From Action Calculi to Linear Logic

Andrew Barber[1], Philippa Gardner[2], Masahito Hasegawa[3] and Gordon Plotkin[1]

[1] Dept. Computer Science, University of Edinburgh, Edinburgh EH9 3JZ, Scotland
[2] Computing Laboratory, University of Cambridge, Cambridge CB2 3QG, England
[3] RIMS, Kyoto University, Kyoto 606-8502, Japan

Abstract. Milner introduced action calculi as a framework for investigating models of interactive behaviour. We present a type-theoretic account of action calculi using the propositions-as-types paradigm; the type theory has a sound and complete interpretation in Power's categorical models. We go on to give a sound translation of our type theory in the (type theory of) intuitionistic linear logic, corresponding to the relation between Benton's models of linear logic and models of action calculi. The conservativity of the syntactic translation is proved by a model-embedding construction using the Yoneda lemma. Finally, we briefly discuss how these techniques can also be used to give conservative translations between various extensions of action calculi.

1 Introduction

Action calculi arose directly from the π-calculus [MPW92]. They were introduced by Milner [Mil96], to provide a uniform notation for capturing many calculi of interaction such as the π-calculus, the λ-calculus, models of distributed migratory systems [CG97,Sew97], the spi-calculus used for describing security protocols [AG97] and the object calculus [AC96]. In this paper, we present a type-theoretic account of action calculi using the well-known propositions-as-types paradigm. In particular, the constants of action calculi are analogues of Aczel's general binding operators [Acz80]. We give a sound and conservative translation of the calculus into a type theory for intuitionistic linear logic. Similar results hold for some extensions of action calculi.

Semantic methods play an essential rôle in our work. Our type theory has a natural interpretation in Power's categorical models [Pow96]. Further, Power's models are (essentially) reducts of Benton's models of linear logic [Ben95]. We are thereby presented with a situation similar to that well-known in other areas, where sound translations between theories are in correspondence with functors between the categories of their models. A leading example is provided by essentially algebraic theories and their models [AR94]. The conservativity of such sound translations can be shown using a model-embedding construction; in our case, using the Yoneda lemma and work of Day [Day73], we show that the sound syntactic embedding of our type theory for action calculi into the type theory of linear logic is conservative. We emphasise that, while the general model-theoretic

ideas underlying this development are well-known, there seems to be no general theory available which, as a particular case, applies to action calculi.

We show that our techniques also apply to three extensions of action calculi: action calculi with code (which, together with a reflexion operator, provide just enough extra structure to support recursion), Milner's higher-order action calculi [Mil94a], which turn out to be equivalent to an extension of Moggi's commutative computational λ-calculus [Mog88] with commutativity (incorporating results of [GH97]), and linear action calculi. We consider the (necessarily sound) translations between these calculi corresponding to the natural embeddings of models; all these translations are conservative.

It remains on-going research to fully understand the dynamics of action calculi and obtain, for example, a general theory of bisimulation. We do not study dynamics in this paper except to observe that reduction dynamics as presented in [Mil96] corresponds to a standard rewriting in the type-theoretic presentation.

We hope that the relation described between action calculi and linear logic will prove fruitful, especially for the further understanding of concurrency. We also hope that the treatment of translations of type theories via reducts of models will help expose a useful paradigm.

Summary: We introduce action calculi in section 2, and give the corresponding type-theoretic account in section 3. Section 4 presents the categorical semantics, showing soundness and completeness. In section 5, we describe Benton's type theory for linear logic and its categorical semantics. Section 6 gives the translation from action calculus to linear logic, which is shown to be conservative in section 7. In section 8, we consider the three extensions of action calculi, and we conclude in section 9.

2 Action Calculi

This section gives a brief presentation of action calculi. Our presentation differs from Milner's original one [Mil96] in its choice of primitives, but is easily shown to be equivalent. We believe this alternative presentation is slightly more natural.

The static part of an action calculus is defined by a set of terms and an equational theory on the terms. It is generated from a *signature* $\mathbb{K} = (\mathsf{P}, \mathcal{K})$, which consists of a set P of *primes* denoted by $p, q, \ldots$ and a set $\mathcal{K}$ of constants called *control operators*. Each control operator is equipped with an *arity* $((m_1, n_1), \ldots, (m_r, n_r)) \to (m, n)$, where the m's and n's are finite sequences of primes, called *tensor arities*; we write ε for the empty sequence, $\otimes$ for concatenation using infix notation, and M for the set of tensor arities. We usually refer to the tensor arities as just arities, where the meaning is clear. We assume a fixed countably infinite set X of *names*, each of which has a prime arity. We let $x, y, \ldots$ range over names, and write x^p to indicate that x has prime arity p.

Definition 1 (Terms)
The terms $a, b, c \ldots$ of the action calculus $\mathsf{AC}(\mathbb{K})$ are constructed from the basic operators: *identity* $\mathbf{id}_m$, *permutation* $\mathbf{p}_{m,n}$, *composition* $\cdot$, *tensor* $\otimes$, *abstraction*

$(x^p)_-$, *datum* $\langle x^p \rangle$ and also from *control operators* K. Every term a has an arity $a : m \to n$, for tensor arities m and n, and the terms and their arities are given using the following rules:

$$\mathbf{id}_m : m \to m \qquad \mathbf{p}_{m,n} : m \otimes n \to n \otimes m$$

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

$$\frac{a : m \to n}{(x^p)a : p \otimes m \to n} \qquad \langle x^p \rangle : \varepsilon \to p$$

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

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

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 $fn(a, b, \ldots)$. Given a possibly empty sequence of names $\vec{x} = x_1^{p_1}, \ldots, x_r^{p_r}$, we write $|\vec{x}|$ 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.

Definition 2 (The Equational Theory AC)
The *equational theory* AC on terms is that generated by the axioms of a strict symmetric monoidal category with symmetry $\mathbf{p}$:

$$\begin{aligned} a \cdot (b \cdot c) &= (a \cdot b) \cdot c & a \cdot \mathbf{id}_n &= a = \mathbf{id}_m \cdot a \\ (a \cdot b) \otimes (c \cdot d) &= (a \otimes c) \cdot (b \otimes d) & \mathbf{id}_m \otimes \mathbf{id}_n &= \mathbf{id}_{m \otimes n} \\ a \otimes (b \otimes c) &= (a \otimes b) \otimes c & a \otimes \mathbf{id}_\varepsilon &= a = \mathbf{id}_\varepsilon \otimes a \end{aligned}$$

$$\mathbf{p}_{m,n} \cdot (b \otimes a) = (a \otimes b) \cdot \mathbf{p}_{m',n'} \qquad \mathbf{p}_{m \otimes n, m'} = (\mathbf{id}_m \otimes \mathbf{p}_{n,m'}) \cdot (\mathbf{p}_{m,m'} \otimes \mathbf{id}_n)$$
$$\mathbf{p}_{m,n} \cdot \mathbf{p}_{n,m} = \mathbf{id}_{m \otimes n}$$

augmented by two naming axioms:

$$(\langle y \rangle \otimes \mathbf{id}_m) \cdot (x)a = a\{\langle y \rangle / \langle x \rangle\} \qquad (\sigma)$$
$$(x)((\langle x \rangle \otimes \mathbf{id}) \cdot a) = a, \text{ where } x \notin fn(a) \qquad (\delta)$$

We write $a = b : m \to n$ in $\mathsf{AC}(\mathbb{K})$ if a and b are terms of $\mathsf{AC}(\mathbb{K})$ with arity $m \to n$, and $a = b$ in AC. We use the abbreviations $(\vec{x})a \stackrel{\text{def}}{=} (x_1) \cdots (x_r)a$ for distinct x_i and $\langle \vec{x} \rangle \stackrel{\text{def}}{=} \langle x_1 \rangle \otimes \cdots \otimes \langle x_r \rangle$, where $(\ )a$ is a and $\langle\ \rangle$ is $\mathbf{id}_\varepsilon$. It is an

immediate consequence of these axioms that $\mathbf{id}_m = (\vec{x})\langle\vec{x}\rangle$ and $\mathbf{p}_{m,n} = (\vec{x}\vec{y})\langle\vec{y}\vec{x}\rangle$, where $|\vec{x}| = m$ and $|\vec{y}| = n$. The *static* part of an action calculus $\mathsf{AC}(\mathbb{K})$ consists of the equivalence classes obtained by quotienting the terms in definition 1 by the equational theory. In [Mil96], the equivalence classes are called *actions*. We overload notation and use actions to also denote the terms, to distinguish them from the type-theoretic terms given in section 3.

REMARK 3
The *dynamic* part of an action calculus $\mathsf{AC}(\mathbb{K})$ is a transitive relation $\searrow$ between the equivalence classes of terms with the same arity which is preserved under tensor, composition, abstraction, and such that there is no a with $\mathbf{id} \searrow a$. Although the dynamics are a key part of action calculi, the results in this paper concentrate on the static part.

3 Type-Theoretic Interpretation

We present a type-theoretic interpretation of action calculi, which gives a general way of describing natural-deduction proofs in a certain logic, using the propositions-as-types paradigm. The underlying logical structure is built using propositions given by the prime arities. The assumptions are split into intuitionistic and linear assumptions; the idea of this division of assumptions is familiar from linear logic [Gir93,BP98,Bar97,Ben95]. For a given action $a : m \to n$ with $fn(a) \subseteq \{\vec{x}\}$, the intuitionistic assumptions account for the free names $\vec{x}$ and the linear assumptions for the domain arity m. The conclusions are lists of propositions (regarded as a strict tensor of prime arities), and correspond to the codomain arity n. The connectives of the logic are determined by the arities associated with the control operators. In particular, the control operators of action calculi correspond to analogues of the *binding operators* of Aczel [Acz80,Plo90], described in the appendix.

Rather than first describing the logic, we proceed directly to the type-theoretic description. The type theory has sequents of the form $\Gamma; \Delta \vdash t : m$, where $\Gamma = x_1^{p_1}, \ldots, x_r^{p_r}$ is a sequence of distinct names from the name-set X with their associated prime arities, $\Delta = w_1{:}q_1, \ldots w_s{:}q_s$ is a sequence of distinct linear variables typed with prime arities, and m is a list of prime arities. We view Γ as an *intuitionistic* context and Δ as a *linear* context, and call the names x^p in X intuitionistic variables. We use $w, v \ldots$ for linear variables, and abbreviate the linear context $w_1{:}q_1, \ldots, w_s{:}q_s$ to $\vec{w}{:}n$, where $n = q_1, \ldots, q_s$. As with action calculi, the type theory is specified by a signature $\mathbb{K} = (\mathsf{P}, \mathcal{K})$. The types are given by the set of primes P, and the terms are generated from the set of control operators $\mathcal{K}$, the set of names X and a countably infinite set of linear variables W.

DEFINITION 4 (TERMS)
The *terms* of the type theory $T(\mathbb{K})$, denoted $t, s \ldots$, are defined by:

$$t ::= w \mid \langle x^p \rangle \mid \mathsf{let}\ \otimes\vec{w}{:}m\ \mathsf{be}\ t\ \mathsf{in}\ t \mid \mathsf{let}\ \langle x^p \rangle\ \mathsf{be}\ t\ \mathsf{in}\ t \mid \otimes(t, \ldots, t) \mid \mathsf{K}((\vec{w}_1{:}m_1)t, \ldots, (\vec{w}_r{:}m_r)t; t)$$

The terms of the form $\mathsf{K}((\vec{w}_1 : m_1)t_1, \ldots, (\vec{w}_r : m_r)t_r; s)$ correspond to the terms arising from Aczel's binding operators. The first r arguments correspond to the arguments described in the appendix, where the notation $(\vec{w}_i : m_i)t_i$ denotes that the linear variables $\vec{w}_i$ are bound in the t_i. From the typing rules, we shall see that the $\vec{w}_i$ contain all the linear variables in t_i. The last argument s is necessary to record the possibility that the operator requires some linear input to be well-formed. We sometimes omit the arity information in the K constructs, the let constructs and the intuitionistic variables when apparent. The two let constructs correspond to linear and intuitionistic cut. The term $\mathsf{let}\ \otimes\vec{w}\ \mathsf{be}\ t\ \mathsf{in}\ s$ binds the sequence of distinct linear variables $\vec{w}$ in s and the term $\mathsf{let}\ \langle x\rangle\ \mathsf{be}\ t\ \mathsf{in}\ s$ binds x in s; we sometimes write these terms as $[\vec{w} := t]t'$ and $[x^p := t]t'$ respectively. The term $\otimes(t_1, \ldots, t_r)$ denotes a tensor of length r; we abbreviate $\otimes[_]$ to $*$ and $\otimes(t_1, t_2)$ to $t_1 \otimes t_2$ as usual. We write $\langle\vec{x}\rangle$ for $\otimes(\langle x_1\rangle, \ldots, \langle x_n\rangle)$ and use $(\mathsf{let}\ \langle\vec{x}\rangle\ \mathsf{be}\ \vec{t}\ \mathsf{in}\ s)$ to denote $[x_1 := t_1]\ldots[x_n := t_r]s$. We identify terms up to α-conversion (that is on linear bound variables and intuitionistic bound variables of the same arity). We use a standard notion of substitution, and write $t\{s/\langle x\rangle\}$ for intuitionistic substitution and $t\{s/w\}$ for linear substitution.

DEFINITION 5
We say that a term can be typed in the type theory $T(\mathbb{K})$ if it can be shown to annotate a sequent using the following rules:

$$\Gamma; v{:}p \vdash v{:}p \qquad \frac{\Gamma; \Delta_1, v{:}p, w{:}q, \Delta_2 \vdash t{:}m}{\Gamma; \Delta_1, w{:}q, v{:}p, \Delta_2 \vdash t{:}m}$$

$$\Gamma_1, x^p, \Gamma_2; _ \vdash \langle x^p\rangle : p \qquad \frac{\Gamma; \Delta_1 \vdash t{:}p \quad \Gamma, x^p; \Delta_2 \vdash s{:}m}{\Gamma; \Delta_1, \Delta_2 \vdash \mathsf{let}\ \langle x^p\rangle\ \mathsf{be}\ t\ \mathsf{in}\ s{:}m}$$

$$\frac{\Gamma; \Delta_i \vdash t_i{:}m_i \quad i = 1, \ldots, r}{\Gamma; \Delta_1, \ldots, \Delta_r \vdash \otimes(t_1, \ldots, t_r) : m_1, \ldots, m_r} \qquad \frac{\Gamma; \Delta_1 \vdash t{:}m \quad \Gamma; \vec{w}{:}m, \Delta_2 \vdash s{:}n}{\Gamma; \Delta_1, \Delta_2 \vdash \mathsf{let}\ \otimes\vec{w}{:}m\ \mathsf{be}\ t\ \mathsf{in}\ s{:}n}$$

$$\frac{\Gamma; \vec{w}_i{:}m_i \vdash t_i{:}n_i \qquad \Gamma; \Delta \vdash s{:}m \qquad i = 1, \ldots, r}{\Gamma; \Delta \vdash \mathsf{K}((\vec{w}_1{:}m_1)t_1, \ldots, (\vec{w}_r{:}m_r)t_r; s) : n}\ (\mathsf{K} \in \mathcal{K})$$

where, in the last rule, K has arity $((m_1, n_1), \ldots, (m_r, n_r)) \to (m, n)$.

The exchange rule (on the first line) is needed to handle the commutativity of the tensor in the type theory – in the action calculus an explicit permutation operator appears instead. Because of this rule, terms do not have unique derivations (in given contexts), though they do have unique typings. As a result various coherence issues arise – see below. An alternative approach, see, for example [BP98], is to "build-in" the exchange rule by allowing permutations of contexts in the other rules.

We give the equalities of the type theory in definition 6. Most of these are familiar from linear logic. However, it is worth noting that the first two axioms enforce the strictness of the tensor. Also note that we can derive terms

$_; \vec{w}:m, \vec{v}:n \vdash s:n,m$ corresponding to the permutation operator using the admissible exchange on the intuitionistic context Γ.

Definition 6 (Equality)
We define an equality judgement $\Gamma;\Delta \vdash t = s : m$, where $\Gamma;\Delta \vdash t:m$ and $\Gamma;\Delta \vdash s:m$, with appropriate reflexivity, transitivity and congruence rules, based on the following axioms:

$$\otimes(\otimes\vec{t}_1,\ldots,\otimes\vec{t}_r) = \otimes(\vec{t}_1,\vec{t}_2,\ldots,\vec{t}_r) \qquad \otimes(t) = t$$
$$\mathsf{let}\ \langle x\rangle\ \mathsf{be}\ \langle y\rangle\ \mathsf{in}\ t = t\{\langle y\rangle/\langle x\rangle\} \qquad [\vec{v}\vec{w} := t\otimes s]t' = [\vec{v} := t][\vec{w} := s]t'$$
$$\mathsf{let}\ \langle x\rangle\ \mathsf{be}\ t\ \mathsf{in}\ \langle x\rangle = t \qquad \mathsf{let}\ \otimes\vec{w}\ \mathsf{be}\ t\ \mathsf{in}\ \otimes\vec{w} = t$$

augmented with the commuting conversions:

$$[x := s][\vec{w} := t']t = [\vec{w} := ([x := s]t')]t \qquad [\vec{w} := t][x := s]t' = [x := s][\vec{w} := t]t'$$
$$[\vec{w} := t][x := t']s = [x := ([\vec{w} := t]t')]s \qquad [\vec{w} := t](s\otimes t') = s\otimes([\vec{w} := t]t')$$
$$[\vec{w} := t](t'\otimes s) = ([\vec{w} := t]t')\otimes s \qquad [\vec{w} := t][\vec{v} := s]t' = [\vec{v} := s][\vec{w} := t]t'$$
$$[\vec{w} := t][\vec{v} := t']s = [\vec{v} := ([\vec{w} := t]t')]s$$

$$[\vec{w} := t]\mathsf{K}((\vec{v}_1)t_1,\ldots,(\vec{v}_r)t_r; s) = \mathsf{K}((\vec{v}_1)t_1,\ldots,(\vec{v}_r)t_r; [\vec{w} := t]s)$$

where in the first two commuting conversions x may not be a free intuitionistic variable of t. Since these terms are well-typed by assumption, these conditions on free variables are the only ones necessary.

3.1 Connection with Action Calculi

We give the formal justification of our assertion that the type theory $\mathsf{T}(\mathbb{K})$ corresponds to the action calculus $\mathsf{AC}(\mathbb{K})$, by defining translations which are sound and inverse to each other up to provable equality (modulo a certain, essentially arbitrary, choice of linear context). First, we give the translation from $\mathsf{AC}(\mathbb{K})$ to $\mathsf{T}(\mathbb{K})$; given an action $a : m \to n$, we define a term $\Phi_\Delta(a)$, which depends on an arbitrary linear context Δ where $|\Delta|$ is the domain arity m.

Definition 7 (The Translation Φ)
For every action $a : m \to n$ with free names in the list Γ and for every linear context Δ with $|\Delta| = m$, we define a term $\Phi_\Delta(a)$ such that $\Gamma;\Delta \vdash \Phi_\Delta(a):n$. The definition is by induction on the structure of a:

$\Phi_\Delta(\mathbf{id}_m) = \otimes(w_1,\ldots,w_r)$, where $m = p_1\ldots p_r$ and $\Delta = w_1 : p_1,\ldots,w_r : p_r$
$\Phi_{w_1:p_1,w_2:p_2}(\mathbf{p}_{p_1,p_2}) = w_2\otimes w_1$
$\Phi_\Delta(a_1\cdot a_2) = \mathsf{let}\ \otimes\vec{v}\ \mathsf{be}\ \Phi_\Delta(a_1)\ \mathsf{in}\ \Phi_{\Delta_1}(a_2)$, where $\Delta_1 = \vec{v}{:}l$ and $a_1 : m \to l$
$\Phi_{\Delta_1,\Delta_2}(a_1\otimes a_2) = \Phi_{\Delta_1}(a_1)\otimes\Phi_{\Delta_2}(a_2)$, where $a_1 : |\Delta_1| \to k$ and $a_2 : |\Delta_2| \to l$
$\Phi_{w:p,\Delta_1}((x^p)a_1) = \mathsf{let}\ \langle x\rangle\ \mathsf{be}\ w\ \mathsf{in}\ \Phi_{\Delta_1}(a_1)$
$\Phi_\varepsilon(\langle x^p\rangle) = \langle x^p\rangle$
$\Phi_\Delta(\mathsf{K}(a_1,\ldots,a_r)) = \mathsf{K}((\vec{v}_1)\Phi_{\Delta_1}(a_1),\ldots,(\vec{v}_r)\Phi_{\Delta_r}(a_r);\otimes\vec{w})$,
where $\Delta = \vec{w} : m$, $\Delta_i = \vec{v}_i{:}m_i$ with the m and m_i's given by the arity of K

In the third and last cases we assume a fixed choice of new variables (in fact all choices result in α-equivalent terms).

The translation from $\mathsf{T}(\mathbb{K})$ to $\mathsf{AC}(\mathbb{K})$ is simpler: given a sequent $\Gamma;\Delta \vdash t{:}n$, we define an action $\Psi(\Gamma;\Delta \vdash t{:}n) : |\Delta| \to n$, with free names in the list Γ as follows (we confuse derivations with sequents).

Definition 8 (The Translation Ψ)
The translation Ψ is defined inductively on derivation of sequents:

$$\Psi(\Gamma; w{:}p \vdash w{:}p) = \mathbf{id}_p$$
$$\Psi(\Gamma; \Delta_1, w{:}q, v{:}p, \Delta_2 \vdash t{:}m) = (\mathbf{id} \otimes \mathbf{p}_{q,p} \otimes \mathbf{id}) \cdot \Psi(\Gamma; \Delta_1, v{:}p, w{:}q, \Delta_2 \vdash t{:}m)$$
$$\Psi(\Gamma_1, x^p, \Gamma_2; _ \vdash \langle x^p \rangle{:}p) = \langle x^p \rangle$$
$$\Psi(\Gamma; \Delta_1, \Delta_2 \vdash \mathsf{let}\ \langle x^p \rangle\ \mathsf{be}\ t\ \mathsf{in}\ s{:}m) =$$
$$(\Psi(\Gamma; \Delta_1 \vdash t{:}p) \otimes \mathbf{id}_{|\Delta_2|}) \cdot (x)\Psi(\Gamma, x^p; \Delta_2 \vdash s{:}m)$$
$$\Psi(\Gamma; \Delta_1, \dots, \Delta_r \vdash \otimes(t_1, \dots, t_r) : m_1, \dots, m_r) = \bigotimes_{i=1,\dots,r} (\Psi(\Gamma; \Delta_i \vdash t_i{:}m_i))$$
$$\Psi(\Gamma; \Delta_1, \Delta_2 \vdash \mathsf{let}\ \otimes\vec{w}{:}m\ \mathsf{be}\ t\ \mathsf{in}\ s{:}n) =$$
$$(\Psi(\Gamma; \Delta_1 \vdash t{:}m) \otimes \mathbf{id}_{|\Delta_2|}) \cdot \Psi(\Gamma; \vec{w}{:}m, \Delta_2 \vdash s{:}n)$$
$$\Psi(\Gamma; \Delta \vdash \mathsf{K}((\vec{w}_1{:}m_1)t_1, \dots, (\vec{w}_r{:}m_r)t_r; s) : n) =$$
$$\Psi(\Gamma; \Delta \vdash s{:}m) \cdot \mathsf{K}(\Psi(\Gamma; \vec{w}_1{:}m_1 \vdash t_1{:}n_1), \dots, \Psi(\Gamma; \vec{w}_r{:}m_r \vdash t_r{:}n_r))$$

A suitable coherence result can be proved, that, up to provable equality, the translation is independent of the derivation chosen.

The following proposition states that the translations are sound and inverse to each other up to provable equality (and modulo the choice of contexts). The action calculus $\mathsf{AC}(\mathbb{K})$ and its corresponding type theory $\mathsf{T}(\mathbb{K})$ are therefore equivalent.

Proposition 9

1. If $a = b : m \to n$ in $\mathsf{AC}(\mathbb{K})$ such that $fn(a, b)$ is contained in Γ, then $\Gamma; \vec{w}{:}m \vdash \Phi_{\vec{w}:m}(a) = \Phi_{\vec{w}:m}(b) : n$ in $\mathsf{T}(\mathbb{K})$ for an arbitrary linear context $\vec{w}{:}m$.
2. If $\Gamma; \Delta \vdash t = s : n$ in $\mathsf{T}(\mathbb{K})$ then $\Psi(\Gamma; \Delta \vdash t{:}n) = \Psi(\Gamma; \Delta \vdash s{:}n) : |\Delta| \to n$ in $\mathsf{AC}(\mathbb{K})$.
3. $\Psi(\Gamma; \Delta \vdash \Phi_\Delta(a){:}n) = a : |\Delta| \to n$ in $\mathsf{AC}(\mathbb{K})$, if $fn(a)$ is contained in Γ.
4. If $\Gamma; \vec{w}{:}m \vdash t{:}n$ in $\mathsf{T}(\mathbb{K})$ then $\Gamma; \vec{w}{:}m \vdash \Phi_{\vec{w}:m}(\Psi(\Gamma; \vec{w}{:}m \vdash t{:}n)) = t{:}n$ in $\mathsf{T}(\mathbb{K})$.

4 Categorical Models

The type theory given in section 3 has categorical models given by Power's elementary control structures [Pow96]. In this section, we define the models, and give an interpretation of the type theory in the models. This interpretation

is sound and complete, by standard term-model arguments providing an initial model. It can be shown that the translations between the type theory and the corresponding action calculi respect their semantics. With this and proposition 9 one sees that our results are the type-theoretic analogue of Power's.

4.1 Action Models

The action models are constructed from a *carrier* $(\mathcal{C}, \mathcal{S}, F)$, where $\mathcal{C}$ is a strict cartesian category which models the free names, $\mathcal{S}$ is a strict symmetric monoidal category which models arbitrary terms of $\mathsf{AC}(\mathbb{K})$, and $F : \mathcal{C} \to \mathcal{S}$ is a strict symmetric monoidal functor which embeds the cartesian structure in the symmetric monoidal structure. An action model (over $\mathbb{K}$) further provides an interpretation function of the prime arities as objects of $\mathcal{C}$, and of control operators as natural transformations.

Definition 10 (Action Models)
An *action model* over signature $\mathbb{K}$, denoted by $\mathcal{A}$, consists of a carrier $(\mathcal{C}, \mathcal{S}, F)$ together with an interpretation function $[\![_]\!]_{\mathsf{P}} : \mathsf{P} \to \mathsf{obj}(\mathcal{C})$, and for each operator K with arity $((m_1, n_1), \ldots, (m_r, n_r)) \to (m, n)$, a natural transformation

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

where $[\![_]\!] : \mathsf{M} \to \mathsf{obj}(\mathcal{C})$ is defined by $[\![p_1, \ldots, p_r]\!] = [\![p_1]\!]_{\mathsf{P}} \times \ldots [\![p_r]\!]_{\mathsf{P}}$ and $(_)' = F(_)$. Where convenient, we omit the subscripts from $[\![_]\!]_{\mathsf{P}}$ and $[\![_]\!]_{\mathcal{K}}$.

Remark 11
In [Pow96], the category $\mathcal{C}$ is the free cartesian category generated from the set of primes P and F is the identity-on-objects functor. We prefer a simpler, more general condition, and note that the term model satisfies the stronger conditions. Power also includes a preorder on the morphisms of $\mathcal{S}$ to model the the action calculi dynamics, which we omit.

Definition 12 (Action morphisms)
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 strict cartesian functor and $f_s : \mathcal{S}_1 \to \mathcal{S}_2$ is a strict symmetric monoidal functor such that: $F_2 \circ f_c = f_s \circ F_1$; for each $p \in \mathsf{P}$ we have $[\![p]\!]_{\mathsf{P}}^{\mathcal{A}_1}; f_c = [\![p]\!]_{\mathsf{P}}^{\mathcal{A}_2}$; and, for each operator K with arity $((m_1, n_1), \ldots, (m_r, n_r)) \to (m, n)$, we have $f_s(([\![\mathsf{K}]\!]_{\mathcal{K}}^{\mathcal{A}_1})_{(_)}(..)) = ([\![\mathsf{K}]\!]_{\mathcal{K}}^{\mathcal{A}_2})_{f_c(_)}(f_s(..))$ in

$$\mathsf{Nat}_{\mathcal{C}_1}(\prod_{i=1,\ldots,r} \mathcal{S}_1(F_1(_) \otimes ([\![m_i]\!]^{\mathcal{A}_1})', ([\![n_i]\!]^{\mathcal{A}_1})'), \mathcal{S}_2(F_2 f_c(_) \otimes ([\![m]\!]^{\mathcal{A}_2})', ([\![n]\!]^{\mathcal{A}_2})')).$$

An action model is *small* when its component categories are small. The *category of small action models*, $\mathbf{Mod}(\mathsf{T}(\mathbb{K}))$, is the category whose objects are the small action models and whose morphisms are the action morphisms, with the obvious identities and composition.

REMARK 13
There are at least two other possible approaches to modelling action calculi categorically. The first of these is to use fibrations, as in the fibrational control structures of [HP95]; the second is to use indeterminates, freely adding morphisms $x : 1 \to [\![p]\!]$ to $\mathcal{C}$ in such a way that the relevant structure is preserved. One advantage of this latter approach is its clear modelling of free names, and in fact it adapts the results of Gardner [Gar98] who adds indeterminates to her closed action calculi setting to recapture the expressiveness of free names. In independent but related work, Pavlović [Pav97] adds indeterminates to his models for the closed action calculi and points out the connection with the standard categorical notion of functional completeness.

4.2 Interpretation

In this section, 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. First, we require some notation. Given the interpretation function $[\![_]\!] : \mathsf{M} \to \mathsf{obj}(\mathcal{C})$, we extend the function to intuitionistic contexts defining $[\![\Gamma]\!] = [\![|\Gamma|]\!]$, to linear contexts defining $[\![\Delta]\!] = F[\![|\Delta|]\!]$, and finally to contexts $\Gamma;\Delta$ defining $[\![\Gamma;\Delta]\!] = F([\![\Gamma]\!]) \otimes [\![\Delta]\!]$. We sometimes omit the semantic brackets for clarity of presentation.

DEFINITION 14 (INTERPRETATION OF $T(\mathbb{K})$)
Given a type theory $\mathsf{T}(\mathbb{K})$ and action model $\mathcal{A}$, the interpretation $[\![_]\!]$ of sequents $\Gamma;\Delta \vdash t{:}m$ in the type theory as morphisms $[\![\Gamma;\Delta]\!] \to [\![m]\!]'$ in $\mathcal{S}$ is defined by induction on the derivation of sequents, where we (again) elide the distinction between derivations and typed terms:

- Axiom: $[\![\Gamma; w{:}p \vdash w{:}p]\!] = F(\mathsf{disc}_{[\![\Gamma]\!]}) \otimes \mathsf{id}_{p'}$, where $\mathsf{disc}_{[\![\Gamma]\!]} : [\![\Gamma]\!] \to 1$ denotes the morphism to the terminal object 1 in $\mathcal{C}$.
- Exchange:

$$\frac{[\![\Gamma;\Delta_1, v{:}p, w{:}q, \Delta_2 \vdash t{:}m]\!] = f}{[\![\Gamma;\Delta_1, w{:}q, v{:}p, \Delta_2 \vdash t{:}m]\!] = (\mathsf{id}_{\Gamma'} \otimes \mathsf{id}_{\Delta_1} \otimes \sigma_{q',p'} \otimes \mathsf{id}_{\Delta_2}); f}$$

  where $\sigma_{p',q'} : p' \otimes q' \to q' \otimes p'$ denotes the permutation natural isomorphism in $\mathcal{S}$.
- Name introduction: $[\![\Gamma_1, x^p, \Gamma_2; _ \vdash x{:}p]\!] = F(\mathsf{disc}_{[\![\Gamma_1]\!]}) \otimes \mathsf{id}_{p'} \otimes F(\mathsf{disc}_{[\![\Gamma_2]\!]})$
- Name elimination:

$$\frac{[\![\Gamma;\Delta_1 \vdash t{:}p]\!] = f \quad [\![\Gamma, x^p;\Delta_2 \vdash s{:}m]\!] = g}{\begin{array}{l}[\![\Gamma;\Delta_1,\Delta_2 \vdash \mathsf{let}\ \langle x^p\rangle\ \mathsf{be}\ t\ \mathsf{in}\ s{:}m]\!] = \\ \qquad (F(\mathsf{copy}_{[\![\Gamma]\!]}) \otimes \mathsf{id}_{\Delta_1} \otimes \mathsf{id}_{\Delta_2}); (\mathsf{id}_{\Gamma'} \otimes f \otimes \mathsf{id}_{\Delta_2}); g\end{array}}$$

  where $\mathsf{copy}_{[\![\Gamma]\!]} : [\![\Gamma]\!] \to [\![\Gamma]\!] \times [\![\Gamma]\!]$ is the diagonal morphism in $\mathcal{C}$.

– Tensor introduction:

$$\frac{[\![\Gamma;\Delta_i \vdash t_i{:}m_i]\!] = f_i}{[\![\Gamma;\Delta_1,\ldots,\Delta_n \vdash \otimes(t_1,\ldots,t_r){:}m_1,\ldots,m_r]\!] = \mathsf{perm};(f_1 \otimes \ldots \otimes f_r)}$$

where perm is the evident permutation and copy morphism $[\![\Gamma;\Delta_1,\ldots\Delta_r]\!] \to [\![\Gamma;\Delta_1]\!] \otimes [\![\Gamma;\Delta_2]\!] \ldots \otimes [\![\Gamma;\Delta_r]\!]$.

– Tensor elimination:

$$\frac{[\![\Gamma;\Delta_1 \vdash t{:}m]\!] = f \quad [\![\Gamma;\Delta_2,\vec{w}{:}m \vdash s{:}n]\!] = g}{\begin{array}{l}[\![\Gamma;\Delta_1,\Delta_2 \vdash \mathsf{let}\ \otimes\vec{w}\ \mathsf{be}\ t\ \mathsf{in}\ s{:}m]\!] = \\ \qquad (F(\mathsf{copy}_{[\![\Gamma]\!]}) \otimes \mathsf{id}_{\Delta_1\otimes\Delta_2});(\mathsf{id}_{\Gamma'} \otimes f \otimes \mathsf{id}_{\Delta_2});(\mathsf{id}_{\Gamma'} \otimes \sigma_{m',\Delta_2});g\end{array}}$$

– Control rule: Given the natural transformation $[\![\mathsf{K}]\!]_{\mathcal{K}} : \prod_{i=1,\ldots,r} \mathcal{S}(F(_) \otimes m_i', n_i') \to \mathcal{S}(F(_) \otimes m', n')$, we have

$$\frac{[\![\Gamma;\vec{w}_i{:}m_i \vdash t_i{:}n_i]\!] = f_i \quad [\![\Gamma;\Delta \vdash s{:}m]\!] = g \quad i = 1\ldots r}{\begin{array}{l}[\![\Gamma;\Delta \vdash \mathsf{K}((\vec{w}_1{:}m_1)t_1,\ldots,(\vec{w}_r{:}m_r)t_r;s)]\!] = \\ \qquad (F(\mathsf{copy}_{[\![\Gamma]\!]}) \otimes \mathsf{id}_\Delta);(\mathsf{id}_{\Gamma'} \otimes g);[\![\mathsf{K}]\!]_{\Gamma'}(f_1,\ldots,f_r)\end{array}}$$

The proof of the soundness of the interpretation is straightforward. Completeness is proved by defining a term model. The basic idea is that the morphisms in the cartesian category are constructed from lists of sequents $\Gamma;_ \vdash t{:}m$, where t does not contain a control operator, and the symmetric monoidal category $\mathcal{S}$ is constructed from arbitrary sequents $_;\Delta \vdash t{:}m$. Due to space restrictions, we do not give the construction; the details can be found in [BGHP96].

PROPOSITION 15

1. (Soundness) $\Gamma;\Delta \vdash t = s : m$ implies $[\![\Gamma;\Delta \vdash t{:}m]\!] = [\![\Gamma;\Delta \vdash s{:}m]\!]$ in any action model.
2. (Completeness) Given derivations $\Gamma;\Delta \vdash t{:}m$ and $\Gamma;\Delta \vdash s{:}m$ in type theory $\mathsf{T}(\mathbb{K})$, if in every action model $[\![\Gamma;\Delta \vdash t{:}m]\!] = [\![\Gamma;\Delta \vdash s{:}m]\!]$ then $\Gamma;\Delta \vdash t = s : m$.
3. (Initiality) There is an initial term model $\mathcal{A}_T$.

5 Linear Logic

The type theory presented in this section is essentially the LNL (Linear and Non-Linear logic) of Benton [Ben95]. It consists of intuitionistic entailments $\Gamma \vdash_{\mathcal{C}} M{:}X$ and linear entailments $\Gamma;\Delta \vdash_{\mathcal{L}} L{:}A$, with operators F and G to pass between the entailment relations.

We assume a set of primitive intuitionistic types P. The sets of intuitionistic types, denoted by $X, Y, \ldots$, and linear types, denoted by $A, B, \ldots$, are given by the grammars

$$X ::= p \in \mathsf{P} \mid 1 \mid X \times X \mid G(A) \qquad A ::= I \mid A \otimes A \mid A \multimap A \mid F(X)$$

LNL also includes an intuitionistic arrow type, although this is actually not necessary to capture linear logic. We also assume a set C of constants, ranged over by c; each c has a linear type A_c. With P this determines the LNL-signature $\mathbb{C} = (\mathsf{P}, \mathsf{C})$. We also assume a set X of intuitionistic variables ranged over by $x, y \ldots$ and a set of linear variables W ranged over by $w, v \ldots$. Now the sets of intuitionistic terms, denoted by $M, N, \ldots$, and linear terms, denoted by $K, L, \ldots$, are defined by:

$$\begin{aligned} M ::= {} & x \mid i \mid \langle M, M\rangle \mid \pi_1 \mid \pi_2 \mid G(L) \\ L ::= {} & c \mid w \mid * \mid \mathsf{let}\ *{:}I\ \mathsf{be}\ L\ \mathsf{in}\ L \mid L \otimes L \mid \mathsf{let}\ w \otimes w{:}A \otimes A\ \mathsf{be}\ L\ \mathsf{in}\ L \\ & \mid \lambda w{:}A.L \mid LL \mid F(M) \mid \mathsf{let}\ F(x){:}F(X)\ \mathsf{be}\ L\ \mathsf{in}\ L \mid \mathsf{derelict}(M) \end{aligned}$$

Definition 16 (The Typing Rules)
The type theory for LNL over LNL-signature $\mathbb{C} = (\mathsf{P}, \mathsf{C})$, denoted by $\mathsf{LNL}(\mathbb{C})$, is described by the following rules:

$$\Gamma; w{:}A \vdash_{\mathcal{L}} w{:}A \qquad \Gamma, x{:}X \vdash_{\mathcal{C}} x{:}X$$

$$\frac{}{\Gamma \vdash_{\mathcal{C}} i{:}1} \qquad \frac{\Gamma \vdash_{\mathcal{C}} M{:}X \times Y}{\Gamma \vdash_{\mathcal{C}} \pi_1 M{:}X}$$

$$\frac{\Gamma \vdash_{\mathcal{C}} M{:}X \quad \Gamma \vdash N{:}Y}{\Gamma \vdash_{\mathcal{C}} \langle M, N\rangle{:}X \times Y} \qquad \frac{\Gamma \vdash_{\mathcal{C}} M{:}X \times Y}{\Gamma \vdash_{\mathcal{C}} \pi_2 M{:}Y}$$

$$\Gamma; _ \vdash_{\mathcal{L}} *{:}I \qquad \frac{\Gamma; \Delta_1 \vdash_{\mathcal{L}} K{:}I \quad \Gamma; \Delta_2 \vdash_{\mathcal{L}} L{:}A}{\Gamma; \Delta_1, \Delta_2 \vdash_{\mathcal{L}} \mathsf{let}\ *\ \mathsf{be}\ K\ \mathsf{in}\ L{:}A}$$

$$\frac{\Gamma; \Delta_1 \vdash_{\mathcal{L}} K{:}A \quad \Gamma; \Delta_2 \vdash_{\mathcal{L}} L{:}B}{\Gamma; \Delta_1, \Delta_2 \vdash_{\mathcal{L}} K \otimes L{:}A \otimes B} \qquad \frac{\Gamma; \Delta_1 \vdash_{\mathcal{L}} K{:}A \otimes B \quad \Gamma; \Delta_2, v{:}A, w{:}B \vdash_{\mathcal{L}} L{:}C}{\Gamma; \Delta_1, \Delta_2 \vdash_{\mathcal{L}} \mathsf{let}\ v \otimes w\ \mathsf{be}\ K\ \mathsf{in}\ L{:}C}$$

$$\frac{\Gamma; \Delta, v{:}A \vdash_{\mathcal{L}} K{:}B}{\Gamma; \Delta \vdash_{\mathcal{L}} \lambda v{:}A.K{:}A \multimap B} \qquad \frac{\Gamma; \Delta_1 \vdash_{\mathcal{L}} K{:}A \multimap B \quad \Gamma; \Delta_2 \vdash_{\mathcal{L}} L{:}A}{\Gamma; \Delta_1, \Delta_2 \vdash_{\mathcal{L}} KL{:}B}$$

$$\frac{\Gamma \vdash_{\mathcal{C}} M{:}X}{\Gamma; _ \vdash_{\mathcal{L}} F(M){:}FX} \qquad \frac{\Gamma; \Delta_1 \vdash_{\mathcal{L}} K{:}FX \quad \Gamma, x{:}X; \Delta_2 \vdash_{\mathcal{L}} L{:}C}{\Gamma; \Delta_1, \Delta_2 \vdash_{\mathcal{L}} \mathsf{let}\ F(x)\ \mathsf{be}\ K\ \mathsf{in}\ L{:}C}$$

$$\frac{\Gamma; _ \vdash_{\mathcal{L}} K{:}A}{\Gamma \vdash_{\mathcal{C}} G(K){:}GA} \qquad \frac{\Gamma \vdash_{\mathcal{C}} M{:}GA}{\Gamma; _ \vdash_{\mathcal{L}} \mathsf{derelict}(M){:}A}$$

$$\Gamma; _ \vdash_{\mathcal{L}} c{:}A_c \ \ (\text{for } c{:}A_c \in \mathsf{C})$$

With action calculi and their corresponding type theories, the tensor is strict. The tensor in LNL is not strict. We interpret the list structure using the non-strict tensor by defining $\otimes(_) = *$ and $\otimes(A_1, \ldots A_r) = (..(A_1 \otimes A_2) \ldots \otimes A_r)$ for $r > 0$. We adapt this definition for lists of terms in the obvious way, and define a term construct $\mathsf{let}\ \otimes\vec{w}\ \mathsf{be}\ t\ \mathsf{in}\ u$ along the same lines.

We include constants for LNL which correspond to the controls operators arising from a signature $\mathbb{K}$. In $\mathsf{T}(\mathbb{K})$, we have control terms that have the form $\mathsf{K}((\vec{w}_1 : m_1)t_1, \ldots, (\vec{w}_r : m_r)t_r; s)$. In LNL, we are able to make use of the linear λ-abstraction to mimic the abstractions in the control term.

DEFINITION 17
LNL *over signature* $\mathbb{K} = (\mathsf{P}, \mathcal{K})$, denoted by $\mathsf{LNL}(\mathbb{K})$, is the type theory $\mathsf{LNL}(\mathsf{P}, \mathsf{C}_\mathcal{K})$ where $\mathsf{C}_\mathcal{K}$ is the set

$$\{c_\mathsf{K} : \otimes_{i=1,\ldots,r}(m_i^\circ \multimap n_i^\circ) \multimap (m^\circ \multimap n^\circ) \mid \mathsf{K} \in \mathcal{K} \text{ and has arity } ((m_1, n_1), \ldots, (m_r, n_r)) \to (m, n)\}$$

in which we write $(p_1, \ldots, p_s)^\circ$ for $\otimes_{i=1,\ldots,s} F(p_i)$.

5.1 Linear Models

Our models of the linear type theory given in definition 16, called LNL models, are adaptations of the models given in [BBdPH93]. It is convenient to work with strict versions of the models; the non-strict case is described in [BGHP96]. The *carrier* of an LNL model is a quadruple $(\mathcal{C}, \mathcal{S}, F, G)$ where $\mathcal{C}$ is a strict cartesian category, $\mathcal{S}$ is a strict symmetric monoidal closed category and $F : \mathcal{C} \to \mathcal{S}$ is a strict symmetric monoidal functor with right adjoint G. Each model has appropriate interpretation functions for the primitive types and constants.

DEFINITION 18 (THE MODELS OF LNL($\mathbb{C}$))
A strict LNL model *over a LNL-signature* $\mathbb{C} = (\mathsf{P}, \mathsf{C})$, denoted by $\mathcal{L}$, is a carrier $(\mathcal{C}, \mathcal{S}, F, G)$, with interpretation functions $[\![_]\!]_\mathsf{P} : \mathsf{P} \to \mathsf{obj}(\mathcal{C})$ and $[\![_]\!]_\mathsf{C}$, where $[\![c]\!]_\mathsf{C} \in \mathcal{S}(I, [\![A_c]\!])$ for $c : A_c$ and $[\![A_c]\!]$ is defined in the obvious way.

DEFINITION 19 (LNL($\mathbb{C}$) MORPHISM)
A strict $\mathsf{LNL}(\mathbb{C})$ morphism $f : \mathcal{L}_1 \to \mathcal{L}_2$ between two strict LNL models over signature $\mathbb{C}$ (with carriers $(\mathcal{C}_1, \mathcal{S}_1, F_1, G_1)$ and $(\mathcal{C}_2, \mathcal{S}_2, F_2, G_2)$ respectively) is a pair (f_c, f_s) where $f_c : \mathcal{C}_1 \to \mathcal{C}_2$ is a strict cartesian functor and $f_s : \mathcal{S}_1 \to \mathcal{S}_2$ is a strict symmetric monoidal closed functor, such that (f_c, f_s) is a map of adjoints from $F_1 \dashv G_1$ to $F_2 \dashv G_2$ (see [Mac71]), for all $p \in \mathsf{P}$ we have $[\![p]\!]_\mathsf{P}^{\mathcal{L}_1}; f_c = [\![p]\!]_\mathsf{P}^{\mathcal{L}_2}$ and for all $c \in \mathsf{C}$, we have $[\![c]\!]_\mathsf{C}^{\mathcal{L}_1}; f_s = [\![c]\!]_\mathsf{C}^{\mathcal{L}_2}$.

Given an action calculus signature $\mathbb{K} = (\mathsf{P}, \mathcal{K})$, the *category of strict small LNL*($\mathbb{K}$) *models*, denoted by $\mathbf{Mod}(\mathsf{LNL}(\mathbb{K}))$, is the category whose objects are strict small $\mathsf{LNL}(\mathbb{C})$ models and whose morphisms are strict $\mathsf{LNL}(\mathbb{C})$ morphisms where $\mathbb{C} = (\mathsf{P}, \mathsf{C}_\mathcal{K})$.

The interpretation of the type theory $\mathsf{LNL}(\mathbb{C})$ in a (strict) $\mathsf{LNL}(\mathbb{C})$ model $\mathcal{L}$ sends derivations of sequents $\Gamma; \Delta \vdash_\mathcal{L} L : A$ to arrows $[\![\Gamma; \Delta]\!] \xrightarrow{[\![\Gamma; \Delta \vdash_\mathcal{L} L : A]\!]} [\![A]\!]$ in $\mathcal{S}$, and derivations of sequents $\Gamma \vdash_\mathcal{C} M : X$ to arrows $[\![\Gamma]\!] \xrightarrow{[\![\Gamma \vdash_\mathcal{C} M : X]\!]} [\![X]\!]$ in $\mathcal{C}$. The type constructors F and G are interpreted using the functors F and G in the model. We omit the interpretation; it is similar in spirit to the one given

in definition 14. Benton has shown that the interpretation is sound. We have shown completeness, by constructing a (strict) initial term model, denoted by $\mathcal{L}_T$.

6 Translation

In this section, we give the translation from $\mathsf{T}(\mathbb{K})$ to $\mathsf{LNL}(\mathbb{K})$ and show how it corresponds to a functor ρ between the categories of their models, following the general ideas of functorial semantics mentioned in the introduction.

Definition 20 (Translation from T(𝕂) to LNL(𝕂))
The translation $(_)^\circ$ from $\mathsf{T}(\mathbb{K})$ to $\mathsf{LNL}(\mathbb{K})$ is defined inductively on the structure of the types and terms, where we use the same linear variable sets in $T(\mathbb{K})$ and $\mathsf{LNL}(\mathbb{K})$, and assume that the intuitionistic variable set of $\mathsf{LNL}(\mathbb{K})$ includes that of $T(\mathbb{K})$:

types	**terms**
$p^\circ = Fp$	$w^\circ = w$
$(p_1 \dots p_r)^\circ = \otimes(p_1^\circ, \dots, p_r^\circ)$	$\langle x^p \rangle^\circ = F(x)$
	$(\mathsf{let}\ \langle x^p \rangle\ \mathsf{be}\ t\ \mathsf{in}\ s)^\circ = \mathsf{let}\ F(x){:}p^\circ\ \mathsf{be}\ t^\circ\ \mathsf{in}\ s^\circ$
	$\otimes(t_1, \dots, t_r)^\circ = \otimes(t_1^\circ, \dots, t_r^\circ)$
	$(\mathsf{let}\ \otimes\vec{w}{:}m\ \mathsf{be}\ t\ \mathsf{in}\ s)^\circ = \mathsf{let}\ \otimes\vec{w}{:}m^\circ\ \mathsf{be}\ t^\circ\ \mathsf{in}\ s^\circ$
	$\mathsf{K}((\vec{w}_1{:}m_i)t_1, \dots, (\vec{w}_r{:}m_r)t_r; s)^\circ =$
	$\mathsf{c_K}\big(\underset{i=1,\dots,r}{\otimes} FG(\lambda\vec{w}_i{:}m_i^\circ.t_i^\circ)\big)\ \ s^\circ$

Now setting $(x_1^{p_1} \dots x_r^{p_r})^\circ = x_1{:}p_1 \dots x_r{:}p_r$ for intuitionistic contexts and $\Delta^\circ = \Delta$ for linear contexts, we have that if $\Gamma; \Delta \vdash t{:}m$ then $\Gamma^\circ; \Delta^\circ \vdash_{\mathcal{L}} t^\circ{:}m^\circ$.

Lemma 21
There is a functor $\rho : \mathbf{Mod}(\mathsf{LNL}(\mathbb{K})) \to \mathbf{Mod}(\mathsf{T}(\mathbb{K}))$.

Proof Given $\mathsf{LNL}(\mathbb{K})$ model $\mathcal{L}$ with carrier $(\mathcal{C}, \mathcal{S}, F, G)$, we define $\rho(\mathcal{L})$ to be the action model having carrier $(\mathcal{C}, \mathcal{S}, F)$ with the same interpretation function on primes and the natural transformation $[\![\mathsf{K}]\!]_{\mathcal{K}}$ for each $\mathsf{K} \in \mathcal{K}$ constructed using $[\![c_\mathsf{K}]\!]_{\mathsf{C}}$ and the isomorphism

$$\mathsf{Nat}_{\mathcal{C}}\Big(\prod_{i=1,\dots,n} \mathcal{S}(F(_) \otimes \alpha_i, \beta_i), \mathcal{S}(F(_) \otimes \alpha, \beta)\Big) \simeq \mathcal{S}\Big(\bigotimes_{i=1,\dots,n} FG(\alpha_i \multimap \beta_i), \alpha \multimap \beta\Big).$$

Given the strict LNL-morphism $(f_c, f_s) : \mathcal{L}_1 \to \mathcal{L}_2$, it is not difficult to show that $(f_c, f_s) : \rho(\mathcal{L}_1) \to \rho(\mathcal{L}_2)$ is an action morphism. □

Thus the functor ρ is a forgetful functor which discards the extra structure of the LNL-models.

PROPOSITION 22
Let $\mathcal{L}$ be a model of $\mathsf{LNL}(\mathbb{K})$, and suppose that $\Gamma;\Delta \vdash t{:}m$. Then

$$[\![\Gamma^\circ;\Delta^\circ \vdash t^\circ{:}m^\circ]\!]^{\mathcal{L}} = [\![\Gamma;\Delta \vdash t{:}m]\!]^{\rho(\mathcal{L})}.$$

COROLLARY 23 (SOUNDNESS)
If $\Gamma;\Delta \vdash t = s{:}m$ in $\mathsf{T}(\mathbb{K})$ then $\Gamma^\circ;\Delta^\circ \vdash_{\mathcal{L}} t^\circ = s^\circ{:}m^\circ$ in $\mathsf{LNL}(\mathbb{K})$.

Proof By proposition 22 if $\Gamma;\Delta \vdash t = s{:}m$ holds in every model of $\mathsf{T}(\mathbb{K})$ then $\Gamma^\circ;\Delta^\circ \vdash_{\mathcal{L}} t^\circ = s^\circ{:}m^\circ$ holds in every model of $\mathsf{LNL}(\mathbb{K})$. The result then follows by the soundness of $\mathsf{T}(\mathbb{K})$ and the completeness of $\mathsf{LNL}(\mathbb{K})$. □

7 Conservativity Result

The conservativity of the syntactic translation $(_)^\circ$ from $\mathsf{T}(\mathbb{K})$ to $\mathsf{LNL}(\mathbb{K})$ is proved by constructing a model of linear logic from an action model in such a way that the structure of the action model is faithfully preserved (Corollary 25). Our construction is based on the fact that the presheaf category of a small (symmetric) monoidal category is its free (symmetric) monoidal cocompletion [Day70,IK86]. Related results are described systematically in [PR97] and used in [GH97] in essentially the same manner.

LEMMA 24
Let $\mathcal{C}$ and $\mathcal{D}$ be small strict symmetric monoidal categories, with a strict symmetric monoidal functor $F : \mathcal{C} \to \mathcal{D}$. Then there exist small strict symmetric monoidal closed categories $\widehat{\mathcal{C}}$ and $\widehat{\mathcal{D}}$, fully faithful strict symmetric monoidal functors $i_{\mathcal{C}} : \mathcal{C} \to \widehat{\mathcal{C}}$ and $i_{\mathcal{D}} : \mathcal{D} \to \widehat{\mathcal{D}}$ together with a strict symmetric monoidal functor $\widehat{F} : \widehat{\mathcal{C}} \to \widehat{\mathcal{D}}$ such that the induced square commutes and $\widehat{F}$ has a right adjoint.

Proof Following [Day70,IK86], we know that the presheaf category $[\mathcal{C}^{\mathrm{op}}, \mathbf{Set}]$ is a free symmetric monoidal cocompletion of $\mathcal{C}$ and the Yoneda embedding is strong symmetric monoidal; the monoidal product in the presheaf category is given by the coend $G \otimes' H = \int^{X,Y} GX \times HY \times \mathcal{C}(_, X \otimes Y)$ and $I' = \mathcal{C}(_, I)$. Note that this definition makes sense in the enriched setting [Kel82], though here we do not need this generality (but see section 9). For our purpose, we need to take the strict equivalent of the presheaf category and the Yoneda embedding, which we shall denote by $\widehat{\mathcal{C}}$ and $i_{\mathcal{C}}$ respectively (and similarly for $\mathcal{D}$). Then F extends to a strict symmetric monoidal functor $\widehat{F} : \widehat{\mathcal{C}} \to \widehat{\mathcal{D}}$ with a right adjoint, where the latter is induced by $[F^{\mathrm{op}}, \mathbf{Set}]$ and the former is given as a left Kan extension. We can choose $\widehat{F}$ so that the induced square strictly commutes. While $\widehat{\mathcal{C}}$ and $\widehat{\mathcal{D}}$ obtained as above are not small, we can cut down them to be small and retain the required structure – note that we only need full subcategories with small sets of objects and arrows; it is routine but lengthy to write down the explicit description of them as small sets. □

COROLLARY 25
For an action model $\mathcal{A}$, there exists an $\mathsf{LNL}(\mathbb{K})$ model $\mathcal{L}$ such that there is a faithful action morphism from $\mathcal{A}$ to $\rho(\mathcal{L})$.

Proof Assume that $\mathcal{A}$ has the carrier $(\mathcal{C}, \mathcal{S}, F)$. We take the carrier of $\mathcal{L}$ to be $(\widehat{\mathcal{C}}, \widehat{\mathcal{S}}, \widehat{F}, G)$ described as above, where G is a right adjoint of $\widehat{F}$. The interpretation function for primes is given by $[\![p]\!]^{\mathcal{L}}_{\mathsf{P}} = i_{\mathcal{C}}([\![p]\!]^{\mathcal{A}}_{\mathsf{P}})$. Given arity $((m_1, n_1), \ldots, (m_r, n_r)) \to (m, n)$, we note an isomorphism

$$\mathsf{Nat}_{\mathcal{C}}(\prod_{i=1,\ldots,r} \mathcal{S}(F(_) \otimes [\![m_i]\!]^{\mathcal{A}'}, [\![n_i]\!]^{\mathcal{A}'}), \mathcal{S}(F(_) \otimes [\![m]\!]^{\mathcal{A}'}, [\![n]\!]^{\mathcal{A}'})) \simeq \widehat{\mathcal{S}}(\bigotimes_{i=1,\ldots,r} \widehat{F}G([\![m_i]\!]^{\mathcal{L}'} \multimap [\![n_i]\!]^{\mathcal{L}'}), [\![m]\!]^{\mathcal{L}'} \multimap [\![n]\!]^{\mathcal{L}'})$$

obtained by applying the Yoneda lemma repeatedly, which gives $[\![_]\!]^{\mathcal{L}}_{\mathcal{C}_{\mathcal{K}}}$ from $[\![_]\!]^{\mathcal{A}}_{\mathcal{K}}$. □

THEOREM 26 (CONSERVATIVITY)
The translation $(_)^\circ$ is conservative.

Proof Suppose that the sequent $\Gamma^\circ; \Delta^\circ \vdash_{\mathcal{L}} t^\circ = s^\circ : m^\circ$ is provable in $\mathsf{LNL}(\mathbb{K})$. Let $\mathcal{A}_T$ be the initial action model, and construct $\mathcal{L}$ as above. Then the equality holds in $\mathcal{L}$, and hence $\Gamma; \Delta \vdash t = s : m$ holds in $\rho(\mathcal{L})$. However, $i_{\mathcal{S}}$ is faithful, and so $\Gamma; \Delta \vdash t = s : m$ holds in $\mathcal{A}_T$, and is therefore provable in $\mathsf{T}(\mathbb{K})$. □

8 Extensions of Action Calculi

We have emphasised the fact that the functor $\rho : \mathbf{Mod}(\mathsf{LNL}(\mathbb{K})) \to \mathbf{Mod}(\mathsf{T}(\mathbb{K}))$ corresponds to the syntactic translation $(_)^\circ : \mathsf{T}(\mathbb{K}) \to \mathsf{LNL}(\mathbb{K})$. We have also shown that this translation is conservative, by constructing a LNL model from an action model which faithfully preserves the structure of the action model. Our techniques can also be used to provide conservative translations for various extensions of action calculi. We consider three extensions: higher-order action calculi introduced by Milner in [Mil94a], action calculi with code, and linear action calculi, which we introduce. We give a brief summary of our results: the full details are given in [BGHP96].

Higher-order action calculi extend action calculi, allowing closures of actions to be created and substituted for free names in other actions.

DEFINITION 27 (HIGHER-ORDER ACTION CALCULI)
The *higher-order action calculus* $\mathsf{HAC}(\mathbb{K})$ is given by extending the definition of action calculi as follows:

1. the set of primes and the set of arities are constructed from the following abstract grammars:

$$\text{set of primes } p ::= p' \in \mathsf{P} \mid m \Rightarrow m$$
$$\text{set of arities } m ::= p \mid m \otimes m \mid \varepsilon$$

where P denotes the set of basic primes specified by the signature, and the tensor is strict;

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

$$\frac{a : m \to n}{\boldsymbol{\lambda}(a) : \varepsilon \to (m \Rightarrow n)} \qquad \mathbf{ap}_{m,n} : (m \Rightarrow n) \otimes m \to n$$

3. the equational theory is generated from the axioms in definition 2, plus the axioms

$$(\boldsymbol{\lambda}(a) \otimes \mathbf{id}) \cdot \mathbf{ap} = a \quad (\beta) \qquad \boldsymbol{\lambda}((\langle x \rangle \otimes \mathbf{id}) \cdot \mathbf{ap}) = \langle x \rangle \quad (\eta)$$
$$(\boldsymbol{\lambda}(a) \otimes \mathbf{id}) \cdot (x)b = b\{\boldsymbol{\lambda}(a)/\langle x \rangle\} \quad (\sigma)$$

In [Mil94a], Milner uses the notation $\ulcorner a \urcorner$ instead of $\boldsymbol{\lambda}(a)$. Hasegawa and Gardner give a type-theoretic formulation of higher-order action calculi [GH97], which is shown to be an extension of Moggi's computational λ-calculus [Mog88] with commutativity. They show that a higher-order model is an action model with carrier $(\mathcal{C}, \mathcal{S}, F)$ such that, for every $B \in \mathcal{S}$, the functor $F(_) \otimes B : \mathcal{C} \to \mathcal{S}$ has a right adjoint.

An alternative way of expressing the higher-order features is to use two extensions to the basic action calculus structure: the *action calculus with code*, whose structure is a fragment of higher-order action calculi, and the *linear action calculus*, which conservatively extends the higher-order action calculus.

Definition 28 (Action Calculi with Code)
An *action calculus with code* $\mathsf{ACC}(\mathbb{K})$ is the fragment of the higher-order action calculus $\mathsf{HAC}(\mathbb{K})$ given by restricting actions of the form $\boldsymbol{\lambda}(a)$ to the case when a has arity $\varepsilon \to m$. More specifically, we write $!m$ for $(\varepsilon \Rightarrow m)$, and extend the terms of action calculi with two extra constructs given by the rules:

$$\frac{a : \varepsilon \to m}{\mathbf{code}(a) : \varepsilon \to !m} \qquad \mathbf{decode}_m : !m \to m$$

The equalities are generated by the axioms in definition 2, plus three extra axioms corresponding to the β, η and σ axioms given in definition 27.

Action calculi with code provide just enough structure to give recursion in the presence of a reflexion operator [Mif96,Has97a]. They are modelled by action models with carrier $(\mathcal{C}, \mathcal{S}, F)$ in which the functor F has a right adjoint.

Linear action calculi extend action calculi with code, by incorporating linear arities, a linear λ-abstraction and application.

Definition 29 (Linear Action Calculi)
The *linear action calculus* $\mathsf{LAC}(\mathbb{K})$ is given by extending the definition of the action calculus with code $\mathsf{ACC}(\mathbb{K})$ as follows:

1. the set of primes is the union of the sets of *intuitionistic* and *linear* primes given, together with the set of arities, by the following grammars:

$$\begin{array}{ll} \text{intuitionistic primes} & p ::= p' \in \mathsf{P} \mid !m \\ \text{linear primes} & l ::= p \multimap m \\ \text{arities} & m ::= p \mid l \mid m \otimes m \mid \varepsilon \end{array}$$

   where P denotes the set of basic primes from the signature, the arities accompanying the names are restricted to the intuitionistic primes, and the tensor is strict;
2. the set of terms is generated by the rules in definitions 1 and 28, plus the rules

$$\frac{a : m \otimes p \to n}{\boldsymbol{\lambda}^L(a) : m \to (p \multimap n)} \qquad \mathbf{ap}^L_{p,n} : (p \multimap n) \otimes p \to n$$

   for any linear or intuitionistic prime p.
3. the equational theory is generated from the axioms in definition 1 and 28, plus the axioms

$$\begin{array}{ll} (\boldsymbol{\lambda}^L(a) \otimes \mathbf{id}) \cdot \mathbf{ap}^L = a & \boldsymbol{\lambda}^L((a \otimes \mathbf{id}) \cdot \mathbf{ap}^L) = a \\ \boldsymbol{\lambda}^L((a \otimes \mathbf{id}) \cdot b) = a \cdot \boldsymbol{\lambda}^L(b) & (x)\boldsymbol{\lambda}^L(a) = \boldsymbol{\lambda}^L((x)a) \end{array}$$

   again for any linear or intuitionistic prime p.

In [BGHP96], we give the type-theoretic formulation of $\mathsf{LAC}(\mathbb{K})$, which is an extension of the type theory in section 3 and corresponds to a strict version of LNL. In the type theory, the last two axioms correspond to moving the intuitionistic and linear let constructs inside the λ-terms. The models for $\mathsf{LAC}(\mathbb{K})$ are given by action models with carrier $(\mathcal{C}, \mathcal{S}, F)$, where $\mathcal{S}$ is closed and F has a right adjoint.

Remark 30
Another linear extension of action calculi with code is possible, more along the lines of Milner's original higher-order action calculi. Instead of the partial closure operation $\boldsymbol{\lambda}^L$ of definition 29, define the closure $\boldsymbol{\lambda}'_L(a) : \epsilon \to (m \multimap n)$ for any action $m \to n$. Given an application operator and axioms corresponding to the β and η axioms of definition 29, this extension gives the minimal extra structure required to obtain a conservativity result over the higher-order action calculi.

The translations, which connect the basic definition of action calculi and its various extensions, can be described by: the evident embeddings α_1 from $\mathsf{AC}(\mathbb{K})$ to $\mathsf{ACC}(\mathbb{K})$ and α_2 from $\mathsf{ACC}(\mathbb{K})$ to $\mathsf{LAC}(\mathbb{K})$; the translation α_3 from $\mathsf{ACC}(\mathbb{K})$ to $\mathsf{HAC}(\mathbb{K})$, which sends $!m$ to $\varepsilon \Rightarrow \alpha_3(m)$; and the translation α_4 from $\mathsf{HAC}(\mathbb{K})$ to $\mathsf{LAC}(\mathbb{K})$, which sends $m \Rightarrow n$ to $!(p_1 \multimap (p_2 \multimap \ldots (p_r \multimap \alpha_4(n)) \ldots))$ where m is $p_1 \otimes \ldots \otimes p_r$ for $r \geq 0$. All these translations are sound, and correspond to functors between the categories of models. Using similar techniques to section 7, we have shown that these translations are sound and conservative, and correspond to functors between the categories of models. The details are given in [BGHP96].

9 Concluding Remarks

We have given a type-theoretic presentation of the static part of action calculi, and shown that it conservatively embeds in a type theory for intuitionistic linear logic by appealing to the corresponding categorical models. Milner defines the dynamics of action calculi using order-enriched categories. It should be possible to extend our results to take account of this notion of dynamics, by using an ordered type theory with suitable order-enriched models; one question that arises is to what extent the controls should be order-enriched.

We have also indicated that our techniques are easily adapted to three extensions of action calculi: the higher-order action calculi introduced by Milner, the action calculi with code and the linear action calculi presented here. Milner has also introduced the *reflexive* action calculi [Mil94b], by adding a reflexion operator to mimic the notion of feedback. Mifsud [Mif96] and Hasegawa [Has97b,Has97a] have given the corresponding categorical models, where the reflexion operator corresponds to the trace operator of Joyal, Street and Verity [JSV96]. It is straightforward to extend the type-theoretic presentation to account for reflexion [Has97a]. We trivially have sound embeddings of each action calculus into its reflexive counterpart, and have sound translations analogous to the $\alpha_1, \ldots, \alpha_4$ given at the end of section 8. Milner has a syntactic proof that the reflexive action calculi conservatively extend action calculi using molecular forms, but it remains an open problem whether any of the other embeddings or translations are conservative. In particular, our semantic techniques do not apply as the presheaf construction does not yield trace operators at higher types.

We have related action calculi with the much-studied world of linear logic. We hope our work will lead to a cross-fertilisation of ideas between these two areas of research, and a further understanding of interactive behaviour.

Acknowledgments We thank John Power for many helpful discussions, and Martin Hyland for pointing out the relevance of the connection between syntactic translations and functors between categories of models.

References

[AC96] M. Abadi and L. Cardelli: *A Theory of Objects*. Monographs in Computer Science, Springer (1996)

[AG97] M. Abadi and A. Gordon: A calculus for cryptgraphic protocols. In *Proc. 4th ACM Conf. Computer and Communications Security*, ACM Press (1997) 36–47

[Acz80] P. Aczel: Frege structures and the notions of proposition, truth and set. In *The Kleene Symposium*, North-Holland (1980) 31–59

[AR94] J. Adámek and J. Rosický: *Locally Presentable and Accessible Categories*. London Mathematical Society Lecture Note Series 189, Cambridge University Press (1994)

[Bar97] A. Barber: *Linear Type Theories, Semantics and Action calculi*. PhD thesis ECS-LFCS-97-371, University of Edinburgh (1997)

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

[BP98] A. Barber and G. Plotkin: Dual intuitionistic linear logic. Submitted (1998)

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

[BBdPH93] N. Benton, G. Bierman, V. de Paiva and J.M.E. Hyland: Linear lambda-calculus and categorical models revisited. In *Proc. Computer Science Logic (CSL'92)*, Springer Lecture Notes in Computer Science 702 (1993) 61–84

[CG97] L. Cardelli and A. Gordon: Mobile ambients. Draft (1997)

[Day70] B.J. Day: On closed categories of functors. In *Midwest Category Seminar Reports IV*, Springer Lecture Notes in Mathematics 137 (1970) 1–38

[Day73] B.J. Day: An embedding theorem for closed categories. In *Category Seminar Sydney*. Springer Lecture Notes in Mathematics 420 (1973) 55–64

[Gar98] P. Gardner: Closed action calculi. To appear in *Theoretical Computer Science* (1998)

[GH97] P. Gardner and M. Hasegawa: Types and models for higher-order action calculi. In *Proc. Theoretical Aspects of Computer Software (TACS'97)*, Springer Lecture Notes in Computer Science 1281 (1997) 583–603

[Gir93] J.-Y. Girard: On the unity of logic. *Annals of Pure and Applied Logic* 59 (1993) 201–217

[Has97a] M. Hasegawa: Recursion from cyclic sharing: traced monoidal categories and models of cyclic lambda calculi. In *Proc. Typed Lambda Calculi and Applications (TLCA'97)*, Springer Lecture Notes in Computer Science 1210 (1997) 196–213

[Has97b] M. Hasegawa: *Models of Sharing Graphs (A Categorical Semantics of Let and Letrec)*. PhD thesis ECS-LFCS-97-360, University of Edinburgh (1997)

[HP95] C. Hermida and A.J. Power: Fibrational control structures. In *Proc. Concurrency Theory (CONCUR'95)*, Springer Lecture Notes in Computer Science 962 (1995) 117–129

[IK86] G.B. Im and G.M. Kelly: A universal property of the convolution monoidal structure. *Journal of Pure and Applied Algebra* 43 (1986) 75–88

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

[Kel82] G.M. Kelly: *Basic Concepts of Enriched Category Theory*. London Mathematical Society Lecture Note Series 64, Cambridge University Press (1982)

[Mac71] S. Mac Lane: *Categories for the Working Mathematician*. Springer Graduate Texts in Mathematics 5 (1971)

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

[Mil94a] R. Milner: Higher-order action calculi. In *Proc. Computer Science Logic (CSL'93)*, Springer Lecture Notes in Computer Science 832 (1994) 238–260

[Mil94b] R. Milner: Action calculi V: reflexive molecular forms (with Appendix by O. Jensen). Unpublished manuscript (1994)

[Mil96] R. Milner: Calculi for interaction. *Acta Informatica* 33(8) (1996) 707–737

[MPW92] R. Milner, J. Parrow and D. Walker: A calculus of mobile processes, part I + II. *Information and Computation* 100(1) (1992) 1–77

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

[Pav97] D. Pavlović: Categorical logic of names and abstraction in action calculi. *Mathematical Structures in Computer Science* 7(6) (1997) 619–637

[Plo90] G. Plotkin: An illative theory of relations. In *Situation Theory and Its Applications, Volume I*, CSLI Lecture Notes Series, Centre for the Study of Language and Information (1990) 133–146

[Pow96] A.J. Power: Elementary control structures. In *Proc. Concurrency Theory (CONCUR'96)*, Springer Lecture Notes in Computer Science 1119 (1996) 115–130

[PR97] A.J. Power and E.P. Robinson: Premonoidal categories and notions of computation. *Mathematical Structures in Computer Science* 7(5) (1997) 453–468

[Sew97] P. Sewell: Global/local subtyping for a distributed π-calculus. Submitted (1997)

Appendix

We explain Aczel's *binding operators* [Acz80,Plo90] using the example of the standard natural deduction rule for $\vee$-elimination:

$$\dfrac{\begin{array}{ccc}(A) & (B) & \\ \vdots & \vdots & \\ C & C & A \vee B\end{array}}{C}$$

where formulae A and B are discharged from the assumptions. The corresponding type-theoretic formulation involves a "cases construction":

$$\dfrac{\Gamma, x{:}A \vdash u{:}C \quad \Gamma, y{:}B \vdash v{:}C \quad \Gamma \vdash t{:}A \vee B}{\Gamma \vdash \mathsf{cases}_{A,B,C}((x{:}A)u, (y{:}B)v, t){:}C}$$

where, the variables x and y are bound in u and v respectively. Note the occurrence of the $\mathsf{cases}_{A,B,C}$ operator. The general rule for such an operator is:

$$\dfrac{\Gamma, \vec{x}_1{:}\vec{A}_1 \vdash t_1{:}B_1 \quad \ldots \quad \Gamma, \vec{x}_r{:}\vec{A}_r \vdash t_r{:}B_r}{\Gamma \vdash \mathsf{K}((\vec{x}_1{:}\vec{A}_1)t_1, \ldots, (\vec{x}_r{:}\vec{A}_r)t_r){:}B}$$

where each $\vec{x}_i{:}\vec{A}_i$ denotes the sequence of distinct variables which are bound in the ith component. These binding operators can be used to give a general account of natural-deduction rules.