Action Structures

Robin Milner

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

December 1992

Abstract

Action structures are proposed as a variety of algebra to underlie concrete models of concurrency and interaction. An action structure is equipped with *composition* and *product* of actions, together with two other ingredients: an indexed family of *abstractors* to allow parametrisation of actions, and a *reaction* relation to represent activity. The eight axioms of an action structure make it an enriched strict monoidal category; however, the work is presented algebraically rather than in category theory.

The notion of action structure is developed mathematically, and examples are studied ranging from the evaluation of expressions to the statics and dynamics of Petri nets. For algebraic process calculi in particular, it is shown how they may be defined by a uniform superposition of process structure upon an action structure specific to each calculus. This allows a common treatment of bisimulation congruence.

The theory of action structures emphasizes the notion of *effect*; that is, the effect which any interaction among processes exerts upon its participants. Effects are degenerate actions, and constitute a sub-actionstructure with special properties which support the general treatment of bisimulation.

Other current work on action structures is outlined, in particular their use for the π-calculus. Challenges are posed for the algebraic theory, including the study of combinations of action structures. Action structures are briefly compared with some other general models.

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

1 Introduction

The purpose of this paper is to find a mathematical structure which can underlie concrete models of concurrent computation, but which is free of the ad hoc details which are often present in such models.

A classic example of such a structure for sequential computation is Scott's notion of domain [10]. Its essence is to understand complete computations by examining the partial computations which approximate them. Indeed domains also help us to understand concurrent computations in this way, as shown by Nielsen, Plotkin and Winskel [17] in an important paper which relates domains both to event structures and to Petri nets [19].

Event structures and Petri nets are good examples of mathematical structures specific to concurrency. But we appear still to lack a canonical structure which is *combinational*, i.e. which explains how processes are synthesized, and which embodies the dynamics of *interaction* among processes. There are algebraic process calculi [2, 11, 15] and algebraic theories of Petri nets [6, 7]; these models have theoretical content which emphasizes combination and interaction. But what they have in common has not been distilled, so that as a family –and even individually– they can appear ad hoc.

It is therefore reasonable to look for a canonical algebraic structure for concurrency, with interaction at its heart; in this paper we propose *action structures* as a candidate. An action structure is an algebra which allows complex actions to be built from simple ones; in addition it has a dynamic component which defines the possibly complex performance of an action.

Central to action structures are two concerns. The first is that an action may be simple in one view, complex in another. Here are two examples, to suggest that this is consistent with common usage of the term "action". First, think of a sequential program accessing the fourteenth member of a linked list. At a low level this consists of many actions –chaining down the list– and this is how an implementer must treat it; at another level it is clearly a single action, an interaction between a program and the memory. As another example, suppose that you treat me to lunch because you want to borrow money from me. The lunch consists of many interactions, undoubtedly a complex pattern! Later, however, you and I will remember it as a single action: you stung me. This relativity of atomicity is well known to systems engineers; one only has to think of the difficult questions associated with transactions in a database. Studies of action refinement in process algebras, e.g. by van Glabbeek and Goltz [9] and Aceto and Hennessy [1], have begun to place relative atomicity on a theoretical footing; one purpose of action structures is to continue the task.

The second concern of action structures is with the *effect* which a shared action exerts upon its participants. Each participant suffers a different effect. In the case of the lunch you are richer, I am stung, the waiter is tired. [1] For a more formal example

[1]The waiter also took part, and illustrates another difficulty with concurrency. Unlike you and me, he will not remember our lunch as an atomic action; for him, it was inextricably interwoven with the serving of other lunches. This shows that the level of atomicity does not only vary from one global view of a system to another; it even varies between parts of the system.

consider the following transition in CCS [15]:

$$\overline{a}x \centerdot P \mid a(y) \centerdot Q \;\longrightarrow\; P \mid \{x/y\}Q,$$

where $\{x/y\}Q$ means the substitution of x for y in Q. In action structures this will, roughly speaking, be modelled by the *reaction*

$$\overline{a}x \times a(y) \searrow \mathbf{1} \times \{x/y\}.$$

In general, a reaction $\alpha \searrow \alpha'$ in an action structure represents part of the performance of an action α; α' may augment the performance by further reaction $\alpha' \searrow \alpha''$, and so on. In the example, a reaction of the product of two actions (itself an action) reduces it to a remainder action which is the product of $\mathbf{1}$ and $\{x/y\}$; these, and their product, are degenerate actions which we call *effects*. They are degenerate in that they perform no further reaction; instead they represent the effect exerted upon P and Q respectively by the performance of the original action, i.e. by the occurrence of a communication through the port a. Note that the unit action $\mathbf{1}$, the effect on P, represents *no* effect; it is only Q that feels an effect, which in this case is a substitution.

Strong motivation for action structures came from the π-calculus [16]. A distinguishing feature of the π-calculus is that port-names, like a in the above, can be transmitted as data; this can create another kind of effect, namely a new liaison between two agents. To illustrate, in the π-calculus we may adapt the previous example as follows:

$$((\boldsymbol{\nu}x)\overline{a}x \centerdot P) \mid a(y) \centerdot Q \;\longrightarrow\; (\boldsymbol{\nu}x)(P \mid \{x/y\}Q).$$

Here the restriction $(\boldsymbol{\nu}x)$ localises the port x to $\overline{a}x \centerdot P$; but an effect of the reaction is to widen the scope of x, so that it becomes a port which can carry information between P and Q. In action structures the corresponding reaction is now roughly speaking

$$((\boldsymbol{\nu}x) \circ \overline{a}x) \times a(y) \searrow (\boldsymbol{\nu}x) \circ (\mathbf{1} \times \{x/y\}),$$

and we see the newly created liaison as a component of the resulting effect.

The π-calculus also gave a second motivation. Through its treatment of names and restriction the π-calculus allows a complex action, such as accessing the n^{th} element of a list, to be programmed in terms of many simple actions; so one would like a version of the π-calculus in which such a complex action can be a single transition of a process. This would correspond to the original π-calculus in the way that SCCS [14], where a transition may involve arbitrarily many "particulate" interactions, corresponds to CCS where it may only involve a single one. This motivation is traced in Appendix A; it should be intelligible without detailed knowledge of the π-calculus, and it naturally leads to the three algebraic operations of an action structure. A main outcome of this paper is that process calculi like CCS, the π-calculus and their variants can be factored into two parts: at the lower level an action structure which is different for each of them, and at the upper level a process structure which is common to them all.

Outline In Section 2 we introduce the definition of action structures. We first present static action structures as an algebraic theory; then we add dynamics, a preorder preserved by the algebraic operations; finally we define a natural notion of homomorphism and substructure.

Section 3 gives several examples of action structures: functions, expression evaluation, process calculi and Petri nets. It also suggests how to treat the notion of *port*, or communication interface, uniformly in action structures. Action structures provide quite a general framework, within which these examples lie comfortably.

In Section 4, we treat the central notion of *effect*, the influence of the performance of an action on its participants. Effects take their place as a sub-actionstructure consisting of the *inert* actions – those which have nothing left to perform. Some of the examples of Section 3 are re-visited to see what the effect structure is in each case.

The next three sections are devoted to building process structure in a uniform way upon an action structure. Sections 5 and 6 deal respectively with the statics and with the dynamics of processes. The latter comes from two sources: from actions, especially the communications between processes, and from the process control operators.

Section 7 presents a non-trivial consequence of the theory: that the natural bisimulation equivalence over processes, presented uniformly with respect to the underlying action structure, is indeed a congruence relation. This crucially depends upon the mathematical properties of effects, and also of a new kind of actions known as *incidents*. An incident-set is a subset of the actions satisfying a simple property. There are many possible incident-sets, and there is a corresponding bisimilarity congruence for each; the larger the incident-set, the finer the congruence. The section ends with a proof that under a simple condition there is a smallest incident-set including a given set of actions.

Section 8 alludes to work by other authors which is related, or with which a relation should be sought, apart from papers already mentioned in the text. The section also describes other research in progress on action structures, and poses some future goals to test both their theory and their application.

Style Although action structures are amenable to category-theoretic treatment, I have chosen not to present them that way. But some of the presentation has been strengthened by categorical thinking with the help of Barry Jay and John Power, to whom I am grateful for useful insights. No doubt further mathematical insight will come from using categories; I hope that others will take up this challenge. One of my aims is to make action structures an example of a good interface through which mathematics can contribute to computer science and vice versa. Such an interface should embody some specific computational intuitions in a mathematical form which is widely accessible.

2 Action structures

An action structure is an algebra with some extra structure. It consists essentially of a strict monoidal category with two additions, one structural and one dynamic. The elements of an action structure are called *actions*; they are its morphisms, in categorical terms. The structural addition is an indexed family of functions over actions called *abstractors*; categorically they are endo-functors. Abstractors allow actions to be parametric upon certain constituents. The dynamic addition is called *reaction*; that is, a preorder over actions which represents activity.

We now give the algebraic formulation of action structures which we shall work with. For discussion of a more categorical and slightly more general formulation see Section 2.7.

2.1 Definition Let X be a set; in this context its members $x, y, \ldots$ will often be called *names*. Then a *static action structure* A over X consists of

- A monoid $(M, +, 0)$ of *arities*, and an assignment to each x of an arity in M. Categorically, M is the monoid of objects.
- For each $m, n \in M$ a set $A_{m,n}$ of *actions*. If $\alpha \in A_{m,n}$ we say that α has *arity* $m \to n$, and write $\alpha : m \to n$.
- Nullary operators (constants) $\mathbf{1}_m$, binary operators $\circ$ and $\times$, and a family $\{ab_x \mid x \in X\}$ of unary operators over A.

The operators are subject to the following rules of arity:

$$\mathbf{1}_m : m \to m \qquad \frac{\alpha : k \to n \qquad \beta : m \to p}{\alpha \times \beta : k{+}m \to n{+}p}$$

$$\frac{\alpha : k \to m \qquad \beta : m \to p}{\alpha \circ \beta : k \to p} \qquad \frac{x : k \qquad \alpha : m \to n}{\mathbf{ab}_x \alpha : k{+}m \to k{+}n}$$

and we freely omit the explicit arity m in $\mathbf{1}_m$, meaning that it can be supplied in any way which is consistent with the context. The operators satisfy eight laws:

C1. $\alpha \circ \mathbf{1} = \alpha = \mathbf{1} \circ \alpha$ C2. $\alpha \circ (\beta \circ \gamma) = (\alpha \circ \beta) \circ \gamma$

P1. $\alpha \times \mathbf{1}_0 = \alpha = \mathbf{1}_0 \times \alpha$ P2. $\alpha \times (\beta \times \gamma) = (\alpha \times \beta) \times \gamma$

PF1. $\mathbf{1} \times \mathbf{1} = \mathbf{1}$ PF2. $(\alpha \circ \beta) \times (\gamma \circ \delta) = (\alpha \times \gamma) \circ (\beta \times \delta)$

AF1. $\mathbf{ab}_x \mathbf{1} = \mathbf{1}$ AF2. $\mathbf{ab}_x(\alpha \circ \beta) = (\mathbf{ab}_x \alpha) \circ (\mathbf{ab}_x \beta)$.

■

Often the arities M will be just the natural numbers N under addition; then the arity of each name x will usually be 1. But we may instead wish the arity of a name to be a *sort*, in some set $\mathcal{S}$; then M will be the free monoid generated by $\mathcal{S}$. More complex arities also arise in constructions of action structures.

The laws not involving the abstractors, i.e. C, P and PF, are just those of a strict monoidal category. PF asserts that $\times$ is a (bi)functor. Similarly, AF asserts that each $\mathbf{ab}_x$ is a functor.

We now proceed to the dynamics.

2.2 Definition A (*dynamic*) *action structure* A over X consists of a static action structure together with a preorder $\searrow$ over each $A_{m,n}$ called *reaction*, such that

1. Each operator preserves reaction, e.g. $\alpha \searrow \alpha'$ implies $\alpha \circ \beta \searrow \alpha' \circ \beta$;

2. $\mathbf{1}$ is inactive, i.e. $\mathbf{1} \searrow \alpha$ implies $\alpha = \mathbf{1}$. ■

Reaction is often the transitive reflexive closure of an elementary relation $\overset{1}{\searrow}$ which may be called *atomic reaction*, representing the occurrence of an elementary event. An example is the interaction between complementary actions $a, \overline{a}$ in CCS; in Section 3.6 we shall see that $a \times \overline{a} \overset{1}{\searrow} \tau$.

One of the aims in studying action structures is how to regard the *composite* reactions in an action structure as the *elementary* steps of a higher-level computational structure. In Section 6 we show how to build familiar process calculi in a uniform way upon an action structure; then the transition relation, already familiar in process calculi, is definable in terms of the lower-level reaction relation.

Indeed, it is quite natural to expect that the higher-level structure will itself be an action structure. This will not be the case for the process calculus reconstruction of Section 6; we try there to keep as close to the original calculi as possible. But in a later paper we shall show how, in a natural reformulation, a process calculus can indeed be treated as an action structure; this means that a process itself is just a possibly complex action.

2.3 Terminology We shall usually consider dynamic, not static, action structures; so we usually omit the adjective "dynamic". In fact, we can think of a static action structure just as an action structure whose reaction is the identity relation.

When the arity monoid M and the name-set X are understood we shall just use A to denote either the action-set or the entire action structure. At other times we may write (M, X, A) and (N, Y, B) for two distinct action structures; we may write $(M, X, A, \searrow)$ if we want to emphasise reaction; we may also write M_A, X_A, $\searrow_A$ to denote the parts of a given action structure A. Such abuses, used with care, will save heavy formality.

2.4 Homomorphisms of action structures A notion of homomorphism between action structures almost determines itself – though there is a little freedom as to how to treat reaction. We adopt the following:

Definition A *homomorphism* $\Phi : A \to B$ between two action structures A and B consists of

- A monoid homomorphism $\Phi : M_A \to M_B$;
- A map $\Phi : X_A \to X_B$ such that $x : m$ implies $\Phi x : \Phi m$;
- A map $\Phi : A \to B$ of actions such that
  - $\alpha : m \to n$ implies $\Phi\alpha : \Phi m \to \Phi n$;

– Φ preserves all the operations $\mathbf{1}$, $\circ$, $\times$ and $\mathbf{ab}_x$;

– $\alpha \searrow_A \alpha'$ implies $\Phi\alpha \searrow_B \Phi\alpha'$.

Φ is an *isomorphism* if there is a homomorphism $\Phi^{-1} : B \to A$ such that both $\Phi \circ \Phi^{-1}$ and $\Phi^{-1} \circ \Phi$ are identity maps on arities, names and actions. ■

As we add further structure to the notion of action structure, we may also refine the notion of homomorphism.

2.5 Definition An action structure B is a *sub-actionstructure* of A if there is a homomorphism $\Phi : B \to A$ such that

1. Φ is injective on X_B, M_B and B;
2. For all $\beta, \beta' \in B$, $\beta \searrow_B \beta'$ iff $\Phi\beta \searrow_A \Phi\beta'$. ■

In defining substructure notions it is common to require the implication in clause 2 only from left to right, and to use the term *full* for a substructure obeying the double implication. But for the present we have no use for the weaker condition, so we adopt the stronger one to avoid tediously repeating the term "full".

2.6 Discussion The aspect of action structures which is least familiar is abstraction. The examples of Section 3 below will show that abstraction permits parametrisation of actions upon entities of some kind; but it also shows that these entities can be quite diverse in nature: ordinary values in one case, portnames in another, places and transitions in the case of Petri nets. So the notion of action structure presents, via abstraction and its axioms, something common to all these parametrisations.

But does it capture *all* that they have in common? This is really a vague question, since we clearly don't want to limit ourselves to the examples of Section 3. Certainly for those applications it will be seen that the names X stand for entities, and that the abstractors bind those names. In fact for each name x there is usually an action $\langle x \rangle$ such that $\langle x \rangle \circ \mathbf{ab}_y \alpha$ imposes the substitution $\{x/y\}$ upon α; to be precise, it represents $\langle x \rangle \times \{x/y\}\alpha$.

We might therefore expect notions such as *free name*, *substitution* and *alpha-conversion* to be captured by the formulation and axioms of action structures. But they are not, for several reasons. First, the present axioms of action structure lie easily in the mind because they have a simple and appealing pattern. Second, they are strong enough to deliver interesting properties of process calculi in a uniform way, which is a main purpose this paper. Third, if we are interested in strengthening the axioms then we should pay equal attention to all the other operations, especially product. For example, we may like to assume that product is commutative "up to isomorphism"; this appears to be at the same level of particularity as assuming alpha-convertibility for abstractors.

One can hope that the "plain" notion of action structure treated here will explain a wide range of phenomena. Naturally one also expects to define stronger notions; among other things, they may capture more clearly the name-binding quality of abstractors.

2.7 Categorical formulation Having discussed how the notion of action structure may be *strengthened*, we now look at the natural formulation in categorical terms, which turns out to be a little *weaker* (i.e. more general).

The axioms AF require the abstractors to be functors. But we have not allowed the functors $\mathbf{ab}_x$ to be quite arbitrary; the rule of arity for $\mathbf{ab}_x\alpha$ in 2.1 asserts that the operation of the functor $\mathbf{ab}_x$ upon objects is taken to be $k+(-)$, where k is fixed by x. Nothing in this paper depends on this relationship between the abstraction functors and the monoidal operation. Thus a more abstract and categorical definition of action structure would omit the ascription of an arity k to each name x, and simply declare the abstractors to be an arbitrary indexed set of functors. In our formulation this would imply a different arity rule for $\mathbf{ab}_x\alpha$, namely

$$\frac{\alpha : m \to n}{\mathbf{ab}_x\alpha : \mathbf{ab}_x m \to \mathbf{ab}_x n}$$

All the results of the present paper would hold with this generalisation. The more specific formulation of Definition 2.1 is an attempt at a comfortable compromise between mathematical generality and the specific interesting applications.

The same concern has guided my use of notation, which diverges in at least two respects from the standard in category theory. First, I have called the monoid of arities $(M, +, 0)$, not $(M, \times, 1)$, while using $\times$ as the monoidal operator upon actions. This partly obscures the fact that the monoidal structure is something enjoyed by objects and morphisms jointly, not separately. But in many important examples the arities are the natural numbers under addition.

Second, with some misgiving I have adopted the forward version of composition, so that for $\alpha : k \to m$ and $\beta : m \to p$ I write $\alpha \circ \beta$, not $\beta \circ \alpha$. It would be confusing to do otherwise in this work, since in Sections 5 and 6 we find that composition $\alpha \circ P$ is closely allied to the prefixing operation of $\alpha \bullet P$ in process algebra. An alternative which some prefer is to use ";" for forward composition, but the use of punctuation in an algebraic expression can be distracting.

3 Examples of action structures

We begin by fitting some widely-known ideas into the framework of action structures, and then proceed to show how these structures can accommodate less standard notions found in concurrency theory.

3.1 Functions Let D be some fixed set of values. Define $\mathsf{FUN} = (M, X, A, \searrow)$ to be the action structure in which

$$\begin{aligned} M &= (\mathsf{N}, +, 0) \\ X &= \emptyset \\ A_{m,n} &= D^m \to D^n \\ \searrow &= \text{the identity relation.} \end{aligned}$$

The operations over A are as follows, where we use "," to concatenate tuples and $\vec{v}$ for a tuple $v_1, \ldots, v_m$ of variables:

$$\begin{array}{rcll} \mathbf{1}_m & \stackrel{\text{def}}{=} & \lambda\vec{v} \,.\, \vec{v} & \\ f \circ g & \stackrel{\text{def}}{=} & \lambda\vec{v} \,.\, g(f\vec{v}) & \\ f \times g & \stackrel{\text{def}}{=} & \lambda\vec{u}, \vec{v} \,.\, (f\vec{u}, g\vec{v}) & \quad (\vec{u}, \vec{v} \text{ all distinct}). \end{array}$$

There are no abstractors, so we only have to verify the axioms of a monoidal category; this is simple and well-known.

3.2 Functions with abstractors Let X be a set, which we shall call the *global variables*, and let $\mathrm{Env} \stackrel{\text{def}}{=} X \to D$, the valuations of global variables in D. Define the action structure $\mathsf{FUN}[X] = (M, X, A, \searrow)$ as follows:

$$\begin{array}{rcl} M & = & (\mathsf{N}, +, 0) \\ X & = & \text{as given} \\ A_{m,n} & = & \mathrm{Env} \to (D^m \to D^n) \\ \searrow & = & \text{the identity relation.} \end{array}$$

The operations are nearly as simple as in FUN; we only have to distribute the parameter $\varrho \in \mathrm{Env}$ properly. Let us write $\varrho_{x \mapsto v}$ for the valuation which maps x to the value v, but otherwise assigns values as ϱ does. Then

$$\begin{array}{rcll} \mathbf{1}_m \varrho & \stackrel{\text{def}}{=} & \lambda\vec{v} \,.\, \vec{v} & \\ (f \circ g)\varrho & \stackrel{\text{def}}{=} & \lambda\vec{v} \,.\, g\varrho(f\varrho\vec{v}) & \\ (f \times g)\varrho & \stackrel{\text{def}}{=} & \lambda\vec{u}, \vec{v} \,.\, (f\varrho\vec{u}, g\varrho\vec{v}) & \quad (\vec{u}, \vec{v} \text{ all distinct}) \\ (\mathbf{ab}_x f)\varrho & \stackrel{\text{def}}{=} & \lambda u, \vec{v} \,.\, (u, f\varrho_{x \mapsto u}\vec{v}) & \quad (u, \vec{v} \text{ all distinct}). \end{array}$$

Thus $(\mathbf{ab}_x f)\varrho$ takes one more argument than $f\varrho$; besides pushing this argument (u) through as a result, it also treats it as the value of x when evaluating f. It is instructive to check the laws AF for abstractors (2.1).

Clearly FUN is isomorphic to $\mathsf{FUN}[\emptyset]$. More generally, whatever X is, there is an injective homomorphism from FUN to $\mathsf{FUN}[X]$; it sends every $f \in D^m \to D^n$ to the corresponding function $g \in \mathrm{Env} \to (D^m \to D^n)$ which is *valuation-insensitive*, i.e. $g\varrho\vec{v}$ does not depend upon ϱ. Thus FUN is a sub-actionstructure of $\mathsf{FUN}[X]$.

3.3 Evaluating expressions We have seen how functions constitute a static action structure. We now wish to consider how the expressions which denote functions constitute a *dynamic* action structure, which is given meaning by a homomorphism to $\mathsf{FUN}[X]$. It must be repeated that we are doing little more than casting familiar ideas in a new form. In effect we are presenting a dynamic variant of a Lawvere algebraic theory, with equations replaced by reactions; see Barr and Wells [3].

Let Exp_m be the class of expressions built from the global variables X and formal (or local) variables $y_1, \ldots, y_m$, by a fixed set of operators standing for operations over D. Thus Exp_m is the smallest set such that

- Each of $y_1, \ldots, y_m$ and each $x \in X$ is an expression in Exp_m;
- If $e_1, \ldots, e_k$ are expressions in Exp_m and $\underline{op}$ is a k-ary operator (standing for a function $op \in D^k \to D$), then $\underline{op}\langle e_1, \ldots, e_k\rangle$ is an expression in Exp_m.

An expression in Exp_m obviously denotes a function in $\mathrm{Env} \to (D^m \to D)$, i.e. an action with arity $m \to 1$ in $\mathsf{FUN}[X]$. Expressions and their evaluation can be treated as a dynamic action structure $\mathsf{EXP}[X] = (M, X, A, \searrow)$, where

$$\begin{array}{rcl} M & = & (\mathsf{N}, +, 0) \\ X & = & \text{as before} \\ A_{m,n} & = & \mathrm{Exp}_m^n \\ \searrow & & \text{defined below.} \end{array}$$

Now, for $e \in \mathrm{Exp}_m$, write $e[\vec{d}]$ to mean the result of replacing the m formal variables $\vec{y}$ simultaneously in e by the m expressions $\vec{d}$; write $\{\vec{c}/\vec{x}\}e$ to mean the result of replacing the k global variables $\vec{x}$ simultaneously in e by the k expressions $\vec{c}$. Each member of $A_{m,n}$ is an n-tuple of expressions, which we shall write as $\langle e_1, \ldots, e_n\rangle$. Then the operations in $\mathsf{EXP}[X]$ are

$$\begin{array}{rcl} \mathbf{1}_m & \stackrel{\text{def}}{=} & \langle y_1, \ldots, y_m\rangle \\ \langle\vec{d}\,\rangle \circ \langle\vec{e}\,\rangle & \stackrel{\text{def}}{=} & \langle\vec{e}\,[\vec{d}\,]\rangle \\ \langle\vec{d}\,\rangle \times \langle\vec{e}\,\rangle & \stackrel{\text{def}}{=} & \langle\vec{d}\,,\, \vec{e}\,[y_{k+1}, \ldots, y_{k+m}]\rangle \\ & & \qquad \text{where } \langle\vec{d}\,\rangle : k \to l,\ \langle\vec{e}\,\rangle : m \to n \\ \mathbf{ab}_x\langle\vec{e}\,\rangle & \stackrel{\text{def}}{=} & \langle y_1\,,\, \{y_1/x\}(\vec{e}\,[y_2, \ldots, y_{m+1}])\rangle. \end{array}$$

There is nothing surprising in this. Note only that the effect of $\mathbf{ab}_x$ is to transform x into the first formal variable, incrementing the indices of the others by 1.

Evaluation is what makes $\searrow_{\mathsf{EXP}}$ non-trivial. First, suppose that for each value $v \in D$ there is a nullary operator $\underline{v}$; we write $\underline{v}\langle\ \rangle$ as $\underline{v}$ and call it a *constant*. For $\alpha, \alpha' : m \to n$, define $\alpha \stackrel{1}{\searrow} \alpha'$ to mean that α' can be gained from α by replacing a single subexpression $\underline{op}\langle\underline{v_1}, \ldots, \underline{v_k}\rangle$ in α by the constant $\underline{op(v_1, \ldots, v_k)}$. Then define reaction to be the transitive reflexive closure of $\stackrel{1}{\searrow}$ so that $\searrow \stackrel{\text{def}}{=} (\stackrel{1}{\searrow})^*$.

Now of course we look for a natural homomorphism $\Phi : \mathsf{EXP}[X] \to \mathsf{FUN}[X]$ sending each expression to its denotation – a function. What is this denotation? For $e \in \mathrm{Exp}_0$ (an expression with no formal variables) it is given by $[\![e]\!]$, where

$$\begin{array}{rcl} [\![x]\!]\varrho & \stackrel{\text{def}}{=} & \varrho x \\ [\![\underline{op}\langle e_1, \ldots, e_k\rangle]\!]\varrho & \stackrel{\text{def}}{=} & op([\![e_1]\!]\varrho, \ldots, [\![e_k]\!]\varrho). \end{array}$$

Then indeed Φ can be defined as the identity upon the arities M and the names X, and upon actions by

$$\Phi(\langle e_1, \ldots, e_n \rangle)\varrho \stackrel{\text{def}}{=} \lambda\vec{v}\,.\,\langle [\![e_1[\underline{\vec{v}}]]\!]\varrho, \ldots, [\![e_n[\underline{\vec{v}}]]\!]\varrho\rangle.$$

It is quite routine to check the Φ is a homomorphism; in particular, that it preserves reaction, i.e.

$$\alpha \searrow_{\mathsf{EXP}[X]} \alpha' \text{ implies } \Phi\alpha = \Phi\alpha'$$

(recalling that $\searrow_{\mathsf{FUN}[X]}$ is the identity).

Action structures are not limited to such familiar things as functions and expressions, as our next few examples show. Moreover, even these familiar cases yield more interest when we come to *effective* action structures in the next section.

3.4 Synchronous CCS (1) Synchronous CCS [14], or SCCS, is a process calculus in which processes $P, Q, \ldots$ are built from actions $\alpha, \beta, \ldots$ by various constructions, among which are *prefixing* $\alpha \bullet P$ and *product* $P \times Q$. The dynamics of processes is given by *labelled transitions* of the form $P \xrightarrow{\alpha} P'$; these are defined by transition rules such as

$$\alpha \bullet P \xrightarrow{\alpha} P \qquad\qquad \frac{P \xrightarrow{\alpha} P' \qquad Q \xrightarrow{\beta} Q'}{P \times Q \xrightarrow{\alpha\times\beta} P' \times Q'}.$$

The actions of SCCS are just elements of the free Abelian group generated by a fixed set Z of (positive) *particles* $a, b, \ldots$. Thus an instance of the product rule is

$$\frac{P \xrightarrow{a\overline{b}c} P' \qquad Q \xrightarrow{b\overline{c}} Q'}{P \times Q \xrightarrow{a} P' \times Q'}$$

because $a\overline{b}c \times b\overline{c} = a$ is a product of actions.

Now it is computationally meaningful (and when we treat π-calculus as a generalisation of SCCS we shall find it *necessary*) to separate the dynamic ingredient of interaction, the cancellation of a particle by its inverse, from the algebraic structure of actions. This is made possible by the dynamic element of action structures.

Let $\overline{Z} \stackrel{\text{def}}{=} \{\overline{a} \mid a \in Z\}$, the negative particles. Ignoring abstractors for now, we get a simple action structure in which reaction represents for the first time the *interaction* between two actions; that is, unlike in previous examples, it is possible for $\alpha \times \beta$ to react when neither α nor β can react independently.

In fact we define the action structure $\mathsf{SCCS} = (M, X, A, \searrow)$, where

$$\begin{array}{rcl} M & = & 0 \\ X & = & \emptyset \\ A & = & \text{multisets over } Z \cup \overline{Z}, \end{array}$$

and $\searrow$ is the least preorder on A such that for all $\alpha \in A$ and $a \in Z$

$$\alpha \cup \{a, \overline{a}\} \searrow \alpha.$$

(Here $\cup$ means union of multisets.) Then the operations are just

$$\begin{aligned} \mathbf{1} &\stackrel{\text{def}}{=} \emptyset \\ \circ, \times &\stackrel{\text{def}}{=} \cup . \end{aligned}$$

One can easily show that SCCS is just the action structure with one object (arity) 0, freely generated by the elementary actions $Z \cup \overline{Z}$ and the reaction $a \times \overline{a} \searrow \mathbf{1}$ for each $a \in Z$.

Of course, this action structure only explains the *actions* of synchronous CCS, not its processes or process combinators. But some of the latter can indeed be explained via action structures, as we now show.

3.5 Synchronous CCS (2) By adding abstractors to the action structure SCCS, we can explain not only the actions of SCCS but also those of its operations which involve the binding of particle names. There are essentially three such operations:

1. Restriction: $P\backslash a$;
2. Renaming: $P[b/a]$;
3. Parametric definition: $F(a) \stackrel{\text{def}}{=} P$.

(The last was not originally in SCCS, but it is natural to include it.) Now we are not yet attempting to define processes over an arbitrary action structure; this will come in Section 5. But here we need to anticipate a little, just to observe that

- we shall allow abstraction of a particle name from a process, $\mathbf{ab}_a P$;
- we shall allow composition between an action and a process, $\alpha \circ P$;
- further, we shall decree that abstraction distributes over this composition, i.e. $\mathbf{ab}_a(\alpha \circ P) = (\mathbf{ab}_a \alpha) \circ (\mathbf{ab}_a P)$.

Note that the last item only makes sense if we can abstract names from actions, i.e. if we extend the action structure SCCS by adding abstractors. The first two items will allow us to write our three operations as follows:

1. Restriction: $\boldsymbol{\nu} \circ \mathbf{ab}_a P$;
2. Renaming: $\langle b \rangle \circ \mathbf{ab}_a P$;
3. Parametric definition: $F \stackrel{\text{def}}{=} \mathbf{ab}_a P$

provided we have in our action structure the special action $\boldsymbol{\nu}$, and for each particle b the action $\langle b \rangle$ which represents the particle as a datum.

To admit abstraction of particles, and these new actions, we now generalize SCCS to $\mathsf{SCCS}[X] = (M, X, A, \searrow)$, where

$$\begin{array}{rcl} M & \stackrel{\text{def}}{=} & (\mathsf{N}, +, 0) \\ X & = & \text{some set of (positive) particles, typically } Z, \end{array}$$

with actions A and reaction $\searrow$ defined as follows. Each action $\alpha : m \to n$ takes the form

$$(\vec{x})\,(\boldsymbol{\nu}\vec{y})\,S\,\langle\vec{z}\rangle \qquad (\,|\vec{x}| = m,\ |\vec{z}| = n\,),$$

where $\vec{x}$, $\vec{y}$ and $\vec{z}$ are names in X, and S is a multiset over $X \cup \overline{X}$. We say that the names $\vec{x}$, $\vec{y}$ and $\vec{z}$ are respectively *imported*, *restricted* and *exported*, the first two being binding operations over the whole action. We shall assume always that the names $\vec{x}, \vec{y}$ are all distinct, and we do not distinguish α from any variant gained by change of bound names or by permutation of restricted names.

We look at several simple cases of actions, before defining operations and reaction. Note that any of the four parts of an action may be empty, and we may then omit that part. Thus:

1. We write $(\,)\,(\,)\,\{x\}\,\langle\,\rangle$ and $(\,)\,(\,)\,\{\overline{x}\}\,\langle\,\rangle$ as x and $\overline{x}$ respectively; they have arity $0 \to 0$, and correspond to the two elementary actions of SCCS (of course, we expect $x \times \overline{x} \searrow \mathbf{1}_0$);

2. $\mathbf{1}_m = (\vec{x})\,\langle\vec{x}\rangle$, whose full form is $(\vec{x})\,(\,)\,\emptyset\,\langle\vec{x}\rangle$;

3. $\boldsymbol{\nu} : 0 \to 1 = (\boldsymbol{\nu}y)\,\langle y\rangle$, whose full form is $(\,)\,(\boldsymbol{\nu}y)\,\emptyset\,\langle y\rangle$;

4. $\langle x\rangle : 0 \to 1 = (\,)\,(\,)\,\emptyset\,\langle x\rangle$, the name x as a datum.

Note that the *datum* $\langle x\rangle$ in clause 4 is distinct from the *action* called x in clause 1.

The operations are as follows. Let

$$\begin{array}{rcll} \alpha & \equiv & (\vec{u})\,(\boldsymbol{\nu}\vec{v})\,R\,\langle\vec{w}\rangle & \qquad (\vec{u}, \vec{v} \text{ not occurring in } \beta) \\ \beta & \equiv & (\vec{x})\,(\boldsymbol{\nu}\vec{y})\,S\,\langle\vec{z}\rangle & \qquad (\vec{x}, \vec{y} \text{ not occurring in } \alpha). \end{array}$$

Then:

$$\begin{array}{rcll} \mathbf{1}_m & \stackrel{\text{def}}{=} & (\vec{x})\langle\vec{x}\rangle & \qquad (|\vec{x}| = m) \\ \alpha \circ \beta & \stackrel{\text{def}}{=} & (\vec{u})\,(\boldsymbol{\nu}\vec{v}, \vec{y})\,R \cup \{\vec{w}/\vec{x}\}S\,\langle\{\vec{w}/\vec{x}\}\vec{z}\rangle & \\ \alpha \times \beta & \stackrel{\text{def}}{=} & (\vec{u}, \vec{x})\,(\boldsymbol{\nu}\vec{v}, \vec{y})\,R \cup S\,\langle\vec{w}, \vec{z}\rangle & \\ \mathbf{ab}_x\alpha & \stackrel{\text{def}}{=} & (x, \vec{u})\,(\boldsymbol{\nu}\vec{v})\,R\,\langle x, \vec{w}\rangle & \qquad (x \text{ not in } \vec{u} \text{ or } \vec{v}). \end{array}$$

Thus we see that abstraction is a combination of import and export.

Finally, reaction is defined by $\searrow \stackrel{\text{def}}{=} (\searrow^1)^*$, where $\alpha \searrow^1 \alpha'$ means that α' can be derived from α by deleting a single pair of complementary particles from its multiset.

Recall that SCCS is freely generated by the actions $a, \overline{a}$ ($a \in Z$). It is easy to check that every action in $\mathsf{SCCS}[X]$ can be expressed in terms of the following:

$$\begin{array}{rcll} x, \overline{x} & : & 0 \to 0 & \text{– interaction } (x \in X) \\ \langle x \rangle & : & 0 \to 1 & \text{– datum } (x \in X) \\ \boldsymbol{\nu} & : & 0 \to 1 & \text{– new name} \\ \boldsymbol{\omega} & : & 1 \to 0 & \text{– old name.} \end{array}$$

These actions have all been already defined except

$$\boldsymbol{\omega} \stackrel{\text{def}}{=} (x)\langle\ \rangle$$

whose full form is $(x)(\ )\emptyset\langle\ \rangle$, which discards a name. (Think of $\boldsymbol{\nu}$ and $\boldsymbol{\omega}$ as "new" and "old", or "create" and "destroy", or "gain" and "lose".) But $\mathsf{SCCS}[X]$ is not *freely* generated by these actions; certain identities hold among them, e.g. $\langle x \rangle \circ \boldsymbol{\omega} = \mathbf{1}_0$. It should not be too hard to find a complete set of identities, so that $\mathsf{SCCS}[X]$ is indeed the action structure freely generated from the above actions divided by the equivalence induced by the identities.

This new presentation of the actions of SCCS does not directly help in applications. But it allows us to fit SCCS into the action structure framework; it also serves as a platform for a generalisation of the π-calculus to be presented in a later paper.

3.6 CCS There are two differences between pure CCS [15] and SCCS which affect the way we should treat them using action structures. The first is the obvious one that SCCS is synchronous, in the sense that in the product $P \times Q$ the processes P and Q must act in synchronisation, while in the parallel composition $P \mid Q$ of CCS either may delay. This is reflected in the first two transition rules for CCS parallel composition:

$$\frac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q} \qquad\qquad \frac{Q \xrightarrow{\alpha} Q'}{P \mid Q \xrightarrow{\alpha} P \mid Q'}.$$

The third transition rule for "|" is

$$\frac{P \xrightarrow{\alpha} P' \qquad Q \xrightarrow{\overline{\alpha}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'},$$

and this reflects the second difference from SCCS; that is, the atomic action τ of CCS represents the occurrence of just a single interaction between processes, while in SCCS the unit action 1 is used to represent the simultaneous occurrence of any number of such interactions. In a sense the latter is more natural, closer to reality, for one needs a central scheduler to implement the interleaving of single interactions in CCS. But CCS was designed as an applicable theory, rather than as a programming language; the expansion theorem of CCS is easy to use precisely because interactions are treated singly. Moreover in [14] it was demonstrated that at the semantic level of observation equivalence it makes no difference whether single interactions of CCS are treated as interleaved or as simultaneous.

If single interactions were allowed to be simultaneous, then the action structure for CCS could be just that for SCCS; the only difference would be that the concurrent composition of CCS is treated as a derived operation in terms of product and delay (see Section 5.1 below). But to treat single interactions as interleaved only requires two slight changes in the action structure $\mathsf{SCCS}[X]$. So we define $\mathsf{CCS}[X]$ as a slight variant of $\mathsf{SCCS}[X]$, as follows. First we adjoin the special particle τ to the particles $X \cup \overline{X}$, and in a typical action $\alpha = (\vec{x})\,(\boldsymbol{\nu}\vec{y})\,S\,\langle\vec{z}\rangle$ we take S to be a multiset of this larger class of particles. Second, we define $\alpha \searrow^1 \alpha'$ iff α' can be derived from α by replacing a single pair of complementary particles in S by τ (instead of deleting them). In Section 7, we shall see how this adaptation allows us to restrict attention to actions containing at most two particles.

3.7 Ports The constructions $\mathsf{SCCS}[Z]$ and $\mathsf{CCS}[Z]$ involved the complementary actions a and $\overline{a}$ ($a \in Z$), with the reaction rule

$$a \times \overline{a} \searrow \mathbf{1}_0 \text{ (or } \tau);$$

since the actions have arity $0 \to 0$ this can also be written

$$\overline{a} \circ a \searrow \mathbf{1}_0 \text{ (or } \tau).$$

But this only models the pure form of CCS, where a or $\overline{a}$ is simply an indivisible action. By contrast, in CCS with value-passing we think of a as a port through which data can flow. How does this fit into the framework of action structures? Well, Example 3.2 above –the action structure $\mathsf{FUN}[X]$ for functions– is about operations upon data; we can think of a datum of arity m (an m-tuple of values) being fed into a function of arity $m \to n$. Now, how should we represent the export of an m-ary datum through a port a, instead of into a function? A natural idea is to have, for each port name a and appropriate arity m, an action

$$\mathbf{out}_m a : m \to 0.$$

Equally naturally, for importing a datum through the port we can introduce an action

$$\mathbf{in}_m a : 0 \to m.$$

Now the actions a and $\overline{a}$ of SCCS and CCS can be regarded as the special case of $\mathbf{in}_m a$ and $\mathbf{out}_m a$ for which $m = 0$ since no data flows in those calculi. Therefore, generalising the reaction rule above, we assert the port reaction rule

$$\mathbf{out}_m a \circ \mathbf{in}_m a \searrow \mathbf{1}_m.$$

This can also be written

$$\mathbf{in}_m a \times \mathbf{out}_m a \searrow \mathbf{1}_m,$$

for it is a simple exercise to show that if $\alpha : m \to 0$ and $\beta : 0 \to n$ in a strict monoidal category, then $\alpha \circ \beta = \alpha \times \beta = \beta \times \alpha$. This rule exactly expresses synchronised communication at a port.

Based upon this idea we shall show in outline how to extend *any* action structure by adding ports through which its "data" may pass. The quotation marks are to remind us that some action structures may be abstract and not concerned with data; nonetheless our construction makes sense formally even in these cases.

In fact let $B = (M, X, B, \searrow)$ be any action structure, and let Z (disjoint from X) be the ports we wish to add. We shall now outline the construction of $B^{(Z)} = (M', X', B', \searrow')$, the action structure B freely extended by addition of the ports Z. First

$$\begin{array}{rcll} M' & = & M \times \mathsf{N} & \text{– Cartesian product of monoids} \\ X' & = & X \cup Z & \text{– a disjoint union.} \end{array}$$

Let $\Phi : X \to X'$ and $\Phi : M \to M'$ be the injections given by $\Phi x = x$ and $\Phi m = (m, 0)$. The arities in X' are $x : \Phi m$ for each $x : m \in X$, and $a : (0,1)$ for each $a \in Z$.

Next, the actions B' are generated by the following:

1. There is an injection $\Phi : B \to B'$ for which $\alpha : m \to n$ in B implies $\Phi\alpha : \Phi m \to \Phi n$ in B'.

2. In addition there are actions corresponding to those at the end of 3.5 above, subject to certain identities:

$$\begin{array}{rcll} \mathbf{in}_m a & : & (0,0) \to (m,0) & \text{– import datum through port } a \in Z \\ \mathbf{out}_m a & : & (m,0) \to (0,0) & \text{– export datum through port } a \in Z \\ \langle a \rangle & : & (0,0) \to (0,1) & \text{– the port } a \in Z \text{ as a datum} \\ \boldsymbol{\nu} & : & (0,0) \to (0,1) & \text{– created port} \\ \boldsymbol{\omega} & : & (0,1) \to (0,0) & \text{– discarded port.} \end{array}$$

Finally, reaction in $B^{(Z)}$ is generated by the rule

$$\Phi\alpha \searrow' \Phi\beta \ \text{ iff } \ \alpha \searrow \beta$$

ensuring that B is a sub-actionstructure of $B^{(Z)}$, together with the rule

$$\mathbf{out}_m a \circ \mathbf{in}_m a \searrow' \mathbf{1}_{(m,0)} \qquad (a \in Z, m \in M).$$

This construction remains to be detailed more precisely. Note in particular that we have *not* just taken the Cartesian product of B with $\mathsf{SCCS}[Z]$; for the port actions $\mathbf{in}_m a$ and $\mathbf{out}_m a$ are indexed by $a \in Z$, but their arities are drawn from B.

Note also that although we allow a port a to appear as a datum $\langle a \rangle$, it may not be exported through another port $b \in Z$; for the arities $\langle a \rangle : (0,0) \to (0,1)$ and $\mathbf{out}_m b : (m,0) \to (0,0)$ forbid the composition $\langle a \rangle \circ \mathbf{out}_m b$. This is exactly the limitation

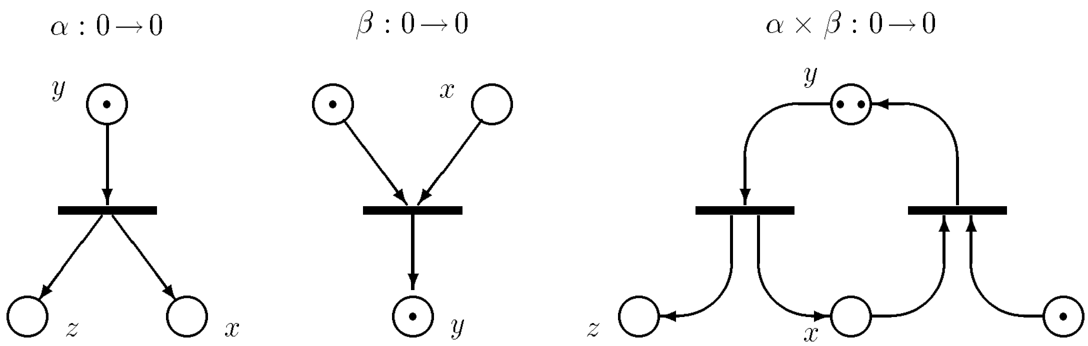


Figure 1: Two PT-nets and their product

of CCS with value-passing – namely, that ports cannot be passed as messages. (Indeed we expect to be able to present an action structure for CCS with value-passing just by extending $\mathsf{FUN}[X]$, for suitable data D, by adding ports in this way.) The π-calculus, whose action structure we shall give in a later paper, corresponds exactly to the removal of this limitation.

3.8 Petri nets The monoidal structure of Petri nets was first demonstrated by Meseguer and Montanari [13]. They model each place-transition net (PT-net) as a monoidal category, whose objects are markings and whose morphisms correspond to firing sequences. It is natural to ask whether the extra ingredients in an action structure can play a rôle in the formulation of Meseguer and Montanari. But they already model the dynamics of a net N by the morphisms of the monoidal category corresponding to N, so there is no rôle for the dynamics of action structures. Nor is there an obvious rôle for abstractors at the level of a single net N.

But there is a natural action structure at a higher level, where each marked PT-net N corresponds to an action. Here the abstractors find a good rôle to play, while an atomic reaction $\searrow^1$ corresponds to the firing of a single transition; if $\alpha \searrow^1 \alpha'$ then α' is the same net as α with a different marking.

This action structure will be treated in detail elsewhere; here we sketch it, for comparison with the previous examples. If X is the name-set then an action of arity $0 \to 0$ is just a marked PT-net some of whose places bear a name from X, no two places bearing the same name. Call this a *place-named* net. Figure 1 shows two place-named nets and their product; the product is formed just by coalescing like-named places.

We now turn to actions of arbitrary arity. An action $\delta : m \to n$ consists of a place-named net N together with a sequence $x_1, \ldots, x_m$ of distinct names representing "bound" or parametric places in the net, and a map (not necessarily injective) from $\{1, \ldots, n\}$ into the places of N indicating the"target" places. Typically we would write $\delta = (x_1, \ldots, x_m)\gamma$ with $\gamma : 0 \to n$. (We allow alpha-conversion of bound names in the usual way.) Figure 2 shows $\gamma : 0 \to 2$ formed from α by targetting two places.

To form the product of two actions $\alpha : 0 \to m$ and $\beta : 0 \to n$ with zero source arity, simply increment the target labels in β by m and proceed to coalesce like-named places

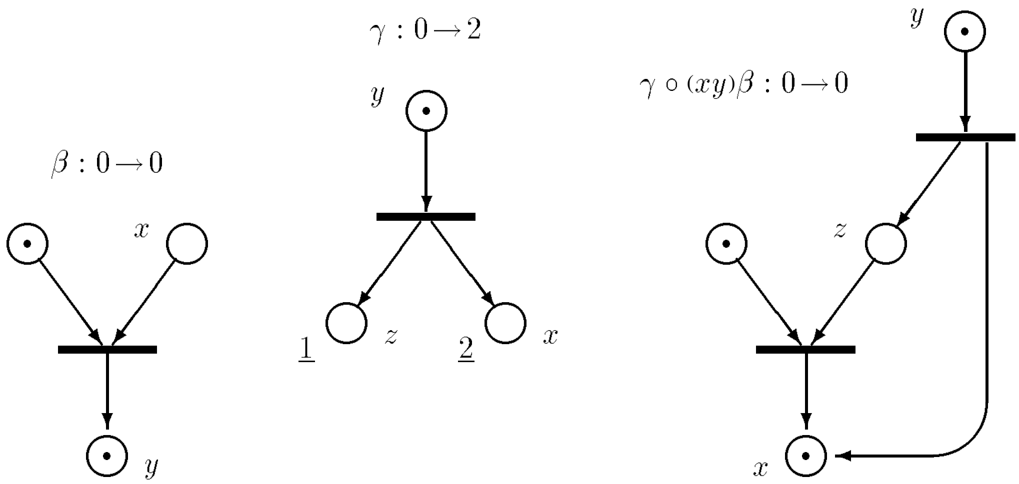


Figure 2: The composition of PT-net actions

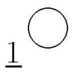
as in Figure 1. To form the product of two arbitrary actions $(\vec{x})\alpha$ and $(\vec{y})\beta$ first ensure by alpha-conversion that the bound names in one action do not occur free or bound in the other, then form $\alpha \times \beta$ as above; the result is $(\vec{x}\vec{y})(\alpha \times \beta)$.

To compose two actions, e.g. to form $\gamma \circ (xy)\beta$ where β and γ are as in Figure 2, first alpha-convert to disjoin bound names as for product; then coalesce each target place in the first action with the corresponding bound place in the second action, removing the target labels of the first action and the bound names of the second.

To form the abstraction $\mathbf{ab}_x\delta : 1{+}m \to 1{+}n$ for arbitrary $\delta : m \to n$, first increment the target labels in δ by one; then target the place named x as 1 (adding an unconnected place x if there is none) and bind x.

Finally, the reaction relation is generated by $\searrow^1$ which represents the firing of a single transition. With α as in Figure 1, we have $\alpha \searrow^1 \alpha'$ where both places x, z are marked in α', and α' is inactive. Moreover it is easily seen that $\alpha \times \beta \searrow^1 \alpha' \times \beta$, but the latter is *not* inactive.

So our action structure contains not only place-named nets, but also nets whose places are targeted and/or bound. Because of this, many operations on nets can be represented algebraically. A simple example is hiding a place, i.e. removing its name and thus fixing its pre- and post-transitions (since it may no longer be coalesced with a like-named place). For let ANON : $0 \to 1$ be the action



consisting of a single place – unnamed, unmarked and targeted. Then to hide the place named z in any $\gamma : 0 \to n$ we simply form ANON $\circ\, (z)\gamma : 0 \to n$. Other simple operations are: increasing the marking at a named place; renaming a place; coalescing two differently-named places. In addition, all place-named nets can be formed by product alone, from simple nets with at most one transition.

A dual action structure to the above allows the naming of transitions, instead of places. Then in forming the product one coalesces like-named transitions. So a named transition may acquire new pre- and post-conditions in this way. Thus the reaction relation $\searrow$ must only represent the firing of *un-named* transitions; this ensures that $\searrow$ is preserved by the action structure operations. Of course transitions, like places, can be hidden, i.e. deprived of their names; this will allow them to fire. We shall not describe this action structure in any further detail. But it is worth noting that, in contrast to the previous, its operations preserve freedom from conflict; so the conflict-free nets form an interesting sub-actionstructure.

Finally, these two action structures can be conflated, allowing naming (hence coalescing) both of transitions and of places.

Much work has been done on algebras of nets. Two good examples are the algebra AFP of Cherkasova and Kotov [7], and the box calculus of Best, Devillers and Hall [6] which is oriented towards defining concurrent programming languages. The action structure approach must be placed in that context, but this task is not addressed here; we have only been concerned to show that action structures yield at least one way of treating Petri nets algebraically. Its use of parametrisation appears to be new.

The relationship must also be examined between the present notion of action structure homomorphism, specialised to Petri nets, and the categorical treatment of nets and their morphisms by Winskel [20] and others.

4 Effects and effective action structures

A central aim of action structures is to make explicit the way in which an interaction exerts an effect upon its participants. Recall from Section 1 our example in the π-calculus; the transition

$$((\boldsymbol{\nu}x)\overline{a}x \centerdot P) \mid a(y) \centerdot Q \;\longrightarrow\; (\boldsymbol{\nu}x)(P \mid \{x/y\}Q)$$

is represented roughly speaking by the reaction

$$((\boldsymbol{\nu}x) \circ \overline{a}x) \times a(y) \searrow (\boldsymbol{\nu}x) \circ (\mathbf{1} \times \{x/y\})$$

in an appropriate action structure.

We shall refer to degenerate actions, such as the result of this reaction, as *effects*; our aim is to see how effects are related to other actions. In passing note that one might have expected the result of a reaction of a product to be itself a product, representing the separate effect which the reaction exerts on each of the participants P and Q. But that is not the case here. A central idea of the π-calculus is that the effect of a communication may be a liaison which did not exist before between a pair of agents, i.e. a restriction; this is an effect upon the product of a pair, not upon its individual members.

What distinguishes effects from other actions? *Inertia*, the inability to contribute to further reaction, is one distinguishing property.

4.1 Definition An action α is *inactive* if $\alpha \searrow \alpha' \Rightarrow \alpha' = \alpha$. An action α is *inert* if, whenever $\beta \circ \alpha \searrow \gamma$, then there exists β' such that $\beta \searrow \beta'$ and $\gamma = \beta' \circ \alpha$. ∎

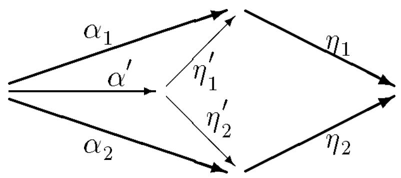


Figure 3: Decomposition for a postcomponent

Thus if α is inert then any reaction of $\beta \circ \alpha$ must be due to β alone.

4.2 Proposition If an action is inert then it is inactive.

Proof **1** is inactive by assumption; so take $\beta = \mathbf{1}$ in the definition. ∎

4.3 Proposition Each unit action **1** is inert. If α and β are inert then so is $\alpha \circ \beta$. ∎

Inertia is *not* symmetric with respect to composition. If α is inert then $\alpha \circ \beta$ may still have reactions not arising from β alone. For example α may be roughly speaking a substitution, which will indeed be an inert action, and the *effect* of this upon β may induce a reaction.

As well as demanding that each effect is an inert action, we also demand certain properites of the *set* of effects in an action structure. This will not fully determine the effects; indeed it will be useful to consider different subsets of actions as effects.

4.4 Definition Let A be an action structure, and E a sub-actionstructure of A. Then E is a *postcomponent* of A if whenever $\alpha = \alpha_1 \circ \eta_1 = \alpha_2 \circ \eta_2$, where $\alpha_i \in A$ and $\eta_i \in E$ ($i = 1, 2$), then there exist $\alpha' \in A$ and $\eta'_1, \eta'_2 \in E$ such that (see Figure 3):

$$\alpha_i = \alpha' \circ \eta'_i \ \ (i = 1, 2) \ \text{ and } \ \eta'_1 \circ \eta_1 \ = \eta'_2 \circ \eta_2 .$$

If further all members of E are inert, then E is an *effect structure* for A, or (A, E) is an *effective action structure*. ∎

In what follows, we shall consistently continue to use $\alpha, \beta, \gamma, \delta$ to range over all actions, and use $\epsilon, \zeta, \eta, \theta$ to range over effects, when the effect structure E is understood.

Note that "postcomponent" is a purely static notion. The condition that E is a postcomponent is about decompositions $\alpha = \alpha' \circ \eta$ of an action, where $\eta \in E$. A few more definitions will clarify the rôle played by decompositions.

4.5 Definition Let E be a sub-actionstructure of the action structure A. Then the pair (α', η) is a *decomposition of* α *for* E if $\alpha = \alpha' \circ \eta$ and $\eta \in E$. We define the following preorder over decompositions for E:

$$(\alpha_1, \eta_1) \leq (\alpha_2, \eta_2) \ \text{ if } \ \alpha_1 = \alpha_2 \circ \eta \ \text{ and } \ \eta \circ \eta_1 = \eta_2 \ \text{ for some } \ \eta \in E.$$

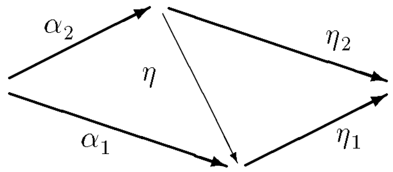


Figure 4: Preordered decompositions: $(\alpha_1, \eta_1) \leq (\alpha_2, \eta_2)$

(See Figure 4.) We say that a decomposition (α^*, η^*) of α for E is *maximal* if $(\alpha', \eta) \leq (\alpha^*, \eta^*)$ for any other decomposition (α', η) of α for E. ■

Now we can make sense of the postcomponent condition as follows: it requires that the decompositions of any action form a directed set under $\leq$. In establishing that a given sub-actionstructure of A is an effect structure, this condition is usually the hardest to check. It is helpful to look for *maximal* decompositions, since they often exist for interesting postcomponents. Their importance is of course as follows:

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

In fact something even stronger may hold, and usually does, for those postcomponents which we wish to use as effect structures. In general, for given α, the decomposition (α, η) may be maximal for some effects η but not for others. But in many effective action structures (A, E) there is a subset of A which behaves better than this:

4.7 Definition Let E be a sub-actionstructure of the action structure A. Then α is *pure for* E if for every $\eta \in E$ the decomposition (α, η) is maximal. We say that the decomposition (α^*, η^*) is *pure for* E if α^* is pure for E and $\eta^* \in E$. ■

In verifying that some given sub-actionstructure E of inert actions is an effect structure it is a good strategy first to identify the pure elements, and then to show that every action α has a pure decomposition, which is of course maximal. The strategy may not always work –there may not be enough pure elements– but it works in two of the cases we examine in Section 4.8 below. We shall also find in Section 6 that the concept of bisimilarity between processes is simpler when the the underlying action structure always has pure decompositions.

Let us denote by P the set of elements pure for E in A. Having identified P as an interesting set of actions, should we expect it to be a sub-actionstructure of A? It turns out to be so in the two examples just mentioned, where pure decompositions always exist. Under one or two extra conditions, which also hold in those examples, the pair (P, E) of sub-actionstructures becomes a *factorisation* of A, in the sense of Freyd and Kelly [8]. I am grateful to John Power for prompting me to look in this direction. There is a duality between the two members of a factorisation; we therefore have –at least in these examples– a duality between pure actions and effects, which may shed extra light on these concepts.

4.8 Examples What possible effect structures exist for action structures in general, and in particular for the examples in Section 3? Certainly the units $E = \{\mathbf{1}_m \mid m \in M\}$ always form an effect structure; this is easy to check. It is not a very interesting one, since every $\alpha : m \to n$ possesses only one effect in E, namely $\mathbf{1}_n$. Indeed, there is advantage in finding a larger effect structure; for we shall see later that this makes bisimilarity a more generous congruence relation over processes.

We turn now to the example action structures of Section 3. Both FUN and $\mathsf{FUN}[X]$ are static, i.e. reaction is the identity, so all actions are inert; therefore these action structures are effect structures for themselves. We would need to know more specific properties of the functions to determine any other effect structure here, apart from the units.

The evaluation of expressions, $\mathsf{EXP}[X]$ in Section 3.3, is immediately more interesting. First, there are many inactive actions which are not inert; $\langle y_1, y_2 \underline{+} \underline{4}\rangle$ is inactive, but if we precompose $\langle \underline{2}, \underline{3}\rangle$ to it we get $\langle \underline{2}, \underline{3} \underline{+} \underline{4}\rangle$ – which is active. However, any action $\langle e_1, \ldots, e_n\rangle$ containing only local variables and constants (i.e. no global variables or non-nullary operators) turns out to be inert, provided it contains each local variable at most once. An example is $\langle y_2, y_1, \underline{4}\rangle$. To see why each y_i should appear at most once, consider $\alpha = \langle y_1, y_1\rangle$; if $\beta = \langle \underline{2} \underline{+} \underline{3}\rangle$, we have

$$\beta \circ \alpha = \langle \underline{2} \underline{+} \underline{3}, \underline{2} \underline{+} \underline{3}\rangle \searrow \langle \underline{5}, \underline{2} \underline{+} \underline{3}\rangle$$

and we cannot find β' for which $\beta \searrow \beta'$ and $\langle \underline{5}, \underline{2} \underline{+} \underline{3}\rangle = \beta' \circ \alpha$. [2]

So do the actions containing only local variables and constants, and each y_i at most once, form an effect structure? They do indeed; but it turns out to be one in which maximal decompositions do not always exist. The reader may enjoy checking these facts, and also verifying that effects cannot contain global variables.

Now consider the subset of these actions containing each y_i *exactly* once. It is easy to see that they form a sub-actionstructure. And in contrast to the previous case we find that pure decompositions always exist, so we have an effect structure by Proposition 4.6. The pure actions are of the form $\langle d_1, \ldots, d_m\rangle$ where no d_j is a constant $\underline{v}$. For a pure decomposition (α^*, η^*) of any $\alpha = \langle e_1, \ldots, e_n\rangle$, take α^* to be some permutation $\langle e_{i_1}, \ldots, e_{i_m}\rangle$ of the expressions e_i excluding any which is a constant.

For SCCS(Section 3.4) it's easy to see that the only effect structure is $\{\mathbf{1}_0\}$; for a non-empty multiset cannot be inert. But $\mathsf{SCCS}[X]$ and $\mathsf{CCS}[X]$ (Sections 3.5 and 3.6) are more interesting. Let E contain all actions $(\vec{x})\,(\boldsymbol{\nu}\vec{y})\,S\,\langle\vec{z}\rangle$ in which the multiset S is empty; then all members of E are certainly inert. Also E is indeed an effect structure. Again, this can be shown by identifying the pure actions; they turn out to

[2] Of course if we take $\beta' = \langle \underline{5}\rangle$ then we *do* have the weaker property $\langle \underline{5}, \underline{2} \underline{+} \underline{3}\rangle \searrow \beta' \circ \alpha$. This suggests that a form of inertia weaker than Definition 4.1 might be useful, with the condition $\gamma = \beta' \circ \alpha$ weakened to $\gamma \searrow \beta' \circ \alpha$. It remains to be seen how much of the ensuing theory goes through with this weaker notion. But it is worth noting that for the other examples discussed in Section 3 the weakening does not seem to increase the set of inert actions.

be all those $(\vec{x})\,(\boldsymbol{\nu}\vec{y})\,S\,\langle\vec{z}\rangle$ in which every bound name (one of $\vec{x}$ or $\vec{y}$) occurs in $\vec{z}$, and every restricted name (one of $\vec{y}$) also occurs in S.

These last two examples of effect structures are strikingly different; so it appears that the notion of effective action structure is quite a broad one.

4.9 Inertia reconsidered According to Definition 4.1, for α to be inert means that α *cannot create reactions if you postcompose it*, i.e. if you form $\gamma \circ \alpha$ for some γ. But our interest in effects is based on the idea that *an effect η can create reactions if you precompose it*, i.e. if you form $\eta \circ \gamma$ for suitable γ. Now it turns out in all our examples that *an effect η can create reactions only if you precompose it*, e.g. not if you juxtapose it as in $\eta \times \gamma$. So why do we not take this stronger condition to be the definition of inertia? Does it make much difference if we do?

The answer is: There may be fewer inert actions with this stronger condition, but it does not affect the concept of effect structure at all. Essentially this is because not only are the individual actions of an effect structure E required to be inert, but also E is required to be a sub-actionstructure.

Let us make this claim precise.

4.10 Definition A *preaction* φ is a unary operation on actions built from operations of the form $\gamma \circ (-)$, $\gamma \times (-)$, $(-) \times \gamma$ and $\mathbf{ab}_x(-)$ where γ is any action. ■

The term "preaction" is chosen to indicate that *precomposition* $\gamma \circ (-)$, but not *postcomposition* $(-) \circ \gamma$, is allowed in forming preactions.

Structural congruence and the reaction relation extend in an obvious way to preactions.

We now define strong inertia in terms of preactions; this contrasts with the definition of inertia, which used only precomposition. So strong inertia clearly implies inertia.

4.11 Definition An action α is *strongly inert* if whenever $\varphi(\alpha) \searrow \beta$, then $\beta = \varphi'(\alpha)$ for some preaction φ' such that $\varphi \searrow \varphi'$. ■

Now we show that it would make no difference to the notion of effect structure, Definition 4.4, if we demanded that its members be *strongly* inert.

4.12 Lemma Every preaction φ can be expressed as $\varphi = \alpha \circ \psi$, where ψ is a preaction formed by the operations $\mathbf{1} \times (-)$, $(-) \times \mathbf{1}$ and $\mathbf{ab}_x(-)$.

Proof By induction on the structure of preactions. To take just the most interesting case, consider the preaction $\gamma \times \varphi$. By assumption $\varphi = \alpha \circ \psi$ where ψ is of the required form, so

$$\begin{aligned}\gamma \times \varphi &= \gamma \times (\alpha \circ \psi) \\ &= (\gamma \times \alpha) \circ (\mathbf{1} \times \psi)\end{aligned}$$

which is of the required form. ■

4.13 Proposition Let E be an effect structure. Then all its members are strongly inert.

Proof Let $\eta \in E$ and let φ be a preaction such that $\varphi(\eta) \searrow \beta$. Now let $\varphi = \alpha \circ \psi$ as in the lemma; so $\alpha \circ \psi(\eta) \searrow \beta$. But $\psi(\eta) \in E$ is inert, so for some α' we have $\alpha \searrow \alpha'$ and $\beta = \alpha' \circ \psi(\eta)$. But $\alpha' \circ \psi$ is a preaction and $\varphi \searrow \alpha' \circ \psi$. So we have shown η to be strongly inert. ■

Thus we have lost nothing in demanding only the weaker condition of inertia in defining an effect structure; moreover this weaker condition is simpler to verify in examples.

Preactions will turn up again in Section 7, where we define the notion of *incident*.

5 Process structure: statics

We can employ action structures in more than one way to model concurrent computation. The first way is to use them to explain the single transitions, of which the behaviour of interactive processes is constituted. At the higher level of the processes themselves these single transitions may be considered atomic, though the detailed performance of an action –i.e. the reaction within it– can be highly complex.

There is a second approach which must be explored: the use of action structures to explain whole processes, as well as to explain their constituent actions at the lower level. This approach will depend upon finding ways of building higher-level action structures from lower-level ones. The possibilities in this direction appear to be rich; there are several candidate constructions to be examined. Indeed we outlined one of them in Section 3.7; the systematic extension of action structures by the addition of ports.

Here, however, we take the first approach. We show how certain control operations can be added to an arbitrary action structure, to yield a process structure. We thus obtain a generic process calculus, and are able carry its theory forward a fair way independently of the underlying action structure.

Thus we bring some unity to the study of process calculi. Specifically, we can unite the study of CCS, SCCS and the π-calculus. Other calculi such as ACP [2] and CSP [11] may also fit into the framework; this remains to be seen. Even though complete unity is difficult to attain, the intrinsic coherence of the notion of action structure carries some weight, as does the quite natural extension to process structure which we shall now exhibit.

5.1 Process syntax We study processes over an arbitrary action structure, whose actions will be denoted by $\alpha, \beta, \ldots$ as before. Processes, denoted by $P, Q, \ldots$, are given by the syntax

$$\begin{array}{rcl} P & ::= & \alpha \circ P \;\big|\; P \times P \;\big|\; \mathbf{ab}_x P \\ & & \big|\; \bullet P \;\big|\; \mathbf{0} \;\big|\; P + P \;\big|\; \partial P \;\big|\; !P \;. \end{array}$$

The first three constructions lift the operations of an action structure to process level, but allowing composition only between an action and a process rather than the composition

$P \circ Q$ of two processes. (The theory of the latter is not quite as simple.)

The fourth construction $\bullet P$ represents what we shall call *committal*. Once we have defined reaction for processes, then we shall find that a process P may be capable of many different reactions of the form

$$P \searrow \alpha \circ \bullet P'$$

which we shall call *commitments*. To establish a link with CCS syntax we shall elide the "$\circ$" before "$\bullet$", writing

$$P \searrow \alpha \bullet P'.$$

Each commitment represents P's ability to perform a single action α, considered atomic at process level, with continuation P'. Moreover, if (α', η) is a decomposition of α then our theory will reflect the intuition that the effect η exerts an influence upon the continuation P'. The need to write down this influence explicitly, as $\eta \circ P'$, is one reason for admitting the composition construction at process level.

The last five constructions from committal ($\bullet$) onwards represent control, and we call them *control terms*; equally we call $\bullet$, $\mathbf{0}$, $+$, $\boldsymbol{\partial}$ and $!$ *control operators*, and refer to them respectively as committal, zero, sum, delay and replication. The last four are drawn from SCCS and the π-calculus; but they have no dependence on the nature of the *actions* in those calculi, and that is why it makes sense to introduce them at a different, higher, level. I don't claim that the five control operators are a complete or fundamental set in any sense; indeed I think the notion of action structure has greater claim to this virtue. But they work well in their parent calculi, so they serve as exemplars for the way in which we uniformly add control structure to action structure. It is worth recalling from [14] that, in the framework of SCCS, the parallel composition operator "|" of CCS can be recovered by the definition

$$P \mid Q \stackrel{\text{def}}{=} P \times \boldsymbol{\partial} Q + \boldsymbol{\partial} P \times Q$$

provided that the CCS prefixing operator $\alpha \bullet P$ is treated as $\alpha \bullet \boldsymbol{\partial} P$.

The dynamics of each operator is presented in Section 6; the remainder of this section is devoted to statics.

5.2 Arity Processes do not constitute an action structure; in this approach they are not actions. Instead, each process has an arity $m \in M$ given by the rules

$$\frac{\alpha : m \to n \qquad P : n}{\alpha \circ P : m} \qquad \frac{P : m \qquad Q : n}{P \times Q : m{+}n} \qquad \frac{x : k \qquad P : m}{\mathbf{ab}_x P : k{+}m}$$

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

These are mostly straightforward. A process with non-zero arity should be thought of as parametric, ready to import its parameter –typically an effect– by precomposition. Note that $!P$ is only allowed when $P : 0$. Why? Well, we expect $!P$ to be behaviourally equivalent to $(P \times\ !P) + \bullet\ !P$, so the arities of these two must be the same. Bearing in mind the rules for sum and committal, this implies that the two arities $P : k$ and $!P : m$ satisfy the equations $m = k{+}m = k$, which implies $m = k = 0$.

5.3 Structural Congruence We impose a structural congruence, $\equiv$, upon the syntax of processes. It is defined by the following laws, i.e. it is the smallest congruence satisfying them:

$$\begin{array}{rcl} P & \equiv & 1 \circ P \\ \alpha \circ (\beta \circ P) & \equiv & (\alpha \circ \beta) \circ P \\ (\alpha \circ P) \times (\beta \circ Q) & \equiv & (\alpha \times \beta) \circ (P \times Q) \\ \mathbf{ab}_x(\alpha \circ P) & \equiv & (\mathbf{ab}_x\alpha) \circ (\mathbf{ab}_x P) \, . \end{array}$$

As far as possible these are the laws of action structures lifted to process level. In particular they allow us to confuse the process-level and action-level operations with impunity.

Note that there are no laws for the control operators. We could have imposed some, e.g. $P + Q \equiv Q + P$, $\mathbf{ab}_x \bullet P \equiv \ \bullet \mathbf{ab}_x P, \ldots$, but we prefer to attain these equations for a *behavioural* (rather than structural) congruence. Thus we have a clean separation in the treatment of action operators and control operators; the latter are done *via* the process dynamics (see Section 6).

We shall need the following notion: An occurrence in P of an action α or process term Q is called a *ready* occurrence if it is not within any control term. For example α occurs readily in $P \times \alpha \bullet Q$ but not in $P + \alpha \bullet Q$.

Now that we have established a suitable connection between composition ($\circ$) of actions and composition of an action with a process, it is a good moment to begin to omit the composition symbol; so henceforth we shall write $\alpha \circ \beta$ as $\alpha\beta$ and $\alpha \circ P$ as αP. We give (elided) composition highest binding power, and all other binary operators lowest binding power; so $\alpha\beta \times \gamma\delta$ means $(\alpha\beta) \times (\gamma\delta)$, $\mathbf{ab}_x\gamma\delta$ means $\mathbf{ab}_x(\gamma\delta)$, $\mathbf{ab}_x\gamma \times \delta$ means $(\mathbf{ab}_x\gamma) \times \delta$ and $\partial P + Q$ means $(\partial P) + Q$.

5.4 Passive processes The dynamics of processes will be a cooperation between reactions due to the underlying action structure, and control reductions. The former can only be contributed by ready occurrences of actions α. Intuitively we can say that a process is passive (cannot react) if no $\alpha \neq \mathbf{1}$ has a ready occurrence. Thus if $\alpha \neq \mathbf{1}$ we would say that $\alpha \bullet P \times \ \bullet Q$ is active; on the other hand $\alpha \bullet P + \ \bullet Q$ is passive, because a control reduction is needed to allow α to react. But we need to refine this intuitive definition of "passive" into one which is preserved by structural congruence. The following meets our purpose:

5.5 Definition The *passive* processes are the smallest set closed under structural congruence such that

- All control terms are passive;
- If P, Q are passive, then so are $P \times Q$ and $\mathbf{ab}_x P$. ∎

(The important feature here is the omission of composition, αP, from the second clause.) The term "passive" is justified if we consider the dynamics of Section 6; for it will follow from Lemma 6.6 that no reaction can occur in an passive process, until a control reduction has occurred.

We shall use $H, J, K, \ldots$ to range over passive processes. The next few results show that any P can be uniquely expressed in the form αH, up to structural congruence.

5.6 Definition If $P \equiv \alpha H$, where H is passive, then the latter is called a *head normal form* (*hnf*) of P. ■

5.7 Proposition Every process possesses a head normal form.

Proof By induction on the structure of process terms. ■

We may take this as a justification of our structural congruence laws (5.3), since they seem to be the minimum which will yield such a result. The next proposition strengthens this claim:

5.8 Proposition Head normal forms are unique up to structural congruence; that is, if $\alpha H \equiv \beta J$ then $\alpha = \beta$ and $H \equiv J$.

Proof We first show that $\alpha H \equiv \beta J$ implies $\alpha = \beta$. For this purpose, we use an interpretation $\mathcal{I}$ of process terms in the underlying action structure; $\mathcal{I}$ is defined in the obvious way for composition, product and abstraction over processes, and takes all control terms to the unit action. Thus

$$\begin{array}{rcll} \mathcal{I}(\alpha P) & \stackrel{\text{def}}{=} & \alpha \mathcal{I}(P) & \\ \mathcal{I}(P \times Q) & \stackrel{\text{def}}{=} & \mathcal{I}(P) \times \mathcal{I}(Q) & \\ \mathcal{I}(\mathbf{ab}_x P) & \stackrel{\text{def}}{=} & \mathbf{ab}_x \mathcal{I}(P) & \\ \mathcal{I}(R) & \stackrel{\text{def}}{=} & \mathbf{1}_m & \quad \text{if } R \text{ is a control term.} \end{array}$$

(In the last case m is the arity of the control term.)

Note that if $P : m$ then $\mathcal{I}(P) : m \to n$ for some n which is determined by the translation. It is easy to check that the four laws of structural congruence preserve $\mathcal{I}$, i.e. $P \equiv Q$ implies $\mathcal{I}(P) = \mathcal{I}(Q)$. From Definition 5.5 it follows that $\mathcal{I}(H) = \mathbf{1}$ for any passive process H. Hence from $\alpha H \equiv \beta J$ we infer

$$\alpha = \mathcal{I}(\alpha H) = \mathcal{I}(\beta J) = \beta.$$

Next we show that $\alpha H \equiv \beta J$ implies $H \equiv J$. For any P, let $\widehat{P}$ be the result of replacing all ready occurrences of actions by the unit action. For example if P is $R \times \alpha \boldsymbol{\partial}(\beta Q)$ then $\widehat{P}$ is $\widehat{R} \times \boldsymbol{\partial}(\beta Q)$. In general the arity of $\widehat{P}$ will differ from that of P, but not in the case of an passive term H – in fact, $\widehat{H} \equiv H$.

Now suppose that $P \equiv Q$, and that this is proved by a single application of one of the four laws of structural congruence to a subterm of P. Then it can be checked that also $\widehat{P} \equiv \widehat{Q}$. Essentially this is because in each law a ready occurrence on one side corresponds uniquely to a ready occurrence on the other side. So in general $P \equiv Q$ implies $\widehat{P} \equiv \widehat{Q}$. Thus from $\alpha H \equiv \beta J$ we infer

$$H \equiv \widehat{\alpha H} \equiv \widehat{\beta J} \equiv J$$

as required. ■

If we consider a process to be a structural congruence class of process terms, then Proposition 5.8 is the most important property of the statics, or structure, of processes. It assures us that even though we have imposed structural congruence we still have structure which is simple and natural. A corollary is the following:

5.9 Corollary Let $P \equiv \alpha P'$. Then

1. If P is γQ then for some β, γ' and Q'

$$Q \equiv \beta Q', \;\; \alpha\gamma' = \gamma\beta \;\text{ and }\; P' \equiv \gamma' Q'.$$

2. If P is $Q \times R$ then for some β, γ, Q' and R'

$$Q \equiv \beta Q', \;\; R \equiv \gamma R', \;\; \alpha = \beta \times \gamma \;\text{ and }\; P' \equiv Q' \times R'.$$

3. If P is $\mathbf{ab}_x Q$ then for some β and Q'

$$Q \equiv \beta Q', \;\; \alpha = \mathbf{ab}_x \beta \;\text{ and }\; P' \equiv \mathbf{ab}_x Q'.$$

4. If P is a control term then for some α'

$$\alpha\alpha' = \mathbf{1} \;\text{ and }\; P' \equiv \alpha' P.$$

Proof In clause 1 for example take Q' to be passive, so that $\beta Q'$ is the unique hnf of Q. ■

Finally, we shall need the following simple proposition about committal:

5.10 Proposition If $\bullet P \equiv \bullet Q$ then $P \equiv Q$.

Proof For any R, let $\widehat{R}$ be the result of simultaneously replacing every ready occurrence of a commitment $\bullet S$ in R by S itself. By means similar to the second half of the proof of Proposition 5.8, we see that $R_1 \equiv R_2$ implies $\widehat{R_1} \equiv \widehat{R_2}$. But $\widehat{\bullet P}$ is P, so the result follows. ■

6 Process structure: dynamics

Our presentation of process dynamics is best introduced by contrast with what was done for CCS and SCCS. Consider the SCCS transition

$$(a \bullet P + b \bullet Q) \times \overline{a} \bullet R \xrightarrow{1} P \times R.$$

According to [14] it is derived by three transition rules, whose general form is:

$$\alpha \bullet P \xrightarrow{\alpha} P \qquad\qquad \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$$

$$\frac{P \xrightarrow{\alpha} P' \qquad Q \xrightarrow{\beta} Q'}{P \times Q \xrightarrow{\alpha\times\beta} P' \times Q'}.$$

Here we wish to derive such transitions in another way; we wish to distinguish the contributions made to any transition by *control* from those made by *action*. We shall use the relations $\underset{c}{\searrow}$ and $\underset{a}{\searrow}$ for these respective contributions. So taking SCCS is the underlying action structure, the above transition will be derived in three stages using the rules given below in 6.2 and 6.3:

$$\begin{array}{rcll}
(a \bullet P + b \bullet Q) \times \overline{a} \bullet R & \underset{c}{\searrow} & a \bullet P \times \overline{a} \bullet R & \text{by a rule for } + \\
& \equiv & (a \times \overline{a})(\,\bullet P \times \,\bullet R) & \\
& \underset{c}{\searrow} & (a \times \overline{a}) \bullet (P \times R) & \text{by a rule for } \bullet \\
& \underset{a}{\searrow} & \mathbf{1} \bullet (P \times R) & \text{by the rule for } \underset{a}{\searrow}.
\end{array}$$

Thus, if *reduction* $\searrow$ over processes is defined as $(\underset{a}{\searrow} \cup \underset{c}{\searrow})^*$ then we have derived $S \searrow \mathbf{1} \bullet (P \times R)$, where S stands for the original system.

We now proceed to the formal definitions.

6.1 Definition A *context* $\mathcal{C}$ is a process term with a single hole, denoted by [ ], to be filled by a 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. ■

Structural congruence can be extended in an obvious way to contexts. $[\,] \times Q$ is an example of a ready context; $[\,] + Q$ is not ready. It is important to note that structural congruence preserves readiness of a context.

6.2 Definition The *action preorder* $\underset{a}{\searrow}$ over processes is the least preorder closed under structural congruence, such that for all ready $\mathcal{C}$

$$\alpha \searrow \alpha' \;\text{ implies }\; \mathcal{C}[\alpha P] \underset{a}{\searrow} \mathcal{C}[\alpha' P].$$

We use $\underset{a}{\searrow}^{1}$ to represent a single use of this rule, so that $\underset{a}{\searrow} \;=\; (\underset{a}{\searrow}^{1})^*$. ■

6.3 Definition The *control preorder* $\searrow_{c}$ over processes is the least preorder closed under structural congruence satisfying the following *control rules*:

$$\begin{array}{llll}
\bullet P \times \bullet Q \searrow_{c} \bullet(P \times Q) & P+Q \searrow_{c} P & \partial P \searrow_{c} P & !P \searrow_{c} P \times\, !P \\
\mathbf{ab}_x \bullet P \searrow_{c} \bullet \mathbf{ab}_x P & P+Q \searrow_{c} Q & \partial P \searrow_{c} \bullet \partial P & !P \searrow_{c} \bullet\, !P
\end{array}$$

and such that for all ready $\mathcal{C}$

$$P \searrow_{c} P' \text{ implies } \mathcal{C}[P] \searrow_{c} \mathcal{C}[P'].$$

We use $\searrow_{c}^{1}$ to represent a single use of one of the control rules in a ready context; then $\searrow_{c} = (\searrow_{c}^{1})^*$. ■

Each rule for sum, delay and replication serves one of two purposes; either it makes some process ready for action, or it makes a commitment. A variety of control operators can be similarly defined. But the rules for committal are somewhat special; they lift commitments of subprocesses to the top level. For example, we obtain a global commitment from two local ones as follows:

$$\alpha \bullet P \times \mathbf{ab}_x(\beta \bullet Q) \searrow (\alpha \times \mathbf{ab}_x \beta) \bullet (P \times \mathbf{ab}_x Q).$$

6.4 Definition The *reduction preorder* $\searrow$ over processes is the least preorder which includes $\searrow_{a}$ and $\searrow_{c}$. Thus $\searrow = (\searrow_{a} \cup \searrow_{c})^*$. ■

Of course $\searrow = (\searrow^{1})^*$ where $\searrow^{1} = (\searrow_{a}^{1} \cup \searrow_{c}^{1})^*$; so $\searrow$ consists just of a sequence of action and control steps.

A moment's thought shows that an action step creates no new possible control steps; so all the control steps can be done first. Hence

6.5 Proposition $\searrow \; = \; \searrow_{c} \searrow_{a}$. ■

One illustration of this result is the SCCS example with which we began this section. It also shows how, by contrast, a control step *can* create new possible action steps; for example, it can make an action occurrence ready.

Now, to prepare for the introduction of various forms of similarity and bisimilarity over processes, we prove two propositions which represent the basic theory of reduction. This theory is very simple, and rests conveniently on the notion of head normal form introduced in the previous section.

6.6 Lemma Let $\alpha H \searrow_{a} \beta J$. Then $\alpha \searrow \beta$ and $H \equiv J$. (Compare Corollary 5.9.)

Proof It is enough to prove the result for $\searrow_{a}^{1}$. In fact it is easiest to prove first that $P \searrow_{a}^{1} P'$ implies

$$P \equiv \gamma R, \;\; P' \equiv \gamma' R \text{ and } \gamma \searrow \gamma' \text{ for some } \gamma, \gamma' \text{ and } R.$$

We prove this by a straightforward induction on the structure of the ready context $\mathcal{C}$ for which $P \equiv \mathcal{C}[\delta S]$, $P' \equiv \mathcal{C}[\delta' S]$ and $\delta \searrow \delta'$.

Now take P and P' to be αH and βJ. Then for some γ, γ', R we have

$$\alpha H \equiv \gamma R, \ \ \beta J \equiv \gamma' R \ \text{ and } \ \gamma \searrow \gamma'.$$

Now let R have hnf $\gamma'' K$. Then from the uniqueness of hnfs it follows that

$$H \equiv K \equiv J \ \text{ and } \ \alpha = \gamma\gamma'' \searrow \gamma'\gamma'' = \beta. \qquad \blacksquare$$

The corresponding lemma for $\underset{c}{\searrow}$ is easier:

6.7 Lemma Let $\alpha H \underset{c}{\searrow} \beta J$. Then there exists γ such that $H \underset{c}{\searrow} \gamma J$ and $\alpha\gamma = \beta$.

Proof Again, it is enough to prove the result for $\overset{1}{\underset{c}{\searrow}}$. There is some ready context $\mathcal{C}$ and some K, Q such that

$$H \equiv \mathcal{C}[K], \ \ K \overset{1}{\underset{c}{\searrow}} Q \ \text{ and } \ \beta J \equiv \alpha\mathcal{C}[Q].$$

Now let $\mathcal{C}[Q]$ have hnf $\gamma J'$. Then $H \overset{1}{\underset{c}{\searrow}} \gamma J'$, and by the uniqueness of hnfs it follows that $\beta = \alpha\gamma$ and $J \equiv J'$. $\blacksquare$

Putting these two lemmas together, with the help of Proposition 6.5 we have proved a characterisation of reduction in terms of hnfs:

6.8 Proposition Let $\alpha H \searrow \beta J$. Then for some γ, $H \underset{c}{\searrow} \gamma J$ and $\alpha\gamma \searrow \beta$. $\blacksquare$

Our second basic result about dynamics is about commitments $R \searrow \gamma \bullet R'$. It is the dynamic analogue of Corollary 5.9, and asserts that commitments of a composite process arise from commitments of its components. It prepares for the proof that bisimilarity is a congruence, Theorem 7.10; readers who are not interested in the detail of that proof can safely skip to 6.10.

6.9 Proposition Let $P \searrow \alpha \bullet P'$. Then

1. If P is βQ then for some γ

$$Q \underset{c}{\searrow} \gamma \bullet P' \ \text{ and } \ \beta\gamma \searrow \alpha.$$

2. If P is $Q \times R$, then for some β, γ, Q' and R'

$$Q \underset{c}{\searrow} \beta \bullet Q', \ \ R \underset{c}{\searrow} \gamma \bullet R', \ \ \beta \times \gamma \searrow \alpha \ \text{ and } \ Q' \times R' \equiv P'.$$

3. If P is $\mathbf{ab}_x Q$ then for some β and Q'

$$Q \underset{c}{\searrow} \beta \bullet Q', \ \ \mathbf{ab}_x \beta \searrow \alpha \ \text{ and } \ \mathbf{ab}_x Q' \equiv P'.$$

Proof For clause 1 consider the hnf of P, hence of R, and apply Proposition 6.8. It will be enough to outline the proof of clause 3; clause 2 is similar. From the assumption, we have $\mathbf{ab}_x Q \searrow_{c} \alpha' \bullet P'$ where $\alpha' \searrow \alpha$. Now by inspection the control reduction must take the form of $n \geq 0$ steps of the form

$$\mathbf{ab}_x Q_i \searrow_{c}^{1} \mathbf{ab}_x Q_{i+1}$$

where $Q \equiv Q_0$, $Q_i \searrow_{c}^{1} Q_{i+1}$ $(0 \leq i < n)$ and $Q_n \equiv \beta \bullet Q'$, followed by a last step of the form

$$\mathbf{ab}_x Q_n \equiv (\mathbf{ab}_x \beta)\, \mathbf{ab}_x(\bullet Q') \searrow_{c}^{1} (\mathbf{ab}_x \beta) \bullet \mathbf{ab}_x Q'.$$

(Note that no further control reduction is possible after this step.) Thus we have $Q \searrow_{c} \beta \bullet Q'$ where $\mathbf{ab}_x \beta = \alpha' \searrow \alpha$ and $\mathbf{ab}_x Q' \equiv P'$ as required. ■

6.10 Commitment and transition We have seen that for a commitment

$$P \searrow \alpha \bullet P'$$

no further control reduction is possible; if also α is inactive, then no further reduction at all can occur.

At this point recall that we intend to model the single transitions of processes as actions in an action structure. (See the beginning of Section 5.) Commitments are the key to this. As a first approximation we can think of the above commitment as a single transition of P, and write it as $P \xrightarrow{\alpha} P'$, recalling that process calculi like CCS are defined in terms of transitions labelled with actions. But this is not quite enough; for we declared that we should make explicit the way in which an action exerts an effect $\eta \in E$ upon the continuation of a process. Moreover the transition labels in CCS, except for τ, represent that part of an action which may interact with or be observed by the environment; but we saw at the end of Section 4 that effects are strongly inert, which implies that they cannot react with the environment. So in proceeding from $P \searrow \alpha \bullet P'$ to $P \xrightarrow{\alpha} P'$ we somehow want to absorb any effect of α into P'. How shall we do this? Since an action does not possess an unique effect, though its effects are directed under precomposition (see after Definition 4.5), the following definition is appropriate:

6.11 Definition The *transition relations* $\{\xrightarrow{\alpha} \mid \alpha \in A\}$ over processes are defined by

$$P \xrightarrow{\alpha} P' \text{ iff, for some } \eta \text{ and } P'',\quad P \searrow \alpha\eta \bullet P'' \text{ and } \eta P'' \equiv P'. \qquad ■$$

These relations behave quite well. The following proposition shows that they are appropriately preserved by composition, abstraction and product of processes, by reaction of actions, and by factorisation of an action through an effect:

6.12 Proposition Let $P \xrightarrow{\alpha} P'$. Then

1. $\gamma P \xrightarrow{\gamma\alpha} P'$;
2. $\mathbf{ab}_x P \xrightarrow{\mathbf{ab}_x\alpha} \mathbf{ab}_x P'$;
3. if also $Q \xrightarrow{\beta} Q'$ then $P \times Q \xrightarrow{\alpha\times\beta} P' \times Q'$;
4. if $\alpha \searrow \alpha'$ then $P \xrightarrow{\alpha'} P'$;
5. if $\alpha = \alpha'\eta$ then $P \xrightarrow{\alpha'} \eta P'$. ∎

The next step is to define a notion of (bi)simulation over processes, which uses these transitions. It takes into account that the rôle of an effect η is not to be observed, but to modify the successor process; thus an action with one effect may be simulated by another with different effect, provided the two actions have the same observable content.

6.13 Definition A binary relation $\mathcal{S}$ over processes is a *simulation* if, whenever $(P, Q) \in \mathcal{S}$ and $P \xrightarrow{\alpha} P'$, then there exist α', ε and Q' such that

$$\alpha = \alpha'\varepsilon, \;\; Q \xrightarrow{\alpha'} Q' \;\text{ and }\; (\varepsilon P', Q') \in \mathcal{S}.$$

If $\mathcal{S}$ is also symmetric then it is called a *bisimulation*.

If the pair (P, Q) is in some simulation then we say that Q *simulates* P and write $P \lesssim Q$. If (P, Q) is in some bisimulation then we say that P and Q are *bisimilar* and write $P \sim Q$. ∎

We shall hold off the theory of $\lesssim$ and $\sim$ until the next section, because it is a special case of what is done there. But it is worth recording here a corollary of the main result:

6.14 Theorem $\lesssim$ is a precongruence, and $\sim$ a congruence, for the algebra of processes; that is, these relations are respectively a preorder and an equivalence, and are preserved by all the operations introduced in Section 5.1.

Proof This follows from Theorem 7.10 by taking K, the incident-set, to be the entire action structure. ∎

It is also worth observing that a simpler notion of simulation suffices, in the case that every action has a pure decomposition (see Definition 4.7):

6.15 Definition $\mathcal{S}$ is a *pure simulation* if, whenever $(P, Q) \in \mathcal{S}$ and $P \xrightarrow{\alpha} P'$ for some pure α, then there exists Q' such that $Q \xrightarrow{\alpha} Q'$ and $(P', Q') \in \mathcal{S}$. If $\mathcal{S}$ is also symmetric, then it is a *pure bisimulation*. ∎

Then under the stated condition this gives rise to exactly the same theory:

6.16 Proposition If every action has a pure decomposition, then $P \lesssim Q$ (*resp*. $P \sim Q$) iff (P, Q) is in some pure simulation (*resp*. bisimulation). ∎

This theory is therefore quite pleasant. But even if we choose a natural effect structure –and whether or not there exist pure decompositions– it turns out that this notion of bisimulation does not yield the familiar equivalence in familiar cases; it is still too strong. We may naturally try to weaken it by restricting the range of α in Definitions 6.13 and 6.15; but this does not always yield a congruence.

The remedy lies in the concept of an *incident*, which is the subject of the following section.

7 Incidents and incident-simulation

We begin this section by showing why bisimilarity, as defined in Definition 6.13, is a stronger equivalence than we would wish.

Recall from Section 3.4 the action structure $\mathsf{SCCS}[Z]$. Choose the effect structure found in Section 4.8, which consists of all actions $(\vec{x})\,(\boldsymbol{\nu}\vec{y})\,S\,\langle\vec{z}\rangle$ whose multiset S is empty. Now $a \times \overline{a}$, or more fully $(\,)(\,)\{a, \overline{a}\}(\,)$, is a different action from $b \times \overline{b}$. So when we consider processes over this action structure we find

$$a \bullet P \times \overline{a} \bullet Q \;\not\sim\; b \bullet P \times \overline{b} \bullet Q,$$

since the former has the transition $\cdots \xrightarrow{a \times \overline{a}} P \times Q$ which cannot be matched by the latter in the sense required for a simulation (6.13). But in SCCS as originally defined the two processes *are* bisimilar, and we would wish them so. (Recall that the product of SCCS is synchronous –unlike in CCS– so a cannot act independently of $\overline{a}$.) In fact they are both bisimilar to $\mathbf{1} \bullet (P \times Q)$.

It is tempting to relax the definition of simulation by restricting consideration to transitions $\xrightarrow{\alpha}$ in which α is inactive, thus excluding things like $a \times \overline{a}$. This indeed works in the case of SCCS. But for *general* conditions under which we can restrict the considered transitions, and still retain the congruential property of bisimulation, we have to be more careful. Indeed, for the synchronous π-calculus (see Appendix A) it does not work to consider only the inactive actions in simulation.

Throughout this section we assume some arbitrary fixed effective action structure (A, E).

7.1 Definition An *incident-set* K is a subset of the actions of A such that

1. For all $\alpha \in A$ and $\eta \in E$ such that $\alpha\eta$ is defined, $\alpha\eta \in K$ iff $\alpha \in K$;
2. For any preaction φ, if $\varphi(\alpha) \searrow \lambda \in K$ then there exists $\kappa \in K$ for which $\alpha \searrow \kappa$ and $\varphi(\kappa) \searrow \lambda$. ∎

Henceforward, when an incident-set K is understood we shall call its members *incidents*, and use ι, κ, λ, μ to range over them. Here, as in the definition of an effect structure, we have imposed the weakest conditions which yield interesting results. For example K need not be a sub-actionstructure.

Clause 2 is the main condition which will ensure congruence. Clause 1 is auxiliary; an important consequence of it is that for a given incident κ, its decompositions (κ', η) –where κ' is also an incident– form a directed set (see after Definitions 4.4 and 4.5):

7.2 Proposition Let $\kappa_1\eta_1 = \kappa_2\eta_2$. Then there exist κ, ε_1 and ε_2 such that

$$\kappa_1 = \kappa\varepsilon_1, \;\; \kappa_2 = \kappa\varepsilon_2 \;\text{ and }\; \varepsilon_1\eta_1 = \varepsilon_2\eta_2. \qquad \blacksquare$$

Note that this result is only concerned with the statics of incidents.

7.3 Examples (1) It is easy to see that in general, in any action structure A, the entire set A of actions is an incident-set. But we look for smaller incident-sets, because we wish to restrict α in the definition of simulation (6.13) to range over an incident-set. These smaller incident-sets can differ strikingly in different cases.

In both $\mathsf{EXP}[X]$ and $\mathsf{SCCS}[X]$ (Sections 3.3 and 3.5) it turns out that the inactive actions constitute an incident-set. (This appears to be mainly because the reaction relation is confluent in both cases.) For $\mathsf{SCCS}[X]$ these are just the actions $(\vec{x})\,(\boldsymbol{\nu}\vec{y})\,S\,\langle\vec{z}\rangle$ in which S contains no complementary pair $\{x, \overline{x}\}$. A smaller incident-set K consists of the inactive actions which also have no restricted particles; that is, if $v \in S$ or $\overline{v} \in S$ then $v \notin \vec{y}$. This is a more appropriate incident-set, since no restricted particle in an action α can ever participate in a reaction of $\varphi(\alpha)$, for any preaction φ. For this K we conjecture that the K-bisimilarity defined in 7.5 below coincides with the original strong congruence for SCCS [14].

7.4 Examples (2) The case of $\mathsf{CCS}[X]$ is interesting. Recall from 3.6 that its reaction rule replaces a complementary pair of particles by the special particle τ, instead of just deleting them. What difference does this make to the possible choice of incident-set? We may take exactly the inactive actions as an incident-set, as for $\mathsf{SCCS}[X]$; this allows an incident to consist of many simultaneous interactions. But due to the presence of the special particle τ we can make the transitions $P \xrightarrow{\kappa} P'$ $(\kappa \in K)$ correspond much more closely to the original transitions of CCS [15] which were labelled by a single particle. We outline how to achieve this.

We have referred to the particle x as *positive* and $\overline{x}$ as *negative*; for now, let us say that τ is *both* positive *and* negative. Define the *weight* of an action to be the pair (p, n), where p (resp. n) is the number of its positive (resp. negative) particles; so each τ counts both in p and in n. Then reaction in $\mathsf{CCS}[X]$ preserves the weight of an action; it follows quite easily from this that the set of inactive actions whose weight does not exceed some fixed bound $(p_{\text{max}}, n_{\text{max}})$ constitutes an incident-set. In particular those with weight at most $(1, 1)$ constitute an incident-set; these are the inactive actions with particle-sets $\emptyset$, $\{\tau\}$, $\{x\}$, $\{\overline{x}\}$ or $\{x, \overline{y}\}$ for $x \neq y$. Again, we get a more appropriate incident-set K if we also exclude actions containing restricted particles. This K is in fact the smallest incident-set containing the action $\{\tau\}$ (see 7.14 below), and the corresponding K-bisimilarity is close to the original strong congruence for CCS; but it is a *stronger* congruence, since it is also preserved by instantiation of port-names. [3]

In our examples so far the natural incidents have been inactive. In general, and especially in the case of the π-calculus and Petri nets, the situation is more delicate; in these cases the inactive actions do not constitute an incident-set, and there are important incident-sets which contain active actions. We leave the details to future writing.

We are now ready to introduce a natural class of preorders and equivalences, one for each incident-set. The properties of an incident-set will ensure that they are congruential.

7.5 Definition Let K be an incident-set. A binary relation $\mathcal{S}$ over processes is a K-*simulation* if, whenever $(P, Q) \in \mathcal{S}$ and $P \xrightarrow{\kappa} P'$, then there exist κ', ε and Q' such that

$$\kappa = \kappa'\varepsilon,\ \ Q \xrightarrow{\kappa'} Q' \ \text{ and } \ (\varepsilon P', Q') \in \mathcal{S}.$$

If $\mathcal{S}$ is also symmetric, then it is called a K-*bisimulation*.

If (P, Q) is in some K-simulation we say that Q K-*simulates* P, and write $P \lesssim_K Q$. If (P, Q) is in some K-bisimulation we say that P and Q are K-*bisimilar*, and write $P \sim_K Q$. ∎

To establish that $P \lesssim_K Q$, or that $P \sim_K Q$, it is convenient to have a more generous class of relations than K-simulations. The following notion is useful:

7.6 Definition If $\mathcal{S}$ is a relation over processes, then $\mathcal{S}^\circ$, the *composition closure* of $\mathcal{S}$, is defined by

$$\mathcal{S}^\circ \stackrel{\text{def}}{=} \{(\alpha P, \alpha Q) \mid \alpha \text{ an action and } (P, Q) \in \mathcal{S}\}.$$

$\mathcal{S}$ is a K-*simulation up to composition* if, whenever $(P, Q) \in \mathcal{S}$ and $P \xrightarrow{\kappa} P'$, then there exist κ', ε, Q' such that

$$\kappa = \kappa'\varepsilon,\ \ Q \xrightarrow{\kappa'} Q' \ \text{ and } (\varepsilon P', Q') \in \mathcal{S}^\circ.$$

If $\mathcal{S}$ is also symmetric then it is called a K-*bisimulation up to composition*. ∎

The importance of this definition is that, to establish (bi)similarity, we only have to exhibit a (bi)simulation up to composition. To be precise:

7.7 Proposition Let $\mathcal{S}$ be a K-simulation up to composition. Then $\mathcal{S}^\circ$ is a K-simulation. Hence if $(P, Q) \in \mathcal{S}$ then $P \lesssim_K Q$, and if in addition $\mathcal{S}$ is symmetric then $P \sim_K Q$.

[3] In the standard presentation of CCS [15] instantiation of port-names is not admitted; in fact the strong congruence $a \bullet \mathbf{0} \mid \overline{b} \bullet \mathbf{0} \sim a \bullet \overline{b} \bullet \mathbf{0} + \overline{b} \bullet a \bullet \mathbf{0}$ holds if $a \neq b$, but it fails when b is instantiated to a.

Proof It is enough to show the first part. Let $\mathcal{S}$ be a K-simulation up to composition and $(\alpha P, \alpha Q)$ be a typical member of $\mathcal{S}^\circ$ with $(P, Q) \in \mathcal{S}$. Let $\alpha P \xrightarrow{\kappa} P'$. Then By 7.1 there exist λ, η and P'' such that

$$P \xrightarrow{\lambda} P'', \ \ \alpha\lambda \searrow \kappa\eta \ \text{ and } \ \eta P' \equiv P''.$$

Since $(P, Q) \in \mathcal{S}$, there exist λ', ε and Q'' such that

$$Q \xrightarrow{\lambda'} Q'', \ \ \lambda'\varepsilon = \lambda \ \text{ and } \ (\varepsilon P'', Q'') \in \mathcal{S}^\circ.$$

It follows from Proposition 6.12 that $\alpha Q \xrightarrow{\alpha\lambda'} Q''$. But $\alpha\lambda'\varepsilon \searrow \kappa\eta$ and η is inert, so there exists κ' such that

$$\alpha\lambda' \searrow \kappa' \ \text{ and } \ \kappa'\varepsilon = \kappa\eta.$$

Now from Proposition 7.2 there exist κ'', ε' and η' such that

$$\kappa' = \kappa''\varepsilon', \ \ \kappa = \kappa''\eta' \ \text{ and } \ \varepsilon'\varepsilon = \eta'\eta.$$

So again from Proposition 6.12 we have $\alpha Q \xrightarrow{\kappa'} Q''$, hence

$$\alpha Q \xrightarrow{\kappa''} \varepsilon' Q'',$$

and since $\kappa = \kappa''\eta'$ it only remains to show that $(\eta' P', \varepsilon' Q'') \in \mathcal{S}^\circ$. But

$$(\eta' P', \varepsilon' Q'') \equiv (\eta'\eta P'', \varepsilon' Q'') \equiv (\varepsilon'\varepsilon P'', \varepsilon' Q'')$$

and we are done, since $\mathcal{S}^\circ$ is composition-closed and $(\varepsilon P'', Q'') \in \mathcal{S}^\circ$. ■

As a simple corollary, we find that $\lesssim$ and $\sim$ are preserved by composition:

7.8 Proposition $P \lesssim_K Q$ implies $\alpha P \lesssim_K \alpha Q$, and $P \sim_K Q$ implies $\alpha P \sim_K \alpha Q$.

Proof If $P \lesssim_K Q$ then (P, Q) is in some K-simulation $\mathcal{S}$, which is *a fortiori* a K-simulation up to composition. But $(\alpha P, \alpha Q) \in \mathcal{S}^\circ$ which is itself a K-simulation by the Proposition. Hence $\alpha P \lesssim_K \alpha Q$. ■

With the help of this we can now establish

7.9 Proposition $\lesssim_K$ is a preorder and $\sim_K$ is an equivalence.

Proof It will be enough to show that $\lesssim_K$ (hence similarly $\sim_K$) is transitive. For this, we show that $\mathcal{S} = \{(P, R) \mid \exists Q.\ P \lesssim_K Q \lesssim_K R\}$ is a K-simulation. So let $P \lesssim_K Q \lesssim_K R$ and $P \xrightarrow{\kappa} P'$. It is enough to find κ'', ε'' and R' for which $\kappa = \kappa''\varepsilon''$, $R \xrightarrow{\kappa''} R'$ and $(\varepsilon'' P', R') \in \mathcal{S}$.

First, there exist κ', ε and Q' for which

$$\kappa = \kappa'\varepsilon, \ \ Q \xrightarrow{\kappa'} Q' \ \text{ and } \ \varepsilon P' \lesssim_K Q'.$$

Further, there exist κ'', ε' and R' for which

$$\kappa' = \kappa''\varepsilon', \;\; R \xrightarrow{\kappa''} R' \;\text{ and }\; \varepsilon' Q' \lesssim_K R'.$$

Now from Proposition 7.8 we know that $\varepsilon'\varepsilon P' \lesssim_K \varepsilon' Q'$; then the required conditions hold if we take $\varepsilon'' = \varepsilon'\varepsilon$. ■

We have now prepared the ground for the main result about incident-simulation.

7.10 Theorem $\lesssim_K$ is a precongruence, and $\sim_K$ is a congruence.

Proof (Outline) We have already shown in Proposition 7.8 that composition preserves these relations. We must now do the same for all other process constructions. In many cases this amounts to proving that an appropriate relation is a (bi)simulation up to composition; the detailed reasoning often follows the pattern of the proof of Prop 7.7.

For example, to show that product preserves $\lesssim_K$ it is be enough to show that

$$\mathcal{S} \stackrel{\text{def}}{=} \{(P \times R,\; Q \times R) \mid P \lesssim_K Q\}$$

is a K-simulation up to composition. For replication one shows that

$$\mathcal{S} \stackrel{\text{def}}{=} \{(R \times\, !P,\; S \times\, !Q) \mid R \lesssim_K S,\; P \lesssim_K Q\}$$

is a K-simulation up to composition, and hence so also is $\mathcal{S} \cup \{(\,!P,\;\; !Q) \mid P \lesssim_K Q\}$ by a similar argument.

The details will be found in Appendix B. ■

Before leaving the topic of K-simulation, we note that that when every action has a pure decomposition, then the definition of pure simulation (6.15), and its essential equivalence to the original notion proved in 6.16, can be extended in an obvious way to a notion of *pure K-simulation* and a similar equivalence.

7.11 Discussion We have achieved our main technical goal as far as processes are concerned, i.e. a uniform way of defining bisimilarity which is guaranteed to be congruential. But what is the variety of incident-sets K? Recall that an incident-set is relative to a given effect structure E. It turns out that, under a modest condition, an incident-set must include the whole of E:

Exercise Let K be an incident-set. Assume that for each $m \in M$ there is a member of K with target arity m. Show that $E \subseteq K$. (*Hint*: use inertia.) ■

Is there in some sense a best incident-set? The smallest one, the empty set, is not at all interesting. We have an interest in incident-sets which contain certain given actions; recall the choice for CCS in 7.4. So we finish this section by showing that given any set B of actions, under a mild condition there is a smallest incident-set which includes B. Since $\sim_K$ increases as K decreases, this appears to give the most generous congruential bisimilarity in which two bisimilar processes certainly "agree" with respect to the incidents in B.

For the rest of the section we continue to keep the action structure (A, E) fixed. We shall also assume that $\searrow$ in A is generated by $\overset{1}{\searrow}$ so that $\searrow \;=\; (\overset{1}{\searrow})^*$.

7.12 Definition A set $S \subseteq A$ *convergent* if no infinite reaction sequence

$$\alpha_0 \overset{1}{\searrow} \alpha_1 \overset{1}{\searrow} \alpha_2 \overset{1}{\searrow} \cdots$$

has infinitely many members in S. ■

Now A may not itself be convergent, but it may possess a convergent incident-set. Actually, in all our examples except for Petri nets, A itself is convergent.

If there were a smallest incident-set including B, it should be something like the intersection of all incident-sets including B. Unfortunately this class does not seem to be closed under intersection, in general. But . . .

7.13 Proposition Let $\mathcal{C}$ be any set of incident-sets containing a convergent member. Then $\bigcap \mathcal{C}$ is also an incident-set.

Proof Clearly $\bigcap \mathcal{C}$ satisfies 7.1(1), so we must show that it also satisfies 7.1(2). Suppose not. Then for some preaction φ, some $\kappa \in \bigcap \mathcal{C}$ and some $\beta \in A$ we have

$$\phi(\beta) \searrow \kappa \text{ but there is no } \lambda \in \bigcap \mathcal{C} \text{ for which } \beta \searrow \lambda \text{ and } \varphi(\lambda) \searrow \kappa.$$

Take any $K \in \mathcal{C}$. We get a contradiction by finding a reaction sequence

$$\beta \;=\; \beta_0 \overset{+}{\searrow} \beta_1 \overset{+}{\searrow} \cdots$$

which passes infinitely often through K. (Here $\overset{+}{\searrow} \;=\; (\overset{1}{\searrow})^+$, the transitive closure of $\overset{1}{\searrow}$.) We also ensure that $\varphi(\beta_i) \searrow \kappa$ for all i.

Suppose $\beta_0, \beta_1, \ldots, \beta_j$ are already determined. By assumption, $\beta_j \notin \bigcap \mathcal{C}$. There are two cases:

1. $\beta_j \in K$. Then for some other $K' \in \mathcal{C}$, $\beta_j \notin K'$. Then choose $\beta_{j+1} \in K'$ such that $\beta_j \searrow \beta_{j+1}$ and $\varphi(\beta_{j+1}) \searrow \kappa$.

2. $\beta_j \notin K$. Then choose $\beta_{j+1} \in K$ with the same conditions.

Clearly adjacent members of this sequence are distinct, so $\beta_j \overset{+}{\searrow} \beta_{j+1}$ as required; also the sequence passes infinitely often through K by construction. ■

7.14 Corollary Let $B \subseteq A$. If there is any convergent incident-set including B, then there is a smallest incident-set including B.

Proof Take $\mathcal{C}$ in Prop 7.13 to be all incident-sets which include B. ■

7.15 Discussion The first topic of this section was simulations and bisimulations defined in terms of incident-sets, the main result being that they are congruential. We conjectured that we also recover in this way the original strong bisimilarities of SCCS and CCS, though with a natural refinement in the latter case. It remains to be seen how many other familiar variants of (bi)similarity can thus be recovered.

The second topic was the structure of the class of incident-sets. The main result here, the existence of a smallest incident-set under certain conditions, does not depend at all upon process structures and the uniform way (defined in Section 5) in which they can be built from action structures. In fact the notion of incident-set appears to be intrinsic to action structure theory. The only part of its definition which is partly motivated by process structure is the notion of preaction, Definition 4.10. It is conceivable that a variation of this definition, extending or limiting the class of operations from which preactions can be built, may be appropriate for other uses of action structures. However, the force of Prop 4.13 was that the same notion of *effect structure* is gained whether we demand inertia (which is defined in terms of composition only) or *strong* inertia (which is defined in terms of preactions). This suggests that the current definition of preaction is a natural one, independently of its use in defining incident-sets and thereby obtaining congruential process bisimilarities.

Be this as it may, note that Corollary 7.14, asserting the existence of a smallest incident-set, does not depend at all upon the definition of preaction.

8 Related work and future directions

General models of concurrency This paper is one of many attempts to find a common framework, or central ideas, in concurrent computing. It would take a long essay to classify these attempts, or even to compare them all to this one. I shall mention only three alternative lines.

The *chemical abstract machine* (CHAM) of Berry and Boudol [5] has common aspects with action structures. Indeed, in particular cases such as the π-calculus action structure the actions, with a multiset of particles as their principal component, are close to what the natural CHAM for the π-calculus looks like. Crudely, one can see action structures as an attempt to make an algebra of CHAMs, with parametrisation. Other differences, e.g. in the dynamics, arise from treating reaction as algebraically as possible. Whatever the difference, the CHAM has been a guiding influence in the present work.

Meseguer proposes conditional rewriting logic [12] as a general framework for unifying the study of concurrency. Action structures are not about logic, but they should be compared with the categories, so-called $\mathcal{R}$-systems, which are models –in the technical sense– of a Meseguer rewrite theory. This is a rather general class, and contains the monoidal categories which represent individual Petri nets [13], already referred to in Section 3.8 above. One may contrast this work with action structures as follows: in an $\mathcal{R}$-system the carrier of the algebra consists of the computations of an individual system (e.g. Petri net), while in an action structure the carrier consists of the systems themselves. Direct comparison is therefore difficult; nonetheless it would be of interest to explore a combination of action structures with Meseguer's logical approach.

Recently Nielsen and Winskel have proposed asynchronous transition systems as a general model of concurrency [18]. These are essentially transition systems with additional structure recording the independence among events. This is one difference from action structures, whose definition makes no mention of independence among reactions, though independence can be examined in particular action structures – e.g. in Petri nets. But other differences are perhaps deeper. A good point to examine is the rôle played by morphisms. A morphism in the category of asynchronous transition systems describes, crudely speaking, how one process (e.g. Petri net) may implement or simulate another. In contrast a homomorphism of action structures describes how one *class* of agents, e.g. π-calculus processes, may implement another, e.g. arithmetic expressions. Both notions deserve study; once again comparison is difficult because the models work at different levels.

One may find this disparity among models bewildering. Are we doing something wrong? I think not; concurrent computing can be viewed in so many ways. This has long been apparent at the level of programming languages, and of concrete calculi; it is no surprise to find it so at the more abstract level of general models. Each model must be analysed deeply in its own terms, and be developed with awareness of the others; we must hope that this leads us to unifying concepts, but we should not expect the search to be quick and easy.

Further study of action structures This paper has introduced action structures, and it has devoted considerable space to an application of them: the definition of process structure over them, and the uniform study of congruence among the resulting processes. The aim was to show how action structures can inform familiar theories, so as to justify their further study.

Several studies of particular action structures remain to be done, or to be reported. I have developed action structures for the π-calculus far enough to understand the notions of effect, pure action and incident in that setting, and this work will be reported separately. Another important instance is Petri nets; it seems that there will be several interestingly related action structures for nets, and this direction remains largely unexplored.

More intrinsic to the concept of action structures is to examine what constructions exist over them. Here one hopes that a particular instance, such as that for CCS with value-passing, can not only be presented monolithically but also built from simpler action structures. This possibility was hinted in Section 3.7, where the uniform addition of ports to an action structure was discussed; that construction remains to be made precise.

What other possible constructions are there? This paper has explained process calculi by superposing process structure upon action structure; can they instead be presented as action structures themselves? This entails enriching an action structure by the addition of control operators such as those introduced in Section 5. In fact it is fairly easy to see how to extend an arbitrary action structure A in a standard way by adding new operators with their dynamics; it is not so easy to see what this extension does to the effect structure, and the other fine structure of A. The success of action structures as a mathematical model will depend upon how well this works out. If it works well then we may fruitfully reformulate existing process calculi as action structures, and the homomorphisms among action structures may become be the basis of a semantic theory.

References

[1] Aceto, L. and Hennessy, M.C., *Towards action-refinement in process algebras*, To appear in Journal of Information and Computation.

[2] Baeten, J.C.M. and Weijland, W.P., **Process Algebra**, Cambridge University Press 1990.

[3] Barr, M. and Wells, C., **Toposes, Triples and Theories**, Springer Verlag, 1985.

[4] Bellin, G. and Scott, P.J. (with introduction by S. Abramsky), *On the π-calculus and linear logic*, Submitted for publication, 1992.

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

[6] Best, E., Devillers, R. and Hall, J.G., *The box calculus: a new causal algebra with multi-label communication*, in **Advances in Petri Nets '92**, ed. G.Rozenberg, Lecture Notes in Computer Science, Springer Verlag, Vol.609, pp21–69, 1992.

[7] Cherkasova, L.A. and Kotov, V.E., *Descriptive and analytical process algebras*, in **Advances in Petri Nets '89**, ed. G Rozenberg, Lecture Notes in Computer Science, Springer Verlag, Vol 424, pp77–104, 1989.

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

[9] van Glabbeek, R. and Goltz, U., *Equivalence notions for concurrent systems and refinement of actions*, Proc. 4th Conference on Mathematical Foundations of Computer Science, Lecture Notes in Computer Science, Springer Verlag, Vol 379, pp237–248, 1988.

[10] Gunter, C.A. and Scott, D.S., *Semantic domains*, in **Handbook of Theoretical Computer Science**, Vol A, pp633–674, Elsevier, 1990.

[11] Hoare, C.A.R., **Communicating Sequential Processes**, Prentice Hall, 1985.

[12] Meseguer, J., *Conditional rewriting logic as a unified model of concurrency*, Journal of Theoretical Computer Science, Vol 96, pp73–155, 1992.

[13] Meseguer, J. and Montanari, U., *Petri nets are monoids*, Journal of Information and Computation, Vol 88, pp105–155, 1990.

[14] Milner, R., *Calculi for synchrony and asynchrony*, Journal of Theoretical Computer Science Vol 25, pp267–310, 1983.

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

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

[17] Nielsen, M., Plotkin, G.D. and Winskel, G., *Petri nets, event structures and domains*, Journal of Theoretical Computer Science, Vol 13, No 1, 1981.

[18] Nielsen, M., and Winskel, G., *Models for concurrency*, to appear in Volume IV of the **Handbook of Logic and the Foundations of Computer Science**, ed. D.Gabbay, Oxford University Press, 1993.

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

[20] Winskel, G., *Petri nets, algebras, morphisms and compositionality*, Journal of Information and Computation, March 1987.

APPENDIX A From π-calculus to action structures

As mentioned in Section 1, part of the motivation for action structures came from the π-calculus. Here we show in particular how the wish for transitions composed of many single interactions leads to the three classes of operation in action structures: *product*, *composition* and *abstraction*. What little we need of the π-calculus is presented here. This appendix, as its title implies, assumes no prior knowledge of action structures.

CCS has the prefix construction $a(y) \bullet P$, meaning "receive any value y at port a and proceed to P". (The variable y is bound.) The π-calculus has the same construction; but here values, variables and ports are all the same thing: names. Therefore ports can be bound, or parametric; this is what allows the π-calculus to model dynamically changing structure.

Consider the parallel composition

$$S \;\equiv\; x(y) \bullet y(u) \bullet P \mid \overline{x}z \bullet Q \mid \overline{z}v \bullet R;$$

here a communication at port x can occur, yielding the transition $S \longrightarrow S'$ where

$$S' \;\equiv\; z(u) \bullet P \mid Q \mid \overline{z}v \bullet R.$$

(For simplicity, we assume y not free in P.) At this point a communication at z can occur, which could not occur before; this yields the transition $S' \longrightarrow S''$ where

$$S'' \;\equiv\; \{v/u\}P \mid Q \mid R.$$

We can say that the communication at x has *enabled* the one at z, through the instantiation of y to z. This enablement is a phenomenon of the π-calculus, not of CCS.

Now the prefixing operator "$\bullet$" imposes *precedence*, i.e. order of occurrence, among actions in both CCS and the π-calculus. But we can ask whether it is so essential in the latter, since the flow of data (names) in communications, which we are calling *enablement*, imposes some –though less– order of occurrence. Another possibility is to admit both precedence and enablement to the calculus.

Let us look at one way of doing this. Use π to stand for a *particle*, i.e. a simple prefix of the form $x(y)$ or $\overline{x}y$, and consider two cases of the construction $\pi_1 \bullet \pi_2 \bullet R$:

$$\begin{aligned} P &\;\equiv\; x(y) \bullet \overline{z}w \bullet P' \\ Q &\;\equiv\; x(y) \bullet \overline{y}w \bullet Q'. \end{aligned}$$

In both cases π_1 *precedes* π_2; but only in Q does π_1 *enable* π_2. So let us allow composite prefixes, as in

$$\begin{aligned} P &\;\equiv\; (x(y), \overline{z}w) \bullet P' \\ Q &\;\equiv\; (x(y), \overline{y}w) \bullet Q' \end{aligned} \tag{1}$$

where the binding of a variable like y is still to the right, but the commutation $\pi, \pi' \equiv \pi', \pi$ is allowed provided it does not make a free variable bound or vice versa. Thus the comma represents enablement.

Such a step, even abandoning "**.**" entirely in favour of enablement, has been suggested by several people. Joachim Parrow, who with David Walker and myself first introduced the π-calculus, mooted the idea some years ago. Samson Abramsky found that proof terms for linear logic could be written in π-calculus more accurately with enablement; Gianluigi Bellin and Philip Scott [4] have pursued this idea further. I am also grateful to Bob Constable for pointing out a natural property of concurrent logic programming which can be loosely paraphrased as follows: A variable becomes accessible for a certain purpose just when it is sufficiently instantiated for that purpose. We may think of instantiation as provision of data; in general terms, the idea of data flow as a form of control is an old and natural one, likely to reappear in different contexts.

We now carry our π-calculus example forward a short way, leading to the operations of action structures. Let us suppose that we extend the π-calculus by allowing a prefix to be any sequence of particles, allowing commutations as described above. We may call this the *synchronous* π-calculus, since it corresponds to the π-calculus just as synchronous CCS [14] corresponds to CCS. Now recall the multiple prefixes in (1); they are

$$\alpha = (x(y), \overline{z}w) \text{ and } \beta = (x(y), \overline{y}w).$$

It is natural to think of α as the juxtaposition, or *product*, of its parts:

$$\alpha = \pi_1 \times \pi_2, \text{ where } \pi_1 = x(y) \text{ and } \pi_2 = \overline{z}w.$$

On the other hand β is not simply a juxtaposition, since one particle enables the other; so we prefer to think of it as a *composition* of its parts:

$$(?) \qquad \beta = \pi_1 \circ \pi_2', \text{ where } \pi_1 = x(y) \text{ and } \pi_2' = \overline{y}w.$$

But this formulation pays little attention to how π_1 binds y in π_2', which is the essence of enablement. The most natural thing, rather than introduce a family of binding operators $x(y)$ –one for each pair of names x and y– is to introduce an singly-indexed *abstraction* operator $\mathbf{ab}_y$. This allows a more accurate expression for β:

$$\beta = \beta_1 \circ \beta_2, \text{ where } \beta_1 = \mathbf{in}_1 x \text{ and } \beta_2 = \mathbf{ab}_y \pi_2'.$$

Here $\mathbf{in}_1 x$, for any name x, is an action which inputs a single name at port x; then the composition ensures that this single name is bound to y in π_2'.

There are some loose ends to tie up to make this presentation formally correct. The general framework is set up in Section 2 of the paper, making the abstractors $\mathbf{ab}_y$ functorial over an action structure considered as a category; the detailed study of action structures for the π-calculus will be done in another paper. The point of the short exercise above was to illustrate the rôle played in an action structure by its three operations: *product*, *composition* and *abstraction*. The dynamical ingredient of action structures, *reaction*, is what allows the particles in the action $\gamma = (x(y), \overline{x}z)$ to annihilate one another, leaving in their place the substitution $\{z/y\}$, which is itself treated as a degenerate kind of action: an *effect*.

APPENDIX B Proof of theorem 7.10

Theorem 7.10 asserts that the K-simulation preorder and the K-bisimulation equivalence are congruential. To prepare for it we we shall need the following development of Proposition 6.9, which shows how an incident-transition of a composite process must arise from incident-transitions of its components:

Proposition Let $P \xrightarrow{\kappa} P'$. Then

1. If P is βQ then for some λ and η
$$Q \xrightarrow{\lambda} Q',\ \ \beta\lambda \searrow \kappa\eta \ \text{ and } \ \eta Q' \equiv P'.$$
2. If P is $Q \times R$, then for some λ, μ, η, Q' and R'
$$Q \xrightarrow{\lambda} Q',\ \ R \xrightarrow{\mu} R',\ \ \lambda \times \mu \searrow \kappa\eta \ \text{ and } \ \eta(Q' \times R') \equiv P'.$$
3. If P is $\mathbf{ab}_x Q$ then for some λ, η and Q'
$$Q \xrightarrow{\lambda} Q',\ \ \mathbf{ab}_x\lambda \searrow \kappa\eta \ \text{ and } \ \eta\, \mathbf{ab}_x Q' \equiv P'.$$

Proof It will be enough to do clause 2. We have $P \searrow \kappa\eta \bullet P''$ for some η and P'' with $\eta P'' \equiv P'$. So by 6.9(2), there exist β, γ, Q' and R' such that

$$Q \searrow_c \beta \bullet Q',\ \ R \searrow_c \gamma \bullet R',\ \ \beta \times \gamma \searrow \kappa\eta \ \text{ and } \ Q' \times R' \equiv P''.$$

The rest follows by applying the second clause of Definition 7.1, first to the preaction $(-) \times \gamma$ to find λ, then to the preaction $\lambda \times (-)$ to find μ. ∎

We are now ready for the theorem.

Theorem $\lesssim_K$ is a precongruence, and $\sim_K$ is a congruence.

Proof It was shown in Proposition 7.8 that composition preserves these relations. We must now do the same for all other process constructions. In many cases this amounts to proving that an appropriate relation is a (bi)simulation up to composition; the detailed reasoning often follows the pattern of the proof of Prop 7.7.

To show that product preserves $\lesssim_K$ it will be enough to show that

$$\mathcal{S} \stackrel{\text{def}}{=} \{(P \times R,\ Q \times R) \mid P \lesssim_K Q\}$$

is a K-simulation up to composition.

Let $P \times R \xrightarrow{\kappa} S$. Assuming $P \lesssim_K Q$, we shall find κ'', η' and T such that $Q \times R \xrightarrow{\kappa''} T$, $\kappa''\eta' = \kappa$ and $(\eta' S, T) \in \mathcal{S}^{\circ}$.

First, from 7.3(2) there exist λ, μ, η, P' and R' for which

$$P \xrightarrow{\lambda} P', \;\; R \xrightarrow{\mu} R', \;\; \lambda\times\mu \searrow \kappa\eta \;\text{ and }\; \eta(P'\times R') \equiv S.$$

Now $P \lesssim_K Q$, so there exist λ', ε and Q' such that

$$Q \xrightarrow{\lambda'} Q', \;\; \lambda'\varepsilon = \lambda \;\text{ and }\; \varepsilon P' \lesssim_K Q'.$$

We deduce $(\lambda'\times\mu)(\varepsilon\times\mathbf{1}) = \lambda\times\mu \searrow \kappa\eta$; but $\varepsilon\times\mathbf{1}$ is inert so there exists κ' for which

$$\lambda'\times\mu \searrow \kappa' \;\text{ and }\; \kappa'(\varepsilon\times\mathbf{1}) = \kappa\eta.$$

From the last equation, by Proposition 7.2 there exist κ'', η'' and η' for which

$$\kappa' = \kappa''\eta'', \;\; \kappa = \kappa''\eta' \;\text{ and }\; \eta''(\varepsilon\times\mathbf{1}) = \eta'\eta.$$

From the above transitions for Q and R, using Proposition 6.12 we deduce

$$Q\times R \xrightarrow{\kappa''} \eta''(Q'\times R'),$$

so we shall take T to be the result of this transition. Then from the above equations we find

$$(\eta' S, T) \;\equiv\; (\eta''(\varepsilon P'\times R'),\; \eta''(Q'\times R'))$$

which is in $\mathcal{S}^\circ$ since $\varepsilon P' \lesssim_K Q'$.

This completes the proof that product preserves incident-simulation, and a similar proof works for abstraction. It remains to show that the control operators preserve it. We consider only commitment and replication; summation and delay are no harder.

For commitment, let $P \lesssim_K Q$ and consider $(\bullet P, \;\bullet Q)$. Let $\bullet P \xrightarrow{\kappa} P'$; we shall show $\bullet Q \xrightarrow{\kappa} Q'$ where $P' \lesssim_K Q'$.

We have

$$\bullet P \searrow \kappa\eta \bullet P'' \;\text{ where }\; \eta P'' \equiv P'.$$

By Proposition 6.8 it follows that for some γ

$$\bullet P \searrow_{\mathsf{c}} \gamma \bullet P'' \;\text{ and }\; \gamma \searrow \kappa\eta.$$

But $\bullet P \searrow_{\mathsf{c}}^{1} \ldots$ is impossible, so $\gamma = \mathbf{1}$ and $\bullet P \equiv \;\bullet P''$, whence by Proposition 5.10 $P \equiv P''$. But $\mathbf{1}$ is inactive, so $\kappa\eta = \mathbf{1}$. Now $\bullet Q \xrightarrow{\mathbf{1}} Q$, i.e. $\bullet Q \xrightarrow{\kappa\eta} Q$, so applying Proposition 6.12(5) we get

$$\bullet Q \xrightarrow{\kappa} \eta Q \;\text{ and }\; P' \equiv \eta P \lesssim_K \eta Q.$$

This completes the proof that committal preserves incident-simulation.

For replication, the main task is to show that

$$\mathcal{S} \stackrel{\text{def}}{=} \{(R\times\, !P,\; S\times\, !Q) \mid R \lesssim_K S,\; P \lesssim_K Q\}$$

is a simulation up to composition – and hence so also is $\mathcal{S} \cup \{(\,!P,\ !Q) \mid P \lesssim_K Q\}$ by a similar argument.

So assume $R \lesssim_K S$ and $P \lesssim_K Q$, and let $R\times\ !P \xrightarrow{\kappa} U$. We shall find κ'', η' and V for which $S\times\ !Q \xrightarrow{\kappa''} V$, $\kappa = \kappa''\eta'$ and $(\eta' U, V) \in \mathcal{S}^\circ$. W.l.o.g. we can assume that (for some $n \geq 0$) the first $n{+}1$ atomic reductions of the transition are an unfolding of $!P$ into n copies of P followed by a commitment:

$$R\times\ !P \searrow^1_{\mathsf{c}} R\times P\times\ !P \searrow^1_{\mathsf{c}} \cdots \searrow^1_{\mathsf{c}} R\times P^n\times\ !P \searrow^1_{\mathsf{c}} R\times P^n\times \bullet\ !P,$$

where P^n means $\overbrace{P\times\cdots\times P}^{n\ times}$ (or the term is absent if $n = 0$).

So $R\times P^n\times \bullet\ !P \xrightarrow{\kappa} U$; then by Propositions 7.3(2) there exist λ, μ, η, R' and P' such that

$$R\times P^n \xrightarrow{\lambda} R',\quad !P \xrightarrow{\mu} P',\quad \lambda\times\mu \searrow \kappa\eta \ \text{ and }\ \eta(R'\times P'') \equiv U. \qquad (2)$$

Considering the first of these transitions, we know that $R\times P^n \lesssim_K S\times Q^n$, so there exist λ', ε and S' for which

$$S\times Q^n \xrightarrow{\lambda'} S',\quad \lambda'\varepsilon = \lambda \ \text{ and }\ \varepsilon R' \lesssim_K S'. \qquad (3)$$

Since $(\lambda'\times\mu)(\varepsilon\times\mathbf{1}) \searrow \kappa\eta$ and $\varepsilon\times\mathbf{1}$ is inert, there exists κ' for which

$$\lambda'\times\mu \searrow \kappa' \ \text{ and }\ \kappa'(\varepsilon\times\mathbf{1}) = \kappa\eta,$$

whence by Proposition 7.2 there also exist κ'', η'' and η' for which

$$\kappa' = \kappa''\eta'',\quad \kappa = \kappa''\eta' \ \text{ and }\ \eta''(\varepsilon\times\mathbf{1}) = \eta'\eta.$$

Considering the second transition of (2), by reasoning similar to the argument for commitment above we find that there exists θ such that

$$\mu\theta = \mathbf{1} \ \text{ and }\ P' \equiv \theta\,!P.$$

Now let us consider a transition for $S\times\ !Q$; from (3) we easily find that

$$S\times\ !Q \xrightarrow{\lambda'\times\mathbf{1}} S'\times\ !Q;$$

also $\lambda'\times\mathbf{1} = (\lambda'\times\mu)(\mathbf{1}\times\theta) \searrow \kappa''\eta''(\mathbf{1}\times\theta)$, so by Prop 6.12(4),(5) we get

$$S\times\ !Q \xrightarrow{\kappa''} \eta''(\mathbf{1}\times\theta)(S'\times\ !Q).$$

Take V to be the result of this transition. Since $\kappa = \kappa''\eta'$, it only remains to prove that $(\eta' U, V) \in \mathcal{S}^\circ$. But

$$\eta' U \equiv \eta'\eta(R'\times P') \equiv \eta''(\varepsilon\times\mathbf{1})(R'\times\theta\,!P) \equiv \eta''(\mathbf{1}\times\theta)(\varepsilon R'\times\ !P),$$

and we are done since $\varepsilon R' \lesssim_K S'$.

This completes the proof that $\lesssim_K$ is a precongruence; an analogous proof shows that $\sim_K$ is a congruence. ■