Categorical logic of names and abstraction in action calculi

D. Pavlović*
COGS, University of Sussex, Brighton, UK
e-mail: duskop@cogs.susx.ac.uk

June 27, 1996

Abstract

Milner's action calculus implements abstraction in monoidal categories, so that familiar λ-calculi can be subsumed together with the π-calculus and the Petri nets. Variables are generalised to *names*: only a restricted form of substitution is allowed.

In the present paper, the well-known categorical semantics of the λ-calculus is generalised to the action calculus. A suitable functional completeness theorem for symmetric monoidal categories is proved: we determine the conditions under which the abstraction is definable. Algebraically, the distinction between the variables and the names boils down to the distinction between the transcendental and the algebraic elements. The former lead to polynomial extensions, like e.g. the ring $\mathbb{Z}[$

1 Introduction

Algebraically, the λ-abstraction arises from a *property* of certain structures — namely, that each polynomial can be reduced to a normal form with a single coefficient. This property is known as *combinatorial* [5, ch. 6] or *functional*

may change during the execution, since different communication channels that may be received can open the different computation paths, or preempt them. This is the idea of a *mobile* process. The corresponding generalisation of the λ-calculus is the π-calculus [22]. The main difference is that a function is *applied* to its input sequentially, while a mobile process can *communicate* with any of the processes running in parallel with it, provided that there is a common communication channel. The function application has been generalised to the communication, which may be non-sequential and non-deterministic.

The main feature of the π-abstraction is that it does not just bind a parameter x, for which the received input is to be substituted, like the λ-abstraction does, but it also specifies a communication channel y, where the sought input must be received. A π-abstraction operator is thus in the form $y(x)$, and it binds x, and adds y as a free channel name, ready for the input. A process prefixed with such an operator will not consume just any argument immediately preceding it, and substitute it for x, like an old λ-term would do. A process/π-term must first find a parallel process (another π-term) sending its output through y. It can be recognized by a prefix in the form $\overline{y}u$, where u is the name being sent. In case several such are running in parallel, prefixed, say, with $\overline{y}u_1$, $\overline{y}u_2$ etc., only one of their outputs/prefixes will be consumed, and the choice will be made nondeterministically. Any of the channels $u_1, u_2 \ldots$ may thus end up being substituted for x.

The *action calculus* provides means for reducing this complex reduction procedure to a familiar abstraction/application routine. There are two dimensions of this apparent syntactic miracle: the *controls*, and the *dynamics*. The controls mark the input and the output channels, and block the reduction unless these coincide. The dynamics is the reduction preorder that, unlike the Church-Rosser situations, cannot be hidden away from semantics. Indeed, upon different communications, processes may reduce in essentially different ways.

However, in the present paper, we shall neglect the dynamic side, and try to show that the remaining *static action calculus* is a kind of "monoidal λ-calculus" plus controls. The dynamics of the full-blown structure is undoubtedly essential ing adds

Several classes of natural examples, modelling static action calculi, are described in subsection 3.3. A different application of the obtained semantics is presented in subsection 3.2: it is shown how allowing the ordinary substitution reduces the action calculus to the cartesian κ-calculus. This sheds some light on the degeneration of the extensional higher-order action calculus to the ordinary λ-calculus, described in [20] — and shows how narrow is the passage from the cartesian to the monoidal abstraction. The simplicity with which the latter has been introduced in the action calculus conceals a genuinely fundamental idea, mostly behind the concept of a name, with its constrained substitution.

While weakening the cartesian setting leaves the abstraction operations virtually unchanged, it has deep repercussions on the substitution, which on its turn weakens the β-reduction. The original constraints, imposed on the substitution in the action calculus, were computationally motivated: if names are channel parameters, then only the proper channel names should be substituted for them, and surely not arbitrary process expressions. Variables, as the value parameters, on the other hand, accomodate substitution of all expressions that can be evaluated.

The algebraic treatment provides different explanations. In algebra, the substitution is implemented by means of extensions: a variable x, freely adjoined to, say, the ring of integers $\mathbb{Z}$, leads to the polynomial extension $\mathbb{Z}[x]$. The fact that x is free for substitution of any element means that it is *transcendental*, unconstrained by any equations over $\mathbb{Z}$. On the other hand, an *algebraic* element, which does satisfy some equations, can only be replaced by elements satisfying the same equations. E.g., if $x^2 - 2 = 0$ holds for x, then only $-x$ can be substituted for it, without invalidating the equation. Of course, this x is just $\sqrt{2}$ and the substitution constraint formally means that the extension $\mathbb{Z}[\sqrt{2}] = \mathbb{Z}[x]/(x^2 - 2)$ has exactly one nontrivial endomorphism fixing $\mathbb{Z}$, induced by the assignment $\sqrt{2} \mapsto -\sqrt{2}$. Conversely, the "name" $\sqrt{2}$ can be viewed as an abbreviation of something like $[x; x^2 = 2]$. We shall later present names exactly in this form.

The idea that names are some algebraic elements, as opposed to variables as transcendental elements, suggests a general treatment of the constrained substitution, along the lines of Galois theory. Luckily, this general treatment need not be developed very far: the names arising in the action calculus turn out to be the algebraic elements of a very special kind, characterised by a simple set of equations (23–25). They are consequences of the conversion rules of the name abstraction (cf. definition 3.2). The point of the functional completeness is that the name abstraction, together with its conversion

the composition is written in the form $f \circ g$, (referring to $(f \circ g)(x) = f(g(x))$) rather than $g \cdot f$.

The notion of a model is defined in the usual way.

(a) *For each* $\alpha : m \longrightarrow n$ *in* $\mathbb{A}[x;Q]$ *there is a unique* $\kappa x.\alpha : k \otimes m \longrightarrow n$ *in* $\mathbb{A}$*, such that*

$$\alpha \quad = \quad \mathsf{ad}(\kappa x.\alpha) \circ (x \otimes m) \tag{5}$$

(b) *The functor* $\mathsf{ad} : \mathbb{A} \longrightarrow \mathbb{A}[x;Q]$ *has a left adjoint* ab*, such that the composite* $\mathsf{ab} \circ \mathsf{ad}$ *is just tensoring with* k*. The unit and the counit of the adjunction are respectively in the forms* $\eta_m = x \otimes m$ *and* $\varepsilon_m = \omega \otimes m$*, for some* $\omega : k \to \top$*.*

(c) $\mathbb{A}[x;Q]$ *is isomorphic with the Kleisli category for a comonad over the endofunctor* $k \otimes (-) : \mathbb{A} \longrightarrow \mathbb{A}$*.*

Proof. **(a)⇒(b)** Consider the maps

$$\mathbb{A}[x;Q]\Big(m,n\Big) \xrightarrow{\kappa x.(-)}$$

commute. In the presence of (a), this commutativity is equivalent to the equation $\kappa x.\alpha \circ \varphi = (\kappa x.\alpha) \circ \mathsf{ab}(\varphi)$. This is the required naturality of (9) in m.

The naturality in n boils down to the equation $\kappa x.\mathsf{ad}(f) \circ \alpha = f \circ \kappa x.\alpha$, for all $f \in \mathbb{A}(n, n')$. But this is again a consequence of the uniquess part of (a).

Hence the adjunction $\mathsf{ab} \dashv \mathsf{ad}$. By correspondence (6) its unit and counit will be as asserted in (b). So it remains to prove that $\mathsf{ab} \circ \mathsf{ad}(f)$ is $k \otimes f$ for all f from $\mathbb{A}$.

Since $\mathbb{A}[x; Q]$ is a monoidal category, any $\varphi : m' \rightarrow m$ in it satisfies

$$(x \otimes m) \circ \varphi = x \otimes \varphi = (k \otimes \varphi) \circ (x \otimes m'). \tag{12}$$

Putting $\varphi = \mathsf{ad}(f)$, we get that

$$\begin{array}{ccc} m' & \xrightarrow{x \otimes m'} & k \otimes m' \\ {\scriptstyle \mathsf{ad}(f)}\downarrow & & \downarrow{\scriptstyle \mathsf{ad}(k \otimes f)} \\ m & \xrightarrow[x \otimes m]{} & k \otimes m \end{array} \tag{13}$$

commutes, since ad is a monoidal functor, identity on objects, and hence $k \otimes \mathsf{ad}$

functor $\mathsf{ad} : \mathbb{A} \longrightarrow \mathbb{A}[x;Q]$ adds a dummy x, *whereas* $\mathsf{ab} : \mathbb{A}[x;Q] \longrightarrow \mathbb{A}$ abstracts *over* x.

$\mathbb{A}$ *is functionally complete with respect to a set of names* $[X;Q]$ *if for all* $x \in X$ *and* $Y \subseteq X \smallsetminus \{x\}$, $\mathbb{A}[Y;Q]$ *is functionally complete with respect to* $[x;Q]$. *Moreover, the abstraction should be uniform, in the sense that the diagram*

$$\begin{array}{ccc} \mathbb{A}[x,Y;Q] & \xrightarrow{\mathsf{ab}_x^Y} & \mathbb{A}[Y;Q] \\ \downarrow{\scriptstyle \mathsf{ad}_Z^{x,Y}} & & \downarrow{\scriptstyle \mathsf{ad}_Z^Y} \\ \mathbb{A}[x,Y,Z;Q] & \xrightarrow[\mathsf{ab}_x^{Y,Z}]{} & \mathbb{A}[Y,Z;Q] \end{array} \tag{14}$$

must commute for all $Y, Z \subseteq X \smallsetminus \{x\}$. *Moreover, the canonical isomorphism*

$$\mathsf{ab}_y^Z \circ \mathsf{ab}_x^{y,Z} \cong \mathsf{ab}_x^Z \circ \mathsf{ab}_y^{x,Z} \quad : \quad \mathbb{A}[x,y,Z;Q] \longrightarrow \mathbb{A}[Z;Q] \tag{15}$$

induced by the fact that both sides are adjoint to $\mathsf{ad}_{x,y}^Z$, *must come from the symmetry* $\tau : k \otimes \ell \xrightarrow{\sim} \ell \otimes k$.

An abstraction situation *is a pair* $(\mathcal{K}, Q)$, *such that every* $\mathcal{K}_\otimes$*-category* $\mathbb{A}$ *is functionally complete with respect to any set of names* $[X;Q]$.

Remarks. This last quantifier over all sets of names is not as extensive as it appears: by proposition 2.2, $\mathbb{A}$ is functionally complete for $[X;Q]$ if and only it is functionally complete for $[X';Q]$, where X' is the image of $type : X \to |\mathbb{A}|$. To check an abstraction situation, one only needs to consider the subsets X' of $|\mathbb{A}|$ (but still for all $\mathcal{K}_\otimes$-categories $\mathbb{A}$, though).

The notion of functional completeness, as defined above, should not be confused with its homonym in duality theory [12], where, say, the boolean algebra 2 is functionally complete because every function $2^n \to 2$ corresponds to a polynomial. In boolean algebras, there are thus enough polynomials to represent all functions, whereas we are concerned with the situations when there are enough constants to represent all polynomials. The term polynomial completeness might be better, but the usage, at least for the cartesian case, seems completely standard.

2.4 Characterising abstraction

Commutative comonoids and cartesianness. Let be the full subcategory of Set^{op} spanned by the natural numbers: a morphism $m \to n$ in is just a function $m \leftarrow n$. Since Set is the coproduct completion of 1, Set^{op} is the product completion [14]. The free cartesian category generated by 1 is thus $\mathsf{Set}_{\mathrm{fin}}^{op}$, and

is the free *strictly* cartesian category over 1, with + as the cartesian product and 0 as the terminal object.

in [7, def. 3.6], categorical in [8, def. 8.1(4)] and [28, def. 4.1]. The following is closer to the former.

Definition 2.5 *Let* ℓ *be a graphic operation with the arity*

$$\frac{b_1 : \ell \otimes m_1 \to n_1 \qquad b_2 : \ell \otimes m_2 \to n_2 \qquad \cdots \qquad b_r : \ell \otimes m_r \to n_r}{\ell(b_1, \ldots, b_r) : \ell \otimes m \longrightarrow n} \tag{17}$$

Given two such, k *and* ℓ*, an arrow* $u : \ell \to k$ *in* $\mathbb{A}$ *is said to be* admissible *if it satisfies*

$$\ell\Big(b_1(u \otimes m_1), \ldots, b_r(u \otimes m_r)\Big) \quad = \quad k(b_1, \ldots, b_r) \circ (u \otimes m). \tag{18}$$

A monoidal functor $\mathbb{M} \to \mathbb{A}$ *is admissible if its image consists of arrows admissible with respect to a given family* $\{\ \ell\}_{\ell \in \mathbb{M}}$*.*

Finally, a comonoid k *in a* $\mathcal{K}_\otimes$*-category* $\mathbb{A}$ *is admissible if every graphic operation* $\in \mathcal{K}$ *induces a unique family making the monoidal functor* $\to \mathbb{A} : 1 \mapsto k$ *admissible. Such an operation is called* k-control*.*

Each morphism $m \leftarrow n$ of decomposes as $m \hookleftarrow m' \simeq n' \twoheadleftarrow n$, where the first and the last components are monotone. It is not hard to see that a monoidal functor $\to \mathbb{A}$ must take every monotone injection $m \hookleftarrow m'$ to an arrow derived from ω and $\otimes$, every monotone surjection $n' \twoheadleftarrow n$ to an arrow derived from Δ and ω and every bijection $m' \simeq n'$ to a composite of the symmetries τ. Since the class of arrows satisfying (18) is clearly closed under the composition, checking whether $\to \mathbb{A}$ is admissible boils down to checking separately the admissibility of the arrows derived from ω and $\otimes$, from Δ and $\otimes$, and from τ, $\otimes$ and $\circ$. These three parts correspond to conditions 1–3 from [7, def. 3.6].

On the other hand, every tensor power k^j of a commutative comonoid k is a commutative comonoid again. Hence the Kleisli categories $\mathbb{A}_{k^j \otimes}$ for all natural numbers j. The mapping $j \mapsto \mathbb{A}_{k^j \otimes}$ determines an indexed category $^{op} \to \mathsf{Cat}$, with the reindexing induced by the precomposition. Condition (18) now appears as the naturality with respect to this reindexing, and a k-control is just an indexed graph operation on this indexed category. The setting described in [8, def. 8.1(4)] and [28, def. 4.1] is built upon this idea[2].

Proposition 2.6 *A* $\mathcal{K}_\otimes$*-category* $\mathbb{A}$ *is functionally complete with respect to the extension by* $[x; Q]$ *of type* k *if and only if*

(i) k *is an admissible commutative comonoid in* $\mathbb{A}$*, and*

(ii) x *is an admissible comonoid homomorphism in* $\mathbb{A}[x; Q]$*.*

[2]However, it seems that these graphic operations as natural transformations must be total and monotone with respect to the reduction preorder, so that, e.g., currying, or replication [16] cannot be treated directly.

Proof. When $\mathbb{A}$ is functionally complete, the commutative comonoid structure $\top \overset{\omega}{\leftarrow} k \overset{\Delta}{\rightarrow} k \otimes k$ is

$$\omega = \kappa x.\mathrm{id}_\top \tag{19}$$

$$\Delta = \kappa x.x \otimes x. \tag{20}$$

Its admissibility is proved as in [7, sec. 4.2].

The fact that x is an admissible comonoid homomorphism from $\top \overset{\mathrm{id}}{\leftarrow} \top \overset{\mathrm{id}}{\rightarrow} \top \otimes \top$ follows directly from (5).

The other way around, assume (i) and (ii). The abstraction κx is defined inductively. An arrow of $\mathbb{A}[$

$$
\begin{aligned}
&= \quad {}_k\Big(\mathsf{ad}(\kappa x.\alpha_1), \ldots\Big) \circ (x \otimes m) \\
&= \quad \Big(\mathsf{ad}(\kappa x.\alpha_1) \circ (x \otimes m), \ldots\Big) \\
&= \quad (\alpha_1, \ldots, \alpha_r).
\end{aligned}
$$

The remaining cases only depend on x being a comonoid homomorphism. The uniqueness part of 2.2(a) follows by a similar inductive argument. □

Corollary 2.7 *$(\mathcal{K}, Q)$ is an abstraction situation if and only if*

(i) *$\mathcal{K}$ makes all objects into admissible commutative comonoids, while*

(ii) *Q makes all names into admissible comonoid homomorphisms.*

3 Action calculi and control structures

3.1 Abstraction elimination

Definition 3.1 *A monoidal category where every object has a commutative comonoid structure is said to be* semi-cartesian.

An action *category is a $\mathcal{K}_\otimes$-category with a distinguished admissible commutative comonoid structure on every object.*

A semi-cartesian category is cartesian if and only if each object carries a unique comonoid structure, and such structures form two natural families, Δ and ω. The naturality means that all morphisms of the category must be comonoid homomorphisms.

In action categories, the *property*

on all $x \in X$ *and all* $\quad \in \mathcal{K}$.

Proof. Θ is the smallest set of equations satisfying condition 2.7(ii). $\mathbb{A}[X;\Theta]$ is thus the free functionally complete extension of the action category $\mathbb{A}$ by X. The equivalent conditions of proposition 2.2 are thus satisfied for every $x \in X$. We show that this setting coincides with the structure of action calculus.

First of all, axiom σ trivially implies equation (5). The converse follows from 2.2(c).

Axioms γ and δ follow from the uniqueness part of 2.2(a). The converse uses the functoriality of ab_x.

Finally, axiom ζ corresponds to condition (15), while (14) remains implicit in the definition of a uniform ab_x in all contexts. □

Control structures [16]. While an action calculus is built up inductively, as the extension of a given $\mathcal{K}_\otimes$-category by names and abstractions, a (static) control structure is a $\mathcal{K}_\otimes$-category *readily given* with the abstraction functors and with names as distinguished arrows. The syntactic concepts, such as context, or substitution, are then reconstructed algebraically.

Corollary 3.4 *A* $\mathcal{K}_\otimes$*-category* $\mathbb{C}$ *is a (static) control structure if and only if it is equivalent with an extension* $\mathbb{A}[X;Q]$ *of a action category* $\mathbb{A}$*, with* $Q \supseteq \Theta$ (23–25).

Proof. By [16, thm. 4.15], $\mathbb{C}$ is a control structure if and only if it is a quotient of an action calculus $\mathbb{B}[X;\mathsf{ab}]$ along an abstraction preserving $\mathcal{K}_\otimes$-functor. Proposition 3.3 thus yields $\mathbb{C}$ as a quotient of $\mathbb{B}[X;\Theta]$. The subcategory $\mathbb{A}$, spanned in $\mathbb{C}$ by the image of $\mathbb{B}$⟫⊣⇐⟨⇒⇐⟩⊣}⇒∈←∈←{⊗∀∀←←⊗⇜⇞⇔⇒|⊗∃⇐⊣⟩⇒ ⊗∀∀∃∀⇤←←⊗{∀ ⊣⟩∃∃

Proposition 3.5 *Let $u : \ell \to k$ be a morphism in an action*

no controls. The action calculi, described in [18, 3.1–3.3], are of this kind: cartesian categories, or their polynomial extensions, supporting the κ-calculus. No controls.

ii. The departure from the cartesian setting is essentially due to introducing controls. In spite of their diversity, all the original action calculi can be subsumed under a simple syntactic construction, *molecular forms* [21] — which actually yields the free extension of a free cartesian category by a given set of controls. Such extensions turn out to be action categories, of course.

Consider a formal expression $(\vec{x})\langle\vec{y}\rangle$, where all x_i from $\vec{x} = x_0 \cdots x_{m-1}$ are distinct, whereas each y_j from of $\vec{y} = y_0 \cdots y_{n-1}$ occurs in $\vec{x}$ as some x_i. Setting $f(j) = i$ whenever $y_j = x_i$,

The first and the most basic concrete example is the mentioned category Rel. Its monoidal structure and the commutative comonoids are induced by the cartesian structure of Set. But note that these inherited comonoids are not the only ones in Rel. In general, binary relations $\omega : k \nrightarrow 1$ and $\Delta : k \nrightarrow k \otimes k$ (i.e. $\omega \subseteq k$ and $\Delta \subseteq k \times k \times k$) will form a commutative comonoid if and only if for every $a \in k$ holds

$$\omega(a) \iff \forall bc.\Delta(a,b,c) \Leftrightarrow a = b = c \tag{26}$$

$$\neg\omega(a) \iff \exists! b.\ \omega(b) \wedge \Delta(a,a,b) \wedge \Delta(a,b,a). \tag{27}$$

These nonstandard comonoid structures offer a choice of nonstandard action category structures on Rel, with varying notions of name, extension etc. In fact, the morphisms from m to n in the extension of Rel by any name $x : 1 \nrightarrow k$ will always be the k-indexed families of relations $m \nrightarrow n$, but the composition and the identities will vary with the comonoid on k. The reader can work this out using 2.2(c), or noticing that the name x, as an arrow of the extension, is the family $\{x_a : 1 \nrightarrow k\}_{a \in k}$, where each $x_a \subseteq k$ is

$$x_a(b) \iff a = b \wedge \omega(a). \tag{28}$$

Which controls can be added to Rel? In principle, this is a bit like asking which operations can be added to the signature of a given algebra. For action categories there is a canonical choice, though. The details will be explained in [27], but let us here just display the
wi0.0791000.67(suitablf)-13999.6(.412Tdf85.44020Td7Rel)Tj/R610.24Tf

the formulas

$$\omega\Delta^o \quad = \quad \omega \otimes \omega \tag{30}$$

$$\exists\varphi : k \nrightarrow k \quad \forall\overline{\omega} : k \nrightarrow 1.$$
$$\Big(\varphi\varphi^o \subseteq \mathrm{id} \ \wedge \ \overline{\omega} \cap \omega = \emptyset \quad \Longrightarrow \quad \overline{\omega}\Delta^o \quad = \quad (\varphi\overline{\omega} \otimes \omega) \cup (\omega \otimes \varphi\overline{\omega})\Big) \tag{31}$$

which boil down to (26–27) when the classical logic is available.

In general, any allegory [6] or cartesian bicategory [2] subsumes an action category: the latter structure even contains commutative comonoids as a part of the definition. A range of familiar examples is obtained: categories of semilattices, total orders, total relations, partial maps... — they all turn out to support the name abstraction, with various classes of controls. The basic *interaction categories* [1] also fall into this group; SProc even appears as a category of relations for a certain regular fibration [26].

At this point, however, it is probably fair to reiterate that these considerations are restricted to the *static* action calculus. The abundance of the examples suggests that the presented theory of abstraction is actually *too* general to really pin down the intended computational meaning of the full action calculus. As pointed out before, an essential part remains to be captured by narrowing down the dynamics. It conceptual importance is can be seen, e.g., in the fact that the intuitive difference of interaction categories and action calculi can only be captured on the level of dynamics, since interaction categories support the static action calculus, but *fail to satisfy the dynamic axioms*. In fact, very few of the mentioned examples, with their natural hom-set orders, satisfy these axioms. Already in Rel, the identities are neither minimal nor maximal with respect to the inclusion. They are maximal among the partial maps, and minimal among the total relations, but more subtle dynamic requirements will perhaps be needed to really capture ideas.

vi. A different class of examples is obtained by generalising Rel as the Kleisli category for the commutative monad [9] $\wp : \mathsf{Set} \to \mathsf{Set}$. The generalisation is based on the following lemma (roughly from [29, 4.7]).

Lemma 3.6 *A monad T on a monoidal category $\mathbb{V}$ is commutative if and only if the Kleisli category $\mathbb{V}_T$ and the canonical functor $\mathbb{V} \to \mathbb{V}_T$ are monoidal.*

Since a monoidal functor preserves comonoids, $\mathbb{V}_T$ will be an action category (semi-cartesian) as soon as $\mathbb{V}$ is.

According to Moggi [23], many notions of computation are naturally presented as strong monads on cartesian categories: the objects are sets of values, and the monad assigns to each of them the corresponding set of computations. If the base category depicts the maps on values, the induced Kleisli category can be thought of as the *category of computations*. The above lemma now implies that such a category of computations is an action category as soon as the corresponding monad is commutative. And the commutativity in this setting corresponds to the invariance of the order of execution.

This correspondence

References

[1] S. Abramsky et al., Interaction categories and the foundations of the typed concurrent programming, in: *Deductive Program Design: Proceedings of the 1994 Marktoberdorf International Summer School*, ed. M. Broy, (Springer 1995), (also on URL `http://theory.doc.ic.ac.uk`)

[2] A. Carboni and R.F.C. Walters, Cartesian bicategories I, *J. Pure Appl. Algebra* 49(1987) 11–32

[3] P.-L. Curien, *Combinateurs Catégoriques, Algorithmes Séquentiels et Programmation Applicative*, Thèse de Doctorat détat (Univ. Paris VII); updated English version: *Categorical Combinators, Sequential Algorithms and Functional Programming*, Progress in Theor. Comp. Sci. (Birkhäuser 1993)

[4] H.B. Curry, Grundlagen der Kombinatorischen Logik, *Amer. J. Math.* 52(1930) 509–536, 789–834

[5] H.B. Curry et al.,

[16] A. Mifsud et. al. Control Structures, *Proceedings of LICS '95* (IEEE 1995)

[17] R. Milner, *Communication and Concurrency* (Prentice Hall 1989)

[18] R. Milner, Action Calculi and the π-Calculus, in: *Proceedings of the NATO Summer School on Logic and Computation*, (Springer 1993)

[19] R. Milner, Action Calculi I: Axioms and Applications, *appeared as:* Action calculi, or syntactic action structures, *Proceedings of MFCS '93*, Lecture Notes in Computer Science 711 (Springer 1993) 105–121

[20] R. Milner, Action Calculi III (URL `http://theory.doc.ic.ac.uk`)

[21] R. Milner, Action Calculi IV (URL `http://theory.doc.ic.ac.uk`)

[22] R. Milner et al., A calculus of mobile processes I and II, *Inform. and Comput.* 100(1992), 1–77

[23] E. Moggi, Notions of computation as monads, *Inform. and Comput.* 93(1991) 55–92

[24] J. Meseguer and U. Montanari, Petri Nets are Monoids, *Inform. and Comput.* 88/2(1990) 105-155

[25] D. Pavlović, Maps II: Chasing diagrams in categorical proof theory, *J. of the IGPL* 4/2(1996), 1–36 (URL `http://www.mpi-sb.mpg.de/igpl/Journal`)

[26] D. Pavlović, Categorical logic of concurrency and interaction I. Synchronous processes, in: *Theory and Formal Methods of Computing 1994*, C.L. Hankin et al., eds. (World Scientific 1995), 105–141

[27] D. Pavlović, Action calculi and computational monads, *in preparation*

[28] J. Power, Elementary Control Structures, *to appear* in the proceedings of CONCUR '96

[29] J. Power and E. Robinson, Premonoidal categories, *manuscript*

[30] V. Pratt, Chu Spaces and their Interpretation as Concurrent Objects, *in:* J. van Leeuwen (ed.), *Computer Science Today: Recent Trends and Developments*, Lecture Notes in Computer Science 1000 (Springer 1995) 392–405

[31] M. Schönfinkel, Über die Bausteine der Mathematischen Logik, *Math. Annalen* 92(1924) 305–316

[32] D. Scott, Relating Theories of the λ-Calculus, in: *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, eds. J.P. Seldin and J.R. Hindley (Academic Press 1980) 403–450