From Process Calculi to Process Frameworks

Philippa Gardner[1]

Abstract. We present two process frameworks: the *action calculi* of Milner, and the *fusion systems* of Gardner and Wischik. The action calculus framework is based on process constructs arising from the π-calculus. We give a non-standard presentation of the π-calculus, to emphasise the similarities between the calculus and the framework. The fusion system framework generalises a new process calculus called the π_F*-calculus*. We describe the π_F-calculus, which is based on different process constructs to those of the π-calculus, and show that the generalisation from the calculus to the framework is simple. We compare the frameworks by studying examples.

Introduction

Our aim in studying process frameworks is to integrate ways of reasoning about related interactive behaviour. The purpose is not to provide a general, all-encompassing system for describing interaction—that would miss the point that there are many kinds of interactive behaviour worth exploring. Rather, the aim is to study interaction based on some fixed underlying process structure. A process framework consists of a structural congruence, which establishes the common process constructs used in the framework, and a reaction relation which describes the interactive behaviour of particular processes specified by a signature. The higher-order term rewriting systems have a similar format; the structural congruence describes the higher-order features given by the λ-terms with β-equality, and a rewriting relation describes the reaction between terms generated from a signature.

In this paper, we describe two process frameworks: the *action calculi* of Milner [Mil96], and the *fusion systems* of Gardner and Wischik [GW99]. The action calculus framework is based on process constructs arising from the π-calculus. We give a non-standard presentation of the π-calculus, to emphasise the similarities between the calculus and the framework. We also present the fusion system framework which generalises a new process calculus, called the π_F*-calculus* [GW00], in much the same way as the action calculus framework generalises the π-calculus. The π_F-calculus is similar to the π-calculus in that its interactive behaviour is based on input and output processes, and different in that its underlying process structure is not the same. We describe the π_F-calculus and the fusion system framework, and illustrate that the generalisation from the calculus to the framework is simple. We also compare the two frameworks by studying specific examples.

[1] Address: Computer Laboratory, University of Cambridge, New Museums Site, Pembroke Street, Cambridge, CB2 3QG. Email: pag20@cl.cam.ac.uk. The author acknowledges support of an EPSRC Advanced Fellowship.

From the π-calculus to action calculi

We describe the transition from the π-calculus to the action calculus framework. By adapting Milner's presentation of the π-calculus in [Mil99], we show that this transition is comparatively simple. Instead of the input and output processes specific to the π-calculus, an action calculus consists of more flexible *control processes*. The interactive behaviour of such control processes is given by a reaction relation specific to the particular action calculus under consideration. The other constructs of an action calculus are essentially the same as those of our π-calculus, except that there is no restriction. Instead the π-process $(\nu x)P$ is interpreted by an action process $\mathsf{new} \cdot (x)P$, where the control process new blocks the access to the name-abstraction. We explore two examples of action calculi: the π action calculus corresponding to the π-calculus, and the λ_v action calculus corresponding to a call-by-value λ-calculus. These examples illustrate that the basic process constructs for action calculi are indeed natural. We also refer to recent results of Leifer and Milner [LM00], who provide general bisimulation congruences for (simple) reactive systems and aim to extend their results to action calculi.

The π_F-calculus

The π_F-calculus is similar to the π-calculus in that it consists of input and output processes that interact with each other. It is different from the π-calculus in the way that these processes interact. In a π-reaction, the intuition is that names are sent along a channel name to replace the abstracted names at the other end. In contrast, the π_F-calculus does not have abstraction. Instead, a π_F-reaction is directionless with names being *fused* together during reaction. This fusion mechanism is described using a *explicit fusion* $\langle z = y \rangle$, which is a process declaring that two names can be used interchangeably. A natural interpretation is that names refer to addresses, with explicit fusions declaring that two names refer to the same address.

The π_F-calculus is similar in many respects to the fusion calculus of Parrow and Victor [PV98], and the χ-calculus of Fu [Fu97]. The key difference is how the name-fusions have effect. In the π_F-calculus, fusions are explicitly recorded and have effect though the structural congruence. This has the consequence that the π_F-reaction is a simple local reaction between input and output processes. In the fusion calculus on the other hand, fusions occur implicitly within the reaction relation. This outcome of this is a non-local reaction relation.

We provide a natural bisimulation congruence for the π_F-calculus. We also have simple embeddings for the π-calculus, the fusion calculus and a call-by-value λ-calculus in the π_F-calculus. Further details can be found in [GW00]. In particular, we show that the fusion-embedding is fully abstract, in the sense that hyper-equivalence for the fusion calculus corresponds to the bisimulation arising from its embedding in the π_F-calculus.

Fusion systems

The generalisation from the π_F-calculus to the fusion system framework is simple. The basic process constructs and the definition of the structural congruence are the same. As with action calculi, fusion systems have the flexibility to describe other forms of interactive behaviour by adapting the input and output processes of the π_F-calculus to more general control processes. The interactive behaviour of the control processes is again given by a reaction relation, which is specific to the particular fusion system under consideration. We explore the π_F fusion system, which corresponds exactly to the π_F-calculus, and the λ_v fusion system which describes a call-by-value λ-calculus. The λ_v fusion system is a subsystem of the π_F fusion system. This simple relationship depends on the explicit fusions. It means that the bisimulation congruences for the π_F-system easily transfer to the λ_v-system. We are currently exploring a link between a bisimulation congruence for the λ_v-system and a corresponding behavioural congruence for the call-by-value λ-calculus. It remains further work to provide general bisimulation results for fusion systems. However, this is not our primary concern. First, we would like to explore specific examples of fusion systems to illustrate the expressiveness of explicit fusions.

Acknowledgements The work on the π_F-calculus and fusion systems is joint with Lucian Wischik.

1 From the π-calculus to action calculi

In [Mil99], Milner presents the π-calculus using a structural congruence which describes the underlying process structure, and a reaction relation which describes the interaction between input and output processes. In particular, he builds processes using abstractions and concretions as interim constructs, and generates the reaction relation by the axiom

$$\overline{x}.P \,|\, x.Q \searrow P \cdot Q,$$

where the operator $\cdot$ is a derived operator expressing the application of a concretion to an abstraction. In contrast, we treat all the constructs in the same way. We consider the application operator as a primitive operator. Although it can be derived in the π-calculus, it cannot be derived in the action calculus framework. We regard abstractions as processes, but instead of the concretion $(\nu z)\langle y\rangle P$ for example, we have the process $(\nu z)(\langle y\rangle \,|\, P)$ where the process $\langle y\rangle$ is called a *datum*. The reason for focusing on datums rather than concretions is that variables of the λ-calculus translate to datums in the corresponding action calculus. The datums therefore seem more fundamental.

In order to define reaction precisely, we require the correct number of datums and abstractions to match. This information is given by the *arity* of the process. In general, the arity information can include quite complex typing information, such as the sorting discipline for the π-calculus given in [Mil99]. In this paper

we keep the arity structure simple, and just use it to count abstractions and datums.

There are two natural approaches for defining simple arities for π-processes. One approach keeps the abstractions and datums separate, so that we can form $(x)P$ and $\langle x \rangle \,|\, P$, but not $(x)(\langle x \rangle \,|\, P)$. In this case, arities are integers with the positive numbers counting the abstractions and the negative numbers the datums. The resulting process algebra corresponds to the π-calculus presented in Milner's book [Mil99]. The second approach mixes abstractions and datums, so that we have for example the process $(x)(\langle x \rangle \,|\, P)$. Arities have the form (m, n) for $m, n, \geq 0$, where m counts the abstractions and n the datums. We choose this second approach here, since it gives a smooth link to the action calculus framework. It remains further work however to fully justify the use of abstracted datums for the π-calculus. They are justified in the action calculus framework.

Following our presentation of the π-calculus, the description of the action calculus framework is comparatively simple. An action calculus essentially consists of the same underlying process constructs as our π-calculus, except that it does not have restriction. The definition of the structural congruence has to be modified however to account for the more general behaviour of application in the framework. Instead of input and output processes, an action calculus consists of general *control processes* of the form $K(P_1, \ldots, P_r)$ where the *control* K denotes some sort of boundary (or firewall) containing the processes P_i. The interactive behaviour of the control processes is described by a reaction relation.

For example, the input and output processes of the π-calculus correspond to action processes of the form $\langle x \rangle \cdot \mathsf{in}(P)$ and $\langle x \rangle \cdot \mathsf{out}(P)$ respectively. The controls in and out are specific to the π action calculus, and provide a boundary inside which reaction does not occur. Restriction is not primitive in the framework. Instead the restriction $(\nu x)P$ of the π-calculus is interpreted by $\mathsf{new} \cdot (x)P$ where new is a control which blocks the access to the abstraction.

1.1 The π-calculus

Throughout this paper, we assume that we have an infinite set N of names ranged over by $u, \ldots, z$, and use the notation $\vec{z}$ to denote a sequence of names and $|\vec{z}|$ to denote the length of the sequence.

Definition 1
The set $\mathcal{P}'_\pi$ of *pre-processes* of the π-calculus is defined by the grammar

$P ::=$	nil	Nil process
	$P \,|\, P$	Parallel composition
	$P \cdot P$	Application
	$\langle x \rangle$	Datum
	$(x)P$	Abstraction
	$(\nu x)P$	Restriction
	$x.P$	Input Process
	$\overline{x}.P$	Output process

The definitions of *free* and *bound* names are standard: the abstraction $(x)P$ and the restriction $(\nu x)P$ bind x in P, and x is free in the processes $\langle x \rangle$, $x.P$ and $\overline{x}.P$. We write $\mathsf{fn}(P)$ to denote the set of free names in P.

The application operator is used to apply the contents of input and output processes during reaction:

$$\overline{x}.P \mid x.Q \searrow P \cdot Q.$$

To define reaction properly, we require the correct number of datums in P and abstractions in Q to match. This information is given by the *arity* of a pre-process. We write $P : (m, n)$ to declare that a pre-process has arity (m, n), where m and n are natural numbers counting the abstractions and datums respectively.

Definition 2
The set $\mathcal{P}_\pi$ of π-*processes* of arity (m, n) is defined inductively by the rules

$$\mathsf{nil} : (0, 0) \qquad \langle x \rangle : (0, 1)$$

$$\frac{P : (m, n) \qquad Q : (k, l)}{P \mid Q : (m + k, n + l)} \qquad \frac{P : (m, k) \qquad Q : (k, n)}{P \cdot Q : (m, n)}$$

$$\frac{P : (m, n)}{(x)P : (m + 1, n)} \qquad \frac{P : (m, n)}{(\nu x)P : (m, n)}$$

$$\frac{P : (m, 0)}{x.P : (0, 0)} \qquad \frac{P : (0, m)}{\overline{x}.P : (0, 0)}$$

Definition 3
The *structural congruence* between π-processes of the same arity, written $\equiv$, is the smallest congruence satisfying the axioms given in figure 1, and which is closed with respect to $\_ \mid \_$, $\_ \cdot \_$, $(x)\_$, $(\nu x)\_$, $x.\_$ and $\overline{x}.\_$.

Notice the side-condition associated with the axiom for the commutativity of parallel composition. It results in the order of abstractions and datums always being preserved, as one would expect. The other rules are self-explanatory. The application axioms are enough to show that application can be derived from the other operators.

One property of the structural congruence is that every π-process is congruent to a *standard form*. Standard forms are processes with the shape

$$(\vec{x})(\nu \vec{z})(\langle \vec{y} \rangle \mid P),$$

where the $\vec{x}$s and $\vec{z}$s are distinct, the $\vec{z}$s are contained in the $\vec{y}$s, and $P : (0, 0)$ does not contain any applications. Standard forms are essentially unique in the sense that, given two congruent standard forms

$$(\vec{x}_1)(\nu \vec{z}_1)(\langle \vec{y}_1 \rangle \mid P_1) \equiv (\vec{x}_2)(\nu \vec{z}_2)(\langle \vec{y}_2 \rangle \mid P_2),$$

Standard axioms for nil, $\_ \mid \_$ and $(\nu x)\_$:

$$
\begin{aligned}
P \mid \mathsf{nil} &\equiv P \\
P \mid \mathsf{nil} &\equiv P \\
(P \mid Q) \mid R &\equiv P \mid (Q \mid R) \\
P \mid Q &\equiv Q \mid P, \quad P : (m, 0), Q : (0, n)
\end{aligned}
$$

$$
\begin{aligned}
(\nu x)(\nu y)P &\equiv (\nu y)(\nu x)P \\
(\nu x)(P \mid Q) &\equiv (\nu x)P \mid Q, \quad x \notin \mathsf{fn}(Q) \\
(\nu x)(P \mid Q) &\equiv P \mid (\nu x)Q, \quad x \notin \mathsf{fn}(P) \\
(\nu x)P &\equiv (\nu y)P\{y/x\}, \quad y \notin \mathsf{fn}(P) \\
(\nu x)\mathsf{nil} &\equiv \mathsf{nil}
\end{aligned}
$$

Abstraction axioms:

$$
\begin{aligned}
(x)P &\equiv (y)P\{y/x\}, \quad y \notin \mathsf{fn}(P) \\
(\nu x)(y)P &\equiv (y)(\nu x)P, \quad y \not\equiv x \\
(x)(P \mid Q) &\equiv (x)P \mid Q, \quad x \notin \mathsf{fn}(Q) \\
(x)(P \mid Q) &\equiv P \mid (x)Q, \quad x \notin \mathsf{fn}(P), P : (0, m)
\end{aligned}
$$

Application axioms:

$$
\begin{aligned}
(\nu x)(P \cdot Q) &\equiv (\nu x)P \cdot Q, \quad x \notin \mathsf{fn}(Q) \\
(\nu x)(P \cdot Q) &\equiv P \cdot (\nu x)Q, \quad x \notin \mathsf{fn}(P) \\
(x)P \cdot Q &\equiv (x)(P \cdot Q), \quad x \notin \mathsf{fn}(Q) \\
(\langle y \rangle \mid Q) \cdot (x)P &\equiv Q \cdot P\{y/x\} \\
Q \cdot P &\equiv Q \mid P, \quad Q : (m, 0), P : (0, n)
\end{aligned}
$$

Fig. 1. The structural congruence between π-processes of the same arity, written $\equiv$, is the smallest equivalence relation satisfying these axioms and closed with respect to contexts.

then $|\vec{x}_1| = |\vec{x}_2|$ and $|\vec{z}_1| = |\vec{z}_2|$, the $\vec{y}_1$ and $\vec{y}_2$ are the same up to α-conversion of the $\vec{x}$s and $\vec{z}$s, and P_1 and P_2 are structurally congruent again up to α-conversion. Using the standard forms, application can be derived from the other operators.

The *reaction relation* between π-processes of the same arity, written $\searrow$, is the smallest relation generated by

$$\overline{x}.P \mid x.Q \searrow P \cdot Q,$$

where P and Q have arities $(0, m)$ and $(m, 0)$ respectively, and the relation is closed with respect to $\_ \mid \_$, $\_ \cdot \_$, $(x)\_$, $(\nu x)\_$ and $\_ \equiv \_$.

In [Mil99], Milner defines a labelled transition system and the corresponding strong bisimulation for the π-calculus, using standard transitions with labels x, $\overline{x}$ and τ. It is straightforward to adapt his definitions to our presentation.

1.2 Action Calculi

An action calculus is generated from the basic process operators of the π-calculus, except restriction, plus control operators specific to the particular action calculus under consideration. However, the definition of the structural congruence cannot be simply lifted from the corresponding definition for the π-calculus. In the π-calculus, application is only used to apply datums to abstractions. In the framework, application is also used to apply control processes to other processes. This additional use of application requires a more general description of its properties. For example, the associativity of application can be derived in the π-calculus, but must be declared explicitly in the framework.

The simplest way to define the structural congruence for the framework is to adapt the basic constructs of the π-calculus to include the *identity* process id_m and the *permutation* process $\mathsf{p}_{m,n}$, where m and n are arities. These constructs can be derived from the other operators: for example, we have $\mathsf{id}_1 \equiv (x)\langle x \rangle$ and $\mathsf{p}_{1,1} \equiv (x, y)\langle y, x \rangle$ where $y \not\equiv x$. We choose to regard them as primitive, since they play a fundamental role in the congruence definition.

An action calculus is specified by a set $\mathcal{K}$ of controls, plus a reaction relation which describes the interaction between control processes. Each control K in $\mathcal{K}$ has an associated arity $((m_1, n_1), \ldots, (m_r, n_r)) \to (k, l)$, which informs us that a control process $K(P_1, \ldots, P_r)$ has arity (k, l) such that P_i has arity (m_i, n_i).

Definition 4
The set $\mathcal{P}'_{\mathsf{AC}}(\mathcal{K})$ of *pre-processes* of an action calculus specified by control set $\mathcal{K}$ is defined by the grammar

P	::=	id_m	Identity
		$\mathsf{p}_{m,n}$	Permutation
		$P \mid P$	Parallel composition
		$P \cdot P$	Application
		$\langle x \rangle$	Datum
		$(x)P$	Abstraction
		$K(P_1, \ldots, P_r)$	Control process

The set $\mathcal{P}_{\mathsf{AC}}(\mathcal{K})$ of *action processes* of arity (m, n), specified by control set $\mathcal{K}$, is defined by the identity and permutation axioms

$$\mathsf{id}_m : (m, m) \qquad \mathsf{p}_{m,n} : (m + n, n + m),$$

the appropriate rules in definition 2, and the control rule

$$\frac{P_i : (m_i, n_i) \quad i \in \{1, \ldots, r\}}{K(P_1, \ldots, P_r) : (k, l)}$$

where control K has arity $((m_1, n_1), \ldots, (m_r, n_r)) \to (k, l)$.

The axioms generating the structural congruence for the framework are a modification of the axioms for the π-calculus given in figure 1 of section 1.1, minus those referring to restriction. The best way to describe the axioms is to focus on the categorical structure of processes. An action process $P : (m, n)$ can be viewed as a morphism in a category with domain m and codomain n. The id_m operator corresponds to the identity of the category, application to sequential composition, parallel to tensor, and $\mathsf{p}_{m,n}$ to permutation. The structural congruence on action processes is defined in figure 2. It is generated by the axioms of a strict symmetric monoidal category and two additional naming axioms.

The axioms of a strict symmetric monoidal category just state that the identity, parallel composition, application and permutation operators behave as we would expect. A possibility helpful intuition is that the processes id_0 and $\mathsf{p}_{0,0}$ correspond to the nil process and

$$\begin{aligned} \mathsf{id}_m &\equiv (\vec{x})\langle\vec{x}\rangle, \quad \vec{x} \text{ distinct }, |\vec{x}| = m > 0 \\ \mathsf{p}_{m,n} &\equiv (\vec{x}, \vec{y})\langle\vec{y}, \vec{x}\rangle, \quad \vec{x}, \vec{y} \text{ distinct }, |\vec{x}| = m, |\vec{y}| = n, \quad m + n > 0. \end{aligned} \tag{1}$$

For example, the last axiom generalises the commutativity of parallel composition for the π-calculus. The first naming axiom is just a special case of the application axiom for the π-calculus given in figure 1, which describes the application of datums to abstractions. The second naming axiom is a generalisation of the structural congruence given in (1) for the identity process. The appropriate π-axioms in figure 1 of section 1.1 are derivable from the axioms given here. With the above interpretation of the identity and permutation processes, the axioms of the framework are also derivable in the π-calculus.

The *reaction relation* $\searrow$ is a binary relation between action processes of the same arity, which is closed with respect to $\_ \mid \_$, $\_ \cdot \_$, $(x)\_$ and $\_ \equiv \_$. We specify whether reaction may occur inside controls. Both examples studied in this section prevent it.

We have a very different type of standard form result for action calculi, since application plays such a prominent role. The standard forms are not special processes, but rather have a completely different notation to processes. We therefore call them *molecular forms*, to emphasise this difference and conform with the literature [Mil96]. The molecular forms are generated by the grammars

$$\begin{aligned} \text{Molecular forms} \quad & P ::= (\vec{x})[M, \ldots, M]\langle\vec{y}\rangle, \qquad \vec{x} \text{ distinct} \\ \text{Molecules} \quad & M ::= \langle\vec{u}\rangle K(P_1, \ldots, P_r)(\vec{v}), \quad \vec{v} \text{ distinct} \end{aligned}$$

Axioms of a strict symmetric monoidal category:

$$\begin{array}{rcl@{\qquad}rcl} P \cdot \mathsf{id}_n & \equiv & P \equiv \mathsf{id}_m \cdot P & P \mid \mathsf{id}_0 & \equiv & P \equiv \mathsf{id}_0 \mid P \\ P \cdot (Q \cdot R) & \equiv & (P \cdot Q) \cdot R & (P \mid Q) \mid R & \equiv & P \mid (Q \mid R) \\ (P_1 \cdot P_2) \mid (Q_1 \cdot Q_2) & \equiv & (P_1 \mid Q_1) \cdot (P_2 \mid Q_2) & \mathsf{id}_m \mid \mathsf{id}_n & \equiv & \mathsf{id}_{m+n} \end{array}$$

$$\begin{array}{rcl} \mathsf{p}_{m,n} \cdot \mathsf{p}_{n,m} & \equiv & \mathsf{id}_{m+n} \\ \mathsf{p}_{m+n,k} & \equiv & (\mathsf{id}_m \mid \mathsf{p}_{n,k}) \cdot (\mathsf{p}_{m,k} \mid \mathsf{id}_n) \\ \mathsf{p}_{m,n} \cdot (P \mid Q) & \equiv & (Q \mid P) \cdot \mathsf{p}_{k,l}\ , \quad P : (m,l), Q : (n,k) \end{array}$$

Naming axioms:

$$\begin{array}{rcl} (\langle y \rangle \mid \mathsf{id}_m) \cdot (x)P & \equiv & P\{y/x\} \\ (x)((\langle x \rangle \mid \mathsf{id}_m) \cdot P) & \equiv & P, \quad x \notin \mathsf{fn}(P) \end{array}$$

Fig. 2. The structural congruence between action processes of the same arity, written $\equiv$, is the smallest equivalence relation satisfying these axioms and closed with respect to contexts.

with r, $|\vec{u}|$ and $|\vec{v}|$ determined by the arity of control K. The $\vec{x}$ and $\vec{y}$ correspond to the abstractions and datums in the standard forms for the π-calculus given in section 1.1, where $(|\vec{x}|, |\vec{y}|)$ is the arity of the molecular form. The $\vec{u}$ are free, and the $\vec{v}$ are distinct and bind to the right. These $\vec{v}$ provide the mechanism for recording the sequential dependency of control processes. The molecular forms have a simple structural congruence, given by α-conversion of bound names and the partial reordering of the molecules when there is no name clash.

The π action calculus The π action calculus is specified by the controls

$$\mathsf{out} : ((0,m)) \to (1,0) \qquad \mathsf{in} : ((m,0)) \to (1,0) \qquad \mathsf{new} : (\,) \to (0,1)$$

The input and output processes correspond to the action processes of the form $\langle x \rangle \cdot \mathsf{in}(P)$ and $\langle x \rangle \cdot \mathsf{out}(P)$ respectively. Restriction is interpreted by action processes of the form $\mathsf{new} \cdot (x)P$, where the new control blocks the access to the abstraction. The reaction relation is generated by the rule

$$\langle x \rangle \cdot \mathsf{out}(P) \mid \langle x \rangle \cdot \mathsf{in}(Q) \quad \searrow \quad P \cdot Q$$

where P and Q must have arities $(m,0)$ and $(0,n)$ respectively and we specify that reaction does not occur inside the controls.

The correspondence is not absolutely exact, since the π-axiom

$$(\nu x)\mathsf{nil} \equiv \mathsf{nil}$$

is not preserved by the translation. However, the translated processes are bisimilar. Apart from this point, it is not difficult to show that the structural congruence and the reaction relation are strongly preserved by the embedding [Mil96]. These results rely on the molecular forms described earlier.

The λ_v action calculus The λ_v action calculus interprets λ-terms as processes of arity $(0,1)$. It is specified by the controls

$$\mathsf{lam} : ((1,1)) \to (0,1) \qquad \mathsf{ap} : (\,) \to (2,1).$$

For example, the λ-term $\lambda x.x$ corresponds to the action process $\mathsf{lam}((x)\langle x\rangle)$, which illustrates the use of abstracted datums. Again we do not permit reaction inside a lam-control. The reaction relation is generated by the two rules

$$\begin{array}{rcl} (\mathsf{lam}(P) \,|\, \mathsf{id}_1) \cdot \mathsf{ap} & \searrow & P \\ (\mathsf{lam}(P) \,|\, \mathsf{id}_m).(x)Q & \searrow & Q\{\mathsf{lam}(P)/\langle x\rangle\}. \end{array}$$

The first axiom describes β-reduction, and the second describes the explicit substitution of a lam-process for abstracted datums. In fact, there is a higher-order extension of action calculi where the lam- and ap-controls are regarded as primitive constructs. The details can be found in [Mil94a,HG97].

There is another extension of the action calculus framework [Mil94b], which includes a reflexion (or feedback) operator

$$\frac{P : (m+1, n+1)}{\uparrow P : (m,n)}$$

The reflexion operator does not have a direct analogy in the π-calculus. It does have a direct analogy in the λ-calculus, in that the λ_v action calculus plus reflexion corresponds to the cyclic λ-calculus of Ariola and Klop [Has97]. The reflexion axioms correspond to adding a trace operator [JSV96] to the symmetric monoidal category. We do not concentrate on the reflexive extension here, since our aim is to illustrate the close connection between the π-calculus and the action calculus framework. It is however of fundamental importance. In particular, the recent bisimulation results of Leifer and Milner rely on this extension.

Leifer and Milner are currently exploring general bisimulation results for (simple) reactive systems [LM00], and aim to extend their results to a class of reflexive action calculi. Such results have also been studied by Sewell [Sew00]. The idea is to automatically generate a labelled transition system from an underlying reaction relation, and yield a corresponding bisimulation congruence. The labelled transitions have the form $P \xrightarrow{C} Q$, with the intuition that C is a small context necessary to complete a redex in P. Leifer and Milner have characterised what it means for these small contexts to exist. Their challenge is to show that they do indeed exist for specific reactive systems such as action calculi.

2 The π_F-calculus

In this section, we describe the π_F-calculus and define a natural bisimulation congruence. The π_F-calculus is similar in many respects to the fusion calculus of Parrow and Victor [PV98], and the χ-calculus of Fu [Fu97], although we did not invent the π_F-calculus with these connections in mind. There is no

abstraction. Instead, the π_F-calculus has symmetric input and output processes, with a reaction relation which fuses the names together using *explicit fusions*. An explicit fusion $\langle z = y \rangle$ is a process which declares that two names can be used interchangeably.

The key difference between the π_F-calculus and the fusion calculus is how the name-fusions have effect. In the π_F-calculus, fusions are explicitly recorded and have effect though the structural congruence. For example, a typical π_F-reaction is

$$x.\langle z \rangle P \mid \overline{x}.\langle y \rangle Q \mid R \searrow \langle z = y \rangle \mid P \mid Q \mid R.$$

The explicit fusion $\langle z = y \rangle$ declares that z and y can be used interchangeably throughout the process. The reaction on the channel x is a local one between the input and output processes. However, its effect is global in that the scope of $\langle z = y \rangle$ affects process R. The scope of an explicit fusion is limited by the restriction operator. In the fusion calculus on the other hand, fusions occur implicitly within the reaction relation. For example, the fusion reaction

$$(\nu y)(x.\langle z \rangle P \mid \overline{x}.\langle y \rangle Q \mid R) \searrow P\{z/y\} \mid Q\{z/y\} \mid R\{z/y\}$$

corresponds to the restriction (with respect to y) of the π_F-reaction given above. This fusion reaction is not a simple local reaction between input and output processes. Instead, the redex is parametrized by R and requires y (or z) to be restricted. The full definition, using many $\vec{y}$s and $\vec{z}$s, is in fact quite complicated.

Definition 5
The set $\mathcal{P}'_{\pi_F}$ of pre-processes of the π_F-calculus is defined by the grammar

$P ::=$	nil	Nil process
	$P \mid P$	Parallel composition
	$P@P$	Connection
	$\langle x \rangle$	Datum
	$\langle x = y \rangle$	Fusion
	$(\nu x)P$	Restriction
	$x.P$	Input Process
	$\overline{x}.P$	Output process

The definitions of *free* and *bound* names are standard. The *restriction* operator $(\nu x)P$ binds x; otherwise x is free. The connection operator is used to connect the contents of input and output processes during reaction:

$$x.P \mid \overline{x}.Q \searrow P@Q.$$

We regard the connection operator as primitive. However, we shall see that it can be derived from the other axioms. It can also be derived in the fusion system framework, so our choice to include it is really not essential.

To define reaction properly, we require the correct number of datums in P and Q to connect together. This information is given by the *arity* of a pre-process. We write $P : m$ to declare that a pre-process has arity m, where m is a natural number used to count datums.

Definition 6
The set $\mathcal{P}_{\pi_F}$ of *processes* of the π_F-calculus with arity m is defined inductively by the rules

$$\mathsf{nil} : 0 \qquad \langle x = y \rangle : 0 \qquad \langle x \rangle : 1$$

$$\frac{P : m \qquad Q : n}{P \mid Q : m + n} \qquad \frac{P : m \qquad Q : m}{P @ Q : 0} \qquad \frac{P : m}{(\nu x) P : m}$$

$$\frac{P : m}{x.P : 0} \qquad \frac{P : m}{\overline{x}.P : 0}$$

Definition 7
The *structural congruence* between processes, written $\equiv$, is the smallest congruence satisfying the axioms given in figure 3, and which is closed with respect to $\_ \mid \_$, $\_ @ \_$, $(\nu x)\_$, $x.\_$ and $\overline{x}.\_$.

The side-condition on the commutativity of parallel composition allows for processes of arity 0 to be reordered, but not arbitrary processes. The connection axioms are simple.

The fusion axioms are similar in spirit to the name-equalities introduced by Honda in his work on a simple process framework [Hon00]. Our intuition is that $\langle x = y \rangle$ is a symmetric relation which declares that two names can be used interchangeably. The fusion $\langle x = x \rangle$ is congruent to the nil process. So too is $(\nu x)\langle x = y \rangle$, since the local name is unused. Notice that the standard axiom $(\nu x)\mathsf{nil} \equiv \mathsf{nil}$ is derivable from these two fusion axioms. The other six fusion axioms describe small-step substitution, allowing us to deduce $\langle x = y \rangle \mid P \equiv \langle x = y \rangle \mid P\{x/y\}$ as well as α-conversion. For example,

	$(\nu x)(\overline{x}.\mathsf{nil})$	
$\equiv$	$(\nu x)(\nu y)(\langle x = y \rangle \mid \overline{x}.\mathsf{nil})$	create fresh local name y as an alias for x
$\equiv$	$(\nu x)(\nu y)(\langle x = y \rangle \mid \overline{y}.\mathsf{nil})$	substitute y for x
$\equiv$	$(\nu y)(\overline{y}.\mathsf{nil})$	remove the now-unused local name x

Just as for the π-calculus, a property of the structural congruence is that every π_F-process is structurally congruent to a *standard form*. Standard forms are processes with the shape

$$(\nu \vec{x})(\langle \vec{y} \rangle \mid P) \mid \langle \vec{u} = \vec{v} \rangle,$$

where the $\vec{x}$s are distinct and contained in the $\vec{y}$s, and $P : 0$ does not contain connections or fusions. The standard form is essentially unique in the sense that, given two congruent standard forms

$$(\nu \vec{x}_1)(\langle \vec{y}_1 \rangle \mid P_1) \mid \langle \vec{u}_1 = \vec{v}_1 \rangle \equiv (\nu \vec{x}_2)(\langle \vec{y}_2 \rangle \mid P_2) \mid \langle \vec{u}_2 = \vec{v}_2 \rangle,$$

then $|\vec{x}_1| = |\vec{x}_2|$, the fusions $\langle \vec{u}_1 = \vec{v}_1 \rangle$ and $\langle \vec{u}_2 = \vec{v}_2 \rangle$ generate the same equivalence relation on names, the $\vec{y}_1$ and $\vec{y}_2$ are the same up to α-conversion of the

Standard axioms for $\_|\_$, $(\nu x)\_$ and nil:

$$\begin{aligned} P \mid \mathsf{nil} &\equiv P \\ (P \mid Q) \mid R &\equiv P \mid (Q \mid R) \\ P \mid Q &\equiv Q \mid P, \quad P : 0 \end{aligned}$$

$$\begin{aligned} (\nu x)(\nu y)P &\equiv (\nu y)(\nu x)P \\ (\nu x)(P \mid Q) &\equiv (\nu x)P \mid Q, \quad x \notin \mathsf{fn}(Q) \\ (\nu x)(P \mid Q) &\equiv P \mid (\nu x)Q, \quad x \notin \mathsf{fn}(P) \end{aligned}$$

Connection axioms:

$$\begin{aligned} (\nu x)(P@Q) &\equiv (\nu x)P@Q, \quad x \notin \mathsf{fn}(Q) \\ (\nu x)(P@Q) &\equiv P@(\nu x)Q, \quad x \notin \mathsf{fn}(P) \\ (\langle x\rangle \mid P)@(\langle y\rangle \mid Q) &\equiv \langle x = y\rangle \mid (P@Q) \\ P@Q &\equiv P \mid Q, \quad P, Q : 0 \end{aligned}$$

Fusion axioms:

$$\begin{aligned} \langle x = x\rangle &\equiv \mathsf{nil} & \langle x = y\rangle \mid y.P &\equiv \langle x = y\rangle \mid x.P \\ \langle x = y\rangle &\equiv \langle y = x\rangle & \langle x = y\rangle \mid \overline{y}.P &\equiv \langle x = y\rangle \mid \overline{x}.P \\ (\nu x)\langle x = y\rangle &\equiv \mathsf{nil} & \langle x = y\rangle \mid \langle y = z\rangle &\equiv \langle x = y\rangle \mid \langle x = z\rangle \\ & & \langle x = y\rangle \mid \langle y\rangle &\equiv \langle x = y\rangle \mid \langle x\rangle \\ & & \langle x = y\rangle \mid z.P &\equiv \langle x = y\rangle \mid z.(\langle x = y\rangle \mid P) \\ & & \langle x = y\rangle \mid \overline{z}.P &\equiv \langle x = y\rangle \mid \overline{z}.(\langle x = y\rangle \mid P) \end{aligned}$$

Fig. 3. The structural congruence between π_F-processes of the same arity, written $\equiv$, is the smallest equivalence relation satisfying these axioms and closed with respect to contexts.

$\vec{x}_1$s and $\vec{x}_2$s, and up to the name-equivalence generated by fusions, and P_1 and P_2 are structurally congruent up to α-conversion and the name-equivalence. We write $E(P)$ for the smallest equivalence relation on names generated by P. This equivalence relation can be inductively defined on the structure of processes; a simple characterisation is given by $(x, y) \in E(P)$ if and only if $P \equiv P \mid \langle x = y \rangle$.

Using the standard forms, it is possible to express the connection operator as a derived operator.

Definition 8
The *reaction relation* between π_F-processes of the same arity , written $\searrow$, is the smallest relation generated by

$$x.P \mid \overline{x}.Q \searrow P@Q,$$

where P and Q have arity m and the reaction is closed with respect to $\_ \mid \_$, $\_@\_$, $(\nu x)\_$ and $\_ \equiv \_$.

2.1 Bisimulation Congruence

A natural choice of labelled transition system (LTS) for the π_F-calculus consists of the usual CCS-style transitions using labels $\overline{x}$, x and τ, accompanied by a definition of bisimulation which incorporates fusions: $PSQ : 0$ implies

$$\text{for all } x \text{ and } y, \text{ if } P \mid \langle x = y \rangle \xrightarrow{\alpha} P_1 \text{ then } Q \mid \langle x = y \rangle \xrightarrow{\alpha} Q_1 \text{ and } P_1 S Q_1.$$

We call this bisimulation the *open bisimulation*, by analogy with the open bisimulation of the π-calculus.

In the definition of open bisimulation, labelled transitions are analysed with respect to the fusion contexts $\_ \mid \langle x = y \rangle$. In fact, we do not need to consider all such contexts. Instead, we introduce the *fusion transitions*, generated by the axiom

$$x.P \mid \overline{y}.Q \xrightarrow{?x=y} P@Q.$$

A fusion transition with label $?x = y$ declares the presence of input and output processes with channel names x and y, although we do not know which name goes with which process. The resulting bisimulation definition is simpler, in that it is enough to analyse the fusion transitions rather than consider all fusion contexts. However, these fusion transitions do seem to provide additional information about the structure of processes. In order to define a bisimulation relation which equals the open bisimulation, we remove this information in the analysis of the labelled transitions: $PSQ : 0$ implies

$$\text{if } P \xrightarrow{?x=y} P_1 \text{ then either } Q \xrightarrow{?x=y} Q_1 \text{ or } Q \xrightarrow{\tau} Q_1, \text{ and } P_1 \mid \langle x = y \rangle \, S \, Q_1 \mid \langle x = y \rangle.$$

A consequence of adding fusion transitions is that we can use standard techniques to show that the resulting bisimulation relation is a congruence, and does indeed equal the open bisimulation.

In [GW00], we also study the more standard definition of bisimulation, which requires that fusion transitions match exactly. We know it yields a congruence which is contained in the open bisimulation. At the moment, we do not know whether the containment is strict. This question relates to an open problem for the π-calculus without replication and summation [2], of whether strong bisimulation is closed with respect to substitution.

The fusion LTS is given in figure 4. The labelled transitions follow the style of transition given in [Mil99], although our transitions are defined for arbitrary processes instead of processes of arity 0. This choice is not essential, since the bisimulation definition only refers to labelled transitions for processes of arity 0. The only additional complexity is that we have two rules for parallel composition, since this operator is only commutative for processes of arity 0. Notice that the structural congruence rule allows fusions to affect the labels: for example, the process $\langle x=y\rangle \,|\, \overline{x}.P$ can undergo the transition $\xrightarrow{\overline{y}}$ as well as $\xrightarrow{\overline{x}}$, because it is structurally congruent to $\langle x=y\rangle \,|\, \overline{y}.P$. Also notice that we do not have an explicit structural rule for the connection operator. Indeed, it is not possible to write such a rule, since the arity information would cause problems.

$$x.P \xrightarrow{x} P \qquad \overline{x}.P \xrightarrow{\overline{x}} P$$

$$x.P \,|\, \overline{y}.Q \xrightarrow{?x=y} P@Q \qquad x.P \,|\, \overline{x}.Q \xrightarrow{\tau} P@Q$$

$$\frac{P \xrightarrow{\alpha} P_1}{P \,|\, Q \xrightarrow{\alpha} P_1 \,|\, Q} \qquad \frac{Q \xrightarrow{\alpha} Q_1}{P \,|\, Q \xrightarrow{\alpha} P \,|\, Q_1}$$

$$\frac{P \xrightarrow{\alpha} Q, \quad x \notin \alpha}{(\nu x)P \xrightarrow{\alpha} (\nu x)Q} \qquad \frac{P \equiv P_1 \xrightarrow{\alpha} Q_1 \equiv Q}{P \xrightarrow{\alpha} Q}$$

Fig. 4. The labelled transition system for arbitrary π_F-processes. We do not distinguish between the labels $?x=y$ and $?y=x$.

Proposition 9
$P \searrow Q$ if and only if $P \xrightarrow{\tau} Q$.

The basic intuition regarding our definition of bisimulation is that two processes are bisimilar if and only if their standard forms have the same outer structure and, in all contexts of the form $\_@\langle\vec{y}\rangle$, if one process can do a labelled transition then so must the other to yield bisimilar processes. In fact, we do not need to consider all such contexts. Instead, it is enough to remove the top-level datums from the standard forms, and analyse the labelled transitions for the resulting processes.

[2] Personal communication with Davide Sangiorgi.

Definition 10 (Fusion Bisimulation)
A symmetric relation S is a *fusion bisimulation* if and only if PSQ implies

1. P and Q have standard forms $(\nu\vec{x})(\langle\vec{y}\rangle \mid P_1) \mid \langle\vec{u}=\vec{v}\rangle$ and $(\nu\vec{x})(\langle\vec{y}\rangle \mid Q_1) \mid \langle\vec{u}=\vec{v}\rangle$ respectively;

2a. if $P_1 \mid \langle\vec{u}=\vec{v}\rangle \xrightarrow{\alpha} P_1'$ where α is x, $\overline{x}$ or τ then $Q_1 \mid \langle\vec{u}=\vec{v}\rangle \xrightarrow{\alpha} Q_1'$ and $P_1' S Q_1'$;

2b. if $P_1 \mid \langle\vec{u}=\vec{v}\rangle \xrightarrow{?x=y} P_1'$ then either $Q_1 \mid \langle\vec{u}=\vec{v}\rangle \xrightarrow{?x=y} Q_1'$ or $Q_1 \mid \langle\vec{u}=\vec{v}\rangle \xrightarrow{\tau} Q_1'$ and $P_1' \mid \langle x=y\rangle \, S \, Q_1' \mid \langle x=y\rangle$;

3. similarly for Q.

Two processes P and Q are *fusion bisimilar*, written $P \sim_f Q$, if and only if there exists a fusion bisimulation S between them. The relation $\sim_f$ is the largest fusion bisimulation. We call it *the* fusion bisimulation, when the meaning is apparent.

This definition of fusion bisimulation is related to Sangiorgi's efficient characterisation of open bisimulation for the π-calculus with matching [San93].

The fusion transitions enable us to use standard techniques for proving that the fusion bisimulation is a congruence. We remove the structural congruence rule, and define an alternative LTS based on the structure of processes. This alternative LTS is given in figure 5. The non-standard rules are the fusion rules. The first two play a similar role to the τ-rules for the π-calculus. The other two express the effect of explicit fusions on labels and the generation of τ-transitions from simple fusion transitions. Notice that we have no rules for the connection operator. The connection between the fusion LTS and the alternative LTS is therefore only valid up to structural congruence.

Using the alternative LTS, we are able to define a bisimulation relation, prove that it is a congruence and show that it equals the fusion bisimulation. The details are given in [GW00]. With this congruence result, it is not difficult to show that fusion bisimulation equals open bisimulation.

3 Fusion systems

The fusion system framework consists of the same basic process constructs as the π_F-calculus. It generalises the input and output processes to more general *control processes*. For action calculi, control processes have the form $K(P_1, \ldots, P_r)$. It is possible to use the same approach for fusion systems. Instead, we choose to focus on control processes of the form $\vec{x}.K(P_1, \ldots, P_r)$, where the control K is a boundary containing the processes P_i and the names $\vec{x}$ provide the interface to the control. With this choice of control process, the connection operator can be derived from the other operators.

A fusion calculus is specified by a set $\mathcal{K}$ of controls, and a reaction relation describing the interaction between controls. Each control K in $\mathcal{K}$ has an associated arity $(m_1, \ldots, m_r) \to k$. The arity tells us how to construct a control process $\vec{x}.K(P_1, \ldots, P_r)$, where the m_i specify the arities of the P_i and $|\vec{x}| = k$.

$$x.P \xrightarrow{x} P \qquad \overline{x}.P \xrightarrow{\overline{x}} P$$

$$\frac{P \xrightarrow{x} P_1 \quad Q \xrightarrow{\overline{y}} Q_1}{P \,|\, Q \xrightarrow{?x=y} P_1 @ Q_1} \qquad \frac{P \xrightarrow{\overline{x}} P_1 \quad Q \xrightarrow{y} Q_1}{P \,|\, Q \xrightarrow{?x=y} P_1 @ Q_1}$$

$$\frac{P \xrightarrow{\alpha} Q \quad (x,y) \in E(P)}{P \xrightarrow{\alpha[y/x]} Q} \qquad \frac{P \xrightarrow{?x=x} Q}{P \xrightarrow{\tau} Q}$$

$$\frac{P \xrightarrow{\alpha} P_1}{P \,|\, Q \xrightarrow{\alpha} P_1 \,|\, Q} \qquad \frac{Q \xrightarrow{\alpha} Q_1}{P \,|\, Q \xrightarrow{\alpha} P \,|\, Q_1}$$

$$\frac{P \xrightarrow{\alpha} Q, \quad x \notin \alpha}{(\nu x)P \xrightarrow{\alpha} (\nu x)Q}$$

Fig. 5. The alternative LTS for arbitrary processes. The judgement $\alpha[y/x]$ denotes the simple substitution of one y for one x, generated by: $x[y/x] = y$; $\overline{x}[y/x] = \overline{y}$; $(?x=z)[y/x] = ?y=z$, and $\alpha[y/x] = \alpha$ when $x \notin \alpha$. Recall that $E(P)$ is the smallest equivalence relation on names generated by P.

Definition 11
The set $\mathcal{P}'_{\mathsf{F}}(\mathcal{K})$ of *pre-processes* of a fusion system specified by control set $\mathcal{K}$ is defined by the grammar

P	::=	nil	Nil process
		$P \,|\, P$	Parallel composition
		$P@P$	Connection
		$\langle x \rangle$	Datum
		$\langle x=y \rangle$	Fusion
		$(\nu x)P$	Restriction
		$\vec{x}.K(P_1, \ldots, P_r)$	Control process

The set $\mathcal{P}_{\mathsf{F}}(\mathcal{K})$ of *fusion processes* with arity m is defined by the rules in definition 6, with the input and output rules generalised to the control rule

$$\frac{P_i : m_i \quad i \in \{1, \ldots, r\}}{\vec{x}.K(P_1, \ldots, P_r) : 0}$$

where control K has arity $(m_1, \ldots, m_r) \to k$ and $|\vec{x}| = k$.

The definitions of *free* and *bound* names are standard. The definition of the *structural congruence* between fusion processes is the same as that given for the π_F-calculus in figure 3, except that it is closed with respect to arbitrary control processes. The *standard forms* are defined in the same way as those for the π_F-calculus given in section 2. We can therefore derive the connection operator from

the other operators. The *reaction relation* $\searrow$ is a binary relation between fusion processes of the same arity, which is closed with respect to $\_\,|\,$, $\_@\_$, $(\nu x)\_$ and $\_ \equiv \_$. We are able to specify whether the reaction relation is closed with respect to the controls.

The π fusion system We define the π fusion system, which is just a reformulation of the π_F-calculus. It is specified by the controls

$$\mathsf{in} : (m) \to 1 \qquad \mathsf{out} : (m) \to 1$$

where the π_F-processes $x.P$ and $\overline{x}.P$ correspond to the control processes of the form $x.\mathsf{in}(P)$ and $x.\mathsf{out}(P)$ respectively. The reaction relation is generated by the rule

$$x.\mathsf{in}(P) \,|\, x.\mathsf{out}(Q) \searrow P@Q,$$

where P and Q have the same arity and reaction does not occur inside the controls. Since the correspondence with the π_F-calculus is exact, the bisimulation results described in section 2.1 easily transfer to the π_F fusion system.

We summarise the embedding results of the π-calculus and the fusion calculus in the π_F fusion system. These results follow immediately from the analogous embeddings in the π_F-calculus [GW00]. The translations of the π-calculus and the fusion calculus into π_F fusion system are characterised by the input and output cases described above, plus the abstraction and concretion cases:

$$\begin{array}{rcll} [\![(\vec{x})P]\!] & = & (\nu\vec{x})(\langle\vec{x}\rangle \,|\, [\![P]\!]) & \text{Abstraction} \\ [\![(\nu\vec{x})\langle\vec{z}\rangle P]\!] & = & (\nu\vec{x})(\langle\vec{z}\rangle \,|\, [\![P]\!]) & \text{Concretion} \end{array}$$

There is a key difference between the embedding of the π-calculus and the embedding of the fusion calculus. For the π-embedding, reaction of a process in the image of $[\![\_]\!]$ necessary results in a process congruent to one in the image. The same is not true with the embedding of the fusion calculus, since for example

$$x.\mathsf{in}(\langle z\rangle \,|\, [\![P]\!]) \,|\, x.\mathsf{out}(\langle y\rangle \,|\, [\![Q]\!]) \quad \searrow \quad \langle z = y\rangle \,|\, [\![P]\!] \,|\, [\![Q]\!],$$

which does not have a corresponding reaction in the fusion calculus. The process on the left is in the image of the fusion calculus under $[\![\_]\!]$, but the one on the right has an unbound explicit fusion and so is not congruent to anything in the image. We do obtain an embedding result, in that by restricting y (or z) we obtain the π_F fusion reaction

$$\begin{array}{rcl} (\nu y)(x.\mathsf{in}(\langle z\rangle \,|\, [\![P]\!]) \,|\, x.\mathsf{out}(\langle y\rangle \,|\, [\![Q]\!])) & \searrow & (\nu y)(\langle z = y\rangle \,|\, [\![P]\!] \,|\, [\![Q]\!]) \\ & \equiv & [\![P]\!]\{z/y\} \,|\, [\![Q]\!]\{z/y\}, \end{array}$$

which does indeed correspond to a fusion reaction. The full translations and embedding results are given for the π_F-calculus in [GW00].

The λ_v fusion system We define the λ_v fusion system, which corresponds to a call-by-value λ-calculus. It is also possible to define fusion systems for the usual untyped λ-calculus, and the simply-typed versions by adding more structure to the arities. These fusion systems require the ability to replicate processes, which we have not used before. We are currently checking that our bisimulation results in section 2.1 hold in the presence of replication. We do not envisage difficulties.

The λ_v fusion system is specified by the controls

$$\mathsf{lam} : (2) \to 1 \qquad \mathsf{ap} : (\,) \to 3.$$

The idea is that the untyped λ-terms correspond to processes of arity 1. A lam-process of the form $u.\mathsf{lam}(P)$ 'locates' the function at u, with the extra arity for the process P corresponding to the abstraction of the λ-term. An ap-control $uvw.\mathsf{ap}$ 'locates' its function at u, its argument at v and its result at w. The reaction relation is generated by

$$u.\mathsf{lam}(P) \mid uvw.\mathsf{ap} \searrow P@\langle vw\rangle.$$

and reaction does not occur inside the lam-control. The translation from lambda terms to fusion processes is defined inductively by

$$\begin{array}{rcll}
[\![x]\!] & \longmapsto & \langle x\rangle & \\
[\![\lambda x.t]\!] & \longmapsto & (\nu u)(\langle u\rangle \mid !u.\mathsf{lam}((\nu x)(\langle x\rangle \mid [\![t]\!]))) & u \notin \mathsf{fn}([\![\lambda x.t]\!]) \\
[\![st]\!] & \longmapsto & (\nu uvw)(\langle w\rangle \mid [\![s]\!]@\langle u\rangle \mid [\![t]\!]@\langle v\rangle \mid uvw.\mathsf{ap}) & u,v,w \notin \mathsf{fn}([\![st]\!])
\end{array}$$

The replication in the translation of the lambda abstraction is used to generate copies of lambda processes via the structural congruence.

The results for the call-by-value λ-calculus are not as straightforward as the results for the π_F-example. To illustrate this, consider the lambda reaction $(\lambda x.x)y \searrow y$ which corresponds to the fusion system reaction

$$(\nu uw)(\langle w\rangle \mid !u.\mathsf{lam}((\nu x)\langle xx\rangle) \mid uyw.\mathsf{ap}) \searrow \langle y\rangle \mid (\nu u)(!u.((\nu x)\langle xx\rangle)).$$

After reaction, the process $(\nu u)(!u.\mathsf{lam}((\nu x)\langle xx\rangle))$ is redundant. This redundancy is expressed by a bisimulation congruence, with process $(\nu u)(!u.\mathsf{lam}((\nu x)\langle xx\rangle))$ bisimilar to nil. Bisimulation is thus required to relate the reaction relation of the λ-calculus with reaction in the λ_v fusion system.

Bisimulation for the λ_v fusion system is easy. It follows directly from bisimulation for the π_F fusion system. This is because the λ_v fusion system can be regarded as a simple subsystem of the π_F fusion system, using the translation

$$\begin{array}{rcl}
[\![u.\mathsf{lam}(P)]\!] & \mapsto & u.\mathsf{in}([\![P]\!]) \\
[\![uvw.\mathsf{ap}]\!] & \mapsto & u.\mathsf{out}(\langle vw\rangle)
\end{array}$$

which trivially preserves the reaction relation. It is future work to show whether the resulting congruence corresponds to a known behavioural congruence for the call-by-value λ-calculus.

It also remains future work to study general bisimulation congruences for fusion systems. One option is to try to generalise the bisimulation results for the π_F and λ_v fusion systems. Another option is to adapt the techniques of Leifer and Milner, in their work on general bisimulation congruences for reactive systems [LM00]. However, this is not our primary concern. We would first like to explore specific examples of fusion systems to illustrate the expressive power of explicit fusions.

References

[Fu97] Y. Fu. A proof-theoretical approach to communication. *ICALP*, LNCS 1256, Springer-Verlag, 1997.

[GW99] P.A. Gardner and L. Wischik. A process framework based on the π_F-calculus. *EXPRESS*, Elsevier Science Publishers, 1999.

[GW00] P.A. Gardner and L. Wischik. Explicit fusions. *MFCS*, 2000. To appear.

[Has97] M. Hasegawa. *Models of Sharing Graphs (A Categorical Semantics of Let and Letrec)*. PhD thesis, ECS-LFCS-97-360, University of Edinburgh, 1997.

[HG97] M. Hasegawa and P. Gardner. Higher order action calculi and the computational λ-calculus. *Theoretical Aspects of Computer Software*, Sendai, 1997.

[Hon00] K. Honda. Elementary structures in process theory (1): sets with renaming. *Mathematical Structures in Computer Science*, 2000. To appear.

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

[LM00] Jamey Leifer and Robin Milner. Deriving bisimulation congruences for reactive systems. *CONCUR*, 2000. To appear.

[Mil94a] R. Milner. Higher order action calculi. *Computer Science Logic*, LNCS 832, Springer-Verlag, 1994.

[Mil94b] R. Milner. Reflexive action calculi. Unpublished manuscript, 1994.

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

[Mil99] R. Milner. *Communicating and mobile systems: the pi calculus*. CUP, 1999.

[PV98] J. Parrow and B. Victor. The fusion calculus: expressiveness and symmetry in mobile processes. *LICS*, Computer Society Press, 1998.

[San93] D. Sangiorgi. A theory of bisimulation for the pi-calculus. *CONCUR*, Springer-Verlag, 1993.

[Sew00] Peter Sewell. From rewrite rules to bisimulation congruences. *Theoretical Computer Science*, 2000. To appear.