Higher-order action calculi

Robin Milner

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

Abstract Action calculi are a broad class of algebraic structures, including a formulation of Petri nets as well as a formulation of the \( \pi \)-calculus. Each action calculus \( \text{HAC}(\mathcal{K}) \) is generated by a particular set \( \mathcal{K} \) of operators called controls. The purpose of this paper is to extend action calculi in a uniform manner to higher-order. A special case is essentially the extension of the \( \pi \)-calculus to higher order by Sangiorgi. To establish a link between the interactive and functional paradigms of computation, a variety of the \( \lambda \)-calculus is obtained as the extension of the smallest action calculus \( \text{HAC}(\emptyset) \).

The dynamics of higher-order action calculi is presented, blending communication—e.g., in process calculi—with reduction as in the \( \lambda \)-calculus. Strong normalisation is obtained for reduction. A set of equational axioms is given for higher-order action calculi. Taking the quotient of \( \text{HAC}(\emptyset) \) by a single extra axiom \( \eta \), a cartesian-closed category is obtained.

An ultimate goal of the paper is to combine process calculi and functional calculi, both in their formulation and in their semantics.

1 Introduction

Action structures [11] are a class of monoidal categories, enriched with abstraction and dynamics, intended as a semantic framework for interactive systems. Action calculi are a special subclass; each action calculus \( \text{AC}(\mathcal{K}) \) is distinguished by its set \( \mathcal{K} \) of controls and their reaction rules. The notion of action calculus [12] is derived partly from the chemical abstract machine of Berry and Boudol [5]. To each control \( K \in \mathcal{K} \) corresponds a type of molecule; the main interest of action calculi is that they are an algebra of molecular forms, which provide a tractable form of syntax for modelling concurrent systems. This paper extends an arbitrary action calculus to higher order, in a way which parallels the extension of the \( \pi \)-calculus [14] to higher order by Sangiorgi [19].

Particular action calculi exist which are generalisations of Petri nets, and of the \( \pi \)-calculus [12, 13]. Thus the extension to higher order adds power to existing models of concurrency. This paper demonstrates that the algebraic treatment of action calculi, including the notion of molecular form, extends smoothly to higher-order calculi which generalise the \( \lambda \)-calculus.

Sections 1–6 contain a review of action calculi, sufficient for present purposes. The first step towards higher order is taken in Section 7 by the addition of two new controls to any action calculus \( \text{AC}(\mathcal{K}) \). These new controls are essentially the "curry" and "apply" operations of a cartesian-closed category for the typed \( \lambda \)-calculus. The reaction rules

\[ 1 \text{This work was done with the support of a Senior Fellowship from the Science and Engineering Research Council, UK.} \]
for these new controls, \( \sigma \)-reduction and \( \beta \)-reduction, are closely related to reduction in the \( \lambda \sigma \) -calculus [1], a \( \lambda \) -calculus with explicit substitutions; so far they are treated on the same footing as the reaction rules pertaining to the originally controls \( \mathcal{K} \).

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

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

Section 10 examines the possibility of making \( \text{HAC}(\mathcal{K}) \) into a cartesian-closed category. The case \( \mathcal{K} = \emptyset \) is especially interesting, since it corresponds essentially to the typed \( \lambda \) -calculus. Indeed, \( \text{HAC}(\emptyset) \) is nearly a cartesian-closed category, but not quite; a quotient is needed by an extra equation representing \( \eta \)-conversion. Indeed, we show that this quotient converts \textit{any} higher-order action calculus into a cartesian-closed category; but in the case of some action calculi with non-trivial interactive dynamics, such as PIC (representing the \( \pi \) -calculus), it is inconsistent with the dynamic behaviour.

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

2 Arities and names

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

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

3 Actions and controls

The principal ingredient of an action calculus is its \textit{actions}. Assuming an arity monoid \( M \) as above, each action \( a \) of the calculus has a \textit{source} arity and a \textit{target} arity in \( M \); if