure of the atomic relation.

Exercise If e is inactive, or even inert, $e\cdot a$ may still have reactions not arising solely from a. Consider the reactions of $\langle u\rangle\cdot a$ in the above example. ■

4.4 Discussion In the original π-calculus every prefix, and therefore every action, consisted of just a single particle. By introducing the action structure PIC or PIC_τ, surely we have gone much further than we need to capture the behaviour of the π-calculus?

In fact PIC and PIC_τ reflect the real performance of distributed systems rather more closely than does the original formulation of π-calculus. For the transition system of the latter explicitly prevents more than a single interaction to occur in a single transition, and to implement this would actually require a global scheduler – something which is absent in a distributed system by definition! But the π-calculus was designed to be an tractable theoretical tool, not to be a faithful model of implementation; its interleaving of actions is what makes the expansion law of the π-calculus [11] easy to use, since the number of terms in the expansion does not explode exponentially. The same point was made for CCS in [10]. Moreover, the interleaving was justified at least for CCS [10] (and the same should hold for π-calculus) by a theorem that at the semantic level of observation equivalence it makes no difference whether single observations are treated as simultaneous or interleaved.

In [8], Paragraph 7.4, an appropriate incident set was defined to recapture the interleaving of actions for CCS; later in this paper we shall similarly use PIC_τ to recapture interleaving in the π-calculus.

4.5 Effects and transitions What should be the effect of performing an action in PIC or PIC_τ, i.e. what effect structure should we adopt? There is no single answer; to simplify the decision, we shall consider only actions whose body contains no i/o or τ particles (so they are certainly inert), i.e. actions of the form

$$(\vec{x})[\,\boldsymbol{\nu}\vec{u}\,]\langle\vec{y}\rangle.$$

Into this class fall the following: $\langle x\rangle$, a free datum; $(x)\langle xx\rangle$, which forms a link by identifying two names; and $[\,\boldsymbol{\nu}x\,]\langle xx\rangle$, which forms a private link. These are important cases; we shall see the last illustrated in Paragraph 4.6. But the class contains some undesirables; one such is $\boldsymbol{\omega}$, which discards a name. It is undesirable because discarding can be made into a postcomponent of *any* action; for example if $a : 1 \to 1$ then we can make $(xy)\langle x\rangle$ (which discards its second argument) into an effect of a by writing

$$a \;=\; a \cdot (x)\langle xx\rangle \cdot (xy)\langle x\rangle.$$

To exclude the discarding of names, we propose that the effect structure E for PIC and PIC_τ consist of all actions

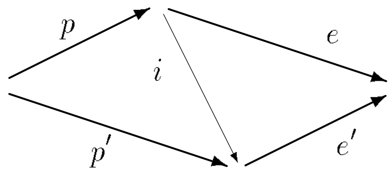
$$(\vec{x})[\,\boldsymbol{\nu}\vec{u}\,]\langle\vec{y}\rangle, \qquad \vec{x}\cup\vec{u}\subseteq\vec{y}.$$

It is routine to verify that E is indeed an effect structure. Furthermore, every action has a pure decomposition for E. (See after Definition 2.8.) In fact the pure actions P are those actions $(\vec{x})S\langle\vec{y}\rangle$ which satisfy two conditions:

- $\vec{y}$ contains no free names, and no repetitions;

- every restricted name in S occurs in at least one i/o particle.

Exercise Show that every a in PIC or PIC_τ can be expressed as $a = p \cdot e$, where $p \in P$ and $e \in E$. Show further that this is unique up to isomorphism; that is, if also $a = p' \cdot e'$ ($p' \in P$ and $e' \in E$) then there exists $i = (\vec{u})\langle\vec{v}\rangle$, where $\vec{v}$ is a permutation of $\vec{u}$, making the following diagram commute:



This makes (P, E) into a Freyd-Kelly factorization in PIC and in PIC_τ [4], considered as categories. ■

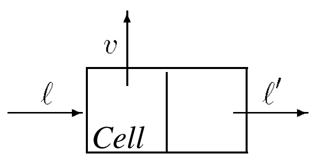
Transitions were introduced in Paragraph 3.3. Recall fron there that if P has a commitment $P \searrow a \bullet P'$, then any decomposition (b, e) of a for E can be used to provide a transition

$$P \xrightarrow{\;b\;} eP'.$$

In the next section we focus on the case when b is a special kind of action known as an *incident*; in such actions, all internal reaction (which could not affect external interaction) has been carried out.

4.6 Illustration Let us now illustrate the power of composite action in PIC, when extended by process structure as described in Section 3. We shall use an encoding of data structures as processes, close to that described in [9]. Specifically, we consider list structures. A single list-cell located at ℓ, containing a value-name v and pointing to a successor cell located at ℓ', may be represented by the action

$$[\,\ell(cn)\,\overline{c}\langle v\ell'\rangle\,] : 0 \to 0.$$



We shall represent it by the process *Cell* of arity 3, parametric upon ℓ, v and ℓ':

$$\mathit{Cell} \stackrel{\text{def}}{=} (\ell v \ell')\,\boldsymbol{\partial}[\,\ell(cn)\,\overline{c}\langle v\ell'\rangle\,] \bullet \mathbf{0}.$$

Similarly, the empty list-cell *Nil* of arity 1, parametric upon its location, is given by

$$\mathit{Nil} \stackrel{\text{def}}{=} (\ell)\,\boldsymbol{\partial}[\,\ell(cn)\,\overline{n}\langle\,\rangle\,] \bullet \mathbf{0}.$$

Now for the present discussion we shall understand a *value*, i.e. a candidate for membership of a list, to be a process *Val* of arity 1, of the form $(v)\,!V$; it is parametric upon its location, and the replication allows its repeated "use". (Note that in this sense *Cell* and *Nil* are not values. But there are variations on this theme, as was indicated in [9].) A located value is naturally written as $\langle v\rangle \mathit{Val}$ (i.e. $\langle v\rangle \cdot \mathit{Val}$), because $\langle v\rangle$ is indeed an action.

So what is the list containing k values $\mathit{Val}_1, \ldots, \mathit{Val}_k$? It is itself a value $L : 1 \to 0$, given by

$$L \;=\; (\ell_0)\,!(\boldsymbol{\nu} v_1\ell_1 \cdots v_k\ell_k)\Big(\langle \ell_0 v_1\ell_1\rangle \mathit{Cell} \otimes \cdots \otimes \langle \ell_{k-1}v_k\ell_k\rangle \mathit{Cell} \otimes \langle \ell_k\rangle \mathit{Nil} \\ \otimes \langle v_1\rangle \mathit{Val}_1 \otimes \cdots \otimes \langle v_k\rangle \mathit{Val}_k\Big).$$

Note this is a *process* expression; for example, restriction on processes was defined at the very end of Section 3. Note also how each value Val_i has been located for inclusion in the list.

Now let us define the single actions *head* and *tail*, which get hold of the first element and the remainder of an arbitrary list value. They are

$$\begin{aligned} \mathit{head} : 1 \to 1 &\stackrel{\text{def}}{=} \mathit{get\,first} \\ \mathit{tail} : 1 \to 1 &\stackrel{\text{def}}{=} \mathit{get\,second}, \end{aligned}$$

where

$$\begin{aligned} \mathit{get} : 1 \to 2 &\stackrel{\text{def}}{=} (\ell)[\,\boldsymbol{\nu} cn\; \overline{\ell}\langle cn\rangle\; c(v\ell')\,]\langle v\ell'\rangle \\ \mathit{first} : 2 \to 1 &\stackrel{\text{def}}{=} (v\ell)\langle v\rangle \\ \mathit{second} : 2 \to 1 &\stackrel{\text{def}}{=} (v\ell)\langle \ell\rangle. \end{aligned}$$

Now let us see how to define processes, given these definitions. Suppose that we have L located at ℓ_0, and we wish to provide its first member as parameter to a process $P : 1$. This can be done by prefixing the action $\langle l_0\rangle \mathit{head}$ to P; then in a single transition this action gains access to the head of L, as follows:

$$\langle \ell_0\rangle \mathit{head}\,\bullet\, P \otimes \langle \ell_0\rangle L \;\xrightarrow{\;b\;}\; (\boldsymbol{\nu} v \cdots)\Big(\langle v\rangle P \otimes \langle v\rangle \mathit{Val}_1 \otimes \langle \ell_0\rangle L \;\cdots\Big).$$

We see that P has indeed "received" v, the location of L's first member Val_1. Note that, due to replication, the located list is still available for further access. The dots $\cdots$ represent material which is irrelevant for present purposes. The action b will normally be an incident (see Section 5); the left-hand side performs a commitment, and the action of this commitment decomposes into (b, e), where the effect e in this case produces the private link v bewteen P and the value Val_1.

Each step down the list L need not involve a separate transition. The following single transition provides access to the fourteenth member of L, where tail^n means the composition of n *tail*s:

$$\langle \ell_0\rangle \mathit{tail}^{13}\mathit{head}\,\bullet\, P \otimes \langle \ell_0\rangle L \;\xrightarrow{\;b'\;}\; (\boldsymbol{\nu} v \cdots)\Big(\langle v\rangle P \otimes \langle v\rangle \mathit{Val}_{14} \otimes \langle \ell_0\rangle L \;\cdots\Big).$$

Let us briefly consider this illustration in broader terms. It naturally prompts the question of why it should be necessary to distinguish process from actions. Why should we not add the control operations $\bullet, \mathbf{0}, +, \boldsymbol{\partial}$ and $\,!\,$ to action structures, combining the notions of process structure and action structure?

One answer is that by distinguishing these notions we get a treatment of processes closer to existing process calculi. However, it may be that process calculi themselves would be enriched by treating them as action structures.

The second answer is that there are some interesting technical problems to resolve before the extension of action structures (by adding control operations) is well understood. Work in this direction will be reported elsewhere.

5 Bisimilarity and Incident Sets

In the first few paragraphs of this section we return to the study of processes over an arbitrary action structure, to review the definitions of bisimilarity and incident sets, given in [8]. Then we look at particular incident sets for the π-calculus.

In Paragraph 3.3 we defined the labelled transition relations $\{\xrightarrow{a} \mid a \in A\}$ on processes over an arbitrary action structure A, with a designated effect structure E. In terms of this, one may define a bisimilarity relation for any subset K of A :

5.1 Bisimulation: Definition A binary relation $\mathcal{S}$ over processes is a K-*simulation* if, for all $(P, Q) \in \mathcal{S}$ and $k \in K$, if $P \xrightarrow{k} P'$ then there exist $k' \in K$,$e \in E$ and Q' such that

$$k = k'e, \quad Q \xrightarrow{k'} Q' \ \text{ and } \ (eP', Q') \in \mathcal{S}.$$

If $\mathcal{S}$ is also symmetric then it is called a K-*bisimulation*. If the pair (P, Q) is in some K-simulation then we say that Q K-*simulates* P and write $P \lesssim_K Q$. If (P, Q) is in some K-bisimulation then we say that P and Q are K-*bisimilar* and write $P \sim_K Q$. ■

It is easy to check that K-similarity is a preorder, and K-bisimilarity is an equivalence, provided K satisfies the condition that $a \in K$ iff $ae \in K$, for all $a \in A$ and $e \in E$ such that the composition ae is defined.

These relations are larger, the smaller we take K to be. We naturally seek general conditions upon K which ensure that the relations will be *congruential*, i.e. preserved by all the process constructions. This concern leads to the notion of *incident*; besides yielding congruential bisimilarity, the notion also seems to be of intrinsic interest in action structures.

We first define two preliminary notions.

5.2 Preactions and strong inertia A *preaction* is a unary operation $\varphi(-)$ over A gained by composing operations of the form $c \cdot (-)$, $c \otimes (-)$, $(-) \otimes c$ and $\mathbf{ab}_x(-)$ where c may be any action. Thus $\mathbf{ab}_x(b \otimes (c \cdot (-)))$ is a preaction, but $b \otimes ((-) \cdot c)$ is not since it uses postcomposition, $(-) \cdot c$.

Recall that an effect e is required to be inert; that is, e cannot "take part" in any reaction when placed in a precomposition context the action $c \cdot (-)$. The first important property of preactions is that, as it turns out, an effect is incapable of partaking in a reaction in *any* preaction context φ (not just precomposition); that is, whenever $\varphi(e) \searrow b$ then there is some preaction φ' such that $\varphi \searrow \varphi'$ and $\varphi'(e) = b$. This property was called *strong inertia* in [8].

Preactions also play an important part in the following:

5.3 Incident sets: Definition A subset K of A is an *incident set* if it satisfies the following two conditions, for every $a \in A$:

- for all $e \in E$ such that ae is defined, $ae \in K$ iff $a \in K$;
- for any preaction φ and any $\ell \in K$, if $\varphi(a) \searrow \ell$ then there exists $k \in K$ such that $a \searrow k$ and $\varphi(k) \searrow \ell$. ■

Due to the first clause, the notion of incident set is relative to a given effect structure. Note that K need not be a sub-actionstructure of A; this will become clear in examples. Further, note that there may be many incident sets relative to a given E; in particular, both A and the empty set are always incident sets.

We now restate the main property of incidents, Theorem 7.9 of [8]:

Theorem If K is an incident set for an arbitrary effective action structure (A, E), then $\lesssim_K$ is a precongruence and $\sim_K$ a congruence in the process structure over A. ■

When a particular incident set K is understood, we shall call its members *incidents*. We shall use k, ℓ to range over incidents.

5.4 The nature of incidents In many action structures the reaction relation $\searrow$ is confluent. When that occurs, it seems that the interesting incident sets contain only inactive actions. All the examples of action structures treated in [8] were confluent except that for Petri nets, for which an incident set was not defined. In particular, readers may check that the incident set given there for CCS contains only inactive actions. (It is also worth noting that it is not a sub-actionstructure; the product of two incidents may indeed be active.)

Now PIC is not confluent, as can be easily seen as follows. Let $a = [\,\overline{x}\langle w\rangle\ \overline{x}\langle z\rangle\ x(y)\,]$; then a has two distinct one-step reactions neither of which can be continued:

$$\begin{array}{ccll} a & \searrow & [\,\overline{x}\langle w\rangle\,] & \not\searrow \\ a & \searrow & [\,\overline{x}\langle z\rangle\,] & \not\searrow \end{array}$$

Indeed we can use this to show that *any* non-empty incident set for PIC must contain active members. For let K be such an incident set, and suppose it contains the one-particle action $\ell = [\,\overline{x}\langle z\rangle\,]$. (The exercise below shows that indeed if $K \neq \emptyset$ then $\ell \in K$.) Let $a = [\,\overline{x}\langle z\rangle\ x(y)\,]$; a is active, and we show $a \in K$. Let φ be the preaction $[\,\overline{x}\langle w\rangle\,] \otimes (-)$. Then $\varphi(a) \searrow \ell \in K$; so there exists $k \in K$ for which $a \searrow k$ and $\varphi(k) \searrow \ell$. But the only action $a' \neq a$ such that $a \searrow a'$ is the unit action $\mathbf{1}_0$, and clearly $\varphi(\mathbf{1}_0) \not\searrow \ell$. Hence $a \in K$, as we claimed.

Exercise If K is a non-empty incident set for PIC, show that $[\,\overline{x}\langle z\rangle\,] \in K$. (*Hint*: Let k be any member of K; consider the preaction $k \otimes [\,x(y)\,] \otimes (-)$. ■

A main aim of this paper is to give interesting and natural incident sets for PIC. First we shall tackle the easier task of finding a very modest incident set for PIC_τ.

5.5 An incident set for PIC_τ Recall that in a PIC_τ reaction the redex is not merely cancelled, but replaced by τ.

Now let us define the *weight* $|\pi|$ of a particle to be 0 if $\pi = \boldsymbol{\nu} x$, 1 if π is an i/o particle, and 2 if $\pi = \tau$. The weight $|a|$ of an action a is the sum of its particle weights. We see easily that $|a \cdot b| = |a| + |b|$, $|a \otimes b| = |a| + |b|$ and $|\mathbf{ab}_x a| = |a|$. Also, reaction preserves weight; this is the important difference between PIC_τ and PIC, and is what allows PIC_τ to have a rather small incident set, as we shall see.

We say that an i/o particle in a is *restricted* if its port is restricted. Now we can define our candidate incident set.

Definition Let K_τ^{PIC} contain every action in PIC_τ whose particle-sequence (apart from restriction particles) is either empty or of the form $\overline{x}\langle\vec{z}\rangle$, or $y(\vec{w})$, or τ, or $\overline{x}\langle\vec{z}\rangle\ y(\vec{w})$, where in each case the i/o particles are unrestricted. ■

Thus each $k \in K_\tau^{\mathsf{PIC}}$ has weight at most 2. Indeed, suppose K is some candidate for an incident set, and we have $\varphi(a) \searrow \ell \in K$. Then $|a| \leq |\ell|$ since reaction preserves weight, so if $a \searrow k \in K$ then also $|k| \leq |\ell|$; thus the presence in K of actions of unbounded weight cannot help to establish K as an incident set.

We can now establish

Proposition K_τ^{PIC} is an incident set for PIC_τ.

Proof It is easy to verify the first clause of Definition 5.3. To verify the second clause, it is enough to verify it for the four basic forms of preaction: $c \cdot (-)$, $c \otimes (-)$, $(-) \otimes c$ and $\mathbf{ab}_x(-)$.

Suppose then that $\varphi(a) \searrow \ell \in K_\tau^{\mathsf{PIC}}$, where φ takes one of these four forms. Note that this reaction takes at most one step. We must find $k \in K_\tau^{\mathsf{PIC}}$ such that $a \searrow k$ and $\varphi(k) \searrow \ell$. We know $|a| \leq |\ell| \leq 2$, so we consider the three possible cases for $|a|$.

If $|a| = 0$ then already $a \in K_\tau^{\mathsf{PIC}}$ and we are done.

If $|a| = 1$ then a has a single i/o particle. If it is free then $a \in K_\tau^{\mathsf{PIC}}$ and we are done. If not, then $\varphi(a)$ also has this restricted particle, and no other i/o particle with the same port; but then $\varphi(a) \notin K_\tau^{\mathsf{PIC}}$, and a one-step reaction cannot reduce it to a member of K_τ^{PIC}, contrary to hypothesis.

If $|a| = 2$ and $a \notin K_\tau^{\mathsf{PIC}}$ (else we are done) there are three cases:

- a has two i/o particles of like sign. Then $\varphi(a)$ has the same property, and no reduction is possible; contradiction.

- a has one free and one restricted i/o particle. Then $\varphi(a)$ has two i/o particles (of which at least one is restricted) having different ports, and no reduction is possible; contradiction.

- a has two restricted i/o particles. Then $\varphi(a)$ has the same, and their interaction reduces $\varphi(a)$ to ℓ. But this interaction can also be done within a, yielding $k \in K_\tau^{\mathsf{PIC}}$ (with a single τ particle) for which $\varphi(k) = \ell$. ∎

This argument would fail if φ could take the form of a postcomposition $(-) \cdot c$; hence the asymmetry in the notion of preaction is important.

The question arises whether this incident set has special status among those possible for PIC_τ. It was demonstrated in [8], Proposition 7.13, that under certain conditions (satisfied by PIC_τ) there is always a smallest incident set which contains certain given actions. Is K_τ^{PIC} the smallest containing τ? For the full PIC_τ, it is not quite, but nearly so; the discrepancy has to do with i/o particles which carry an empty vector of names. But the assiduous reader may enjoy the following

Exercise Show that in the monadic PIC_τ (where every i/o particle carries a single datum name) K_τ^{PIC} as defined above is the smallest incident set. *Hint*: By similar methods to the previous exercise, first show that if $\tau \in K$ then every action of weight 1 is in K; then proceed to consider those of weight 2. ∎

How does the bisimilarity induced by K_τ^{PIC} relate to the congruences defined in terms of bisimilarity for the original π-calculus [11]? It turns out to be a little stronger; but it is also more natural in one sense. For it is obtained by a direct definition, whereas the original bisimilarities were not quite congruential; the congruences were defined as "bisimilarity under all substitutions". The relative merits of these congruences remain to be studied; one also hopes that action structures can throw some light upon the structure of the family of strong bisimilarities, not just for π-calculus but more generally.

5.6 An incident set for monadic PIC We now turn to the more challenging question of incident sets for PIC. Here, the absence of τ means that we no longer have a way to bound the weight of actions in an incident. Instead rely on another intuition, to reduce the class of possible incidents. Suppose a is in the context of some preaction φ and we are interested in the performance of $\varphi(a)$, e.g. its multistep reduction to an incident. Now a may contribute in two ways to this reduction; it may react internally, or it may interact with φ. Looking at a in isolation, it has two kinds of internal reaction; those which may preclude (or be precluded by) a's possible interactions with its environment, and those which neither preclude nor are precluded by such external interactions.

For the latter kind of internal reaction, nothing (i.e. no external interaction) is lost by letting them be performed earlier rather than later. By confining our attention (in forming an incident set) to actions which have carried out *all* such internal reactions, we limit the size of the incident set without endangering the essential second clause of Definition 5.3. A further consideration is that an incident should contain no particle which cannot eventually take part in some reaction.

The definitions which follow should be understood with these ideas in mind. Recall that we assume every bound name (in the actions under discussion) to be distinct from every other bound name, and from all names occurring free.

Definition The set of names *reachable* in a given action $a \in \mathsf{PIC}$ is the smallest such that

- all the imported and free names of a are reachable;

- if z is reachable and $z(\vec{w})$ or $\overline{z}\langle\vec{w}\rangle$ occurs in a then the names $\vec{w}$ are reachable.

An i/o particle is *reachable* in a if its port is reachable; an action is *reachable* if all its i/o particles are reachable. ■

Example Let

$$a \;=\; (x)[\,\boldsymbol{\nu} u\; x(y)\; u(v)\; \overline{u}\langle y\rangle\; \overline{v}\langle z\rangle\,]\langle yu\rangle;$$

then only the names x, y and the particle $x(y)$ are reachable. Note that a contains a redex $u(v)\,\overline{u}\langle y\rangle$ which has an unreachable port. If we reduce it we have

$$a \searrow^{1} (x)[\,\boldsymbol{\nu} u\; x(y)\, \overline{y}\langle z\rangle\,]\langle yu\rangle;$$

the reaction has created a reachable particle $\overline{y}\langle z\rangle$ from one which was previously unreachable, and has reduced the unreachable action a to a reachable one. ■

Surprisingly enough, the status of reachable actions differs according to whether we are considering full PIC, or only monadic PIC. We can prove the following:

5.7 Theorem The set K^{PIC} of reachable actions is an incident set in monadic PIC. ■

I also conjecture, but have not proved, that K^{PIC} is the smallest incident set in monadic PIC which includes the effects. If so, we may claim that it is the most natural incident set.

The proof of the theorem is omitted here. It follows similar lines to the proof of Theorem 7.4, concerning an incident set for the reflexive π-calculus. Of course, one immediately asks why K^{PIC} is *not* an incident set for full PIC. In Paragraph 5.8 below, we see that the discrepancy arises from an apparent anomaly of the binding convention in PIC.

The discrepancy may seem puzzling, if we recall from [9] that the polyadic π-calculus is definable from the monadic calculus. But that definition relies upon the sequentiality imposed by the prefixing construction of the π-calculus, and prefixing is absent in PIC! Further work is needed, specifically on the question of extending action structures by the addition of control operations, before this matter is fully understood.

In Section 7 below we enrich PIC to remove the anomaly. It seems that for many applications the enrichment will not affect the behaviour of processes; for others it may even yield an improved model of behaviour.

Readers who are interested in the detail of incident sets will gain insight from the following exercise, which shows clearly how the notion of incident separates proper from improper transitions.

Exercise Consider again the list-processing illustration in Paragraph 4.6. Consider the commitments of $(\boldsymbol{\nu}\ell_0)(\,\langle\ell_0\rangle\mathit{head}\bullet P \otimes \langle\ell_0\rangle L\,)$, where now the access to the list is restricted. Show that there is a large number of possible commitments of the form

$$(\boldsymbol{\nu}\ell_0)\Big(\,\langle\ell_0\rangle\mathit{head}\bullet P \otimes \langle\ell_0\rangle L\,\Big) \searrow a\bullet Q,$$

but only one in which a an incident. Show that this incident decomposes into two actions; the incident $[\,\boldsymbol{\nu}cn\,]$ which discards some "used" names, and the effect

$$[\,\boldsymbol{\nu}\ell_0\;\boldsymbol{\nu}v_1\ell_1\cdots v_k\ell_k\,]\langle\ell_0v_1v_1\ell_1\cdots v_k\ell_k\rangle$$

which transmits private names to Q. ■

5.8 An anomaly? In PIC, though not in monadic PIC, the interplay between binding and reaction has a surprising consequence which can be illustrated by a simple example. Let

$$b \;=\; [\,v(x)\;\underline{\overline{u}\langle x,y\rangle\;u(z,w)}\;\overline{v}\langle w\rangle\,]\langle x\rangle,$$

or by rearrangement

$$b \;=\; [\,u(z,w)\;\underline{\overline{v}\langle w\rangle\;v(x)}\;\overline{u}\langle x,y\rangle\,]\langle x\rangle,$$

where the two redexes in b are shown by underlining. These two redexes are disjoint (i.e. they involve distinct particles), so one may expect that they can be reduced in arbitrary order. But here are the two reaction sequences for b:

$$\begin{array}{lll} b & \searrow & [\,\underline{v(x)\,\overline{v}\langle w\rangle}\,]\langle x\rangle \searrow \langle y\rangle \\ b & \searrow & [\,u(z,w)\,\overline{u}\langle w,y\rangle\,]\langle w\rangle \quad \not\searrow^{1}. \end{array}$$

The second sequence can go no further! There is no redex, due to PIC's binding convention. Yet if we allowed a reaction, then the result would be $\{{}^{w}\!/z, {}^{y}\!/w\}\langle w\rangle = \langle y\rangle$ as in the first case. (Of course care is needed with such substitutions; their effect is different according to the sequence in which they are applied. But that is not the issue here.) This suggests that in avoiding such redexes PIC we have denied certain reactions which are intuitively acceptable. [3]

Our example can now be used to show that K^{PIC} is indeed not an incident set for full PIC. Let c be a slight variant of b, as follows:

$$c \;=\; [\,\boldsymbol{\nu}v\;v(x)\;\underline{\overline{u}\langle x,y\rangle\;u'(z,w)}\;\overline{v}\langle w\rangle\,].$$

[3] Even in monadic PIC, an action may have two disjoint redexes, but after one is reduced the other is no longer a redex; consider $[\,v(x)\,\overline{u}\langle x\rangle\;u(y)\,\overline{v}\langle y\rangle\,]$. But in monadic PIC it turns out that if both can be reduced in *one* order, then both can be reduced in the other order. This is what lies behind Theorem 5.7, and fails for polyadic PIC.

Now let φ be the preaction $\langle u\rangle\mathbf{ab}_{u'}(-)$; then (as in the first reaction sequence for b)

$$\varphi(c) \searrow [\,\boldsymbol{\nu} v\,]\langle u\rangle.$$

Since $[\,\boldsymbol{\nu} v\,]\langle u\rangle \in K^{\mathsf{PIC}}$, we look for $k \in K^{\mathsf{PIC}}$ such that

$$c \searrow k \ \text{ and } \ \varphi(k) \searrow [\,\boldsymbol{\nu} v\,]\langle u\rangle.$$

But $c \notin K^{\mathsf{PIC}}$, and its only reaction is

$$c \searrow c' \ \not\searrow^{1}, \ \text{ where } c' \ = \ [\,\boldsymbol{\nu} v\ u'(z,w)\,\overline{u}\langle w,y\rangle\,];$$

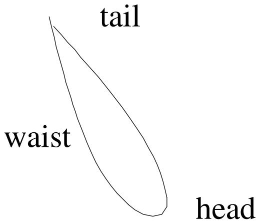
furthermore $\varphi(c') = [\,\boldsymbol{\nu} v\ u(z,w)\,\overline{u}\langle w,y\rangle\,]\langle u\rangle$ has no redex, so cannot reduce to $[\,\boldsymbol{\nu} v\,]\langle u\rangle$. So K^{PIC} is not an incident set for full PIC.

This anomaly of PIC, if it is one, is removed by the notion of reflexive substitution which we give in Paragraph 7.2 below.

6 A Graphical Action Structure

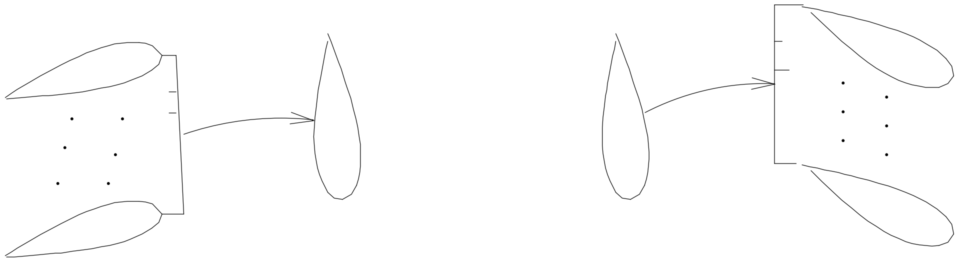
The action structure PIC has been presented in a very syntactic way. This helps formal manipulation, but some essential aspects are still hidden. Here we show that both the statics and the dynamics of PIC correspond to simple graphical ideas. This graphical presentation also helps us to understand the notions of effect and incident. Further, it suggests an enrichment of PIC which has appealing properties.

6.1 Bare π-nets We begin with a simple kind of graph called a *bare π-net*. It has *nodes* and *directed arcs*. Each node is drawn as a pointed oval with a *head*, a *waist* and a *tail* as shown.



Each directed arc is either a *source* or a *target* arc. We first consider *monadic* arcs. A monadic source arc is drawn from the head of a node (the *source* of the arc) to the waist of a node (the *port*); a monadic target arc is drawn from the waist of a node (the *port* of the arc) to the tail of a node (the *target*).

In general, an arc may be *polyadic*; that is, a source arc may have an ordered sequence of k source nodes, and a target arc may have an ordered sequence of k distinct target nodes, for any $k \geq 0$:

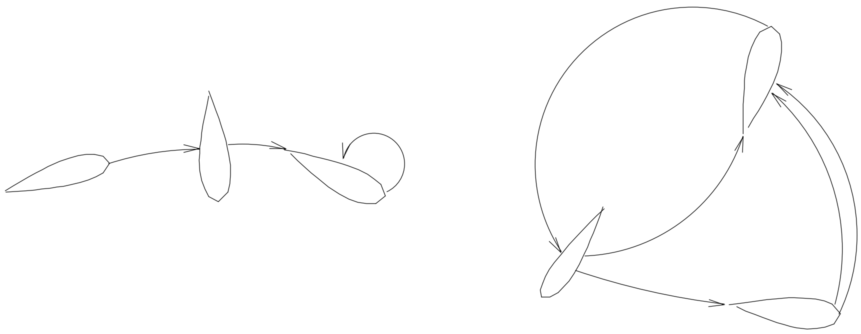


Note that the arc exists even when $k = 0$. A node may be used repeatedly in the rôles of source, target or port of the same or different arcs, subject to the following constraints:

- Target nodes of the same arc are distinct (as already stated);
- Each node is a target of at most one arc;
- There are no cycles consisting only of target arcs.

We say a (bare) π-net is *monadic* if all its arcs are monadic. From now on we shall use only monadic π-nets in our illustrations; Readers will find the ideas simpler to grasp if they think only of monadic π-nets, but may rest assured that the arguments go through unchanged for polyadic π-nets.

Here are two admissible bare monadic π-nets:



Here are two which are inadmissible, because they have target-arc cycles:

A little more terminology will be useful later. If a node m is the port of a target arc then we say it *carries* the arc and its target nodes; simliarly, If m is the port of a source arc then it carries the arc and its source nodes.

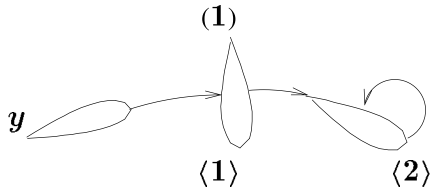
6.2 Actions and π-nets We now define π-nets, and show how to represent the actions of PIC as π-nets.

Suppose that Z is the set of names free in an action $a : h \to k$ of PIC. Then the π-net representing a consists of a certain bare π-net together with

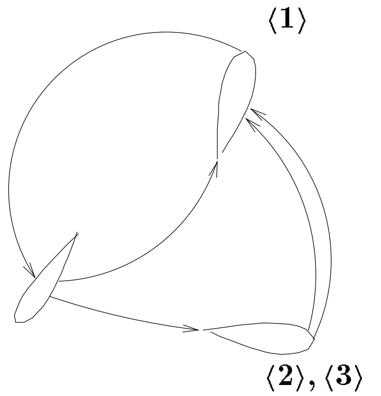
- an injective map from $Z \cup \{1, \ldots, h\}$ into the non-target nodes;

- a map (not necessarily injective) from $\{1, \ldots, k\}$ into the nodes.

Here then are two π-nets, obtained by equipping our two admissible bare π-nets with appropriate maps. The first map is indicated by *tail-labels* such as $\boldsymbol{y}$ or $(\mathbf{1})$, the second by *head-labels* such as $\langle \mathbf{1} \rangle$. Note that our definition ensures that a node may not bear a tail-label if it is a target of an arc.



$$\boldsymbol{a} : \mathbf{1} \to \mathbf{2} \;=\; (\boldsymbol{x})[\, \overline{\boldsymbol{x}}\langle \boldsymbol{y} \rangle \; \boldsymbol{x}(\boldsymbol{z}) \; \overline{\boldsymbol{z}}\langle \boldsymbol{z} \rangle \,]\langle \boldsymbol{x}\boldsymbol{z} \rangle$$



$$\boldsymbol{b} : \mathbf{0} \to \mathbf{3} \;=\; [\, \boldsymbol{\nu x} \; \boldsymbol{x}(\boldsymbol{y}) \; \overline{\boldsymbol{x}}\langle \boldsymbol{y} \rangle \; \boldsymbol{x}(\boldsymbol{z}) \; \overline{\boldsymbol{y}}\langle \boldsymbol{z} \rangle \; \overline{\boldsymbol{y}}\langle \boldsymbol{z} \rangle \,]\langle \boldsymbol{y}\boldsymbol{z}\boldsymbol{z} \rangle$$

For each of these π-nets we have shown the PIC action which it stands for. To translate an arbitrary PIC action a to a π-net is quite simple. Let $a = (\vec{x})S\langle \vec{y} \rangle : h \to k$. Then:

- create a node $\boldsymbol{z}$ (i.e. having this tail-label) for each name z occurring free or bound in a;

- for $1 \le j \le k$ give node $\boldsymbol{y}_j$ the head-label $\langle \boldsymbol{j} \rangle$;

- for each particle $\overline{u}\langle \vec{v} \rangle$ in S draw a source arc with source nodes $\vec{\boldsymbol{v}}$ and port node $\boldsymbol{u}$;

- for each particle $u(\vec{v})$ in S draw a target arc with port node $\boldsymbol{u}$ and target nodes $\vec{\boldsymbol{v}}$, removing the latter tail-labels;

- for each particle $\boldsymbol{\nu} u$ in S remove the tail-label of node $\boldsymbol{u}$;

- for $1 \leq i \leq h$, replace the tail-label of node $\boldsymbol{x}_i$ by $\langle \boldsymbol{i} \rangle$.

Call the result *draw*(a). It is easy to check that *draw*(a) is a π-net on the names free in a. In particular it has no target-arc cycles because a contains no cycle of input particles each binding the next.

It is also easy to define an inverse translation

$$\textit{write} : \pi\text{-nets} \rightarrow \mathsf{PIC};$$

to form *write*(N), first introduce a new name z for every unnamed node in N; then construct the particles – one i/o particle for each arc, and one restriction particle for each non-target node with no tail-label. We omit the remaining details, since they are obvious. The absence of target-arc cycles ensures that the particles in *write*(N) can be written in an appropriate linear order.

Now it is easy to show that *write*$($*draw*$(a)) = a$. It is not quite true in general that *draw*$($*write*$(N)) = N$. But the only discrepancy is when N contains a node which has a name $\boldsymbol{z}$ (say) as tail-label, but bears no arcs and no head-label. Let us call such a node *vacuous*. A vacuous node has no significance; it affects neither the algebraic operations nor the dynamics (both defined below). So we shall not distinguish two π-nets differing only in their vacuous nodes. Then indeed

$$\mathsf{PIC} \quad \underset{\overleftarrow{\textit{write}}}{\overset{\textit{draw}}{\longrightarrow}} \quad \pi\text{-nets}$$

is a bijection.

In what follows we use m, n, p to stand for nodes, and try to avoid using these letters for natural numbers.

6.3 Operations and dynamics The operations of composition, product and abstraction on π-nets are simply described. The first two involve the *coalescing* of two nodes; this means forming a single node bearing the arcs and labels of both the given nodes. Whenever this is done, it is ensured that the resulting π-net is still legal.

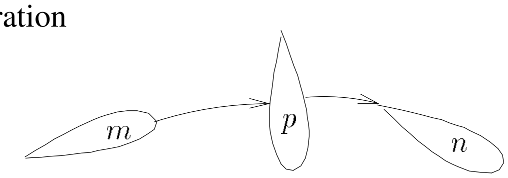
Let $M_1 : h_1 \rightarrow k_1$ and $M_2 : h_2 \rightarrow k_2$ be π-nets.

- To form $M_1 \cdot M_2$, for each i ($1 \leq i \leq k_1 = h_2$) coalesce the node in M_1 whose head-label is $\langle \boldsymbol{i} \rangle$ with the node in M_2 whose tail-label is $(\boldsymbol{i})$, and remove these labels. Also coalesce nodes in M_1 and in M_2 which share a name $\boldsymbol{z}$ as tail-label.

- To form $M_1 \otimes M_2$, first increment all tail-labels $(\boldsymbol{i})$ and head-labels $\langle \boldsymbol{j} \rangle$ in M_2 by h_1 and k_1 respectively. Also, as above, coalesce any pair of nodes with the same tail-label $\boldsymbol{z}$.

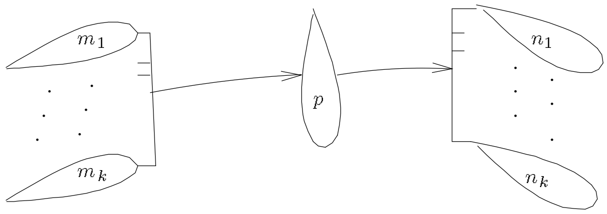
- To form $\mathbf{ab}_x M_1$, first increment all tail-labels $(\boldsymbol{i})$ and head-labels $\langle \boldsymbol{j} \rangle$ in M_1 by 1. Then assign head-label $\langle \mathbf{1} \rangle$ to the node with tail-label $\boldsymbol{x}$, and replace its tail-label by $(\mathbf{1})$.

Finally we define reaction. Consider the monadic case first. A *redex* of a π-net M consists of a configuration



where the nodes m and n are distinct, and there is no path of target arcs from n to m. Then there is a one-step reduction $M \searrow^1 M'$, where M' is derived from M just by removing the two arcs shown and coalescing m with n. We call m, n and p respectively the *source*, the *target* and the *port* of the redex.

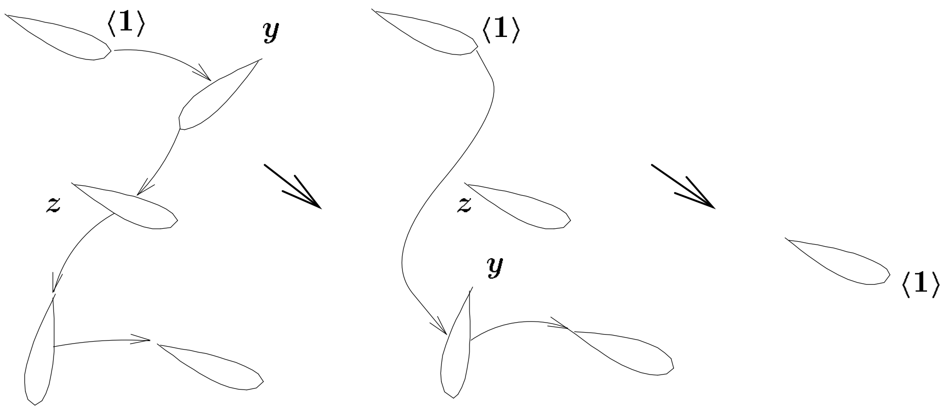
The polyadic case is a simple generalization. A redex is of the following form, where the source nodes are distinct from the target nodes and there is no path of target arcs from any n_j to any m_i:



The reduction again consists of removing the two arcs and coalescing m_i with n_i, for each i. (Note that reduction is not vacuous when $k = 0$, since the arcs are removed.)

Having thus defined redexes and the one-step reaction relation, we take reaction $\searrow$ over π-nets to be the transitive reflexive closure of $\searrow^1$. It is worth noting that reaction does not depend at all upon labelling; in effect we have defined it for bare π-nets.

Here is an illustration of a two-step reaction:



$$[\,\nu x\ \overline{y}\langle x\rangle\ \overline{z}\langle y\rangle\ z(u)\ u(v)\,]\langle x\rangle \;\searrow\; [\,\nu x\ \overline{y}\langle x\rangle\ y(v)\,]\langle x\rangle \;\searrow\; [\,\nu x\,]\langle x\rangle$$

This completes the definition of π-nets as an action structure. It is now a routine matter to check that the bijection consisting of the two translations *draw* and *write* is actually an isomorphism of action structures; that is, it preserves both the operations and the dynamics in both directions. Thus π-nets are an alternative presentation of PIC.

Recall that the effect structure of PIC consists of all actions of the form $(\vec{x})[\,\boldsymbol{\nu}\vec{y}\,]\langle\vec{z}\rangle$ where $\vec{y} \cup \vec{z} \subseteq \vec{x}$. In terms of π-nets, these are just arc-free nets where every node has a head-label.

6.4 Reflexive π-nets Having seen that the action structure of π-nets is isomorphic to PIC, one naturally asks what is gained by this new presentation. One answer is that it gives a less formal, more geometric, view of reaction. We may compare it with the geometry of interaction of Lafont [6], following Girard [5]; it differs notably in that reaction in PIC is not a confluent reduction relation. Another answer is that the geometric view may suggest generalizations or variations of PIC. We now examine one generalization.

Three related constraints have been placed upon π-nets:

C1 A π-net must have no target-arc cycles.

C2 In a one-step reaction, the source nodes must be distinct from target nodes.

C3 In such a reaction there must be no target-arc path from a target to a source.

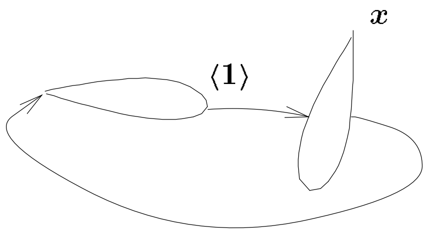
Condition C1 arises from the binding convention in PIC. C2 may also be considered to arise from binding, for it corresponds to disallowing

$$x(\vec{y})\,\overline{x}\langle\vec{z}\rangle$$

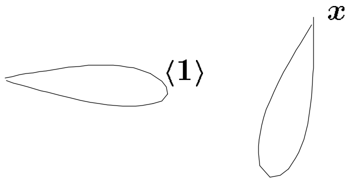
as a redex if $\vec{y}$ and $\vec{z}$ have any name in common. In reducing a redex such as $x(y)\,\overline{x}\langle y\rangle$ apparently the two particles would simply to cancel each other, causing the name y – previously bound – to become free; it is quite unclear what this could mean. Finally, condition C3 prevents a reaction from creating a target-arc cycle and thus violating C1.

We now define *reflexive* π-nets to be a more general class of nets upon which condition C1 is not imposed. Operations upon reflexive π-nets are defined exactly as before, and so is the reaction relation except that conditions C2 and C3 are no longer imposed. It is routine to check that this gives us another action structure, of which the (non-reflexive) π-nets form a sub-actionstructure. We shall shortly give it a syntactic formulation, and it will be called RPIC. But first let us focus on its most interesting aspect which is absent from PIC.

Consider the following (non-reflexive) π-net:



It is the action $[\,x(y)\,\overline{x}\langle y\rangle\,]\langle y\rangle$, and in PIC it has no reaction; but under the reaction relation which we have defined over reflexive π-nets it reduces to



The port node x has become vacuous, so the result is the action $[\,\boldsymbol{\nu}y\,]\langle y\rangle$, or just $\boldsymbol{\nu}$. So apparently in RPIC, just as the reaction of $x(y)\,\overline{x}\langle z\rangle$ produces the substitution $\{{}^{z}\!/y\}$ when $z \neq y$, so the reaction of $x(y)\,\overline{x}\langle y\rangle$ produces the restriction $\boldsymbol{\nu}y$. This reaction is geometrically sensible, since it arises simply from removing a constraint previously placed upon the reaction rule; but its implication is remarkable, for it suggests that restriction is formally a kind of *self*-substitution (but a kind which is unfamiliar). The substitution $\{{}^{z}\!/y\}$ means that y should be taken to denote z; so if we follow the geometry's suggestion then the self-substitution $\{{}^{y}\!/y\}$ should mean that y denotes itself, i.e. *denotes nothing else*. Though fanciful, this aligns with the intuition that a restricted name is indeed distinct from every other name. It explains why we have used the term *reflexive* for the wider class of nets.

In the next section we set out to make self-substitution not only geometrically sensible but also formally sensible.

7 Reflexive π-calculus

In this section we put together two strands of the development so far. In Section 5 we studied incident sets for PIC_τ and PIC; in the latter case we found an incident set for the monadic sub-actionstructure of PIC, but met a difficulty with the full structure. In Section 6 the graphical characterisation of PIC suggested a richer action structure, reflexive PIC or RPIC, which does not arise from the formal presentation.

Here we present RPIC formally, a surprisingly easy task once we establish the idea of reflexive substitution. Then we find a natural and elegant incident set for RPIC.

7.1 The action structure RPIC: statics An action of RPIC takes the form

$$(\vec{x})M\langle\vec{y}\rangle,$$

where the body M is a *multiset* of particles, in contrast to PIC where the body is a *partial sequence* of particles. (The particles are unchanged.) Associated with this difference is a difference of binding convention. In RPIC, as in PIC, there is just one binding occurrence of each bound name x in an action a; but the scope of x is now the whole of a, whereas in PIC it extends only to the right of the binding occurrence.

The effect is to admit cyclic binding dependency among particles. Thus the actions

$$[\,x(y)\;y(z)\;z(x)\,]\;,\;\;[\,x(x)\,]$$

are admissible in RPIC; neither of them has any free names.

At present we seek no direct significance for these cyclic dependencies; they should be treated for now as a formal extension of PIC which allows simple technical development. For one thing, multisets are easier to handle that partial sequences; for another, there will be (as we said) a natural incident set – and this arises because RPIC does not suffer from an apparent anomaly created by the binding rules of PIC.

The operations of composition, product and abstraction are defined as for PIC, but with multiset union in place of the concatenation of partial sequences. If $a = (\vec{u})M\langle\vec{v}\rangle$ and $b = (\vec{x})N\langle\vec{y}\rangle$ then

$$\begin{aligned} a \cdot b &\stackrel{\text{def}}{=} (\vec{u})\, M \cup \sigma N\, \langle \sigma\vec{y}\rangle \qquad (\sigma = \{\vec{v}/\vec{x}\}) \\ a \otimes b &\stackrel{\text{def}}{=} (\vec{u}\vec{x})\, M \cup N\, \langle \vec{v}\vec{y}\rangle \\ \mathbf{ab}_x a &\stackrel{\text{def}}{=} (x\vec{u})\, M\, \langle x\vec{v}\rangle. \end{aligned}$$

7.2 The action structure RPIC: dynamics In RPIC any complementary pair of particles

$$\overline{u}\langle\vec{v}\rangle\; u(\vec{w})$$

in the body of an action a is a redex of a; the contrast with PIC is that members of $\vec{v}$ may be among the bound names $\vec{w}$. By analogy with PIC we allow the one-step reduction

$$a \searrow^1 \{\vec{v}/\vec{w}\} * \widehat{a},$$

where $*$ represents the action of *reflexive substitution* (of $\vec{v}$ for $\vec{w}$) and $\widehat{a}$ stands for a with the redex deleted. Informally, reflexive substitution performs literal replacement of names, but introduces a restriction particle whenever a name is required (via a cycle) to "replace itself".

In defining the operation $\{\vec{y}/\vec{x}\} * a$ we assume always that the names $\vec{x}$ are distinct and not bound in a. But $\vec{y}$ may include members of $\vec{x}$ and even bound names of a; reflexive substitution operates upon a fixed presentation of an action, with no alpha-conversion.

First, define ${}^{y/x}(-)$ to mean the literal replacement of the name y for the name x; for example ${}^{y/x}((y)\langle x\rangle) = (y)\langle y\rangle$. Then reflexive substitution is given recursively by

$$\begin{aligned} \{\,\} * a &\stackrel{\text{def}}{=} a \\ \{z/w, \vec{y}/\vec{x}\} * a &\stackrel{\text{def}}{=} \begin{cases} \{{}^{z/w}\vec{y}/\vec{x}\} * {}^{z/w}a & (z \neq w) \\ \{\vec{y}/\vec{x}\} * (\boldsymbol{\nu} w)a & (z = w) \end{cases} \end{aligned}$$

To ensure that this is a proper definition, it is important to check that it yields the same result (up to alpha-conversion) no matter in which order the elements of a substitution $\{\vec{y}/\vec{x}\}$ are applied. This can be proved by induction on the size of the substitution. The reader may check, for example, that

$$\{y/x, x/y\} * [\,\overline{y}\langle z\rangle\,]\langle x\rangle \;=\; \{y/y\} * [\,\overline{y}\langle z\rangle\,]\langle y\rangle \;=\; [\,\boldsymbol{\nu} y\; \overline{y}\langle z\rangle\,]\langle y\rangle,$$

and that if we reorder the substitution elements we get

$$\{x/y, y/x\} * [\,\overline{y}\langle z\rangle\,]\langle x\rangle \;=\; \{x/x\} * [\,\overline{x}\langle z\rangle\,]\langle x\rangle \;=\; [\,\boldsymbol{\nu} x\; \overline{x}\langle z\rangle\,]\langle x\rangle$$

which is the same up to alpha-conversion.

7.3 Discussion It is a routine matter now to verify that the action structure of reflexive π-nets and the action structure RPIC are isomorphic. The isomorphism is just as described in Paragraph 6.2 for the non-reflexive case. It remains to be seen, by experimenting with applications, whether modelling with RPIC is significantly different from PIC in cases where the richer constructions of RPIC as not needed. Indeed, it is not even clear when and where this extra richness will be useful in practice. What then are the advantages of RPIC?

First, though its significance is not clear, we have found a new characterization of restriction; the reduction

$$[\, u(x)\, \overline{u}\langle x\rangle \,]\langle x\rangle \searrow \boldsymbol{\nu}$$

shows it arising from an interaction, rather than as an independent notion.

Second, the theory becomes easier in at least one important matter. For not only do we now find a natural incident set where we failed for PIC, but we also find this easier to verify than in the case of monadic PIC because we can use graph-theoretic methods without being hampered by awkward constraints.

We now set about this. First, we rephrase the notion of reachability (Paragraph 5.6) in graphical terms. The *reachable* nodes and arcs of a reflexive π-net are the least set such that

- every node with a target-label is reachable;
- every node and arc carried by a reachable node is reachable.

(This is a special case of a more general definition given later.) Further, a reflexive π-net is *reachable* if all its arcs are reachable.

The remainder of this section is devoted to a proof of the following

7.4 Theorem The set K^{RPIC} of reachable reflexive π-nets is an incident set for RPIC. ■

We need some definitions and lemmas to prove this result. First we define a notion of residual of a redex.

7.5 Redexes and residuals A *redex* of $a \in \mathsf{RPIC}$ consists of a source arc and a target arc with a common port, and their associated source and target nodes. This is just as in Paragraph 6.3 for π-nets, but without constraints. Two redexes are *disjoint* if their source arcs differ and their target arcs differ.

We use r, s, t to stand for redexes. The one-step reaction or *firing* corresponding to a redex r is written

$$a \overset{r}{\searrow} a^r,$$

where the *reductum* a^r is gained from a by removing the source and target arcs of r and coalescing each source node with the corresponding target node.

Given a redex r of a, each node n in a corresponds in an obvious way to a node n^r in a^r called its *residual* (under r). Thus if m and n are a corresponding source and target of the redex r we have $m^r{=}n^r$, but otherwise distinct nodes have distinct residuals. Similarly each arc π in a except the source and target arcs of r has a residual arc π^r in a^r; distinct arcs always have distinct residuals. A collection of arcs and nodes in a, such as a redex, also has a residual under r (the collection of residuals of its members). The following is easy to prove in RPIC (but is false in PIC, as shown in Paragraph 5.8):

Lemma A If s is a redex disjoint from r then its residual s^r is a redex of a^r. Moreover the two reductions commute, i.e. $(a^r)^{s^r} = (a^s)^{r^s}$. This result is shown as b in the diagram:

$$\begin{tikzcd} & a \arrow[dl, "r"'] \arrow[dr, "s"] & \\ a^r \arrow[dr, "s^r"'] & & a^s \arrow[dl, "r^s"] \\ & b & \end{tikzcd}$$

■

7.6 Reachability Let P be a subset of the nodes of an action a. The set of P-*reachable* nodes of a is the smallest set which includes P and all target-labelled (i.e. non-restricted) nodes, and such that if m is P-reachable and m carries n then n is P-reachable.

When $P = \{p\}$ we say p-*reachable*, and also talk of *reachable from* P or p, or (when P is empty) just *reachable*. It is clear that m is P-reachable iff it is reachable from some $p \in P$. We say an arc is P-*reachable* in a if it is carried by a P-reachable node. A node or arc which is not P-reachable is P-*unreachable*.

Intuitively, a reachable (i.e. $\emptyset$-reachable) arc is a particle which can eventually react with its environment without a performing any internal reactions. The number of external reactions needed to "reach" it is equal to the length of a path of (reachable) arcs leading to it from a target-labelled node.

We say a redex is P-*reachable* in a if its port is P-reachable.

There is a delicate interplay between reachable and unreachable redexes and nodes. If n is unreachable, then the firing of an *unreachable* redex can make n reachable; for the reduction may coalesce an unreachable node with a reachable one. In contrast, however, the firing of a *reachable* redex cannot improve reachability:

Lemma B Let r be a P-reachable redex of a, and n be P-unreachable. Then n^r is P^r-unreachable in a^r.

Proof Prove the contrapositive; suppose n^r is P^r-reachable in a^r, and construct a path reaching n from P in a. ■

For our present purposes we also need the idea of a set of nodes which can only be reached from each other. We say that a set N of nodes in a is *isolated* if whenever $n \in N$ is P-reachable then $P \cap N \neq \emptyset$. In other words, every $n \in N$ is unreachable, and if it is p-reachable then $p \in N$.

As an example, for any P and a the set of P-unreachable nodes is isolated in a.

Let us say that r is an N-*redex* if its port is in N, and π is an N-*arc* if it is carried by a member of N. The importance of isolation lies in the following:

Lemma C Let $a \searrow \ell$ where ℓ is a reachable action. Suppose that N is isolated in a, and that there is at least one N-arc. Then there is an N-redex r such that

$$a \overset{r}{\searrow} a^r \searrow \ell.$$

Proof Since N is isolated, the given N-arc is unreachable. So a is not reachable, and the reduction must be of length at least 1. The proof is by induction on the length of the reduction

$$a \overset{s}{\searrow} b \searrow \ell.$$

If the first redex s is an N-redex we are done. If not then it is P-reachable, where P is the complement of N in the node-set of a. But every node of N is P-unreachable in a, so by Lemma B every node of N^s is P^s-unreachable in b. In other words N^s is isolated in b; also there exists an N^s-arc. Hence by induction we have

$$a \overset{s}{\searrow} b \overset{t}{\searrow} c \searrow \ell,$$

where t is an N^s-redex in b. But s coalesces no nodes of N, so t is the residual r^s of an N-redex r in a. But r and s are disjoint redexes of a, so by Lemma A we therefore have also

$$a \overset{r}{\searrow} a^r \overset{s^r}{\searrow} c \searrow \ell$$

as required. ■

7.7 Preactions We now need to establish some properties of arbitrary preactions. In each case it will be enough to verify them for the four basic forms of preaction give Paragraph 5.2.

Let φ be any preaction and a any action. For each node n of a we define its *image* a^φ in $\varphi(a)$ in the obvious way. Note that if φ is a precomposition then distinct nodes of a may have the same image. Similarly each arc π of a has an image π^φ; distinct arcs always have distinct images.

Any collection of arcs and nodes in a has an image in $\varphi(a)$. In particular

Lemma D Let r be a redex in a. Then its image r^φ is a redex in $\varphi(a)$, and $\varphi(a) \overset{r^\varphi}{\searrow} \varphi(a^r)$. ■

This property happens to hold even for *post*composition, not just for preactions, though we shall not need that fact. The next property is more crucial, because it holds for preactions but not for postcomposition:

Lemma E Let N be isolated in a. Then N^φ is isolated in $\varphi(a)$. Further, if s is an N^φ-redex in $\varphi(a)$ then s is the image of an N-redex in a. ■

The proof of this is straightforward for the four basic preaction forms, and hence by an inductive argument for all preactions. Notice that it fails if φ is a postcomposition. Take for example $a = [\,\boldsymbol{\nu} x\ \overline{x}\langle y\rangle\,]\langle x\rangle$, and $\varphi(-) = (-)\cdot b$ where $b = (x)[\,\overline{z}\langle x\rangle\ x(w)\,]$. Now $\{x\}$ is isolated in a, but it is not isolated in

$$\varphi(a) \;=\; [\,\boldsymbol{\nu} x\ \overline{x}\langle y\rangle\ \overline{z}\langle x\rangle\ x(w)\,],$$

because z is free and carries x. Furthermore, $\varphi(a)$ has an x-redex which is the image of no redex in a. The postcomposition of b allows it to "gain access" to the isolated nodes (names) of a.

At last we are ready to prove the main theorem.

7.8 Proof of Theorem 7.4 It is easy to verify that K^{RPIC} satisfies the first clause of Definition 5.3 (Incident sets). For the second clause, we assume that $\varphi(a) \searrow \ell \in K^{\mathsf{RPIC}}$, and must find $k \in K^{\mathsf{RPIC}}$ for which $a \searrow k$ and $\varphi(k) \searrow \ell$.

We proceed by induction on the size (number of arcs) of a. If $a \in K^{\mathsf{RPIC}}$ we are done. If not then a has an unreachable arc π. Now let N be the unreachable nodes in a, so that π is an N-arc. Since N is isolated in a, from Lemma E we deduce that N^{φ} is isolated in $\varphi(a)$, and π^{φ} is an N^{φ}-arc.

By Lemma C there is an N^{φ}-redex s for which

$$\varphi(a) \overset{s}{\searrow} \varphi(a)^{s} \searrow \ell.$$

By Lemma E, $s = r^{\varphi}$ where r is an N-redex in a. So let

$$a \overset{r}{\searrow} a';$$

then by Lemma B again $\varphi(a)^{s} = \varphi(a')$, so

$$\varphi(a') \searrow \ell.$$

Hence by induction there exists $k \in K^{\mathsf{RPIC}}$ such that

$$a' \searrow k \ \text{ and } \ \varphi(k) \searrow \ell,$$

and we are done since $a \overset{r}{\searrow} a'$. ■

References

[1] Austry, D. and Boudol, G., *Algèbre de processus et synchronisation*, Journal of Theor. Comp. Science, Vol 30, pp90–131, 1984.

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

[3] Brookes, S.D., Hoare, C.A.R. and Roscoe, A.W., *A theory of communicating sequential processes*, Journal of ACM, Vol 31, pp560–599, 1984.

[4] Freyd, P.J. and Kelly, G.M., *Categories of continuous functors, I*, Journal of Pure and Applied Algebra, Vol 2, pp169–191, 1972.

[5] Girard, J.-Y., *Linear logic*, J. Theoretical Computer Science, Vol 50, pp1–102, 1987.

[6] Lafont, Y., *Interaction nets*, Proc. 17th ACM Symposium on Principles of Programming Languages, 1990.

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

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

[9] Milner, R., *The polyadic π-calculus: a tutorial*, Research Report LFCS–91–180, Laboratory for Foundations of Computer Science, Computer Science Department, Edinburgh University, 1991. (To appear in the Proceedings of the International Summer School on Logic and Algebra of Specification, Marktoberdorf, August 1991.)

[10] Milner, R., *Calculi for synchrony and asynchrony*, Journal of Theoretical Computer Science, Vol.25, pp 267–310, 1983.

[11] 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.

[12] Nielsen, M., Plotkin, G. and Winskel, G., *Petri nets, event structures and domains*, Journal of Theoretical Computer Science, Vol 13, pp85–108, 1981.

[13] Reisig, W., **Petri Nets**, EATCS Monographs on Theoretical Computer Science, ed. W.Brauer, G.Rozenberg, A.Salomaa, Springer Verlag, 1983.