Types and Models for Higher-Order Action Calculi

Philippa Gardner[1] and Masahito Hasegawa[2]

[1] Computing Laboratory, University of Cambridge
New Museums Site, Cambridge, CB2 3QG, UK
email: pag20@cl.cam.ac.uk
[2] Department of Computer Science, University of Edinburgh
The King's Buildings, Edinburgh, EH9 3JZ, Scotland
email: mhas@dcs.ed.ac.uk

Abstract. Milner introduced action calculi as a framework for representing models of interactive behaviour. He also introduced the higher-order action calculi, which add higher-order features to the basic setting. We present type theories for action calculi and higher-order action calculi, and give the categorical models of the higher-order calculi. As applications, we give a semantic proof of the conservativity of higher-order action calculi over action calculi, and a precise connection with Moggi's computational lambda calculus and notions of computation.

1 Introduction

Milner introduced *action calculi* as a framework for representing models of interactive behaviour [Mil96]. He also introduced two conservative extensions: *higher-order* action calculi [Mil94a], which add higher-order features to the basic setting, and *reflexive* action calculi [Mil94b], which give recursion in the presence of the higher-order features. Various examples, which explore the role of action calculi as a general framework, include the π-calculus [Mil96], Petri nets [Mil96] and their higher-order extensions [Mil94a], call-by-name and call-by-value versions of PCF [Jen95], and a type theory with ML-style references [CJLS97]. It remains an on-going research area to fully understand the dynamics of action calculi and obtain, for example, a general theory of bisimulation.

This paper focusses on higher-order action calculi; analogous results for the reflexive extension are given in [Has97b]. We solve two open problems: the connection between higher-order action calculi and typed λ-calculi, and the identification of the categorical models for the higher-order case. Milner first postulated the relationship between higher-order action calculi and typed λ-calculi in [Mil94a]. He showed how to obtain a cartesian closed category from a higher-order action calculi with an extra axiom corresponding to η-equality. However, he also observed that this extra axiom collapses the higher-order structure, in that the conservativity result over action calculi is lost.

One important fact about higher-order action calculi is that we can only substitute names and codes of the form $\ulcorner a \urcorner$, where a is a process. This restriction

is necessary, because arbitrary processes can cause side-effects and hence should not be duplicated nor discarded freely. We give a type-theoretic presentation of higher-order action calculi, where variables and λ-abstractions are the only terms that are substituted and the dynamics corresponds to term rewriting.

We also identify a class of categorical models of higher-order action calculi, by extending Power's models of action calculi [Pow96]. We use a cartesian category for interpreting the names and the codes, a symmetric monoidal category with a local preorder for interpreting arbitrary processes and their dynamics, and a symmetric monoidal adjunction for describing the interplay between the two categories. We give a sound interpretation of our higher-order type theory in these models. To show completeness, we require an extra axiom which corresponds to a η axiom for names.

There are two immediate applications of these results. First, we give a simple semantic proof of the conservativity of (static) higher-order action calculi over action calculi using standard results from category theory. Second, we observe that our type theory is very similar to Moggi's *computational λ-calculus* [Mog88], and make this intuition precise by relating the class of the models of higher-order theories and Moggi's semantic framework called *notions of computation* [Mog91]. Our results also clarify the difference between the two approaches. Moggi describes computational behaviour using monads in a category, whereas the dynamics of action calculi are interpreted by preorders.

This paper is organized as follows. In Section 2, we review action calculi and higher-order action calculi. Section 3 describes the type theories, and Section 4 the corresponding models. We give two applications of these results. Section 5 gives a semantic proof of conservativity of higher-order action calculi over action calculi, and in Section 6 we relate our type theories with Moggi's work by comparing the semantic models. We conclude in Section 7 by showing how our results fit the broader picture.

2 Preliminaries

In this section we review action calculi and higher-order action calculi. Most of the ideas can be found in Milner's original work [Mil96,Mil94a], although there are a few novel features including an additional axiom for higher-order action calculi.

2.1 Action Calculi

An action calculus is defined by a set of terms, an equational theory on the terms and a preorder on the equivalence classes of terms, called the *reaction relation* or the *dynamics*. It is generated from a *signature* $\mathbb{K} = (\mathsf{P}, \mathcal{K})$, which consists of a set P of *(basic) primes* denoted by $p, q, \ldots$ and a set $\mathcal{K}$ of *control operators*. Each control operator K of $\mathcal{K}$ is equipped with an arity rule $((m_1, n_1), \ldots (m_r, n_r)) \to (m, n)$, where the m's and n's are finite sequences of primes called *(tensor) arities*; we write ϵ for the empty sequence, (infix) $\otimes$ for concatenation and write

M for the set of arities. We overload primes with arities of length one, and call them *prime arities*. We assume a fixed denumerable set X of *names*, each of which is equipped with a prime. We let $x, y, \ldots$ range over names, and sometimes write x^p to indicate that x has the prime arity p.

Definition 1. Terms $a, b, c \ldots$ are constructed from the basic operators: *identity* $\mathbf{id}_m$, *composition* $\cdot$, *tensor* $\otimes$, *permutation* $\mathbf{p}_{m,n}$, *abstraction* (x^p), *datum* $\langle x^p \rangle$ and the *control operators* K. A term a is assigned a pair of arities (m, n), for which we write for $a : m \to n$, using the following rules:

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

$$\mathbf{p}_{m,n} : m \otimes n \to n \otimes m \qquad \frac{a : m \to n}{(x^p)a : p \otimes m \to n} \qquad \langle x^p \rangle : \epsilon \to p$$

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

where, in the derivation of $\mathsf{K}(a_1, \ldots, a_r)$, the arity rule of the control operator $\mathsf{K} \in \mathcal{K}$ is $((m_1, n_1), \ldots, (m_r, n_r)) \to (m, n)$. $\square$

Notation. We omit the arity subscripts on the basic operators when apparent. The notions of *free* and *bound name* are standard: (x) binds x and $\langle x \rangle$ represents a free occurrence of x. We write $a\{b/\langle x \rangle\}$ to denote the usual capture-avoiding substitution. The set of names free in $a, b, \ldots$ is denoted by $\mathsf{fn}(a, b, \ldots)$. Given a (possibly empty) sequence of names $\boldsymbol{x} = x_1^{p_1}, \ldots, x_r^{p_r}$ (we use bold letters like $\boldsymbol{x}$, $\boldsymbol{y}$ for sequences of names), we write $|\boldsymbol{x}|$ for $p_1 \otimes \ldots \otimes p_r$. All terms and expressions used are well formed, and all equations are between terms of the same arity. We let $(\boldsymbol{x})a$ denote $(x_1) \cdots (x_r)a$, where the x_i's are distinct, and let $\langle \boldsymbol{x} \rangle$ denote $\langle x_1 \rangle \otimes \cdots \otimes \langle x_r \rangle$, where $(\ )a$ is a and $\langle\ \rangle$ is $\mathbf{id}_\epsilon$.

Definition 2. The equational theory AC is the congruence relation on terms generated by the axioms of a strict monoidal category with a symmetry $\mathbf{p}$:

$$\begin{array}{ll}
a \cdot \mathbf{id} = a = \mathbf{id} \cdot a & a \cdot (b \cdot c) = (a \cdot b) \cdot c \\
a \otimes \mathbf{id}_\epsilon = a = \mathbf{id}_\epsilon \otimes a & a \otimes (b \otimes c) = (a \otimes b) \otimes c \\
\mathbf{id} \otimes \mathbf{id} = \mathbf{id} & (a \cdot b) \otimes (a' \cdot b') = (a \otimes a') \cdot (b \otimes b') \\
\mathbf{p}_{m,n} \cdot (b \otimes a) = (a \otimes b) \cdot \mathbf{p}_{m',n'} & \mathbf{p}_{l \otimes m,n} = (\mathbf{id}_l \otimes \mathbf{p}_{m,n}) \cdot (\mathbf{p}_{l,n} \otimes \mathbf{id}_m) \\
\mathbf{p}_{m,n} \cdot \mathbf{p}_{n,m} = \mathbf{id}_{m \otimes n} &
\end{array}$$

together with the following two axioms (σ) and (δ):

$$\begin{array}{lrcl}
(\sigma) & (\langle y \rangle \otimes \mathbf{id}_m) \cdot (x)a & = & a\{\langle y \rangle / \langle x \rangle\} \\
(\delta) & (x)((\langle x \rangle \otimes \mathbf{id}_m) \cdot a) & = & a \quad \text{where } x \notin \mathsf{fn}(a)
\end{array}$$

$\square$

It is an immediate consequence of these axioms that $\mathbf{id}_m = (\boldsymbol{x})\langle\boldsymbol{x}\rangle$ and $\mathbf{p}_{m,n} = (\boldsymbol{xy})\langle\boldsymbol{yx}\rangle$, where $|\boldsymbol{x}| = m$ and $|\boldsymbol{y}| = n$. Our axiomatisation is slightly different from the original presentation [Mil96] in the choice of primitive term constructors and axioms, but it is routine to see that these two versions are equivalent.

Definition 3. The *actions* are given by the equivalence classes obtained by quotienting the terms (Definition 1) by the equational theory AC (Definition 2). We overload notation and use $a, b, c, \ldots$ to denote actions as well as terms. An *action calculus* $\mathsf{AC}(\mathbb{K})$ consists of the actions and a preorder $\searrow$ on actions which preserves the arities and is closed under composition, tensor and abstraction, such that $\mathbf{id} \searrow a$ implies $a = \mathbf{id}$. The preorder $\searrow$ is also called the *dynamics* or the *reaction relation* of the action calculus. □

A *static* action calculus is an action calculus with the trivial dynamics (i.e. the identity relation on actions).

Example 4. A simple example of non-deterministic behaviour can be described in an action calculus by a control operator $\mathsf{plus} : ((m,n),(m,n)) \to (m,n)$ with the dynamics generated by the rules:

$$\mathsf{plus}(a,b) \ \searrow \ a \quad \text{and} \quad \mathsf{plus}(a,b) \ \searrow \ b.$$

This behaviour is analogous to a choice operator $+$ in process calculi with reduction rules $P + Q \to P$ and $P + Q \to Q$. We refer to [Mil96] for more interesting examples, including the π-calculus and Petri nets, which explore the use of action calculi as a framework for interactive behaviour. The simple example given here is enough to illustrate the ideas presented in this paper. □

2.2 Higher-Order Action Calculi

Higher-order action calculi [Mil94a] extend action calculi so that actions as well as names are substituted for names. The idea is that given an action $a : m \to n$, we package up the action in a *code* $\ulcorner a \urcorner : \epsilon \to (m \Rightarrow n)$, which uses the arity $(m \Rightarrow n)$ to keep a record of the arity structure of a. Such a "closure" of an action can be copied and discarded via substitution, and unpacked using an *application* control **ap**.

Definition 5. The *higher-order action calculus* $\mathsf{HAC}(\mathbb{K})$, where $\mathbb{K} = (\mathsf{P}, \mathcal{K})$ is a signature, is given by extending the definition of action calculi as follows:

1. the set of *higher-order primes* and the set of *higher-order arities* are constructed from the following grammars:

(basic) primes	$p_0, q_0 \ldots$	$\in$	P
higher-order primes	$p, q \ldots$	$::=$	$p_0 \mid m \Rightarrow n$
higher-order arities	$m, n \ldots$	$::=$	$p \mid m \otimes n \mid \epsilon$

2. the set of terms is generated by the rules in Definition 1, plus the rules

$$\frac{a : m \to n}{\ulcorner a \urcorner : \epsilon \to m \Rightarrow n} \qquad \mathbf{ap} : (m \Rightarrow n) \otimes m \to n$$

3. the equational theory is the set of equations upon terms generated from the axioms in Definition 2, plus the higher-order axioms

$$\begin{array}{lrcl} (\sigma_{\text{code}}) & (\ulcorner a \urcorner \otimes \mathbf{id}) \cdot (x)b & = & b\{\ulcorner a \urcorner / \langle x \rangle\} \\ (\beta) & (\ulcorner a \urcorner \otimes \mathbf{id}) \cdot \mathbf{ap} & = & a \\ (\eta_{\text{name}}) & \ulcorner (\langle x \rangle \otimes \mathbf{id}) \cdot \mathbf{ap} \urcorner & = & \langle x \rangle \end{array}$$

4. the dynamics is defined in the same way as the dynamics for action calculi.

□

In addition to the original axioms in [Mil94a], we include the axiom η_{name}. Notice that, for a code $\ulcorner a \urcorner$, we can prove $\ulcorner (\ulcorner a \urcorner \otimes \mathbf{id}) \cdot \mathbf{ap} \urcorner = \ulcorner a \urcorner$ using the β-axiom. The axiom η_{name} states the analogous equality for higher-order names. With this axiom, names and codes thus behave in a similar fashion. In section 4, we give further justification for η_{name} by studying the models.

Remark 6. In [Mil94a], Milner shows that adding a "strong" η axiom of the form $\ulcorner (a \otimes \mathbf{id}) \cdot \mathbf{ap} \urcorner = a$ collapses the structure, in the sense that $\mathsf{HAC}(\mathbb{K})$ with the strong η-axiom is not a conservative extension of $\mathsf{AC}(\mathbb{K})$. In section 5, we prove the conservativity result for our weaker axiom η_{name}. □

While it is possible to regard the higher-order axioms as a part of dynamics and study the properties of the reduction theory (as given in [Mil94a]), we concentrate in this paper on the study of equational theory, since our primary concern is to identify the semantic models of the calculi.

3 Type Theory

We introduce type theories corresponding to the closed fragments of action calculi and higher-order action calculi. In particular, let-bindings are used to represent the sharing of terms and the equational theory distinguishes between those terms which can be copied and discarded at will, the *values*, and those which cannot. Since action calculi are completely determined by the closed fragment [Gar95,Gar96], our type theories are as expressive as the corresponding action calculi. It is possible to work with type theories which relate to action calculi directly (see Remark 16, [BGHP96] and [BGHP97]); the type theories given here are simpler.

3.1 First-Order Theory

Let us fix a signature $\mathbb{K} = (\mathsf{P}, \mathcal{K})$ as introduced in the last section. The type theory $\mathsf{T}(\mathbb{K})$, to be introduced below, consists of sequents of the form $\Gamma \vdash t : n$,

where the *context* $\Gamma = x_1^{p_1}, \ldots, x_m^{p_m}$ is a sequence of distinct names (sometimes written $\boldsymbol{x}^m$), t is a term as introduced below and n is an arity. This sequent corresponds to a closed action $a : |\Gamma| \to n$ in $\mathsf{AC}(\mathbb{K})$, where $|\Gamma|$ denotes $p_1 \otimes \ldots \otimes p_m$.

Definition 7. The set of *type-theoretic terms* (usually just called *terms*) over $\mathbb{K}$ is defined by the following grammar:

$$t \ ::= \ x \mid 0 \mid t \otimes t \mid \mathsf{let}\ \boldsymbol{x}\ \mathsf{be}\ t\ \mathsf{in}\ t \mid \mathsf{K}((\boldsymbol{x})t, \ldots, (\boldsymbol{x})t; t)$$

where we assume that $t \otimes 0 \equiv 0 \otimes t \equiv t$ and $s \otimes (t \otimes u) \equiv (s \otimes t) \otimes u$ for terms s, t and u (strict associativity). □

The term $\mathsf{K}((\boldsymbol{x}_1)t_1, .., (\boldsymbol{x}_r)t_r; t)$ binds the variables from sequence $\boldsymbol{x}_i$ in t_i. Plotkin has pointed out that this can be viewed as a variant of Aczel's general binding operators [Acz80], and this issue is discussed in [BGHP97]. See Example 15 for a concrete example. The term $\mathsf{let}\ \boldsymbol{x}\ \mathsf{be}\ s\ \mathsf{in}\ t$ binds the variables from sequence $\boldsymbol{x}$ in t. We allow α-conversion, and write $t\{u/x\}$ for the standard capture-free substitution.

Definition 8. A term is *well-typed* if it can be shown to annotate a sequent using the following rules.

$$\Gamma, x^p \vdash x : p \qquad \Gamma \vdash 0 : \epsilon \qquad \frac{\Gamma \vdash s : m \quad \Gamma \vdash t : n}{\Gamma \vdash s \otimes t : m \otimes n} \qquad \frac{\Gamma \vdash s : m \quad \Gamma, \boldsymbol{x}^m \vdash t : n}{\Gamma \vdash \mathsf{let}\ \boldsymbol{x}\ \mathsf{be}\ s\ \mathsf{in}\ t : n}$$

$$\frac{\Gamma, \boldsymbol{x}_i^{m_i} \vdash s_i : n_i \ (1 \le i \le r) \quad \Gamma \vdash t : m}{\Gamma \vdash \mathsf{K}((\boldsymbol{x}_1)s_1, \ldots, (\boldsymbol{x}_r)s_r; t) : n} \qquad \frac{\Gamma, x^p, y^q, \Gamma' \vdash t : m}{\Gamma, y^q, x^p, \Gamma' \vdash t : m}$$

where, in the derivation of $\mathsf{K}(\ldots)$, K has arity rule $((m_1, n_1), \ldots, (m_r, n_r)) \to (m, n)$. □

The equality of the type theory should present no surprises; it gives the behaviour of the let-bindings, which describes the sharing of resources. For example, the term $\mathsf{let}\ x\ \mathsf{be}\ t\ \mathsf{in}\ x \otimes x$ describes a term t which is shared by two x's. It does not equal the term $t \otimes t$ in general.

Definition 9. The *equality judgement* $\Gamma \vdash s = t : n$, where $\Gamma \vdash s : n$ and $\Gamma \vdash t : n$, is defined as a congruence relation on well-typed terms of the same arity under the same context generated from the following axioms.

$$\begin{array}{lcl}
\mathsf{let}\ x\ \mathsf{be}\ y\ \mathsf{in}\ t & = & t\{y/x\} \\
\mathsf{let}\ \boldsymbol{x}\ \mathsf{be}\ t\ \mathsf{in}\ \boldsymbol{x} & = & t \\
\mathsf{let}\ \boldsymbol{x}\ \mathsf{be}\ (\mathsf{let}\ \boldsymbol{y}\ \mathsf{be}\ s\ \mathsf{in}\ t)\ \mathsf{in}\ u & = & \mathsf{let}\ \boldsymbol{y}\ \mathsf{be}\ s\ \mathsf{in}\ \mathsf{let}\ \boldsymbol{x}\ \mathsf{be}\ t\ \mathsf{in}\ u \\
\mathsf{let}\ \boldsymbol{x}, \boldsymbol{y}\ \mathsf{be}\ s \otimes t\ \mathsf{in}\ u & = & \mathsf{let}\ \boldsymbol{x}\ \mathsf{be}\ s\ \mathsf{in}\ \mathsf{let}\ \boldsymbol{y}\ \mathsf{be}\ t\ \mathsf{in}\ u \\
s \otimes (\mathsf{let}\ \boldsymbol{x}\ \mathsf{be}\ t\ \mathsf{in}\ u) & = & \mathsf{let}\ \boldsymbol{x}\ \mathsf{be}\ t\ \mathsf{in}\ s \otimes u \\
(\mathsf{let}\ \boldsymbol{x}\ \mathsf{be}\ s\ \mathsf{in}\ t) \otimes u & = & \mathsf{let}\ \boldsymbol{x}\ \mathsf{be}\ s\ \mathsf{in}\ t \otimes u \\
\mathsf{K}((\boldsymbol{x}_1)t_1, \ldots, (\boldsymbol{x}_r)t_r; \mathsf{let}\ \boldsymbol{x}\ \mathsf{be}\ s\ \mathsf{in}\ t) & = & \mathsf{let}\ \boldsymbol{x}\ \mathsf{be}\ s\ \mathsf{in}\ \mathsf{K}((\boldsymbol{x}_1)\mathsf{t}_1, \ldots, (\boldsymbol{x}_\mathsf{r})\mathsf{t}_\mathsf{r}; \mathsf{t})
\end{array}$$

□

Note that both sides of an axiom must have the same type under the same context. For instance, in the fourth axiom, the variables in the $\boldsymbol{x}$ cannot be free in t (written $\boldsymbol{x} \notin \mathsf{fn}(t)$), and in the last axiom $\boldsymbol{x} \notin \bigcup_{i=1}^{r} \mathsf{fn}((\boldsymbol{x}_i)t_i)$.

We give the formal justification of our assertion that the type theory $\mathsf{T}(\mathbb{K})$ corresponds to closed actions of $\mathsf{AC}(\mathbb{K})$, by giving translations between these two systems which are sound and inverse to each other.

Definition 10. For every $a : m \to n$ in $\mathsf{AC}(\mathbb{K})$ and sequence of distinct names $\boldsymbol{x}$ with $|\boldsymbol{x}| = m$ which are not free in a, we define a term $\Phi_{\boldsymbol{x}}(a)$ by induction on the structure of a:

$$\begin{aligned}
\Phi_{\boldsymbol{x}}(\mathbf{id}_m) &= \boldsymbol{x} \\
\Phi_{\boldsymbol{x}}(a \cdot b) &= \mathsf{let}\ \boldsymbol{y}\ \mathsf{be}\ \Phi_{\boldsymbol{x}}(a)\ \mathsf{in}\ \Phi_{\boldsymbol{y}}(b), \quad \text{where } \boldsymbol{y} \notin \mathsf{fn}(b) \\
\Phi_{\boldsymbol{x},\boldsymbol{y}}(a \otimes b) &= \Phi_{\boldsymbol{x}}(a) \otimes \Phi_{\boldsymbol{y}}(b) \\
\Phi_{\boldsymbol{x},\boldsymbol{y}}(\mathbf{p}_{m,n}) &= \boldsymbol{y} \otimes \boldsymbol{x}, \quad \text{where } |\boldsymbol{x}| = m \text{ and } |\boldsymbol{y}| = n \\
\Phi_{x^p,\boldsymbol{x}}((y^p)a) &= \Phi_{\boldsymbol{x}}(a)\{x/y\} \\
\Phi_{\emptyset}(\langle x^p \rangle) &= x \\
\Phi_{\boldsymbol{x}}(\mathsf{K}(a_1, \ldots, a_r)) &= \mathsf{K}((\boldsymbol{x}_1)\Phi_{\boldsymbol{x}_1}(a_1), \ldots, (\boldsymbol{x}_r)\Phi_{\boldsymbol{x}_r}(a_r); \boldsymbol{x})
\end{aligned}$$

□

Proposition 11.

1. *For $a : m \to n$ such that $\mathsf{fn}(a)$ is contained in $\boldsymbol{y}$ (with $|\boldsymbol{y}| = l$), we have $\boldsymbol{y}^l, \boldsymbol{x}^m \vdash \Phi_{\boldsymbol{x}}(a) : n$.*
2. *If $a = b : m \to n$ and $\mathsf{fn}(a, b)$ is contained in $\boldsymbol{y}$ (with $|\boldsymbol{y}| = l$), we have $\boldsymbol{y}^l, \boldsymbol{x}^m \vdash \Phi_{\boldsymbol{x}}(a) = \Phi_{\boldsymbol{x}}(b) : n$.* □

Definition 12. Given a sequent $\Gamma \vdash t : n$ in $\mathsf{T}(\mathbb{K})$, we define a closed term $\Psi(\Gamma \vdash t : n) : |\Gamma| \to n$ by induction on the derivation of sequents[1], where we write Δ_m for the action $(\boldsymbol{x})\langle \boldsymbol{x}\boldsymbol{x} \rangle : m \to m \otimes m$, and $\boldsymbol{\omega}_m$ for $(\boldsymbol{x})\mathbf{id}_\epsilon : m \to \epsilon$:

$$\begin{aligned}
\Psi(\Gamma, x^p \vdash x : p) &= \boldsymbol{\omega}_{|\Gamma|} \otimes \mathbf{id}_p \\
\Psi(\Gamma \vdash 0 : \epsilon) &= \boldsymbol{\omega}_{|\Gamma|} \\
\Psi(\Gamma \vdash s \otimes t : n_1 \otimes n_2) &= \Delta_{|\Gamma|} \cdot (\Psi(\Gamma \vdash s : n_1) \otimes \Psi(\Gamma \vdash t : n_2)) \\
\Psi(\Gamma \vdash \mathsf{let}\ \boldsymbol{y}\ \mathsf{be}\ s\ \mathsf{in}\ t : n) &= \Delta_{|\Gamma|} \cdot (\mathbf{id}_{|\Gamma|} \otimes \Psi(\Gamma \vdash s : m)) \cdot \Psi(\Gamma, \boldsymbol{y}^m \vdash t : n) \\
\Psi(\Gamma \vdash \mathsf{K}((\boldsymbol{x}_1)s_1, \ldots; t) : n) &= \Delta_{|\Gamma|} \cdot (\mathbf{id}_{|\Gamma|} \otimes \Psi(\Gamma \vdash t : m)) \cdot \\
&\qquad (\boldsymbol{x})\mathsf{K}((\langle \boldsymbol{x} \rangle \otimes \mathbf{id}_{m_1}) \cdot \Psi(\Gamma, \boldsymbol{x}_1^{m_1} \vdash s_1 : n_1), \ldots) \\
\Psi(\Gamma, y^q, x^p, \Gamma' \vdash t : n) &= (\mathbf{id}_{|\Gamma|} \otimes \mathbf{p}_{q,p} \otimes \mathbf{id}_{|\Gamma'|}) \cdot \Psi(\Gamma, x^p, y^q, \Gamma' \vdash t : n)
\end{aligned}$$

□

[1] We are being slightly loose with notation. We should use a notation which incorporates derivations of sequents, and show that every derivation of a sequent gives rise to equal actions via Ψ.

Proposition 13.

1. *Given* $\Gamma \vdash t : n$, *it follows that* $\Psi(\Gamma \vdash t : n)$ *is a closed term of arity* $|\Gamma| \to n$.
2. *Given* $\Gamma \vdash s = t : n$, *it follows that* $\Psi(\Gamma \vdash s : n) = \Psi(\Gamma \vdash t : n)$. □

Theorem 14.

1. *Given* $\Gamma \vdash t : n$, *we have* $\Gamma \vdash \Phi_\Gamma(\Psi(\Gamma \vdash t : n)) = t : n$ *in* $\mathsf{T}(\mathbb{K})$.
2. *Given a closed term* $a : m \to n$ *and a fresh distinct sequence of names* $\boldsymbol{x}^m$, *we have* $\Psi(\Phi_{\boldsymbol{x}}(a)) = a$. □

These results establish the connection between the *static* action calculus and our type theory. It is routine to extend them to include the dynamics. The dynamics on the type theory $\mathsf{T}(\mathbb{K})$ is a preorder $\searrow$ on the equivalence classes of the well-typed terms which respects the types and contexts, and is closed under all term constructions except controls, such that $\boldsymbol{x} \searrow t$ implies $t = \boldsymbol{x}$. It is easy to see that the dynamics of an action calculus determines that of the corresponding type theory, and vice versa. Such dynamics can be regarded as a graph rewriting system on sharing graphs (see [Has97b,Mil96]).

Example 15. The control plus for non-determinism in Example 4 can be accommodated in our type theory as

$$\frac{\Gamma, \boldsymbol{x}^m \vdash s : n \quad \Gamma, \boldsymbol{y}^m \vdash t : n \quad \Gamma \vdash u : m}{\Gamma \vdash \mathsf{plus}((\boldsymbol{x})s, (\boldsymbol{y})t; u) : n}$$

with dynamics given by

$$\mathsf{plus}((\boldsymbol{x})s, (\boldsymbol{y})t; u) \searrow \mathsf{let}\ \boldsymbol{x}\ \mathsf{be}\ u\ \mathsf{in}\ s \quad \text{and} \quad \mathsf{plus}((\boldsymbol{x})s, (\boldsymbol{y})t; u) \searrow \mathsf{let}\ \boldsymbol{y}\ \mathsf{be}\ u\ \mathsf{in}\ t.$$

□

Remark 16. An alternative choice is to work with a *two-context* type theory, whose sequents have the form $\Gamma; \Sigma \vdash t : n$. Such a type theory is introduced in [BGHP96,BGHP97]. It has a direct connection with the corresponding action calculus, in that the above sequent corresponds to an action $a : |\Sigma| \to n$ with free names contained in Γ. It also gives a simple connection with intuitionistic linear type theories with two contexts [Ben95,Bar96]. From the results in [Gar95], we know that these type theories are equivalent. □

3.2 Higher-Order Theory

We extend the type theory $\mathsf{T}(\mathbb{K})$ above to incorporate higher-order features. As before, we fix a signature $\mathbb{K} = (\mathsf{P}, \mathcal{K})$.

Definition 17. The *higher-order theory* $\mathsf{HT}(\mathbb{K})$ is given by extending the definition of the first-order theory $\mathsf{T}(\mathbb{K})$ as follows:

1. the types are the higher-order arities given in Definition 5;
2. the terms are those for $\mathsf{T}(\mathbb{K})$ plus *lambda abstraction* $\lambda \boldsymbol{x}.t$ and *application* st where s and t are terms; the associated typing rules are

$$\frac{\Gamma, \boldsymbol{x}^m \vdash t : n}{\Gamma \vdash \lambda \boldsymbol{x}.t : m \Rightarrow n} \qquad \frac{\Gamma \vdash s : m \Rightarrow n \quad \Gamma \vdash t : m}{\Gamma \vdash st : n}$$

3. the equality judgement is defined as Definition 9, with additional axioms:

$$\begin{array}{lrcl} (\sigma_v) & \mathsf{let}\ x\ \mathsf{be}\ \lambda \boldsymbol{y}.s\ \mathsf{in}\ t & = & t\{\lambda \boldsymbol{y}.s/x\} \\ (\beta) & (\lambda \boldsymbol{x}.s)t & = & \mathsf{let}\ \boldsymbol{x}\ \mathsf{be}\ t\ \mathsf{in}\ s \\ (\eta_0) & \lambda \boldsymbol{x}.y\boldsymbol{x} & = & y \\ & & & \\ & (\mathsf{let}\ \boldsymbol{x}\ \mathsf{be}\ s\ \mathsf{in}\ t)u & = & \mathsf{let}\ \boldsymbol{x}\ \mathsf{be}\ s\ \mathsf{in}\ tu \\ & s(\mathsf{let}\ \boldsymbol{x}\ \mathsf{be}\ t\ \mathsf{in}\ u) & = & \mathsf{let}\ \boldsymbol{x}\ \mathsf{be}\ t\ \mathsf{in}\ su \end{array}$$

□

It is straightforward to extend the translations between $\mathsf{T}(\mathbb{K})$ and the closed actions of $\mathsf{AC}(\mathbb{K})$ to translations between $\mathsf{HT}(\mathbb{K})$ and the closed actions of $\mathsf{HAC}(\mathbb{K})$, which are sound and inverse to each other. We define the translations below. It is easy to extend the results given in Proposition 11, Proposition 13 and Theorem 14 to the higher-order setting.

Definition 18.
For every action $a : m \to n$ in $\mathsf{HAC}(\mathbb{K})$ and a sequence of distinct names $\boldsymbol{x}$ with $|\boldsymbol{x}| = m$ which are not free in a, we define a term $\Phi_{\boldsymbol{x}}(a)$ by induction on the structure of a, using definition 10 and the additional cases:

$$\begin{array}{l} \Phi_{\emptyset}(\ulcorner a \urcorner) = \lambda \boldsymbol{x}.\Phi_{\boldsymbol{x}}(a) \ \text{ where } \boldsymbol{x} \notin \mathsf{fn}(a) \\ \Phi_{y,\boldsymbol{x}}(\mathbf{ap}) = y\boldsymbol{x} \end{array}$$

Given a sequent $\Gamma \vdash t : n$ in $\mathsf{HT}(\mathbb{K})$, we define a closed action $\Psi(\Gamma \vdash t : n) : |\Gamma| \to n$ by induction on the derivation of sequents, using definition 12 and the additional cases:

$$\begin{array}{ll} \Psi(\Gamma \vdash \lambda \boldsymbol{y}.t : m \Rightarrow n) & = (\boldsymbol{x})^{\ulcorner}(\langle \boldsymbol{x} \rangle \otimes \mathbf{id}_m) \cdot \Psi(\Gamma, \boldsymbol{y}^m \vdash t : n)^{\urcorner} \ \text{ where } |\boldsymbol{x}| = |\Gamma| \\ \Psi(\Gamma \vdash st : n) & = \Delta_{|\Gamma|} \cdot (\Psi(\Gamma \vdash s : m \Rightarrow n) \otimes \Psi(\Gamma \vdash t : m)) \cdot \mathbf{ap} \end{array}$$

□

4 Categorical Models

We base our models of higher-order action calculi on Power's models of action calculi [Pow96]. Power's models consist of a cartesian category for interpreting the names, a symmetric monoidal category for interpreting arbitrary actions, and a functor for relating the cartesian category with the symmetric monoidal category. We extend Power's idea to the higher-order case, where the cartesian

category interprets the values (the names and codes), and adjunctions describe the interplay between the two categories. This approach enables us to describe the models of action calculi and higher-order action calculi in a simple uniform way. We also note that similar structure has been used for describing models of intuitionistic linear type theory [Ben95,Bar96] in a slightly different but related context (see [BGHP96], [BGHP97] and Section 7).

4.1 Models of The First-Order Theories

First we describe the models for action calculi, which we call *action models*. The *carrier* of an action model is a triple $(\mathcal{C}, \mathcal{S}, \mathcal{F})$, where $\mathcal{C}$ is a strict cartesian category, $\mathcal{S}$ is a strict symmetric monoidal category, and $\mathcal{F} : \mathcal{C} \to \mathcal{S}$ is an identity-on-objects strict symmetric monoidal functor. Therefore we identify $\mathrm{Obj}(\mathcal{C})$ with $\mathrm{Obj}(\mathcal{S})$: $\mathcal{F}$ maps the terminal object of $\mathcal{C}$ to the unit object of $\mathcal{S}$, and binary products to monoidal products, preserving the symmetry. In addition, each action model is specified by a signature $\mathbb{K}$ and provides an interpretation function of the prime arities as objects of $\mathcal{C}$, and of control operators as natural transformations.

Remark 19. In [Pav96], Pavlovic gives related models of action calculi, consisting of a single symmetric monoidal category with a sub-cartesian category. □

Definition 20. An *action model* over signature $\mathbb{K}$, denoted by $\mathcal{A}$, is a carrier $(\mathcal{C}, \mathcal{S}, \mathcal{F})$ equipped with a function $[\![\_]\!]_{\mathsf{P}} : \mathsf{P} \to \mathrm{Obj}(\mathcal{C})$, and for each operator K with arity rule $((m_1, n_1), \ldots, (m_r, n_r)) \to (m, n)$, a natural transformation

$$[\![\mathsf{K}]\!]_{\mathcal{K}} : \mathcal{S}(\mathcal{F}(\_) \otimes [\![m_1]\!], [\![n_1]\!]) \times \ldots \times \mathcal{S}(\mathcal{F}(\_) \otimes [\![m_r]\!], [\![n_r]\!]) \to \mathcal{S}(\mathcal{F}(\_) \otimes [\![m]\!], [\![n]\!])$$

where $[\![m]\!]$ is defined inductively by $[\![\epsilon]\!] = I$ (the terminal object of $\mathcal{C}$, equivalently the unit object of $\mathcal{S}$) and $[\![p \otimes m]\!] = [\![p]\!]_{\mathsf{P}} \otimes [\![m]\!]$. An action model is *faithful* if the functor $\mathcal{F}$ is faithful. It is *small* if the categories $\mathcal{C}$, $\mathcal{S}$ are small. □

Notation. Where convenient, we omit the subscripts from $[\![\_]\!]_{\mathsf{P}}$ and $[\![\_]\!]_{\mathcal{K}}$. We sometimes write $[\![\_]\!]^{\mathcal{A}}$ to emphasise the particular action model $\mathcal{A}$ under consideration.

Definition 21. An *action morphism* $f : \mathcal{A}_1 \to \mathcal{A}_2$ between two action models over signature $\mathbb{K}$ is a pair (f_c, f_s) where $f_c : \mathcal{C}_1 \to \mathcal{C}_2$ is a functor preserving finite products strictly, and $f_s : \mathcal{S}_1 \to \mathcal{S}_2$ is a strict symmetric monoidal functor such that $\mathcal{F}_2 \circ f_c = f_s \circ \mathcal{F}_1$, for each $p \in \mathsf{P}$ we have $f_c([\![p]\!]_{\mathsf{P}}^{\mathcal{A}_1}) = [\![p]\!]_{\mathsf{P}}^{\mathcal{A}_2}$, and, for each operator K with arity rule $((m_1, n_1), \ldots, (m_r, n_r)) \to (m, n)$, we have $f_s(([\![\mathsf{K}]\!]_{\mathcal{K}}^{\mathcal{A}_1})_{(\_)}(..)) = ([\![\mathsf{K}]\!]_{\mathcal{K}}^{\mathcal{A}_2})_{f_c(\_)}(f_s(..))$. □

The *category of small action models*, $\mathbf{AMod}(\mathbb{K})$, is the category whose objects are the small action models, and whose morphisms are the action morphisms, with the obvious identities and composition.

Now we give the interpretation of the type theory $\mathsf{T}(\mathbb{K})$ in an arbitrary action model $\mathcal{A}$, and state the soundness and completeness results. Given the interpretation of arities $[\![\_]\!] : \mathsf{M} \to \mathsf{Obj}(\mathcal{C})$, we extend this to contexts by defining $[\![\Gamma]\!] = [\![|\Gamma|]\!]$.

Definition 22. Given type theory $\mathsf{T}(\mathbb{K})$ and action model $\mathcal{A}$, the *interpretation* $[\![\_]\!]$ of sequents $\Gamma \vdash t : m$ in the type theory as morphisms $[\![\Gamma]\!] \to [\![m]\!]$ in $\mathcal{S}$ is defined by induction on the derivation of sequents:

$$
\begin{array}{ll}
[\![\Gamma, x^p \vdash x : p]\!] & = \mathcal{F}(\pi'_{[\![\Gamma]\!],[\![p]\!]}) \\
[\![\Gamma \vdash 0 : \epsilon]\!] & = \mathcal{F}(!_{[\![\Gamma]\!]}) \\
[\![\Gamma \vdash s \otimes t : m \otimes n]\!] & = \mathcal{F}(\Delta_{[\![\Gamma]\!]}); ([\![\Gamma \vdash s : m]\!] \otimes [\![\Gamma \vdash t : n]\!]) \\
[\![\Gamma \vdash \mathsf{let}\ \boldsymbol{x}\ \mathsf{be}\ s\ \mathsf{in}\ t : n]\!] & = \mathcal{F}(\Delta_{[\![\Gamma]\!]}); (id_{[\![\Gamma]\!]} \otimes [\![\Gamma \vdash s : m]\!]); [\![\Gamma, \boldsymbol{x}^m \vdash t : n]\!] \\
[\![\Gamma \vdash \mathsf{K}((\boldsymbol{x}_1)s_1, \ldots, (\boldsymbol{x}_r)s_r; t) : n]\!] & = \\
\multicolumn{2}{r}{\mathcal{F}(\Delta_{[\![\Gamma]\!]}); (id_{[\![\Gamma]\!]} \otimes [\![\Gamma \vdash t : m]\!]);} \\
\multicolumn{2}{r}{[\![\mathsf{K}]\!]_{[\![\Gamma]\!]}([\![\Gamma, \boldsymbol{x}_1^{m_1} \vdash s_1 : n_1]\!], \ldots, [\![\Gamma, \boldsymbol{x}_r^{m_r} \vdash s_r : n_r]\!])} \\
[\![\Gamma, y^q, x^p, \Gamma' \vdash t : n]\!] & = (id_{[\![\Gamma]\!]} \otimes c_{[\![q]\!],[\![p]\!]} \otimes id_{[\![\Gamma']\!]}); [\![\Gamma, x^p, y^q, \Gamma' \vdash t : n]\!]
\end{array}
$$

where π', $!$ and Δ are the projection, terminal map and diagonal map in the cartesian category $\mathcal{C}$ respectively, and c is the symmetry of $\mathcal{S}$. □

The proof of the soundness of the interpretation is routine. Completeness is proved by defining a term model constructed from $\mathsf{T}(\mathbb{K})$ (or the closed fragment of $\mathsf{AC}(\mathbb{K})$ [Gar95,Pow96]). The basic idea is that the morphisms in the cartesian category $\mathcal{C}$ are constructed from (the equivalence classes of) sequents of the form $\Gamma \vdash \boldsymbol{x} : m$ and the symmetric monoidal category $\mathcal{S}$ is constructed from (the equivalence classes of) arbitrary sequents $\Gamma \vdash t : m$. The functor $\mathcal{F}$ is the obvious inclusion functor from $\mathcal{C}$ to $\mathcal{S}$. The interpretation $[\![\_]\!]_{\mathsf{P}}$ and $[\![\_]\!]_{\mathcal{K}}$ are then determined routinely.

Theorem 23.

1. *(Soundness) Given $\Gamma \vdash s = t : n$ in $\mathsf{T}(\mathbb{K})$, we have $[\![\Gamma \vdash s : n]\!] = [\![\Gamma \vdash t : n]\!]$ in any action model.*
2. *(Completeness) Given derivations $\Gamma \vdash s : n$ and $\Gamma \vdash t : n$ in $\mathsf{T}(\mathbb{K})$, if $[\![\Gamma \vdash s : n]\!] = [\![\Gamma \vdash t : n]\!]$ in every action model then $\Gamma \vdash s = t : n$.*
3. *(Initiality) There is an initial term model $\mathcal{A}_0$ in $\mathbf{AMod}(\mathbb{K})$.* □

The results above only deal with the static part of the type theory (hence action calculi). We can give a semantic interpretation of dynamics in an action model as a *local preorder* on the symmetric monoidal category, such that the arrows coming from the cartesian category are minimal.

Example 24. To interpret an action calculus with non-determinism primitive plus in Example 4, one may choose $\mathcal{C}$ as (the strict equivalent of) the category of sets and functions, and $\mathcal{S}$ as (the strict equivalent of) the category of sets and total relations; $\mathcal{F}$ is then the obvious inclusion functor. The interpretation of plus is

simply given by the union of relations. We can then interpret the dynamics by the local preorder of the inclusion of relation; obviously a total relation is minimal with respect to this preorder if and only if it is a function, i.e. it comes from $\mathcal{C}$. A key feature of action calculi is that the dynamics are not usually closed under the control operators. Notice that in this model the preorder is closed under the natural transformation, so the model does not capture this feature. □

4.2 Higher-Order Extension

The intuition behind the definition of an action model with carrier $(\mathcal{C}, \mathcal{S}, \mathcal{F})$ is that the cartesian category $\mathcal{C}$ describes the behaviour of names in action calculi. In the higher-order case, the codes (λ-terms) have a similar behaviour to the names in that they can be substituted for names of the appropriate arity. Our definition of higher-order models extends action models to reflect this fact, by requiring that the functor $\mathcal{F}(\_) \otimes X : \mathcal{C} \to \mathcal{S}$ has a right adjoint representing the arrow construction, which naturally extends the definition of exponents in cartesian closed categories given by right adjoint of the product functors $(\_) \times X$.

Definition 25. A *higher-order action model* over signature $\mathbb{K}$ is an action model with carrier $(\mathcal{C}, \mathcal{S}, \mathcal{F})$ such that the functor $\mathcal{F}(\_) \otimes X : \mathcal{C} \to \mathcal{S}$ has a (chosen) right adjoint $X \Rightarrow \_ : \mathcal{S} \to \mathcal{C}$ for each object X, where the definition of $[\![m]\!]$ is adapted to the higher-order case by an additional requirement that $[\![m \Rightarrow n]\!] = [\![m]\!] \Rightarrow [\![n]\!]$. Again, a higher-order action model is *faithful* when the functor $\mathcal{F}$ is faithful, and it is *small* if the categories $\mathcal{C}$ and $\mathcal{S}$ are small. □

We write $\mathbf{ap} : (X \Rightarrow Y) \otimes X \to Y$ (in $\mathcal{S}$) for (components of) the counit of the adjunction, and $f^* : A \to X \Rightarrow B$ in $\mathcal{C}$ for the adjunct of $f : A \otimes X \to B$ in $\mathcal{S}$, i.e. f^* is the unique arrow satisfying $(\mathcal{F}(f^*) \otimes X); \mathbf{ap} = f$.

Definition 26. Let $\mathcal{H}_1$ with carrier $(\mathcal{C}, \mathcal{S}, \mathcal{F})$ and $\mathcal{H}_2$ with carrier $(\mathcal{C}', \mathcal{S}', \mathcal{F}')$ be higher-order action models over the same signature $\mathbb{K}$. A *higher-order action morphism* $f : \mathcal{H}_1 \to \mathcal{H}_2$ is an action morphism (f_c, f_s) such that it is a map of adjoints ([ML71], p. 97) from $\mathcal{F}(\_) \otimes X \dashv X \Rightarrow (\_)$ to $\mathcal{F}'(\_) \otimes f_s(X) \dashv f_s(X) \Rightarrow' (\_)$: that is, $f_c(X \Rightarrow (\_)) = f_s(X) \Rightarrow' f_s(\_)$ holds and the following diagram commutes for each X, A and B.

$$\begin{array}{ccc}
\mathcal{S}(\mathcal{F}(A) \otimes X, B) & \xrightarrow{\quad (_)^* \quad} & \mathcal{C}(A, X \Rightarrow B) \\
{\scriptstyle f_s(_)}\downarrow & & \downarrow{\scriptstyle f_c(_)} \\
\mathcal{S}'(f_s(\mathcal{F}(A) \otimes X), f_s(B)) & & \mathcal{C}'(f_c(A), f_c(X \Rightarrow B)) \\
\| & & \| \\
\mathcal{S}'(\mathcal{F}'(f_c(A)) \otimes' f_s(X), f_s(B)) & \xrightarrow{\quad (_)^* \quad} & \mathcal{C}'(f_c(A), f_s(X) \Rightarrow' f_s(B))
\end{array}$$

□

The *category of small higher-order action models*, $\mathbf{HAMod}(\mathbb{K})$, is the category whose objects are the small higher-order action models, and whose morphisms are the higher-order action morphisms, with the obvious identities and composition.

The interpretation of the type theory $\mathsf{HT}(\mathcal{K})$ in an arbitrary higher-order action model $\mathcal{H}$ is given as an extension of Definition 22 to account for λ-abstraction and application.

Definition 27. Given type theory $\mathsf{HT}(\mathbb{K})$ and higher-order action model $\mathcal{H}$, the *interpretation* $[\![\_]\!]$ of sequents $\Gamma \vdash t : m$ in the type theory as morphisms $[\![\Gamma \vdash t : m]\!] : [\![\Gamma]\!] \to [\![m]\!]$ in $\mathcal{S}$ is defined by induction on the structure of the derivation of sequents, as given in definition 22 and the additional cases:

$$\begin{aligned} [\![\Gamma \vdash \lambda \boldsymbol{x}.t : m \Rightarrow n]\!] &= \mathcal{F}([\![\Gamma, \boldsymbol{x}^m \vdash t : n]\!]^*) \\ [\![\Gamma \vdash st : n]\!] &= \mathcal{F}(\Delta_{[\![\Gamma]\!]}); ([\![\Gamma \vdash s : m \Rightarrow n]\!] \otimes [\![\Gamma \vdash t : m]\!]); \mathbf{ap} \end{aligned}$$

□

While soundness of the interpretation is routinely checked, the construction of the term model from the type theory (or the closed fragment of the higher-order action calculus) requires some careful calculation. In this case the cartesian category part is given by the values: that is, the variables and λ-abstractions. The non-trivial part is to check that $X \Rightarrow (\_)$ does give a right adjoint of $\mathcal{F}(\_) \otimes X$, which requires the axiom η_0 (η_{name} in higher-order action calculi). Without this axiom, $X \Rightarrow (\_)$ fails to be a functor, though still it is a semifunctor, and in fact gives rise to a right semiadjoint [Hay85] of $\mathcal{F}(\_) \otimes X$. In this sense, the semantic role of the axiom η_0 (η_{name}) is similar to that of the η axiom of the simply typed lambda calculus.

Theorem 28.

1. *(Soundness) Given $\Gamma \vdash s = t : n$ in $\mathsf{HT}(\mathbb{K})$, we have $[\![\Gamma \vdash s : n]\!] = [\![\Gamma \vdash t : n]\!]$ in any higher-order action model.*
2. *(Completeness) Given derivations $\Gamma \vdash s : n$ and $\Gamma \vdash t : n$ in $\mathsf{HT}(\mathbb{K})$, if $[\![\Gamma \vdash s : n]\!] = [\![\Gamma \vdash t : n]\!]$ in every higher-order action model then $\Gamma \vdash s = t : n$ in $\mathsf{HT}(\mathbb{K})$.*
3. *(Initiality) There is an initial term model $\mathcal{H}_0$ in $\mathbf{HAMod}(\mathbb{K})$.* □

Remark 29. In a higher-order action model, it is easy to show using the Yoneda lemma that

$$\begin{aligned} & \mathbf{Sets}^{\mathcal{C}^{\mathrm{op}}}(\textstyle\prod_{i=1,\ldots,r} \mathcal{S}(\mathcal{F}(\_) \otimes A_i, B_i), \mathcal{S}(\mathcal{F}(\_) \otimes A, B)) \\ \simeq\ & \mathcal{S}(\textstyle\bigotimes_{i=1,\ldots,r} \mathcal{F}(A_i \Rightarrow B_i) \otimes A, B) \\ \simeq\ & \mathcal{C}(\textstyle\prod_{i=1,\ldots,r} (A_i \Rightarrow B_i), A \Rightarrow B) \end{aligned}$$

This fact means that it is possible to interpret the control operators as morphisms rather than natural transformations in the higher-order setting. Syntactically,

this means that we can replace parameterized controls by non-parameterized (higher-order) ones without changing the equational theory. We do not know how this affects the dynamics in general. □

Example 30. The semantic model of the calculus with non-determinism primitive plus, given in Example 24, is in fact a higher-order action model: let $X \Rightarrow Y$ be the set of total relations from X to Y, and for a total relation $R : A \otimes X \to B$ (the tensor product is just the direct product of sets) its adjunct $R^* : A \to X \Rightarrow B$ (in $\mathcal{C}$, hence a function) is determined by $xR^*(a)b$ if and only if $(a, x)Rb$. The above remark about the controls tells us that the interpretation of plus can be given as a total relation from $([\![m]\!] \Rightarrow [\![n]\!]) \otimes ([\![m]\!] \Rightarrow [\![n]\!]) \otimes [\![m]\!]$ to $[\![n]\!]$; explicitly it is given by the relation R where $(r, s, x)Ry$ if and only if $x(r \cup s)y$. □

5 A Semantic Proof of Conservativity

Since the higher-order theories (higher-order action calculi or our higher-order type theories) are obtained from the first-order ones by adding new constructs and additional axioms, there is an obvious sound translation from the first-order theories to the higher-order ones. A non-trivial fact is that this translation is (statically) conservative. In [Mil94a], Milner gives a syntactic proof by appealing to a normal form result on higher-order action calculi, using the so-called higher-order molecular forms. It is easy to extend his syntactic result to incorporate our axiom η_{name}.

Here we give another proof, using the properties of our semantic models. The conservativity result is achieved by constructing a higher-order action model into which the term model of the first-order theory faithfully embeds. In the case of standard algebraic type theory, the corresponding result is well-known. Given a cartesian category $\mathcal{C}$, the presheaf category $[\mathcal{C}^{\mathrm{op}}, \mathbf{Sets}]$ is a cartesian closed category and the Yoneda embedding is fully faithful and preserves products. Applying this fact to the term model for a first-order algebraic theory, it follows that the algebraic type theory can be faithfully embedded in the corresponding higher-order type theory.[2] Additional care is required for our setting, since we have a symmetric monoidal category and a functor, as well as a cartesian category, and our theory contains parameterized control operators which do not exist in standard algebraic theories.

We start with a known fact about the Yoneda construction (free cocompletion) on symmetric monoidal categories [Day70]. A systematic account can be found in [PR96].

Lemma 31. *Let $\mathcal{C}$, $\mathcal{D}$ be small (strict) symmetric monoidal categories and $\mathcal{F} : \mathcal{C} \longrightarrow \mathcal{D}$ an identity-on-objects strict symmetric monoidal functor. There exists a small (strict) symmetric monoidal closed category $\bar{\mathcal{C}}$, a small (strict) symmetric monoidal category $\bar{\mathcal{D}}$ and fully faithful strict symmetric monoidal functors $in_{\mathcal{C}}$:*

[2] For achieving the *fullness*, a little bit more categorical machinery (the gluing construction) is required, but here we do not need it for our purpose.

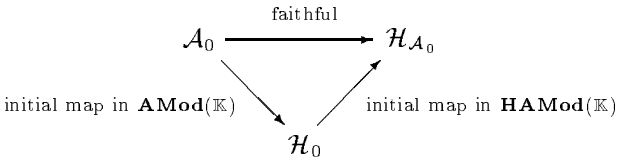
$\mathcal{C} \longrightarrow \bar{\mathcal{C}}$ and $in_{\mathcal{D}} : \mathcal{D} \longrightarrow \bar{\mathcal{D}}$, together with an identity-on-objects strict symmetric monoidal functor $\bar{\mathcal{F}} : \bar{\mathcal{C}} \longrightarrow \bar{\mathcal{D}}$ such that $\bar{\mathcal{F}} \circ in_{\mathcal{C}} = in_{\mathcal{D}} \circ \mathcal{F}$ and $\bar{\mathcal{F}}$ has a right adjoint. Moreover $in_{\mathcal{C}}$ is dense.

Proof: Let $\bar{\mathcal{C}}$ be the presheaf category $[\mathcal{C}^{\mathrm{op}}, \mathbf{Sets}]$ and $in_{\mathcal{C}}$ be the Yoneda embedding (which is dense [ML71]). It is well-known that $\bar{\mathcal{C}}$ is the free symmetric monoidal cocompletion of $\mathcal{C}$ and $in_{\mathcal{C}}$ is strict symmetric monoidal [Day70,IK86]. (If $\mathcal{C}$ is strict, let $\bar{\mathcal{C}}$ be the strict equivalent of $[\mathcal{C}^{\mathrm{op}}, \mathbf{Sets}]$.) Then $\mathcal{F}$ extends to a strict symmetric monoidal functor $\hat{\mathcal{F}} : \bar{\mathcal{C}} \longrightarrow [\mathcal{D}^{\mathrm{op}}, \mathbf{Sets}]$ with a right adjoint $U = [\mathcal{F}^{\mathrm{op}}, \mathbf{Sets}]$, so that $\hat{\mathcal{F}}$ strictly commutes with $\mathcal{F}$. Although $\hat{\mathcal{F}}$ may not be the identity on objects, we can factorize it as $\hat{\mathcal{F}} = J \circ \bar{\mathcal{F}}$ so that $\bar{\mathcal{F}} : \bar{\mathcal{C}} \longrightarrow \bar{\mathcal{D}}$ is the identity on objects and $J : \bar{\mathcal{D}} \longrightarrow [\mathcal{D}^{\mathrm{op}}, \mathbf{Sets}]$ is fully faithful. A right adjoint of $\bar{\mathcal{F}}$ is then given by $U \circ J : \bar{\mathcal{D}} \longrightarrow \bar{\mathcal{C}}$. The categories obtained above are not small, but we can cut down them to be small and still retain the required structure. □

Proposition 32. *Given an action model $\mathcal{A}$ over signature $\mathbb{K}$, there is a higher-order action model $\mathcal{H}_{\mathcal{A}}$ over the same signature $\mathbb{K}$ such that there is a faithful action morphism from $\mathcal{A}$ to $\mathcal{H}_{\mathcal{A}}$.*

Proof: We just apply the previous lemma. Let $(\mathcal{C}, \mathcal{S}, \mathcal{F})$ be the carrier of $\mathcal{A}$. Since $\bar{\mathcal{C}}$ is cartesian closed and $\bar{\mathcal{F}}$ has a right adjoint (say U), the functor $\bar{\mathcal{F}}(\_) \otimes X$ also has a right adjoint $X \Rightarrow (\_) \;=\; (U(\_))^X$. We show that this carrier $(\bar{\mathcal{C}}, \bar{\mathcal{S}}, \bar{\mathcal{F}})$ gives a higher-order action model. We choose the interpretation of arities $[\![\_]\!]'$ so that $[\![m]\!]' = in_{\mathcal{C}}([\![m]\!])$ for first-order arity m. Now we consider the interpretation of controls. For simplicity, we consider just the case of single parameter. Assume that there is a family of functions $K_X : \mathcal{S}(\mathcal{F}(X) \otimes A, B) \longrightarrow \mathcal{S}(\mathcal{F}(X) \otimes C, D)$ (natural in X in $\mathcal{C}$). Since $in_{\mathcal{S}}$ is fully faithful strict symmetric monoidal and $\bar{\mathcal{F}} \circ in_{\mathcal{C}} = in_{\mathcal{S}} \circ \mathcal{F}$, this induces a family of functions $K'_X : \bar{\mathcal{S}}(\bar{\mathcal{F}}(in_{\mathcal{C}}(X)) \otimes in_{\mathcal{S}}(A), in_{\mathcal{S}}(B)) \longrightarrow \bar{\mathcal{S}}(\bar{\mathcal{F}}(in_{\mathcal{C}}(X)) \otimes in_{\mathcal{S}}(C), in_{\mathcal{S}}(D))$ natural in X in $\mathcal{C}$. Since $in_{\mathcal{C}}$ is dense, we can extend K' to a family of functions $K''_X : \bar{\mathcal{S}}(\bar{\mathcal{F}}(X) \otimes in_{\mathcal{S}}(A), in_{\mathcal{S}}(B)) \longrightarrow \bar{\mathcal{S}}(\bar{\mathcal{F}}(X) \otimes in_{\mathcal{S}}(C), in_{\mathcal{S}}(D))$ natural in X in $\bar{\mathcal{C}}$. Thus given an interpretation of a control in $\mathcal{A}$, we have an interpretation of the control in $\mathcal{H}_{\mathcal{A}}$ of the carrier $(\bar{\mathcal{C}}, \bar{\mathcal{S}}, \bar{\mathcal{F}})$. And $(in_{\mathcal{C}}, in_{\mathcal{S}})$ is a faithful action morphism. □

Taking $\mathcal{A}$ as the term model $\mathcal{A}_0$, we obtain a higher-order action model $\mathcal{H}_{\mathcal{A}_0}$ into which $\mathcal{A}_0$ faithfully embeds. Because $\mathcal{H}_0$ is initial in $\mathbf{HAMod}(\mathbb{K})$, there is a unique higher-order action morphism from $\mathcal{H}_0$ to $\mathcal{H}_{\mathcal{A}_0}$. On the other hand, since $\mathcal{A}_0$ is initial in $\mathbf{AMod}(\mathbb{K})$ and also $\mathbf{HAMod}(\mathbb{K})$ is a subcategory of $\mathbf{AMod}(\mathbb{K})$, the following diagram commutes in $\mathbf{AMod}(\mathbb{K})$:



Therefore the unique action morphism from $\mathcal{A}_0$ to $\mathcal{H}_0$ must be faithful.

Theorem 33. *The higher-order theory* $\mathsf{HT}(\mathbb{K})$ *is a conservative extension of the first-order theory* $\mathsf{T}(\mathbb{K})$. □

6 Notions of Computation

We show that the higher-order action calculi conservatively extend Moggi's computational λ-calculus [Mog88], with an additional axiom for commuting let-bindings. Whilst it is possible to give a syntactic proof of this result, we give a simpler comparison by showing that the models of higher-order action calculi correspond to Moggi's semantic framework called *notions of computation* [Mog91].

Independently, Pavlovic has also mentioned the connection between his models of action calculi and notions of computation [Pav96].

6.1 The Computational Lambda Calculus

We reproduce the computational lambda calculus (λ_c-calculus) below, using the simply typed version as found in [MOTW95].

Definition 34. The *computational lambda calculus* is determined by the following data.

1. [Types] The set of *types* over the set P (called base types) is defined by the grammar
$$\sigma ::= p \in \mathsf{P} \mid \sigma \Rightarrow \sigma$$
2. [Terms and Values] The sets of *terms* and *values* over the set of variables V is defined by the grammar
$$\begin{array}{l} \text{terms } t ::= x \in V \mid \lambda x.t \mid tt \mid \mathsf{let}\ x\ \mathsf{be}\ t\ \mathsf{in}\ t \\ \text{values } v ::= x \mid \lambda x.t \end{array}$$
3. [Typing judgements] We say that a term is well-typed if it can be shown to annotate a sequent using the rules
$$\frac{}{\Gamma, x^\sigma \vdash_c x : \sigma} \qquad \frac{\Gamma, x^\sigma \vdash_c t : \tau}{\Gamma \vdash_c \lambda x.t : \sigma \Rightarrow \tau}$$
$$\frac{\Gamma \vdash_c s : \sigma \Rightarrow \tau \quad \Gamma \vdash_c t : \sigma}{\Gamma \vdash_c st : \tau} \qquad \frac{\Gamma \vdash_c s : \sigma \quad \Gamma, x^\sigma \vdash_c t : \tau}{\Gamma \vdash_c \mathsf{let}\ x\ \mathsf{be}\ s\ \mathsf{in}\ t : \tau}$$
$$\frac{\Gamma, x^\sigma, y^{\sigma'}, \Gamma' \vdash_c t : \tau}{\Gamma, y^{\sigma'}, x^\sigma, \Gamma' \vdash_c t : \tau}$$

4. [Equality judgements] We define an equality judgement $\Gamma \vdash_c s = t : \sigma$, where $\Gamma \vdash_c s : \sigma$ and $\Gamma \vdash_c t : \sigma$, as the congruence relation generated by the axioms:

(β_v)	$(\lambda x.t)v$	$=$	$t\{v/x\}$
(η_v)	$\lambda x.vx$	$=$	v, for x not free in v
(let_v)	$\mathsf{let}\ x\ \mathsf{be}\ v\ \mathsf{in}\ t$	$=$	$t\{v/x\}$
(id)	$\mathsf{let}\ x\ \mathsf{be}\ t\ \mathsf{in}\ x$	$=$	t
(comp)	$\mathsf{let}\ y\ \mathsf{be}\ (\mathsf{let}\ x\ \mathsf{be}\ s\ \mathsf{in}\ t)\ \mathsf{in}\ u$	$=$	$\mathsf{let}\ x\ \mathsf{be}\ s\ \mathsf{in}\ \mathsf{let}\ y\ \mathsf{be}\ t\ \mathsf{in}\ u$
(let.1)	et	$=$	$\mathsf{let}\ z\ \mathsf{be}\ e\ \mathsf{in}\ zt$
(let.2)	ve	$=$	$\mathsf{let}\ x\ \mathsf{be}\ e\ \mathsf{in}\ vx$

where e ranges over non-values: that is, applications and let-blocks.

□

It is not hard to see that the obvious translation from the computational lambda calculus (with the base types P) to our higher-order type theory $\mathsf{HT}(\mathbb{K})$ (with $\mathbb{K} = (\mathsf{P}, \mathcal{K})$) is sound. However, this translation is not conservative. To achieve conservativity, we need to strengthen the equational theory of the computational lambda calculus, by assuming an additional axiom for *commuting* let-bindings.

Definition 35. The *commutative* computational lambda calculus is obtained by adding the following axiom:

$$\text{(comm)} \quad \mathsf{let}\ x\ \mathsf{be}\ s\ \mathsf{in}\ \mathsf{let}\ y\ \mathsf{be}\ t\ \mathsf{in}\ u\ =\ \mathsf{let}\ y\ \mathsf{be}\ t\ \mathsf{in}\ \mathsf{let}\ x\ \mathsf{be}\ s\ \mathsf{in}\ u$$

(As in other axioms, both sides of the axiom must have the same type under the same context, therefore x cannot be free in t and y cannot be free in s.) □

Theorem 36. $\mathsf{HT}(\mathbb{K})$ *is a conservative extension of the commutative computational lambda calculus.* □

This theorem can be proved syntactically by comparing the normal forms in these calculi. Although there seems to be no simple confluent terminating rewriting system for the commutative computational λ-calculus, it is possible to define a notion of normal form which closely resembles Milner's higher-order molecular forms [Mil94a]. Rather than presenting this syntactic proof, we give a semantic proof of conservativity by relating the models.

6.2 Comparison with λ_c-Models

We show the connection between faithful higher-order action models and Moggi's models of the computational lambda calculus, called λ_c-*models* [Mog88]. This observation relies on fairly standard category theory, and is a special instance of the results given in [PR96]. To recall from Moggi's work, a monad (T, η, μ) satisfies the *mono requirement* if each component of η is a monomorphism. A

tensorial strength for a monad T on a symmetric monoidal category is a natural transformation with components $\theta_{A,B} : A \otimes TB \to T(A \otimes B)$ subject to the coherence conditions as found in [Koc70,Mog88,Mog91]. It is *commutative* if the evident two natural transformations from $TA \otimes TB$ to $T(A \otimes B)$ agree.

Definition 37. A λ_c-*model* is a cartesian category $\mathcal{C}$ with a strong monad (T, η, μ) which satisfies the mono requirement and has *Kleisli exponents*: that is, for each object A, the functor $\mathcal{J}(A \times \_) : \mathcal{C} \to \mathcal{C}_T$ has a (chosen) right adjoint $A \Rightarrow (\_)$, where $\mathcal{C}_T$ is the Kleisli category of T and $\mathcal{J} : \mathcal{C} \to \mathcal{C}_T$ is given by $\mathcal{J}(f) = f; \eta$. A λ_c-model is said to be *commutative* if the tensorial strength is commutative. □

Proposition 38. *To give a carrier $(\mathcal{C}, \mathcal{S}, \mathcal{F})$ such that $\mathcal{F}$ is faithful and $\mathcal{F}(\_) \otimes X$ has a right adjoint for each X is to give a commutative λ_c-model in which cartesian products are strictly associative.* □

Sketch of Proof: Given such a carrier $(\mathcal{C}, \mathcal{S}, \mathcal{F})$, we have a commutative strong monad on $\mathcal{C}$ as the composition of $\mathcal{F}$ with its right adjoint, which satisfies the mono requirement and has Kleisli exponents. Conversely, given a commutative λ_c-model T over a cartesian category $\mathcal{C}$, we have a carrier $(\mathcal{C}, \mathcal{C}_T, \mathcal{J})$ which satisfies the properties above. Moreover these constructions are inverse to each other. (See [PR96] for a detailed account.) □

Together with the observation in Remark 29, we can specify a higher-order action model as a commutative λ_c-model with the interpretation of primes and controls.

Theorem 39. *To give a faithful higher-order action model over the signature $\mathbb{K} = (\mathsf{P}, \mathcal{K})$ is to give a commutative λ_c-model in which products are strictly associative, an object $[\![p]\!]_{\mathsf{P}}$ for each $p \in \mathsf{P}$ and an arrow*

$$[\![\mathsf{K}]\!]_{\mathcal{K}} : \prod_{i=1,\ldots,r} ([\![m_i]\!] \Rightarrow [\![n_i]\!]) \to [\![m]\!] \Rightarrow [\![n]\!]$$

for each control K *with arity rule* $((m_1, n_1), \ldots, (m_r, n_r)) \to (m, n)$ *where* $[\![m]\!]$ *is defined inductively as in the higher-order action models.* □

Our result shows that the models of a *static* higher-order action calculus correspond to commutative λ_c-models, with interpretation functions for the primes and controls. To interpret the dynamics, we require an additional preorder enrichment of the Kleisli category, which is not present in Moggi's work.

Example 40. The model of plus (Example 4) given in Example 24 can be understood as a commutative λ_c-model: the corresponding monad is the non-empty powerset monad on the category of sets and functions. The interpretation $[\![\mathsf{plus}]\!]$ is given routinely, again by the union of relations. The local preorder on the Kleisli category (the category of sets and total relations) is derived from the subset inclusion on powersets. □

This example, used repeatedly in this paper, gives an elementary account of non-determinism. We can construct more sophisticated models of non-determinism along this line using various powerdomain monads. (In particular, the connection with models of the π-calculus studied in [FMS96,Sta96] requires further investigation.)

7 Conclusions and Related Work

In this paper we have given type-theoretic and categorical presentations of Milner's higher-order action calculi. We have also given a semantic proof of the conservativity of (static) higher-order action calculi over action calculi, and have related our higher-order models with Moggi's λ_c-models. We hope that the type-theoretic and categorical presentations of action calculi studied here will provide useful criteria for assessing which dynamic relations describe interesting interactive behaviour.

Our results form part of a broader picture. First, there is work by Barber, Gardner, Hasegawa and Plotkin [BGHP96,BGHP97], which links action calculi with intuitionistic linear type theory [Bar96,Ben95], and proves conservativity results for various extensions of action calculi incorporating the results given here. This work focusses more extensively on relating type theories by comparing their categorical models. (See also [BW96] for a related work on linear type theory and notions of computation.)

Second, it is natural to extend our theories by replacing the let-bindings in the type theories by recursive letrec-bindings: that is, introducing *cyclic sharing* as studied in graph rewriting theory, see for instance [AA95]. In action calculi, the corresponding construct is given by adding a *reflexion* operator to action calculi [Mil94b,Mif96]. The semantic models of cyclic sharing can be described in terms of *traced symmetric monoidal categories* [JSV96]. In [Has97a,Has97b], Hasegawa studies such structures and shows the connection with traditional models of recursion.

Third, whilst in this paper symmetric monoidal categories are sufficient for describing models of action calculi, Power and Robinson focus on *premonoidal categories* [PR96] for expressing computational behaviour, in which the tensor product need not be bifunctorial. They argue that this weaker structure is more natural for describing computational behaviour such as imperative features. It might be fruitful to seek the premonoidal version of action calculi as well as type theories, although it remains to be seen whether their weaker structure can be justified from the action calculi perspective.

Acknowledgements

We wish to thank Andrew Barber, Gordon Plotkin and John Power for many helpful discussions.

References

[Acz80] P. Aczel, Frege structures and the notions of proposition, truth and set. In *The Kleene Symposium*, North-Holland, 1980.

[AA95] Z. M. Ariola and Arvind, Properties of a first-order functional language with sharing. *Theoretical Computer Science* 146, pages 69-108, 1995.

[BGHP96] A. Barber, P. Gardner, M. Hasegawa and G. Plotkin, Action calculi, the computational λ-calculus and linear logic. Draft, 1996.

[BGHP97] A. Barber, P. Gardner, M. Hasegawa and G. Plotkin, From action calculi to linear logic. Submitted, 1997.

[Bar96] A. Barber, Dual intuitionistic linear logic. Technical report ECS-LFCS-96-347, LFCS, University of Edinburgh, 1996.

[BW96] N. Benton and P. Wadler, Linear logic, monads, and the lambda calculus. In *Proc. 11th Symposium on Logic in Computer Science (LICS'96)*, 1996.

[Ben95] N. Benton, A mixed linear non-linear logic: proofs, terms and models. In *Proc. Computer Science Logic 1994 (CSL'94)*, Springer LNCS 933, pages 121-135, 1995.

[CJLS97] A. Compagnoni, O. Jensen, J. Leifer and P. Sewell, Action calculus semantics for a functional programming language with mutable store. Draft, 1997.

[Day70] B. J. Day, On closed categories of functors. *Midwest Category Seminar Reports IV*, Springer LNM 137, pages 1-38, 1970.

[FMS96] M. P. Fiore, E. Moggi and D. Sangiorgi, A fully-abstract model for the π-calculus. In *Proc. 11th Symposium on Logic in Computer Science (LICS'96)*, 1996.

[Gar95] P. Gardner, A name-free account of action calculi. In *Proc. 11th Int. Conf. Mathematical Foundations of Programming Semantics (MFPS'95)*, Electronic Notes in Computer Science 1, Elsevier, 1995.

[Gar96] P. Gardner, Closed action calculi. 1996. To appear in *Theoretical Computer Science*.

[Has97a] M. Hasegawa, Recursion from cyclic sharing: traced monoidal categories and models of cyclic lambda calculi. In *Proc. Third Int. Conf. Typed Lambda Calculi and Applications (TLCA'97)*, Springer LNCS 1210, pages 196-213, 1997.

[Has97b] M. Hasegawa, *Models of Sharing Graphs (A Categorical Semantics of Let and Letrec)*. PhD thesis, LFCS, University of Edinburgh, 1997.

[Hay85] S. Hayashi, Adjunction of semi-functors: categorical structures in non-extensional lambda-calculus. *Theoretical Computer Science* 41, pages 95-104, 1985.

[IK86] G. B. Im and G. M. Kelly, A universal property of the convolution monoidal structure. *Journal of Pure and Applied Algebra* 43, pages 75-88, 1986.

[Jen95] O. Jensen, Talk at a Newton Institute Workshop on Games, Processes and Logic. Cambridge, November 1995.

[JSV96] A. Joyal, R. Street and D. Verity, Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society* 119(3), pages 447-468, 1996.

[Koc70] A. Kock, Strong functors and monoidal monads. Various Publications Series 11, Aarhus Universitet, 1970.

[ML71] S. Mac Lane, *Categories for the Working Mathematician.* Graduate Texts in Mathematics 5, Springer-Verlag, 1971.

[MOTW95] J. Maraist, M. Odersky, D. Turner and P. Wadler, Call-by-name, call-by-value, call-by-need, and the linear lambda calculus. In *Proc. 11th Int. Conf. Mathematical Foundations of Programming Semantics (MFPS'95)*, Electronic Notes in Computer Science 1, Elsevier, 1995.

[MMP95] A. Mifsud, R. Milner and A. J. Power, Control structures. In *Proc. 10th Symposium on Logic in Computer Science (LICS'95)*, 1995.

[Mif96] A. Mifsud, *Control Structures*. PhD thesis, LFCS, University of Edinburgh, 1996.

[Mil94a] R. Milner, Higher-order action calculi. In *Proc. Computer Science Logic 1992 (CSL'92)*, Springer LNCS 832, pages 238-260, 1994.

[Mil94b] R. Milner, Action calculi V: reflexive molecular forms (with Appendix by O. Jensen). Third draft, July 1994.

[Mil96] R. Milner, Calculi for interaction. *Acta Informatica* 33(8), pages 707-737, 1996.

[Mog88] E. Moggi, Computational lambda-calculus and monads. Technical report ECS-LFCS-88-66, LFCS, University of Edinburgh, 1988.

[Mog91] E. Moggi, Notions of computation and monads. *Information and Computation* 93, pages 55-92, 1991.

[Pav96] D. Pavlovic, Categorical logic of names and abstraction in action calculi. 1996. To appear in *Mathematical Structures in Computer Science*.

[PR96] A. J. Power and E. P. Robinson, Premonoidal categories and notions of computation. 1996. To appear in *Mathematical Structures in Computer Science*.

[Pow96] A. J. Power, Elementary control structures. In *Proc. 6th Int. Conf. Concurrency Theory (CONCUR'96)*, Springer LNCS 1119, pages 115-130, 1996.

[Sta96] I. Stark, A fully abstract domain model for the π-calculus. In *Proc. 11th Symposium on Logic in Computer Science (LICS'96)*, pages 36-42, 1996.