Action Calculi III : Higher-order Calculi

Robin Milner
Edinburgh, November 1993

1 Introduction

This paper extends an arbitrary action calculus [11] to higher order, in a way which parallels the extension of the π-calculus [13] to higher order by Sangiorgi [18]. Action calculi are a subclass of action structures [10], which in turn are an enriched class of monoidal categories. Each action calculus $\mathsf{AC}(\mathcal{K})$ is distinguished by its set $\mathcal{K}$ of *controls* and their *reaction rules*. The notion of action calculus is derived partly from the chemical abstract machine of Berry and Boudol [5]. To each control $K \in \mathcal{K}$ corresponds a type of *molecule*; the main interest of action calculi is that they are an algebra of *molecular forms*, which provide a non-free but algebraically tractable form of syntax for modelling concurrent systems.

Particular action calculi exist which are generalisations of Petri nets, and of the π-calculus; see [11, 12]. Thus the extension to higher order adds power to existing models of concurrency. This paper demonstrates that the algebraic treatment of action calculi, including the notion of molecular form, extends smoothly to higher-order calculi which generalise the λ-calculus.

Sections 1–6 contain a review of action calculi, sufficient for present purposes. The first step towards higher order is taken in Section 7 by the addition of two new controls to any action calculus $\mathsf{AC}(\mathcal{K})$. These new controls are essentially the "curry" and "apply" operations of a cartesian-closed category for the typed λ-calculus. The reaction rules for these new controls, σ-reduction and β-reduction, are closely related to reduction in the $\lambda\sigma$-calculus [1], a λ-calculus with explicit substitutions; so far they are treated on the same footing as the reaction rules pertaining to the originally controls $\mathcal{K}$.

The next step, in Section 8, is to replace σ-reduction by its induced congruence relation, σ-conversion. This leads essentially to the familiar λ-calculus, but enriched by the presence of $\mathcal{K}$-molecules and the (possibly non-confluent) $\mathcal{K}$-reaction relation.

In Section 9 we first determine a condition, called *stability*, under which β-reduction commutes with $\mathcal{K}$-reaction. This justifies our final step towards an equational higher-order calculus, which is to replace β-reduction by its induced congruence, β-conversion. Taking the quotient by this congruence yields our higher-order action calculus which we call $\mathsf{HAC}(\mathcal{K})$. When $\mathcal{K}$ is a control-set suitable for the π-calculus, we end up with an enriched form of Sangiorgi's higher-order π-calculus.

Section 10 examines the possibility of making $\mathsf{HAC}(\mathcal{K})$ into a cartesian-closed category. If $\mathcal{K} = \emptyset$ then $\mathsf{HAC}(\mathcal{K})$ is nearly the standard cartesian-closed category for typed λ-calculus, but not quite. To reach it, a further quotient is needed, by an extra equation representing η-conversion. Indeed, we show that this quotient converts *any* higher-order action calculus into a cartesian-closed category; but in the case of some action calculi with non-trivial interactive dynamics, such as PIC (representing the π-calculus), it is inconsistent with the dynamic behaviour.

This result characterizes –in the categorical framework– the distinction between communicational behaviour on the one hand and purely functional computation on the other.

2 Arities and names

We assume a monoid $(M, \otimes, \epsilon)$ of objects, which are called *arities*. We shall use k, ℓ, m, n to range over M. Two simple examples of M are the natural numbers under addition, and sequences of basic sorts (e.g. INT, BOOL) under concatenation. We shall have more complex cases, but we shall want to assume of M that it *has primes*, i.e. that every arity factors uniquely into prime arities. More precisely, we say that $p \in M$ is *prime* if $p \neq \epsilon$, and $p = m_1 \otimes m_2$ implies that $m_1 = \epsilon$ or $m_2 = \epsilon$. Then we assume that every $m \in M$ is a product of primes, and that if $p_1 \otimes \cdots \otimes p_r = q_1 \otimes \cdots \otimes q_s$ where each p_i and q_j is prime, then $r = s$ and $p_i = q_i$ $(1 \leq i \leq r)$. Thus when M is sequences of basic sorts, the prime arities are the basic sorts INT, BOOL, ...; when M is the natural numbers under addition, 1 is the only prime.

Any action calculus presupposes an arity monoid M with primes, and has a set X of names. We shall use u, v, w, x, y, z to range over X. We assume that to each name $x \in X$ is associated a prime arity $p \in M$; we write $x : p$. We further assume that each prime arity is associated with infinitely many names. We shall write $\vec{x}$ for a vector $x_1 \cdots x_r$ of names; if $x_i : p_i$ for $1 \leq i \leq r$ then we define $|\vec{x}| \stackrel{\text{def}}{=} p_1 \otimes \cdots \otimes p_r$.

3 Actions and controls

The principal ingredient of an action calculus is its *actions*. Assuming an arity monoid M as above, each action a of the calculus has a *source* arity and a *target* arity in M; if these are m and n we write $a : m \to n$. In fact an action calculus is a special kind of category whose objects are arities and whose arrows are actions.

Each action calculus $\mathsf{AC}(\mathcal{K})$ is determined by a set $\mathcal{K}$ of *controls*, together with their reaction rules. Each control $K \in \mathcal{K}$ allows the construction of an action $K(a_1, \ldots, a_r)$ from actions $a_1, \ldots, a_r$, subject to an arity rule which may be presented in the form

$$\frac{a_1 : m_1 \to n_1 \quad \cdots \quad a_r : m_r \to n_r}{K(a_1, \ldots, a_r) : m \to n} \ (\chi)$$

where the side-condition χ may constrain the values of r, m_i, n_i, m, n; this determines the arity of the construction $K(a_1, \ldots, a_r)$ in terms of those of its arguments. We often write the construction as $K\vec{a}$; we may refer to it as a K-*action*, or $\mathcal{K}$-*action*, or *control action*.

There are two forms of presentation for action calculi. In the Section 4 we give the first form, which we call the *molecular* form; it is direct, in that it needs no knowledge of the algebraic theory of action structures. Then in Section 5 we shall demonstrate that action calculiobey simple algebraic laws. In fact they are a subclass of action structures; they satisfy the eight standard equations of action structures, but they also satisfy four further equations called the *concrete axioms*. Thus in Section 6 we come to the second form of presentation of action calculi; namely, that $\mathsf{AC}(\mathcal{K})$ is the free interpretation of these twelve axioms generated by $\mathcal{K}$.

4 Action calculi: molecular form

Let arities M, names X and a control set $\mathcal{K}$ be given. We proceed to define the actions of $\mathsf{AC}(\mathcal{K})$ inductively, at the same time as its so-called molecules.

An *action* $a : m \to n$ takes the form

$$(\vec{x})\,\mu_1,\,\ldots,\,\mu_h\,\langle\vec{u}\rangle \qquad (|\vec{x}| = m,\ |\vec{u}| = n);$$

the sequence $\mu_1,\,\ldots,\,\mu_h$ is called the *body*, and consists of molecules. We shall let a, b, c stand for actions.

In turn, a *molecule* μ takes the form

$$\langle\vec{v}\rangle\, K\vec{b}\,(\vec{y}) \qquad (|\vec{v}| = k, |\vec{y}| = \ell)$$

where $K\vec{b} : k \to \ell$ is a $\mathcal{K}$-action. We shall let κ, λ, μ stand for molecules.

The *subactions* of an action a comprise a itself and the subactions of each action in a molecule of a. We sometimes use contextual notation $a = \mathcal{C}[b]$ to indicate an occurrence of b as a subaction of a. In the action $a = (\vec{x})\,\vec{\mu}\,\langle\vec{u}\rangle$ the *imported* names $\vec{x}$ (which must be distinct) are binding; the (distinct) names $\vec{y}$ of the molecule $\mu = \langle\vec{v}\rangle\,K\vec{b}\,(\vec{y})$ are also binding. The scope of each binding extends to its right, to the end of the smallest subaction containing it. Thus if μ is a molecule of a then the scope of $\vec{y}$ extends to the end of a, while the scope of any name bound within a member of $\vec{b}$ is confined to that subaction; however, any name free in $\vec{b}$ is also free in μ.

Alpha conversion is allowed; using it, we adopt the convention that no bound name occurs outside its scope, either free or bound, in the context in which it is used. The body $\mu_1,\,\ldots,\,\mu_h$ of an action a is a *partial sequence*, which means that the commutation $\lambda, \mu = \mu, \lambda$ of a pair of neighbouring molecules is allowed provided that neither of them binds a name occurring in the other.

The simplest action calculus is of course $\mathsf{AC}(\emptyset)$. There are no molecules, so every action has an empty body and takes the simple form

$$(\vec{x})\,\langle\vec{u}\rangle\ .$$

One may think of actions in $\mathsf{AC}(\emptyset)$ as pure wiring; they are the scaffolding which in richer action calculi is used to connect molecules. Our uniform extension of action calculi to higher order will convert $\mathsf{AC}(\emptyset)$ into $\mathsf{HAC}(\emptyset)$, which turns out to be just the pure simply-typed λ-calculus.

As another example, consider the action calculus for the π-calculus, $\mathsf{PIC} = \mathsf{AC}(\boldsymbol{\nu}, \mathbf{in}, \mathbf{out})$, with natural numbers as arities. Its three controls

$$\boldsymbol{\nu} : 0 \to 1,\ \ \mathbf{in} : 1 \to m,\ \ \mathbf{out} : 1{+}m \to 0$$

are *constants*, since they take no action arguments. The corresponding molecules are of the form

$$\langle\,\rangle\boldsymbol{\nu}(x),\ \langle x\rangle\mathbf{in}(\vec{y}),\ \langle x\vec{z}\rangle\mathbf{out}(\,)\ ,$$

more traditionally written as

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

On the other hand, the action calculus $\mathsf{PIC}(\mathbf{box})$ has an additional control **box** which is not a constant; it allows the construction $\mathbf{box}\,a : 1 \to n$ for any $a : 1 \to n$. The corresponding molecule is $\langle x\rangle\,\mathbf{box}\,a\,(\vec{z})$, where $|\vec{z}| = n$. In the original π-calculus the corresponding construction is the input prefix $x(\vec{y}).P$ (which delays the performance of P), where a corresponds to $(\vec{y})P$. But note that the action calculus is richer, since it allows a to export results (named $\vec{z}$), which was not allowed for P.

This completes the molecular presentation of action calculi, except for the dynamics (the reaction relation) which we defer to Section 6.

5 Action calculi as action structures

We begin by reviewing the definition of action structures, as they were introduced in [10]. A *static action structure* over the name set X is a strict monoidal category equipped with an endofunctor $\mathbf{ab}_x$ for each $x \in X$. Its objects (arities) constitute a monoid $(M, \otimes, \epsilon)$, and its arrows are actions. The basic operations over actions are the identities $\mathbf{id}_m$, composition $\cdot$, tensor product $\otimes$ and abstractors $\mathbf{ab}_x$ $(x \in X)$. They obey the following *arity rules*:

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

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

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

The basic axioms of action structures are as follows:

Action structures – basic equations	
$a \cdot \mathbf{id} = a = \mathbf{id} \cdot a$	$a \cdot (b \cdot c) = (a \cdot b) \cdot c$
$a \otimes \mathbf{id}_\epsilon = a = \mathbf{id}_\epsilon \otimes a$	$a \otimes (b \otimes c) = (a \otimes b) \otimes c$
$\mathbf{id} \otimes \mathbf{id} = \mathbf{id}$	$(a \cdot b) \otimes (c \cdot d) = (a \otimes c) \cdot (b \otimes d)$
$\mathbf{ab}_x \mathbf{id} = \mathbf{id}$	$\mathbf{ab}_x(a \cdot b) = (\mathbf{ab}_x a) \cdot (\mathbf{ab}_x b)$.

A (*dynamic*) action structure also possesses a *reaction relation* $\searrow$, a pre-order which is preserved by composition, product and abstraction and such that $a \searrow a'$ only when a and a' have the same source and target arity. We also require the identities $\mathbf{id}_m$ to be minimal for $\searrow$.

We shall use homomorphisms and isomorphisms of action structures in this paper, as they were defined in [10]. A static homomorphism $\Phi : A \to B$ between two action structures A and B with the same name-set X (we need not consider the case where the name-sets of A and B differ) is a functor which also preserves product and abstraction. A static isomorphism is a pair of static homomorphisms which are mutually inverse. If $a \searrow a' \Rightarrow \Phi a \searrow \Phi a'$, then Φ is a (dynamic) homomorphism; similarly for (dynamic) isomorphisms.

We now return to the molecular form of action calculi; we shall define over them the basic action structure operations. For this purpose we shall use substitutions like $\sigma = \{\vec{u}/\vec{v}\}$, the simultaneous replacement of the names $\vec{u}$ for the (distinct) names $\vec{v}$, first using alpha conversion to change any bound uses of the former. We shall write $\vec{\lambda}, \vec{\mu}$ for the concatenation of two action bodies $\vec{\lambda}$ and $\vec{\mu}$.

Suppose $a = (\vec{x})\vec{\lambda}\langle\vec{u}\rangle$ and $b = (\vec{y})\vec{\mu}\langle\vec{v}\rangle$, with suitable arities; then the operations are as follows:

$$\begin{aligned}
\mathbf{id}_m &\stackrel{\text{def}}{=} (\vec{x})\,\langle\vec{x}\rangle \qquad (|\vec{x}| = m)\\
a \cdot b &\stackrel{\text{def}}{=} (\vec{x})\,\vec{\lambda}, \sigma\vec{\mu}\,\langle\sigma\vec{v}\rangle \qquad (\sigma = \{\vec{u}/\vec{y}\})\\
a \otimes b &\stackrel{\text{def}}{=} (\vec{x}\vec{y})\,\vec{\lambda}, \vec{\mu}\,\langle\vec{u}\vec{v}\rangle\\
\mathbf{ab}_x a &\stackrel{\text{def}}{=} (x\vec{x})\,\vec{\lambda}\,\langle x\vec{u}\rangle\ .
\end{aligned}$$

Thus $a \otimes b$ is simply the juxtaposition of the two actions; no order is dictated by the concatenation of the bodies, since the convention of disjoint bound names ensures that $\vec{\lambda}, \vec{\mu} = \vec{\mu}, \vec{\lambda}$ in this case. But in $a \cdot b$ the substitution σ may make $\vec{\lambda}$ bind some names in $\sigma\vec{\mu}$, so that $\vec{\lambda}, \sigma\vec{\mu} \neq \sigma\vec{\mu}, \vec{\lambda}$.

It is routine to check that $\mathsf{AC}(\mathcal{K})$ is an action structure, i.e. that the operations which we have defined satisfy the basic axioms.

We naturally ask the question: Is there an elementary set of actions in terms of which we can express all the actions of $\mathsf{AC}(\mathcal{K})$, using the basic operations and the controls $\mathcal{K}$? The answer is pleasantly simple. Define

$$\begin{aligned}
\langle x\rangle : \epsilon \to p &\stackrel{\text{def}}{=} (\,)\,\langle x\rangle \qquad (x : p)\\
\boldsymbol{\omega} : p \to \epsilon &\stackrel{\text{def}}{=} (x)\,\langle\,\rangle
\end{aligned}$$

where p is prime (note the empty bodies); then indeed all other actions are definable from these. This fact will be important later. $\langle x\rangle$ represents the name x used as a datum.

We shall need one or two further defined forms: first a (more familiar) form of abstraction $(x)a$ which discards the first datum (i.e. $\langle x\rangle$) exported by $\mathbf{ab}_x a$; then tupling and multiple abstraction; then the permutation actions $\mathbf{i}_{km}$. They are as follows:

$$\begin{aligned}
(x)a &\stackrel{\text{def}}{=} \mathbf{ab}_x a \cdot (\boldsymbol{\omega} \otimes \mathbf{id}_n) \qquad (a : m \to n)\\
\langle x_1 \cdots x_r\rangle &\stackrel{\text{def}}{=} \langle x_1\rangle \otimes \cdots \otimes \langle x_r\rangle\\
(x_1 \cdots x_r)a &\stackrel{\text{def}}{=} (x_1)\cdots(x_r)a\\
\mathbf{i}_{km} &\stackrel{\text{def}}{=} (\vec{x}\vec{y})\langle\vec{y}\vec{x}\rangle \qquad (|\vec{x}| = k, |\vec{y}| = m)\ .
\end{aligned}$$

Note that $\mathbf{i}_{km} : k \otimes m \to m \otimes k$ is an isomorphism of arities; one can easily show that $\mathbf{i}_{mk}$ is indeed its inverse.

6 Action calculi, algebraic form: $\mathsf{AC}(\mathcal{K})$=$\mathcal{T}(\mathcal{K})/\mathsf{AC}$

We have seen that all actions in $\mathsf{AC}(\mathcal{K})$ are definable from the data $\langle x \rangle$ and the discard $\boldsymbol{\omega}$, using the basic operations and the controls $\mathcal{K}$. But $\mathsf{AC}(\mathcal{K})$ is not the action structure *freely generated* by these things; certain identities, such as $\langle x \rangle \cdot \boldsymbol{\omega} = \mathbf{id}_\epsilon$, hold which are not consequences of the basic axioms. In fact, the following so-called *concrete* axioms are also satisfied. They are expressed in terms of the derived form of abstraction and the permutations, which were introduced in the previous section.

The theory AC – equations

$\boldsymbol{\gamma}$:	$(x)a = \boldsymbol{\omega} \otimes a$	(x not free in a)
$\boldsymbol{\delta}$:	$(x)(\langle x \rangle \otimes \mathbf{id}) = \mathbf{id}$	
$\boldsymbol{\zeta}$:	$\mathbf{i}_{km} \cdot (b \otimes a) = (a \otimes b) \cdot \mathbf{i}_{\ell n}$	$(a : k \to \ell,\ b : m \to n)$
$\boldsymbol{\sigma}$:	$(\langle y \rangle \otimes \mathbf{id}_k) \cdot (x)a = \{y/x\}a$	$(a : k \to \ell)$.

In the presence of the basic equations, these four axioms are equivalent to the six originally given in [11]. In particular one can deduce alpha-conversion, $\boldsymbol{\alpha} : (x)a = (y)\{y/x\}a$ (y not free in a). I believe that each of the new axioms is independent. Note that $\boldsymbol{\sigma}$ was called $\boldsymbol{\beta}$ in [11], because it is the vestige of β-conversion in which only names can replace names. But our higher-order calculi will have full-blooded β-conversion; the present rule is called $\boldsymbol{\sigma}$ since it turns out to have more to do with substitution.

We are now ready for the second way of presenting $\mathsf{AC}(\mathcal{K})$. Let $\mathcal{T}(\mathcal{K})$ be the terms built from the generators $\langle x \rangle$ and $\boldsymbol{\omega}$ by the basic operations and $\mathcal{K}$-actions. We shall let r, s, t range over terms. We shall often use the name of an equational axiom such as $\boldsymbol{\sigma}$, or the name of a theory such as AC, to stand for the congruence over $\mathcal{T}(\mathcal{K})$ which is induced by the equation or equations together with the basic action structure axioms. we are concerned with quotients of $\mathcal{T}(\mathcal{K})$ by several such congruences. In fact the fundamental result about action calculi [11] is simply:

6.1 Theorem (Molecular Form) The action calculus $\mathsf{AC}(\mathcal{K})$ is statically isomorphic with $\mathcal{T}(\mathcal{K})/\mathsf{AC}$. ■

We omit the proof. We shall not here need the formal definition of the isomorphism, but shall just suggest it by a few examples. Here are some terms representing members of $\mathcal{T}(\mathcal{K})/\mathsf{AC}$, with their corresponding molecular forms:

$$\begin{array}{rcl} K\vec{t} & \longmapsto & (\vec{x})\,\mu\,\langle\vec{y}\rangle, \ \text{ where } \vec{t} \longmapsto \vec{a} \text{ and } \mu = \langle\vec{x}\rangle K\vec{a}(\vec{y}) \\ \boldsymbol{\omega} = (x)\mathbf{id}_\epsilon & \longmapsto & (x)\,\langle\,\rangle \\ (x)t & \longmapsto & (x\vec{x})\,\vec{\lambda}\,\langle\vec{y}\rangle, \ \text{ where } t \longmapsto (\vec{x})\,\vec{\lambda}\,\langle\vec{y}\rangle\ . \end{array}$$

In passing, note that the isomorphisms $\mathbf{i}_{km}$ make an action calculus a *symmetric* monoidal category; see for example Asperti and Longo [3].

We now turn to the dynamics of an action calculus. In view of the isomorphism of the Molecular Form theorem, we may choose to define the reaction relation $\searrow_{\mathcal{K}}$ for $\mathsf{AC}(\mathcal{K})$ using either the molecular or the algebraic form. In the latter case, it is typically generated by a few rules in terms of $\mathcal{K}$-actions. For $\mathsf{PIC} = \mathsf{AC}(\boldsymbol{\nu}, \mathbf{in}, \mathbf{out})$, for example, let us first define $\mathbf{out}_x \stackrel{\text{def}}{=} (\langle x\rangle \otimes \mathbf{id}) \cdot \mathbf{out}$ and $\mathbf{in}_x \stackrel{\text{def}}{=} \langle x\rangle \cdot \mathbf{in}$; then there is just one reaction rule:

$$\mathbf{out}_x \otimes \mathbf{in}_x \ \searrow\ \mathbf{id} \ .$$

In the molecular form, the same relation is gained by taking the transitive reflexive closure of a reduction rule defined as follows: an adjacent pair of molecules $\overline{x}\langle\vec{z}\rangle, x(\vec{y})$ is a redex, and its reduction consists of removing the pair and applying the substitution $\{\vec{z}/\vec{y}\}$ to the rest of the action.

In general there is the option to allow, or to forbid, reaction to occur within a molecule, for example within the construction **box** a of the extended PIC. This question is discussed in [12]; we shall also find it relevant in Section 9 with respect to the code molecules which we are now about to introduce.

7 First step to higher order: $\mathsf{HAC}^{\sigma\beta}(\mathcal{K}) = \mathsf{AC}(\mathcal{K}^{\Rightarrow})$

There are three main steps from $\mathsf{AC}(\mathcal{K})$ (with arities M and names X) to the corresponding higher-order action calculus, which we shall call $\mathsf{HAC}(\mathcal{K})$. In the first step we form an ordinary action calculus $\mathsf{AC}(\mathcal{K}^{\Rightarrow})$ with a slightly larger control set. For this, we make a few preparations:

- Extend M to $M^{\Rightarrow}$ by freely adding the binary constructor $\Rightarrow$ (exponentiation) of arities. Then if M has primes P, $M^{\Rightarrow}$ has primes

  $$P^{\Rightarrow} \stackrel{\text{def}}{=} P \cup \{m \Rightarrow n \mid m, n \in M^{\Rightarrow}\} \ .$$

  Thus the arities are just the familiar types of a typed λ-calculus with tupling.

- Extend X to $X^{\Rightarrow}$, by introducing an infinity of names at each new prime arity $m \Rightarrow n$. We continue to use u, v, w, x, y, z for names, now ranging over the whole of $X^{\Rightarrow}$.

- Extend the range of arities for (some of) the control actions. There is some latitude here. The choice has important consequences for practical applications; it also affects the final step of our construction of $\mathsf{HAC}(\mathcal{K})$ in Section 9. [1]

[1] In PIC, for example, the arities of the input and output controls are $\mathbf{in} : p \to m$ and $\mathbf{out} : p \otimes m \to \epsilon$ (where of course $p = 1$ and $\epsilon = 0$ if M is the natural numbers); in $\mathsf{AC}(\mathcal{K}^{\Rightarrow})$ we may naturally wish to extend the range of m to $M^{\Rightarrow}$, to allow actions to pass along channels.

But what does it mean to extend the range of p to the all primes of $M^{\Rightarrow}$, not just of M; i.e. to allow $p = k \Rightarrow \ell$? In terms of what follows, it means that (the code of) an action can itself be a channel. But in

After these preparations, we add two new controls to $\mathcal{K}$. We define

$$\mathcal{K}^{\Rightarrow} \stackrel{\text{def}}{=} \mathcal{K} \cup \{\ulcorner\ \urcorner, \mathbf{ap}\}\ ;$$

then our extended action calculus is $\mathsf{AC}(\mathcal{K}^{\Rightarrow})$. The first new control obeys the arity rule

$$\frac{a : m \to n}{\ulcorner a \urcorner : \epsilon \to m \Rightarrow n}$$

and we call $\ulcorner a \urcorner$ the *code* of a. The second new control is a constant called *application*:

$$\mathbf{ap} : (m \Rightarrow n) \otimes m \to n\ .$$

These are familiar constructions in a cartesian-closed category; later we develop this connection.

The essential point is that we can now define two reaction relations, $\searrow_\sigma$ and $\searrow_\beta$, which correspond to familiar notions of reduction in λ-calculus. They are defined to be the smallest pre-orders satisfying the following two rules, and preserved by all operations *including* the control constructions: [2]

The theory $\mathsf{HAC}^{\sigma\beta}$ – reactions		
$\searrow\boldsymbol{\sigma}$:	$(\ulcorner b \urcorner \otimes \mathbf{id}_k) \cdot (x)a \searrow_\sigma \{b/x\}a$	$(a : k \to \ell)$
$\searrow\boldsymbol{\beta}$:	$(\ulcorner a \urcorner \otimes \mathbf{id}_k) \cdot \mathbf{ap} \searrow_\beta a$	$(a : k \to \ell)\ .$

We have to explain the substitution $\{b/x\}$, for $x : m \Rightarrow n$; it means that any free occurrence $\langle x \rangle$ of x in a –and this is the only possible kind of free occurrence– is replaced by $\ulcorner b \urcorner$. (Our convention implies that no name free in b can thereby become bound.) Thus $\searrow_\sigma$ deals with the distribution of a code to wherever it may be applied, and $\searrow_\beta$ decodes or applies it. The reaction relation for $\mathsf{AC}(\mathcal{K}^{\Rightarrow})$ is then

$$(\searrow_\sigma \cup \searrow_\beta \cup \searrow_\mathcal{K})^*\ ;$$

we name the calculus $\mathsf{HAC}^{\sigma\beta}(\mathcal{K})$ to emphasize the reaction rules which have been added to the dynamics of $\mathcal{K}$.

Turning to the molecular form, what do the new molecules look like? They are

$$\ulcorner a \urcorner(z)\ ,\quad \langle \vec{x} \rangle \mathbf{ap}_z(\vec{y})\ ,$$

π-calculus, values such as integers are modelled as actions; so we have a way to model the *computation* of channels as values within action calculi. There are difficult technical questions here, one of which we meet in Section 9. Indeed the very occurrence of a reaction in PIC depends upon the identity of channels, and the identity (or equality) of actions is in general an unsettled matter! For now, we merely note that out framework does not exclude such considerations.

[2] We may omit the requirement that the controls preserve these relations; this would correspond to *lazy* reduction in λ-calculus, where one does not reduce terms inside λ-abstractions. But we prefer to make the controls preserve the relations $\searrow_\sigma$ and $\searrow_\beta$ (in contrast with $\searrow_\mathcal{K}$), because in higher-order action calculi we wish to identify actions which are mutually β-reducible.

the second form being short for $\langle z\vec{x}\rangle$**ap**$(\vec{y})$. For example, consider

$$\ulcorner a\urcorner \cdot (z)\big(\langle zx\rangle \cdot \mathbf{ap} \otimes \langle zy\rangle \cdot \mathbf{ap}\big) \; ;$$

its molecular form is

$$b \; = \; [\, \ulcorner a\urcorner(z), \; \langle x\rangle \mathbf{ap}_z(\vec{u}), \; \langle y\rangle \mathbf{ap}_z(\vec{v}) \,]\, \langle \vec{u}\vec{v}\rangle \; .$$

(We shall sometimes group the molecules of an action by using square brackets, for clarity.) We can think of the molecule $\ulcorner a\urcorner(z)$ as a local declaration, making z stand for $\ulcorner a\urcorner$ within the action b.

We shall now describe how $\searrow_\sigma$ and $\searrow_\beta$ act on molecular forms. They are well-defined, of course, because of the isomorphism assured by the Molecular Form theorem. But molecular forms are a direct presentation of an action calculus, so it makes sense to define over them two single-step *reduction* relations $\longrightarrow_\sigma$ and $\longrightarrow_\beta$, whose transitive reflexive closures $\longrightarrow_\sigma^*$ and $\longrightarrow_\beta^*$ are exactly $\searrow_\sigma$ and $\searrow_\beta$.

7.1 Definition ($\boldsymbol{\sigma}$-reduction) In $\mathsf{HAC}^{\sigma\beta}(\mathcal{K})$, the reduction relation $\longrightarrow_\sigma$ is the smallest relation satisfying the following conditions.

First, let $\mu = \ulcorner c\urcorner(z)$ be a code molecule of a; then $a \longrightarrow_\sigma a'$ holds if a' is derived from a in either of the following ways:

1. Let $z_1, \ldots, z_n$ be new names for some $n \geq 0$; replace μ by $\ulcorner c\urcorner(z_1), \ldots, \ulcorner c\urcorner(z_n)$ and replace each bound occurrence of z in a by some z_i.

2. If a has a unique bound occurrence of z, and this lies within b_i in some other molecule $\langle \vec{x}\rangle\, K\vec{b}\,(\vec{y})$ of a, then move μ to the body of b_i.

Second, if $a \longrightarrow_\sigma a'$ then $\mathcal{C}[a] \longrightarrow_\sigma \mathcal{C}[a']$. ■

In the first rule note that if z has any bound occurrence then $n \neq 0$; if not, then the choice $n = 0$ allows μ to be deleted. In the second rule, note that the other molecule may also be a code molecule.

The following holds:

7.2 Proposition $\longrightarrow_\sigma^* \; = \; \searrow_\sigma$. ■

The reduction relation $\longrightarrow_\sigma$ is akin to the notion of explicit substitution [1]. One may complain that the single steps of $\longrightarrow_\sigma$ are not small enough for a truly satisfactory notion of explicit-substitution reduction. Indeed there are natural ways to refine it, though we need not examine them here.

The relation $\longrightarrow_\sigma$ is not strongly normalizing, since it allows arbitrary proliferation of useless code molecules (in rule 1). This is easily corrected, by demanding that each z_i be used at least once (so that if z has *no* free occurrences then the choice $n = 0$ is forced). The resulting relation $\longrightarrow_\sigma'$ is easily found to be both confluent and terminating, and it provides a unique σ-normal form for actions up to *σ-conversion* (the congruence induced by $\searrow_\sigma$). As a very simple example, the σ-normal form of $[\ \ulcorner b \urcorner(z),\ \langle x \rangle \mathbf{ap}_z(\vec{u})\ ]\,\langle z \rangle$ is

$$[\ \ulcorner b \urcorner(z_1),\ \langle x \rangle \mathbf{ap}_{z_1}(\vec{u}),\ \ulcorner b \urcorner(z_2)\ ]\,\langle z_2 \rangle\ ;$$

note that each bound occurrence of z is now accompanied by "its own" copy of the code molecule defining z.

Turning to $\longrightarrow_\beta$, a *β-redex for z* consists of two molecules; a code molecule for z and an $\mathbf{ap}_z$ molecule containing the unique bound occurrence of z. For example the action displayed above has a β-redex for z_1. Formally:

7.3 Definition (β-reduction) In $\mathsf{HAC}^{\sigma\beta}(\mathcal{K})$, the reduction relation $\longrightarrow_\beta$ is the smallest relation satisfying the following conditions:

First, suppose

$$a\ =\ (\ldots)[\ \vec{\kappa},\ \ulcorner b \urcorner(z),\ \langle \vec{u} \rangle \mathbf{ap}_z(\vec{x}),\ \vec{\lambda}\ ]\,\langle \vec{w} \rangle$$

where z occurs nowhere else, and $b = (\vec{y})[\ \vec{\mu}\ ]\langle \vec{v} \rangle$. Let $\sigma = \{\vec{u}/\vec{y}\}$ and $\tau = \{\sigma\vec{v}/\vec{x}\}$; then $a \longrightarrow_\beta a'$, where

$$a'\ =\ (\ldots)[\ \vec{\kappa},\ \sigma\vec{\mu},\ \tau\vec{\lambda}\ ]\,\langle \tau\vec{w} \rangle\ .$$

Second, if $a \longrightarrow_\beta a'$ then $\mathcal{C}[a] \longrightarrow_\beta \mathcal{C}[a']$. ∎

Thus the code $\ulcorner b \urcorner$ of a β-redex in a is unwrapped and instantiated; in turn, it determines an instantiation of the remainder of a.

The following holds:

7.4 Proposition $\longrightarrow_\beta^*\ =\ \searrow_\beta$. ∎

Now it is easy to see that reduction by $\longrightarrow_\beta$ is strongly normalizing in $\mathsf{HAC}^{\sigma\beta}(\mathcal{K})$, since it always reduces the number of molecules. Also it is easily shown to be weakly confluent; putting these two together, we deduce that it is confluent.

We also conjecture that the combined reduction relation $\longrightarrow_{\sigma\beta}\ =\ \longrightarrow_\sigma' \cup \longrightarrow_\beta$ is strongly normalizing and confluent. However, this question remains open. Indeed the corresponding strong normalization problem for the typed $\lambda\sigma$-calculus is open; [3] Ritter [17] partially solves the problem, showing strong normalization for a reduction strategy in which the mixture of $\longrightarrow_\sigma$ and $\longrightarrow_\beta$ steps is constrained. It is unclear whether the present problem is simpler in any way.

We have exhibited both σ-reduction and β-reduction for $\mathsf{HAC}^{\sigma\beta}(\mathcal{K})$, in order to show how the dynamics of λ-calculus can be captured rather accurately in the framework of action calculi. But for the higher-order calculi we are primarily interested in β-reduction; therefore our next step will be to replace σ-reduction by the equation for σ-conversion.

[3] I am grateful to Eike Ritter for pointing out this fact to me, and to both him and Therèse Hardin for useful discussions on this question.

8 Second step to higher order: $\mathsf{HAC}^{\beta}(\mathcal{K}) = \mathsf{HAC}^{\sigma\beta}(\mathcal{K})/\boldsymbol{\sigma}$

We have obtained a generalization of the $\lambda\sigma$-calculus as an ordinary action calculus, $\mathsf{HAC}^{\sigma\beta}(\mathcal{K})$=$\mathsf{AC}(\mathcal{K}^{\Rightarrow})$, simply by adding codes and application as new controls with appropriate reaction rules. Thus we leave ordinary action calculi for the first time when we replace the reaction rule $\searrow\!\boldsymbol{\sigma}$ by an equation:

$$(\ulcorner b \urcorner \otimes \mathbf{id}_k) \cdot (x)a \;=\; \{{}^{b}\!/\!{}_{x}\}a \qquad (a : k \to \ell)\ .$$

Now this equation can be absorbed into our existing axiom $\boldsymbol{\sigma}$, provided that we generalize the datum y in that equation to a *higher-order datum* d. To be precise, a higher-order datum is either a name x or a code $\ulcorner b \urcorner$. Indeed, the essence of the term "higher-order" is that functions –or more generally actions– can be manipulated as data. We shall let d, e, f stand for higher-order data. (We are using angle brackets $\langle \cdot \rangle$ for the inclusion of data into actions, but they can be omitted in the case of codes.)

With this new convention, we introduce the following:

The theory HAC^{β}– equations and reactions		
$\boldsymbol{\gamma}$:	$(x)a = \boldsymbol{\omega} \otimes a$	(x not free in a)
$\boldsymbol{\delta}$:	$(x)(\langle x \rangle \otimes \mathbf{id}) = \mathbf{id}$	
$\boldsymbol{\zeta}$:	$\mathbf{i}_{km} \cdot (b \otimes a) = (a \otimes b) \cdot \mathbf{i}_{\ell n}$	$(a : k \to \ell,\ b : m \to n)$
$\boldsymbol{\sigma}$:	$(\langle d \rangle \otimes \mathbf{id}_k) \cdot (x)a = \{{}^{d}\!/\!{}_{x}\}a$	$(a : k \to \ell)$
$\searrow\!\boldsymbol{\beta}$:	$(\ulcorner a \urcorner \otimes \mathbf{id}_k) \cdot \mathbf{ap} \searrow_{\beta} a$	$(a : k \to \ell)$.

Now we may take the quotient of $\mathcal{T}(\mathcal{K}^{\Rightarrow})$ by the equations in this table (i.e. everything except $\searrow\!\boldsymbol{\beta}$); this is the same as taking the quotient of $\mathsf{HAC}^{\sigma\beta}(\mathcal{K})$ by $\boldsymbol{\sigma}$. What is the corresponding molecular presentation?

The obvious candidate is to take the σ-normal forms in the molecular presentation of $\mathsf{HAC}^{\sigma\beta}(\mathcal{K})$. These were discussed in Section 7; they have the property that each code molecule binds just one occurrence of a name, and resides "as close to" that occurrence as possible. But this is ungainly; it immediately suggests that we abolish code molecules and simply allow codes to replace names as data in the molecular form.

We therefore define the higher-order action calculus $\mathsf{HAC}^{\beta}(\mathcal{K})$ in molecular form, by close analogy with the way $\mathsf{AC}(\mathcal{K})$ was defined in Section 4. An *action* $a : m \to n$ of $\mathsf{HAC}^{\beta}(\mathcal{K})$ takes the form

$$(\vec{x})\ \mu_1,\ \ldots,\ \mu_h\ \langle \vec{d} \rangle \qquad (|\vec{x}| = m,\ |\vec{d}| = n)\ ,$$

while a molecule μ takes the form

$$\langle \vec{e} \rangle\ K \vec{b}\, (\vec{y}) \qquad (|\vec{e}| = k, |\vec{y}| = \ell)$$

where $K \in \mathcal{K} \cup \{\mathbf{ap}\}$. Thus the only differences from $\mathsf{AC}(\mathcal{K})$ are that the data vectors $\langle \vec{d} \rangle, \langle \vec{e} \rangle$ may contain both names and codes, and that $\mathbf{ap}$ molecules may occur. Note that the general form of an **ap** molecule is $\langle f\vec{e}\rangle\,\mathbf{ap}\,(\vec{y})$, where the data f and $\vec{e}$ are respectively the operator and the operands of the application.

The action structure operations over these forms are also exactly analogous. Suppose $a = (\vec{x})\vec{\lambda}\langle\vec{d}\rangle$ and $b = (\vec{y})\vec{\mu}\langle\vec{e}\rangle$, with suitable arities; then the operations are as follows:

$$\begin{array}{rcll}
\mathbf{id}_m & \stackrel{\text{def}}{=} & (\vec{x})\,\langle\vec{x}\rangle & (|\vec{x}| = m) \\
a \cdot b & \stackrel{\text{def}}{=} & (\vec{x})\,\vec{\lambda}, \sigma\vec{\mu}\,\langle\sigma\vec{e}\rangle & (\sigma = \{\vec{d}/\vec{y}\}) \\
a \otimes b & \stackrel{\text{def}}{=} & (\vec{x}\vec{y})\,\vec{\lambda}, \vec{\mu}\,\langle\vec{d}\vec{e}\rangle & \\
\mathbf{ab}_x a & \stackrel{\text{def}}{=} & (x\vec{x})\,\vec{\lambda}\,\langle x\vec{d}\rangle\;. &
\end{array}$$

Thus what was achieved gradually by an explicit substitution molecule $\ulcorner b \urcorner(z)$ is now achieved by a direct replacement $\{\ulcorner b \urcorner/z\}$.

Now the fundamental result about higher-order action calculi is also analogous:

8.1 Theorem ($\boldsymbol{\beta}$-Molecular Form) $\mathsf{HAC}^{\beta}(\mathcal{K})$ is statically isomorphic with $\mathsf{HAC}^{\sigma\beta}(\mathcal{K})/\boldsymbol{\sigma}$ (equivalently, with $\mathcal{T}(\mathcal{K}^{\Rightarrow})/\mathsf{HAC}^{\beta}$). ■

Let us now turn to dynamics, and consider first the reaction rule $\searrow\boldsymbol{\beta}$. Because of the isomorphism, we can now ask how this relation may be characterized by a single-step reduction relation upon molecular forms.

8.2 Definition ($\boldsymbol{\beta}$-reduction) In $\mathsf{HAC}^{\beta}(\mathcal{K})$, the reduction relation $\longrightarrow_{\beta}$ is the smallest relation satisfying the following conditions: First, suppose

$$a \;=\; (\ldots)\,[\;\vec{\kappa},\;\langle\vec{d}\rangle\mathbf{ap}_{\ulcorner b \urcorner}(\vec{x}),\;\vec{\lambda}\;]\,\langle\vec{f}\rangle$$

and $b = (\vec{y})[\,\vec{\mu}\,]\langle\vec{e}\rangle$. Let $\sigma = \{\vec{d}/\vec{y}\}$ and $\tau = \{\sigma\vec{e}/\vec{x}\}$; then $a \longrightarrow_{\beta} a'$, where

$$a' \;=\; (\ldots)\,[\;\vec{\kappa},\;\sigma\vec{\mu},\;\tau\vec{\lambda}\;]\,\langle\tau\vec{f}\rangle\;.$$

Second, if $a \longrightarrow_{\beta} a'$ then $\mathcal{C}[a] \longrightarrow_{\beta} \mathcal{C}[a']$. ■

The following again holds:

8.3 Proposition $\longrightarrow_{\beta}^{*} \;=\; \searrow_{\beta}$. ■

In the absence of explicit substitution, the behaviour of reduction (i.e. just β-reduction) is clearer to establish. Of course, the strong normalization of $\longrightarrow_{\beta}$ is no longer the trivial matter that it was in $\mathsf{HAC}^{\sigma\beta}(\mathcal{K})$, since the substitutions involved in $\longrightarrow_{\beta}$ are no longer just names for names. Nonetheless, we can establish the following:

8.4 Theorem ($\boldsymbol{\beta}$-Termination and Confluence) $\longrightarrow_{\beta}$ is both strongly normalizing and confluent.

Proof For strong normalization, one can adapt a standard method for the typed λ-calculus; see for example [4]. The adaptation requires some care, but we omit the details here.

For confluence, it is simplest first to prove weak confluence, namely that if $a \longrightarrow_\beta a_1$ and $a \longrightarrow_\beta a_2$ then there exists a' for which $a_1 \longrightarrow_\beta{}^* a'$ and $a_2 \longrightarrow_\beta{}^* a'$. But by a well-known argument, weak confluence with strong normalization implies confluence. ∎

9 Third step to higher order: $\mathsf{HAC}(\mathcal{K}) = \mathsf{HAC}^\beta(\mathcal{K})/\boldsymbol{\beta}$

The ground is well prepared for our final step to higher-order. We first replace the rule $\searrow\boldsymbol{\beta}$ by an equation, so that we obtain a purely equational theory:

The theory HAC – equations

$$
\begin{array}{lll}
\boldsymbol{\beta}: & (\ulcorner a \urcorner \otimes \mathbf{id}_k) \cdot \mathbf{ap} \;=\; a & (a : k \to \ell) \\
\boldsymbol{\gamma}: & (x)a = \boldsymbol{\omega} \otimes a & (x \text{ not free in } a) \\
\boldsymbol{\delta}: & (x)(\langle x \rangle \otimes \mathbf{id}) = \mathbf{id} & \\
\boldsymbol{\zeta}: & \mathbf{i}_{km} \cdot (b \otimes a) = (a \otimes b) \cdot \mathbf{i}_{\ell n} & (a : k \to \ell,\ b : m \to n) \\
\boldsymbol{\sigma}: & (\langle d \rangle \otimes \mathbf{id}_k) \cdot (x)a = \{{}^{d}\!/\!{}_{x}\}a & (a : k \to \ell)\ .
\end{array}
$$

We may take the quotient of $\mathcal{T}(\mathcal{K}^{\Rightarrow})$ by the congruence induced by these equations; this is the same as taking the quotient of $\mathsf{HAC}^\beta(\mathcal{K})$ by $\boldsymbol{\beta}$. As we did in the second step, we again ask: What is the corresponding molecular presentation? But this time the answer is simpler.

The results of the previous section ensure that two actions of $\mathsf{HAC}^{\sigma\beta}(\mathcal{K})$ are interconvertible by $\searrow_\beta$ iff they have the same $\longrightarrow_\beta$-normal form, and that every action a has a unique such normal form; we shall denote it by $\underline{a}$. These normal forms are exactly those molecular forms of $\mathsf{HAC}^\beta(\mathcal{K})$ which contain no β-redex $\langle \vec{d} \rangle \mathbf{ap}_{\ulcorner b \urcorner}(\vec{x})$, i.e. in which every $\mathbf{ap}$ molecule takes the form $\langle \vec{d} \rangle \mathbf{ap}_z(\vec{x})$, having a name (not a code) as its principal datum. Furthermore, a datum $d = \ulcorner a \urcorner$ is in normal form if a is. We therefore *define* $\mathsf{HAC}(\mathcal{K})$ to be the algebra of these molecular normal forms. Some of the operations do not preserve normality, since $\{{}^{\vec{d}}\!/\!{}_{\vec{x}}\}a$ may not be normal even when $\vec{d}$ and a are normal. For example, we must define composition in $\mathsf{HAC}(\mathcal{K})$ as follows: if $a = (\vec{x})\vec{\lambda}\langle \vec{d} \rangle$ and $b = (\vec{y})\vec{\mu}\langle \vec{e} \rangle$ then

$$
a \cdot b \;\stackrel{\text{def}}{=}\; \underline{(\vec{x})\,\vec{\lambda}, \sigma\vec{\mu}\,\langle \sigma\vec{e} \rangle} \qquad (\sigma = \{{}^{\vec{d}}\!/\!{}_{\vec{y}}\})\ .
$$

Once this is done, we arrive finally at the following:

9.1 Theorem (Higher-order Molecular Form) $\mathsf{HAC}(\mathcal{K})$ is statically isomorphic with $\mathsf{HAC}^\beta(\mathcal{K})/\boldsymbol{\beta}$ (equivalently, with $\mathcal{T}(\mathcal{K}^{\Rightarrow})/\mathsf{HAC}$). ∎

Another way of stating this result is to say that the β-normal form operation $(\underline{\ \ })$ is a static homomorphism from $\mathsf{HAC}^\beta(\mathcal{K})$ to $\mathsf{HAC}(\mathcal{K})$. This is a pleasant outcome, but we also want it to be a *dynamic* homomorphism; that is, it should map the reaction relation $\searrow_{\beta\mathcal{K}} = (\searrow_\beta \cup \searrow_\mathcal{K})^*$ of $\mathsf{HAC}^\beta(\mathcal{K})$ into the appropriate reaction relation $\searrow$ for $\mathsf{HAC}(\mathcal{K})$. We shall find this to be the case, under reasonable conditions.

We shall assume for $\mathsf{HAC}^\beta(\mathcal{K})$ a single-step relation $\longrightarrow_\mathcal{K}$ such that $\longrightarrow_\mathcal{K}^* = \searrow_\mathcal{K}$. (One can take $\longrightarrow_\mathcal{K} = \searrow_\mathcal{K}$ of course; but normally $\longrightarrow_\mathcal{K}$ would represent a single quantum of reaction.) Now since $\mathsf{HAC}(\mathcal{K})$ works with β-normal forms, the following is a natural definition:

9.2 Definition (Reaction) We define a single-step relation $\longrightarrow$ and a reaction relation $\searrow$ over $\mathsf{HAC}(\mathcal{K})$ as follows:

$$\begin{array}{lcl} \longrightarrow & \stackrel{\text{def}}{=} & \longrightarrow_\mathcal{K} \longrightarrow_\beta^* \text{ restricted to } \beta\text{-normal forms} \\ \searrow & \stackrel{\text{def}}{=} & \longrightarrow^* . \end{array}$$

■

Now the crucial property which we shall assume for $\mathsf{HAC}^\beta(\mathcal{K})$ is defined thus:

9.3 Definition (Stability) We say that $\longrightarrow_\mathcal{K}$ is *stable* if, whenever $a \longrightarrow_\beta a'$ and $a \longrightarrow_\mathcal{K} b$, then there exists b' for which $a' \longrightarrow_\mathcal{K} b'$ and $b \longrightarrow_\beta^* b'$.

$$\begin{array}{ccc} a & \longrightarrow_\beta & a' \\ \downarrow_\mathcal{K} & & \downarrow_\mathcal{K} \\ b & \longrightarrow_\beta^* & b' \end{array}$$

■

The intuition is that a quantum of $\mathcal{K}$-reaction which can occur before β-reduction can occur equally well after it (and hence one changes nothing by working only with β-normal forms).

We can now prove the desired result:

9.4 Theorem (Dynamic Homomorphism) Let $\longrightarrow_\mathcal{K}$ be stable in $\mathsf{HAC}(\mathcal{K})$. Then the β-normal form operation $\underline{(\ )}$ is a dynamic homomorphism from $\mathsf{HAC}^\beta(\mathcal{K})$ to $\mathsf{HAC}(\mathcal{K})$; that is,

$$a \searrow_{\beta\mathcal{K}} b \;\Rightarrow\; \underline{a} \searrow \underline{b} .$$

Proof Since $\searrow_{\beta\mathcal{K}} = (\longrightarrow_\beta \cup \longrightarrow_\mathcal{K})^*$, it is enough to prove that

$$\begin{array}{lcl} a \longrightarrow_\beta b & \Rightarrow & \underline{a} \searrow \underline{b} \\ a \longrightarrow_\mathcal{K} b & \Rightarrow & \underline{a} \searrow \underline{b} . \end{array}$$

The former is obvious, because $a \longrightarrow_\beta a' \Rightarrow \underline{a} = \underline{a'}$. For the latter, since $a \longrightarrow_\beta^* \underline{a}$, by stability (iterated) we have that for some β'

$$\underline{a} \longrightarrow_\mathcal{K} b' \text{ and } b \longrightarrow_\beta^* b' ;$$

but since $\longrightarrow_\beta$ is confluent we deduce that $b' \longrightarrow_\beta^* \underline{b}$, hence $\underline{a} \longrightarrow \underline{b}$, hence $\underline{a} \searrow \underline{b}$. ∎

Since we have now completed the extension of arbitrary action calculi to higher-order, we must ensure that the character of the original calculus has been preserved. But this is easy, via molecular forms. For every molecular form of $\mathsf{AC}(\mathcal{K})$ is indeed a molecular form of $\mathsf{HAC}(\mathcal{K})$, and the algebra is consistent; thus we have an injection $\Phi : \mathsf{AC}(\mathcal{K}) \rightarrow \mathsf{HAC}(\mathcal{K})$, and moreover it is easy to check that $a \searrow b$ iff $\Phi a \searrow \Phi b$. This is summarised as follows:

9.5 Theorem (Conservative Extension) For any $\mathcal{K}$, $\mathsf{AC}(\mathcal{K})$ is a sub-actionstructure of $\mathsf{HAC}(\mathcal{K})$ (in the sense of [10]). ∎

To conclude this section, we look for sufficient conditions under which $\longrightarrow_\mathcal{K}$ is stable. (We do not attempt a full analysis.) First consider the case of PIC, where a redex for $\longrightarrow_\mathcal{K}$ consists of an instance of

$$\langle z \rangle \mathbf{out}_x,\ \mathbf{in}_x(y)\ ,$$

and reduction consists of removing the redex and applying $\{z/y\}$ to the rest of the action. By "instance" we refer to instantiation of free names; if a name is higher-order, it may instantiate to a code.

A preparatory step in extending an action calculus to higher order is to extend (some of) the admissible arities of its controls. In Section 7 we mentioned that there is some freedom here. Indeed, the choice is relevant to stability. For suppose that the arity of z in the above redex-pattern may be an exponential, $m \Rightarrow n$; then

$$\langle \ulcorner b \urcorner \rangle \mathbf{out}_x,\ \mathbf{in}_x(y)$$

is a redex. Also, it remains a redex after any β-reduction within $\ulcorner b \urcorner$. This redex is what allows processes to flow along channels in the π-calculus, so it is important to allow it! In contrast, suppose that the arity of the channel name x is allowed to be exponential; then

$$\langle z \rangle \mathbf{out}_{\ulcorner c \urcorner},\ \mathbf{in}_{\ulcorner c \urcorner}(y)$$

is a redex – but ceases to be so (i.e. is no longer an instance of the redex-pattern) after a β-reduction within an occurrence of $\ulcorner c \urcorner$! In this case $\longrightarrow_\mathcal{K}$ is not stable; we would therefore have to work harder to prove that $\mathsf{HAC}(\mathcal{K})$ is homomorphic image of $\mathsf{HAC}^\beta(\mathcal{K})$, if indeed it is so. Fortunately, for PIC it appears quite reasonable to restrict x to be a first-order name; that is, in extending PIC to higher-order we assign to **in** and **out** the arities $1 \Rightarrow m$ and $1 \otimes m \Rightarrow \epsilon$, not $p \Rightarrow m$ and $p \otimes m \Rightarrow \epsilon$ for arbitrary p.

More generally, if $\longrightarrow_\mathcal{K}$ can be described in terms of redex-patterns of molecules, where reduction consists of transforming the redex and invoking a substitution, then we ask that the redex-pattern should contain no free repeated higher-order names.

(Metavariables over actions such as "a", "b" may also occur in a redex pattern, and for similar reasons we ask that they should not be repeated.)

But there is another issue, concerning another freedom in extension to higher-order. Codes $\ulcorner a \urcorner$ are introduced in the extension, and we are free to allow or to forbid the reduction of a $\mathcal{K}$-redex within a code. If we allow it, then since β-reduction can cause codes to be copied we shall lose stability in general; we can only establish a weaker form of stability corresponding to the diagram

$$\begin{array}{ccc} a & \rightarrow_\beta & a' \\ \downarrow_{\mathcal{K}} & & \downarrow_{\mathcal{K}}{}^* \\ b & \rightarrow_\beta{}^* & b' \end{array}$$

and this is not enough to prove the Dynamic Homomorphism theorem. Again, fortunately it is reasonable to forbid $\mathcal{K}$-reduction within a code (though also interesting to permit it).

Under the two constraints we have mentioned, we can easily prove stability for PIC, and for its extensions which allow guarding and replication [12]. The same is also true for NETC [11]. We conjecture that the two above-mentioned constraints, stated more formally, will always ensure stability.

10 Cartesian closure: the theory HAC$\boldsymbol{\eta}$

In this section we investigate the relationship between the theory HAC and the theory of typed λ-calculus with β- and η-conversion. We first state some simple consequences of HAC:

10.1 Lemma The following are consequences of HAC:

(1) $(\vec{x})(a \cdot b) = (\vec{x})a \cdot b \qquad (\vec{x}$ not free in $b)$
(2) $(\vec{x})(a \otimes b) = (\vec{x})a \otimes b \qquad (\vec{x}$ not free in $b)$
(3) $\ulcorner a \urcorner \cdot \boldsymbol{\omega} = \mathbf{id}_\epsilon$.

■

Let $A_{m,n}$ be the actions of arity $m \rightarrow n$ in HAC($\mathcal{K}$). Can we find a correspondence

$$A_{k \otimes m, n} \underset{\mathbf{uncurry}}{\overset{\mathbf{curry}}{\rightleftarrows}} A_{k, m \Rightarrow n}$$

which is an isomorphism for HAC($\mathcal{K}$)? Define

$$\begin{array}{rcll} \mathbf{curry}(a) : k \rightarrow m \Rightarrow n & \stackrel{\text{def}}{=} & (\vec{x})\ulcorner(\langle\vec{x}\rangle \otimes \mathbf{id}_m) \cdot a\urcorner & (a : k \otimes m \rightarrow n) \\ \mathbf{uncurry}(c) : k \otimes m \rightarrow n & \stackrel{\text{def}}{=} & (c \otimes \mathbf{id}_m) \cdot \mathbf{ap} & (c : k \rightarrow m \Rightarrow n)\ . \end{array}$$

where $\vec{x}$ is not free in a. Then one half of the isomorphism is readily deduced:

10.2 Lemma $\mathsf{HAC} \vdash \textbf{uncurry}(\textbf{curry}(a)) = a$.

Proof Working in HAC:

$$\begin{array}{rcll}
\textbf{uncurry}(\textbf{curry}(a)) & = & ((\vec{x})\ulcorner(\langle\vec{x}\rangle \otimes \textbf{id}_m)\cdot a\urcorner \otimes \textbf{id}_m)\cdot \textbf{ap} & \\
& = & (\vec{x})(\ulcorner(\langle\vec{x}\rangle \otimes \textbf{id}_m)\cdot a\urcorner \otimes \textbf{id}_m)\cdot \textbf{ap}) & \text{by Lemma 10.1(1),(2)} \\
& = & (\vec{x})(((\langle\vec{x}\rangle \otimes \textbf{id}_m)\cdot a) & \text{by } \boldsymbol{\beta} \\
& = & (\vec{x})(\langle\vec{x}\rangle \otimes \textbf{id}_m)\cdot a & \text{by Lemma 10.1(1)} \\
& = & a & \text{by } \boldsymbol{\delta}
\end{array}$$

■

On the other hand it is *not* the case that $\mathsf{HAC} \vdash \textbf{curry}(\textbf{uncurry}(c)) = c$. In fact we shall shortly see that such a consequence would be unfortunate in $\mathsf{HAC}(\mathcal{K})$ when $\mathcal{K}$ is non-empty.

For this purpose, we shall now consider the theory $\mathsf{HAC}\boldsymbol{\eta}$, gained by adding to HAC the axiom

$\boldsymbol{\eta}$-Conversion
$\boldsymbol{\eta}: \quad \ulcorner(c \otimes \textbf{id}_k)\cdot \textbf{ap}\urcorner = c \qquad (c : \epsilon \to k \Rightarrow \ell)$

10.3 Proposition $\mathsf{HAC}\boldsymbol{\eta}$ entails the isomorphism

$$A_{k\otimes m,n} \underset{\textbf{uncurry}}{\overset{\textbf{curry}}{\rightleftarrows}} A_{k,m\Rightarrow n}$$

Proof Lemma 10.2 is one half; for the other half, working in $\mathsf{HAC}\boldsymbol{\eta}$:

$$\begin{array}{rcll}
\textbf{curry}(\textbf{uncurry}(c)) & = & ((\vec{x})\ulcorner(\langle\vec{x}\rangle \otimes \textbf{id}_m)\cdot(c \otimes \textbf{id}_m)\cdot \textbf{ap}\urcorner & \\
& = & ((\vec{x})\ulcorner(\langle\vec{x}\rangle \cdot c \otimes \textbf{id}_m)\cdot \textbf{ap}\urcorner & \\
& = & (\vec{x})(\langle\vec{x}\rangle \cdot c) & \text{by } \boldsymbol{\eta} \\
& = & (\vec{x})\langle\vec{x}\rangle \cdot c & \text{by Lemma 10.1(1)} \\
& = & c & \text{by } \boldsymbol{\delta}\ .
\end{array}$$

■

We have thus shown that the quotient of any higher-order action calculus by η-conversion is a monoidal-closed category (see Asperti and Longo [3]). To show that it is cartesian-closed, it remains to prove that it is cartesian; this means that there is a unique action from each arity m to ϵ, and that surjective pairing exists. So we define the following:

10.4 Definition

$$\begin{array}{rrcll}
\text{(Discard)} & \boldsymbol{\omega} : m \to \epsilon & \stackrel{\text{def}}{=} & \boldsymbol{\omega} \otimes \cdots \otimes \boldsymbol{\omega} & (m = p_1 \otimes \cdots \otimes p_r) \\
\text{(Copy)} & \textbf{copy}_m : m \to m \otimes m & \stackrel{\text{def}}{=} & (\vec{x})\langle\vec{x}\vec{x}\rangle & \\
\text{(Project)} & \textbf{left} : m \otimes n \to m & \stackrel{\text{def}}{=} & \textbf{id}_m \otimes \boldsymbol{\omega} & \\
\text{(Project)} & \textbf{right} : m \otimes n \to n & \stackrel{\text{def}}{=} & \boldsymbol{\omega} \otimes \textbf{id}_n & \\
\text{(Pair)} & (b, c) : k \to m \otimes n & \stackrel{\text{def}}{=} & \textbf{copy}_k \cdot (b \otimes c) & (b : k \to m,\ c : k \to n)\ .
\end{array}$$

■

Now we state without proof some properties which hold independently of $\boldsymbol{\eta}$:

10.5 Lemma The following are consequences of HAC:

(1) $\mathbf{copy} \cdot (a \otimes \boldsymbol{\omega}) = a$
(2) $\mathbf{copy} \cdot (\boldsymbol{\omega} \otimes a) = a$
(3) $\mathbf{copy}_{m\otimes n} \cdot (\mathbf{left} \otimes \mathbf{right}) = \mathbf{id}_{m\otimes n}$.

■

Now we prove a crucial consequence of the $\boldsymbol{\eta}$ rule:

10.6 Lemma If $b : \epsilon \to n$ then $\mathsf{HAC}\boldsymbol{\eta} \ \vdash \ b \cdot (\vec{x})\ulcorner\langle\vec{x}\rangle\urcorner = \ulcorner b \urcorner$.

Proof Using $\boldsymbol{\eta}$ in reverse (with $m = \epsilon$) we have

$$\begin{array}{rcll} b \cdot (\vec{x})\ulcorner\langle\vec{x}\rangle\urcorner & = & \ulcorner b \cdot (\vec{x})\ulcorner\langle\vec{x}\rangle\urcorner \cdot \mathbf{ap}\urcorner & \\ & = & \ulcorner b \cdot (\vec{x})(\ulcorner\langle\vec{x}\rangle\urcorner \cdot \mathbf{ap})\urcorner & \text{by Lemma 10.1(1)} \\ & = & \ulcorner b \cdot (\vec{x})\langle\vec{x}\rangle\urcorner & \text{by } \boldsymbol{\beta} \\ & = & \ulcorner b \urcorner & \text{by } \boldsymbol{\delta} \ . \end{array}$$

■

We are now able to establish the cartesian property:

10.7 Proposition The following (cartesian laws) are provable in $\mathsf{HAC}\boldsymbol{\eta}$:

(1) $a = \boldsymbol{\omega} \qquad (a : m \to \epsilon)$
(2) $(a, b) \cdot \mathbf{left} = a$
(3) $(a, b) \cdot \mathbf{right} = b$
(4) $(a \cdot \mathbf{left}, a \cdot \mathbf{right}) = a$.

Proof First, by postcomposing $\boldsymbol{\omega}$ to Lemma 10.6 (with $n = \epsilon$) and applying Lemma 10.1(3) we get $b = \mathbf{id}_\epsilon$ for any $b : \epsilon \to \epsilon$. We deduce the first result by noting $a : m \to \epsilon = (\vec{x})(\langle\vec{x}\rangle \cdot a)$ using Lemma 10.1(1), taking b to be $\langle\vec{x}\rangle \cdot a$ and noting that $(\vec{x})\mathbf{id}_\epsilon = \boldsymbol{\omega}$.

For the left projection, we have

$$\begin{array}{rcll} (a, b) \cdot \mathbf{left} & = & \mathbf{copy} \cdot (a \otimes b) \cdot (\mathbf{id} \otimes \boldsymbol{\omega}) & \\ & = & \mathbf{copy} \cdot (a \otimes \boldsymbol{\omega}) & \text{by the first result} \\ & = & a & \text{by Lemma 10.5(1)} \end{array}$$

and the right projection is similar.

Finally, note that $\ulcorner b \urcorner \cdot \mathbf{ap} = b$ by $\boldsymbol{\beta}$, for any $b : \epsilon \to \ell$; so by postcomposing $\mathbf{copy} \cdot (\mathbf{ap} \otimes \mathbf{ap})$ to Lemma 10.6, and using $\boldsymbol{\sigma}$, $\boldsymbol{\beta}$ and Lemma 10.1(1), we get

$$b \cdot \mathbf{copy} = b \otimes b \ .$$

So for arbitrary $a : k \to m \otimes n$, we have $a = (\vec{x})(\langle\vec{x}\rangle \cdot a)$ by Lemma 10.1(1) ($\vec{x}$ not free in a), so by taking $b = (\langle\vec{x}\rangle \cdot a)$ and then abstracting we deduce

$$a \cdot \mathbf{copy} = \mathbf{copy} \cdot (a \otimes a) \ ;$$

finally, by postcomposing **left** $\otimes$ **right** to this, with the help of Lemma 10.5(3) we obtain

$$(a \cdot \textbf{left}, a \cdot \textbf{right}) = a$$

as required. ■

Propositions 10.3 and 10.7 establish the equations of a cartesian-closed category (see for example Lambek and Scott [8]). So we have proved the following:

10.8 Theorem (Cartesian-closure) $\mathsf{HAC}(\mathcal{K})/\boldsymbol{\eta}$, or equivalently $\mathcal{T}(\mathcal{K})/\mathsf{HAC}\boldsymbol{\eta}$, is a cartesian-closed category. ■

Thus, in the case $\mathcal{K} = \emptyset$ we appear to arrive at the standard $\boldsymbol{\beta\eta}$ theory of the typed λ-calculus, presented categorically. The terms have free variables, so the extra structure of abstractors, data and substitution is also present. [4]

For $\mathcal{K} \neq \emptyset$ we have also arrived at a cartesian-closed category, but in general we have severely damaged $\mathsf{HAC}(\mathcal{K})$ by taking the quotient by $\boldsymbol{\eta}$. In fact, we no longer have a conservative extension of the original $\mathsf{AC}(\mathcal{K})$, in strong contrast with Theorem 9.5. For we have identified all actions with arity $m \to \epsilon$; for example in PIC, the process

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

which "receives any channel name y along the channel x and then sends z along y", is identified with the empty process! The reason this is unsatisfactory is that the particles of PIC such as $x(y)$, though inactive by themselves, represent the power to react in the presence of other particles such as $\overline{x}\langle w\rangle$ (which may become available via the product operation $\otimes$). Thus the $\boldsymbol{\eta}$ rule appears to be incompatible with interactivity (one might say: with side-effects), and a central purpose of action calculi is to understand interactivity.

Even though $\boldsymbol{\eta}$ must be rejected in general, one might hope to find a weaker rule $\boldsymbol{\eta}'$ say, such that $\mathsf{HAC}(\mathcal{K})/\boldsymbol{\eta}'$ preserves the interactivity of $\mathsf{HAC}(\mathcal{K})$ even though its subcategory $\mathsf{HAC}(\emptyset)/\boldsymbol{\eta}'$ becomes cartesian-closed. But this too is doomed to failure. For to achieve the latter we must have (in particular) $\mathsf{HAC}{+}\boldsymbol{\eta}' \vdash \textbf{ap} \cdot \boldsymbol{\omega} = \boldsymbol{\omega}$; in the case $\textbf{ap} : (\epsilon \Rightarrow \epsilon) \to \epsilon$ this implies $\textbf{ap} = \boldsymbol{\omega}$. Then in higher-order PIC for example, apply this to $\ulcorner a \urcorner$ where $a = [\ \overline{y}\langle z\rangle\ ]$ (in molecular form); we get $[\ \overline{y}\langle z\rangle\ ] = [\ \ ]$! [5]

This is not to say that $\boldsymbol{\eta}$ is inappropriate for *every* $\mathcal{K}$. For example, the restriction operator $\boldsymbol{\nu}$ is a control which has no dynamics but represents restriction of access.

[4] If we take the arities to be $M^{\Rightarrow}$ where $(M, +, 0)$ is the monoid of natural numbers under addition, then the "obvious" model of $\mathsf{HAC}\boldsymbol{\eta}$ is as follows; specify an arbitrary set V of values, and define sets $\{D_m \mid m \in M^{\Rightarrow}\}$ thus: D_0 = a singleton, $D_1 = V$, $D_{m\otimes n} = D_m \times D_n$, $D_{m\Rightarrow n} = D_n^{D_m}$; let Env (the environments) be all functions from X to $\bigcup_m D_m$ which respect arity; then the arrows from m to n are just the functions from Env to $D_n^{D_m}$ with finite support (i.e. depending only on a finite portion of the environment).

[5] Consider this analogy with a more familiar phenomenon: $\lambda f \lambda x\, f x$ and $\lambda f \lambda x\, (\lambda y\, f x)(f x)$ will mean the same in extensional models of λ-calculus; but in a world (such as Standard ML) where the argument supplied for f may have side-effects they behave differently.

For the case $\mathcal{K} = \{\boldsymbol{\nu}\}$ it seems that the effect of $\boldsymbol{\eta}$ is entirely beneficial – it forces identifications such as

$$[\ \boldsymbol{\nu} x\ ]\langle\ \rangle\ = \mathbf{id}_{\epsilon}\ ,$$

a kind of garbage collection (x being totally inaccessible). This suggests a connection with the $\boldsymbol{\nu}$-calculus of Pitts and Stark [16] which is worth pursuing.

11 Open questions

Although a general pattern of higher-order interactive systems has been drawn, several questions have been raised in the process. Most of the following open questions have been mentioned in the text:

1. The question of strong normalization for $\sigma\beta$-reduction $\longrightarrow_{\sigma\beta}$ remains open (end of Section 7). The corresponding question is also open for the $\lambda\sigma$-calculus [1], though Ritter [17] has a partial solution. It is not clear whether or not the problem for action calculi is easier; in any case, the present work adds some incentive to find a full solution.

2. In raising an action calculus $\mathsf{AC}(\mathcal{K})$ to higher-order, there is some freedom as to how much to extend the admissible arities for the controls $\mathcal{K}$(start of Section 7). What implications does this freedom hold for the expressive power of the resulting higher-order calculus?

3. The freedom just mentioned also affects stability of $\mathcal{K}$-reduction (Definition 9.3), and hence the proof of the Dynamic Homomorphism Theorem (9.4). Is there a wide class of reaction rules which guarantee stability? Does the theorem hold in cases with only a weaker form of stability?

4. A particularly intriguing case of the above is when $\mathcal{K}$-reduction is allowed inside codes (end of Section 9). Since codes can be copied, a $\mathcal{K}$-reduction which is done before copying can be thought of as a *shared* reduction. Remembering that $\mathcal{K}$-reduction is not confluent in general, what sense do these shared reductions make?

5. There have been striking technical advances recently in understanding reduction for λ-calculus, with respect to sharing (e.g. Asperti and Laneve [2]) and local reduction (e.g. Danos and Regnier [6]). Does the present framework allow these ideas to be extended to interactive systems?

6. Though η-conversion is inadmissible in general for higher-order action calculi, it is certainly admissible for some (Section 10); which ones? Otherwise can we quotient by some extensional congruence, perhaps based upon bisimilarity, defined in general terms over all higher-order action calculi? (This question has not even been solved for ordinary action calculi.)

12 Practical development

Recently Benjamin Pierce [15] has explored the design of a programming language based upon the π-calculus. The benefit of such a language should be somewhat like that of a pure functional programming language like Miranda or Haskell; to the extent that it has an understood theory, there is no need for an independent semantic definition of such a language. One can also expect less adhoc methods of reasoning about programs. (Of course, at present this is more true of functional programming than it is of π-calculus programming!) Pierce has begun to find some useful and natural programming styles arising in his experiments.

The molecular forms of action calculi, and especially higher-order action calculi, can been seen as a contribution to this exploration. In fact, a programmer would find nothing surprisingly new in programming with molecular forms. We already pointed out that the molecule $\ulcorner b \urcorner(z)$ is like a local declaration:

$$\ulcorner b \urcorner(z) \;\longmapsto\; \textbf{bind}\ \ulcorner b \urcorner\ \textbf{to}\ z\ \textbf{in}\ \cdots\ .$$

Of course this is an "explicit substitution" molecule, which was removed in the passage from $\mathsf{HAC}^{\sigma\beta}(\mathcal{K})$ to $\mathsf{HAC}^{\beta}(\mathcal{K})$, but there is no harm in keeping it in a programming language. More significantly, an **ap** molecule in $\mathsf{HAC}(\mathcal{K})$ is also a local declaration:

$$\langle \vec{d} \rangle \mathbf{ap}_z(\vec{x}) \;\longmapsto\; \textbf{bind}\ z\langle d_1, \ldots, d_r \rangle\ \textbf{to}\ \ x_1, \ldots, x_s\ \textbf{in}\ \cdots\ ;$$

this is the kind of quasi-function application used in Standard ML, where it is understood that the call of z can well produce side-effects as well as –and in our case even after!– producing results to be called $x_1, \ldots$.

Thus it appears that the effect of action calculi is not so much to suggest new programming constructs or styles, but to take familiar programming constructs and make them part of the underlying theory. It was not planned this way; it therefore seems significant that familiar programming constructs provide normal forms in a rather natural algebraic theory. A fruitful study would be to compare the constructs of Facile [7], a language for combining functional and communicational programming, with the molecular forms; I believe that there will be some interesting correspondences.

Because of this convergence of practical and theoretical constructions, one may expect theoretical development to feed all the more directly into language design and system analysis. (One could say "program analysis", but action calculi are intended for descriptive as well as for prescriptive purposes.) One important direction for development is the use of type structure – or arity structure – based upon π-calculus sortings, to predict (by static analysis) certain forms of good behaviour in communicating systems. A paradigm here is how static type-analysis in Standard ML (for example) can predict that certain evaluations will be free of side-effects and can therefore be understood better, or compiled more efficiently.

Another practical development may be to combine various concurrent paradigms with each other, and with higher-order. Recall from [11] that Petri nets constitute an action calculus; within this framework, one can experiment with receiving a parametric

Petri net along a channel, instantiating it, composing it with some other net, marking it, and executing the result!

References

[1] M. Abadi, L. Cardelli, P.-L. Curien, J.-J. Lévy, *Explicit Substitutions*, ACM POPL Conference, San Francisco, 1990.

[2] Asperti, A. and Laneve, C., *Interaction systems*, Proc. International Workshop on Higher-order algebra, Logic and Term rewriting, Amsterdam, September 1993.

[3] Asperti, A. and Longo, G., **Categories, Types and Structures**, MIT Press, 1991.

[4] Barendregt, H., **The Lambda Calculus**, North Holland 1981.

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

[6] Danos, V. and Regnier, L., *Local and asynchronous beta-reduction*, Proc. LICS, Montreal, 1993.

[7] Giacalone, P., Mishra, p. and Prasad, S., *Facile: a symmetric integration of concurrent and functional programming*, International Journal of Parallel Programming, Vol 18, pp121–160, 1989.

[8] Lambek, J. and Scott, P.J., **Introduction to Higher-order Categorical Logic**, Cambridge University Press,1986.

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

[10] Milner, R., *Action structures and the π-calculus*, to appear in Proceedings of NATO Advanced Study Institute on **Proof and Computation** (held at Marktoberdorf, 1993), about 60 pages. (This paper combines of two previous reports: *Action structures*, Research Report ECS–LFCS–92–249, and *Action structures for the π-calculus*, Research Report ECS–LFCS–93–264, Laboratory for Foundations of Computer Science, Computer Science Department, Edinburgh University.

[11] Milner, R., *Action calculi I: axioms and applications*, Computer Science Department, Edinburgh University, 1993. Also appeared as *Action calculi, or syntactic action structures*, Proceedings of MFCS '93, Lecture Notes in Computer Science Vol 711, Springer-Verlag, pp105–121, 1993.

[12] Milner, R., *Action calculi II: π-nets with boxes and replication*, Computer Science Department, Edinburgh University, 1993.

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

[14] Petri, C.A., Fundamentals of a theory of asynchronous information flow, Proc. IFIP Congress '62, North Holland, pp386–390, 1962.

[15] Pierce, B.C., *Programming in the π-calculus*, Draft, Computer Science Department, University of Edinburgh, 1993.

[16] Pitts, A.M. and Stark, I.D.B., *Observable properties of higher-order functions that dynamically create local names, or: What's new?*, Proc. MFCS'93, Lecture Notes in Computer Science Vol 711, Springer-Verlag, pp122–141, 1993.

[17] Ritter, E., *Normalization for Typed Lambda Calculi with Explicit Substitution*, Paper presented at the 1993 Annual Conference of the European Association for Computer Science Logic, Swansea 1993.

[18] Sangiorgi, D., *Expressing mobility in process algebras: first-order and higher-order paradigms*, PhD Thesis, CST-99-93, Computer Science Department, University of Edinburgh, 1992.