To appear in Express'99. Available online from
http://www.cl.cam.ac.uk/~ljw1004

Symmetric action calculi

Lucian J. Wischik and Philippa Gardner, May 1999 *

There exist many different calculi for rescribing interactive and sequential behaviour. The goal of *action calculi* [1] is to unify these calculi at a syntactic and operation level. We introduce a symmetric variant which extends the reach of action calculi to cover for example the Fusion calculus [2] and Yoshida's process graphs [3]. These symmetric action calculi conservatively extend the reflexive action calculi [4] and have close links to category theory.

Background

Action calculi and their extensions [4] have constructs for *names* and *name-abstraction* in common. The intention is that these naming constructs should represent the naming constructs in other calculi. In addition, each action calculus has a number of *controls* to express the features specific to a particular calculus. For instance, the controls **out** and **in** are used to express the output and input actions of the π-calculus [5]. Finally, the controls may *react* together. The reaction between **out** and **in** corresponds to reaction in the π-calculus.

Action calculi may be represented graphically (see Figure 1). They also have an inductive algebraic definition as a strict monoidal category with the addition of the naming constructs and controls.

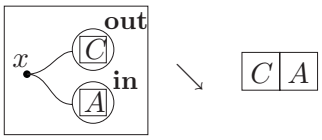
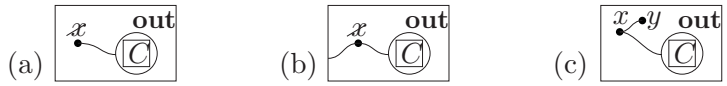


Figure 1: This action-calculus reaction corresponds to the π-calculus reaction $\overline{x}.C|x.A \searrow A@C$, where A and C are abstractions and concretions respectively. With A and C not so constrained, the same reaction rule also describes reaction in the π_I-calculus [6] and the Fusion calculus.

*Computing Laboratory, University of Cambridge, ljw1004@cam.ac.uk, Philippa.Gardner@cl.cam.ac.uk. Wischik is supported by an EPSRC studentship. Gardner is supported by an EPSRC Advance Fellowship.

Symmetric action calculi

We introduce a novel form of action calculi, known as the *symmetric action calculi*, which differ from conventional action calculi in their naming constructs. In particular, the symmetric calculi have operators for *names*, *co-names* and *restriction* (see Figure 2). Conventional action calculi meanwhile combine the last two into a single abstraction operator. In this respect, the move from conventional action calculi to symmetric mirrors that from the π-calculus to the Fusion calculus [2].



$(\nu x)(\overline{x}.C)$ $\quad$ $(x)(\overline{x}.C)$ $\quad$ $x{=}y \mid \overline{x}.C$

Figure 2: (a) Restriction is a primitive operator in symmetric action calculi. Graphically we indicate a restricted (local) name with a line through it. (b) Abstraction is composed from two parts: a *co-name*, illustrated by the wire going into the name, and then restriction. (c) Fusion. (In non-symmetric action calculi, restriction is expressed more indirectly with a 'new' control; and fusions cannot be directly expressed.)

In the symmetric action calculi and in the Fusion calculus one can express the *fusion* of two names and, more generally, equivalence relations over names. In the Fusion calculus, these fusions have effect only as part of the labelled transition system:

$$(x)(y{=}x.P) \;\rightarrow\; P\{y/\!\!/x\}.$$

In the symmetric action calculi the fusions are terms in the calculus, giving such equations as

$$Q \mid y{=}x \quad = \quad Q\{y/\!\!/x\} \mid y{=}x \quad = \quad Q\{x/\!\!/y\} \mid y{=}x.$$

This allows for small-step equational reasoning (see for example Figure 3). As illustrated below with Yoshida's process graphs and the reflexive action calculi, fusions can give simpler rules for equality.

$$(\nu x)(x) \quad \overset{(a)}{=} \quad (\nu xy)(x \mid y{=}x) \quad \overset{(b)}{=} \quad (\nu xy)(y \mid y{=}x) \quad \overset{(c)}{=} \quad (\nu y)(y)$$

Figure 3: Fusions in terms allow for small-step equational reasoning. Here we illustrate the three small steps that make up alpha-conversion: (a) A new local name y is introduced as an alias for x. (b) Because $y = x$, we are able to substitute y for any occurence of x. (c) Finally, in a reverse of the first step, we remove the now-unused local name x.

Figure 4: The *connection* operator in Yoshida's process graphs partitions the named nodes and immediately renames them. In the graph illustrated, x and y have been related and renamed z. Instead of trying to do all this in a single connection operator, it is more convenient merely to fuse the names. Renaming can then be left to the standard naming equations shown in Figure 3.

Process Graphs

Yoshida's *process graphs* [3] extend Lafont's interaction nets [7] to allow for non-determinism. The operations and equational theory on these graphs are intuitive but difficult to state formally. We express the graphs as symmetric action calculi. As illustrated in Figure 4, fusions allow for a simpler statement of the equational theory.

Reflexive action calculi

The reflexion operator in Milner's reflexive action calculi has an awkward definition, illustrated in Figure 5. We note that the reflexion operator is essentially a special case of fusion and can be defined more conveniently as such: (1) Reflecting an output y to an input x results in a fusion $x = y$; (2) Fusions are then handled using the straightforward small-step equalities illustrated in Figure 3.

We have used this symmetric form of reflexion to prove that the symmetric action calculi conservatively extend the reflexive action calculi. The key extension is the presence of fusions within terms. We are currently considering other calculi with such fusions.

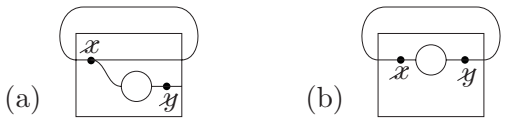


Figure 5: These reflected graphs have very different normal forms in the reflexive action calculi: the first uses an additional **new** control to describe the local name x, and the second is expressed with a *syntactic* (non capture-avoiding) substitution $\{y/x\}^*$. In the symmetric action calculi both reflections can be expressed uniformally—the first with the identity fusion $x{=}x$, and the second with the fusion $y{=}x$.

Other work

It seems likely that fusions can be mimicked in action calculi with an additional control and its reactions (similar to Honda and Yoshida's *equator* [8]). However, this would not be in keeping with the original aim—to use the action calculi naming constructs to represent those in other calculi.

Mirroring earlier work of Gardner [9] and Pavlovic [10], we have developed a name-free algebra for symmetric action calculi. The categorical description of a simpler symmetric algebra, without controls, has already been studied [11]. Our additional axioms for controls indicate their *dinaturality* with respect to the naming structure. This suggests a firm categorical underpinning for symmetric calculi and is a subject of further research.

A general bisimulation result for conventional action calculi remains elusive. We hope however that, with names and co-names as barbs, a barbed bisimulation result for the symmetric action calculi might prove easier.

References

[1] R. Milner. Calculi for interaction. *Acta Informatica*, 33(8), 1996.

[2] Parrow and Victor. The fusion calculus: Expressiveness and symmetry in mobile processes. In *LICS: IEEE Symposium*, 1998.

[3] N. Yoshida. Graph notation for concurrent combinators. *Lecture Notes in Computer Science (LNCS)*, 907:393–412, May 1995.

[4] R. Milner. Action calculi V: Reflexive Action Calculi. MS., 1994.

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

[6] D. Sangiorgi. Pi-calculus, internal mobility and agent-passing calculi. *Theoretical Computer Science*, 167(2), 1996.

[7] Yves Lafont. Interaction combinators. *Information and Computation*, 137(1):69–101, 1997.

[8] K. Honda and N. Yoshida. On reduction-based process semantics. *LNCS*, 761, 1994.

[9] Philippa Gardner. A name-free account of action calculi. In *Proceedings MFPS '95. Electronic Notes in Theoretical Computer Science 1*, 1995.

[10] Duško Pavlović. Categorical logic of names and abstraction in action calculi. *Mathematical Structures in Computer Science*, 7(6), 1997.

[11] F. Gadducci and R. Heckel. An inductive view of graph transformation. *LNCS*, 1376:223–237, 1998.