Strong Bisimulation for the Explicit Fusion Calculus

Lucian Wischik[1] and Philippa Gardner[2]

[1] University of Bologna, Italy. lu@wischik.com
[2] Imperial College, London. pg@doc.ic.ac.uk

Abstract. The pi calculus holds the promise of compile-time checks for whether a given program will have the correct interactive behaviour. The theory behind such checks is *bisimulation*. In the synchronous pi calculus, it is well-known that the various natural definitions of (strong) bisimulation yield different relations. In contrast, for the asynchronous pi calculus, they collapse to a single relation. We show that the definitions transfer naturally from the pi calculus to the explicit fusion calculus (a symmetric variant of the synchronous pi calculus), where they also collapse, and yield a simpler theory.
The important property of explicit fusions is that an explicit fusion in parallel with a term allows fused names to be substituted for each other. This means that parallel contexts become as discriminating as arbitrary contexts, and that *open bisimilarity* is more natural for the explicit fusion calculus than it was for the pi calculus. This is significant because 'open' is the principle behind automated bisimilarity-checkers.

1 Introduction

The past few years have seen much interest in the *pi calculus*, as a foundation for a new generation of programming languages for distributed and interactive computation: for example, JoCAML [10,8] and Polyphonic C# [3] are inspired by the join primitives which in turn arose from the pi calculus; and BPML [6] and Microsoft Biztalk [9,25] are business process languages partially inspired by the pi calculus. At the syntactic level, the pi calculus consists of simple primitives for rendezvous between concurrent processes – simpler than threads, mutexes, events and pipes, for instance, and more suited to message-based interactions like HTTP. Semantically, the pi calculus holds the promise of *behavioural types* – ie. compile-time verification that a piece of code obeys its intended protocol [7].

The standard paradigm for behavioural comparison in the pi calculus is *bisimulation*: this is a game where one party tries to make an interaction that the other party cannot match, and vice versa. If the two parties can match each other exactly, they are said to be *bisimilar*. Two factors complicate the theory of bisimilarity for the pi calculus. First, there are several natural definitions of bisimilarity. Second, one of the implementable form of bisimilarity (*symbolic* or *efficient*, based on *open bisimilarity* [24]) loses elegance through its need for *distinctions*, which indicate when particular names can never become equal.

I. Walukiewicz (Ed.): FOSSACS 2004, LNCS 2987, pp. 484–498, 2004.

Since 1997 there has been interest in symmetric generalisations of the pi calculus based on *fusions* [12,22,17]. These feature a symmetric interaction between sender and receiver, where the sent name becomes 'fused' or equal to the received name. (In contrast, pi calculus is asymmetric in that the sent name replaces the received name.) The stated motivations for fusion calculi were to simplify the basic pi primitives [22], to model graph reduction [12,17], and to model concurrent constraint programming [26]. In particular, we introduced the explicit fusion calculus to simplify action graphs, which lead to Milner's current work on bi-graphs [21]. We implemented explicit fusions in the Fusion Machine [15], and they were also adapted by the ongoing Microsoft 'Highwire' project.

The original contribution of this paper is to show how the four standard definitions of strong bisimulation congruence, familiar from the pi calculus, can be applied *without modification* to the explicit fusion calculus. (By contrast, other fusion calculi have used unfamiliar customised bisimulations). We show that the definitions collapse to a single relation in our fusion calculus – even though they yield different relations in the pi calculus. The reason for the collapse is that distinctions are no longer needed. (The collapsed relation turns out to be the same as those yielded by the customised bisimulations used for other fusion calculi [17].)

Bisimulation relations describe when two processes are behaviourally equivalent. The simplest definition of behavioural equivalence is: 'two processes are equivalent if and only if they have the same behaviour in all contexts.' But the infinite quantification over contexts makes this definition impractical. Instead, one looks for a relation that has a co-inductive definition, and hopes that it coincides with the contextual definition. There are actually four key ways to define behavioural equivalence for process calculi: depending on whether the relation is closed under initial contexts (*'shallow' congruence*) or under subsequently-changing contexts as well (*reduction-closed congruence*); and orthogonally whether we just observe the channels over which messages are sent (*barbed*) or also record the message and the resulting state (*ground*).

In the pi calculus, there is also a co-inductive definition of ground congruence, called *open bisimulation* [24]. It is a comparatively complicated definition, which uses *distinctions* to keep track of those names that must be kept apart (distinct from each other) in the bisimulation analysis. For example, in the transition

$$(x)(\overline{u}\,x \mid P) \xrightarrow{(x)\overline{u}\,x} P \tag{1}$$

the name x can never be substituted by another name; the distinctions keep track of this fact. The relationship between the bisimulations for the pi calculus are summarised below; interestingly, in the *asynchronous* variant of the pi calculus, the congruences collapse [2]:

$$\begin{array}{ccccc} \text{shallow-ground} & = & \text{reduction-closed ground} & = & \text{open} \\ & & \cap & & \\ & & \text{reduction-closed-barbed} & & \\ & & \cap & & \\ & & \text{shallow-barbed} & & \end{array}$$

We study analogous strong bisimulation relations for the *explicit fusion calculus* [17]. Like all fusion calculi, the explicit fusion calculus has symmetric non-binding input and output actions. What sets it apart is its simple local reaction between input and output actions:

$$\overline{u}\,x.P \mid u\,y.Q \;\;\rightarrow\;\; x{=}y \mid P \mid Q. \tag{2}$$

Here the *explicit fusion* $x{=}y$ allows x and y to be used interchangeably throughout the rest of the term. This 'interchange' power of fusions is what simplifies open bisimulation. (Also, the locality of this reaction allows us to use the same bisimulation definitions as the pi calculus; other fusion calculi use a non-local reaction and so need customised versions of bisimulation).

Recall that for Transition 1 in the pi calculus, x could never be substituted by another name, and so had to be kept distinct. But in the explicit fusion calculus, x can be substituted (e.g. by a parallel context $x{=}y \mid \_$), and so distinctions are not needed. Without distinctions, open bisimulation degenerates to just closure under substitution. And all reduction-closed congruences are by nature closed under substitution, and therefore coincide with open bisimulation. (Distinctions are 'not needed' only because fusions limit expressiveness – they make it impossible to generate two names that will always be distinct [4].)

Here we do not study weak bisimulation congruences. Many have been studied by Fu [13,14] for the chi calculus. An interesting open problem is to explore such congruences for the explicit fusion calculus, particularly since *equators* [18] in a weak setting are similar to explicit fusions [19,16].

Structure. The structure of the paper is as follows. Section 2 presents the explicit fusion calculus and compares it to other fusion calculi. Section 3 gives the definitions of barbs and labelled transitions for the calculus. Sections 4 and 5 define strong bisimulation and provide the results: Section 4 concentrates on the reduction-closed congruences, and Section 5 relates these to shallow congruence. The results in Section 4 were first reported in Wischik's doctoral dissertation [27].

This paper has many definitions – an inherent necessity in the current project, of showing how the standard definitions coincide for the explicit fusion calculus. We have been parsimonious, using just two intermediate definitions in our proofs.

$\downarrow$	Barbs (Def. 1), standard from the pi calculus
$\xrightarrow{\alpha}$	Labelled transitions (Def. 2), standard from pi
$\xrightarrow{\alpha}_e$	'Efficient' transitions (Def. 3), used as intermediate in proofs
$\xrightarrow{\alpha}_s$	Structural transitions (Def. 4), must be customised for each calculus
$\sim_b$	Reduction-closed barbed congruence (Def. 6), standard from pi
$\sim_g$	Reduction-closed ground congruence (Def. 7), standard from pi
$\overset{\text{io}}{\sim}_g$	'Inside-outside' bisimulation (Def. 8), used as intermediate in proofs
$\overset{\text{e}}{\sim}_g$	Efficient bisimulation (Def. 9), must be customised for each calculus
$\sim_{gs}$	Shallow ground congruence (Def. 11), standard from pi
$\sim_{bs}$	Shallow barbed congruence (Def. 11), standard from pi

2 Explicit Fusion Calculus

The explicit fusion calculus is defined in Table 1. It is like the pi calculus but with two differences: first, it includes explicit fusions $x{=}y$; second, it uses non-binding input actions $u\,x.P$ instead of binding input $u(x).P$. In the explicit fusion calculus, reaction between non-binding output and non-binding input gives rise to explicit fusions:

$$\overline{u}\,x.P \mid u\,y.Q \mid R \;\rightarrow\; x{=}y \mid P \mid Q \mid R.$$

The effect of the fusion $x{=}y$ is global in scope: the x and the y can be used interchangeably throughout the entire process, including R. To limit the scope of the fusion, we use restriction. For example, restricting x in the above expression we obtain

$$(x)(x{=}y \mid P \mid Q \mid R) \;\equiv\; P\{y/x\} \mid Q\{y/x\} \mid R\{y/x\}.$$

Explicit fusions allow for the substitutive effects of reaction to be delayed, rather than requiring the substitution be performed globally and immediately. This is reminiscent of the use of explicit *substitutions* in the lambda calculus [1], used in implementations to delay the substitutive effects of beta reduction.

We can emulate the pi-calculus reaction $\overline{u}\,x.P \mid u(y).Q \rightarrow P \mid Q\{x/y\}$, by binding y in the input action:

$$\begin{array}{lcll}
\overline{u}\,x.P \mid (y)(u\,y.Q) & \equiv & (y)(\overline{u}\,x.P \mid u\,y.Q) & \text{(assuming } y \notin \mathrm{fn}\,P\text{)} \\
& \rightarrow & (y)(x{=}y \mid P \mid Q) & \\
& \equiv & (y)(x{=}y \mid P \mid Q\{x/y\}) & \text{(substitutive effect of fusion)} \\
& \equiv & (y)(x{=}y) \mid P \mid Q\{x/y\} & \text{(scope intrusion)} \\
& \equiv & P \mid Q\{x/y\}. &
\end{array}$$

Alpha-renaming can also be deduced from the laws of structural congruence: $(x)P \equiv (x)((y)(x{=}y)|P) \equiv (xy)(x{=}y|P\{y/x\}) \equiv (y)((x)(x{=}y)|P\{y/x\}) \equiv (y)P\{y/x\}$.

We remark upon what distinguishes explicit fusions from other fusion calculi. They lack explicit fusions, and cannot therefore allow $x{=}y$ to be left behind after reaction. Instead, they allow reaction only in the presence of a restricted x or y – so the interchange power of the fusion can be immediately and fully discharged:

$$(x)(\overline{u}\,x.P \mid u\,y.Q \mid R) \;\rightarrow\; P\{y/x\} \mid Q\{y/x\} \mid R\{y/x\}.$$

The reaction is non-local, unlike reaction in the pi calculus and explicit fusion calculus. But despite the difference in operational semantics, they end up yielding the same bisimulation congruence as the explicit fusion calculus [17]. This congruence has already been axiomatised for finite processes [22]. We remark that the full polyadic reaction rule, using many xs and ys, is more complicated. It is also not clear how to implement the rule, since practical implementations (eg. Pict, CML) keep no record to distinguish a locally-generated (restricted) name from one that is pre-existing (free).

Table 1. The explicit fusion calculus

The **terms** P and **contexts** E in the explicit fusion calculus are

$$\begin{array}{lcl} P & ::= & \mathbf{0} \mid x{=}y \mid \overline{u}\,\widetilde{x}.P \mid u\,\widetilde{x}.P \mid (x)P \mid P|P \mid !P \\ E & ::= & \_ \mid \overline{u}\,\widetilde{x}.E \mid u\,\widetilde{x}.E \mid (x)E \mid P|E \mid E|P \mid !E \end{array}$$

The **structural congruence** on terms $\equiv$ is the smallest equivalence satisfying the following axioms, and closed with respect to contexts:

$$\begin{array}{llll} P|(Q|R) \equiv (P|Q)|R & P|Q \equiv Q|P & P|\mathbf{0} \equiv \mathbf{0} & !P \equiv P|!P \\ (x)(P|Q) \equiv (x)P \mid Q \text{ if } x \notin \mathrm{fn}(Q) & & (x)(y)P \equiv (y)(x)P & \\ x{=}y \mid y{=}z \equiv x{=}z \mid y{=}z & x{=}y \equiv y{=}x & x{=}x \equiv \mathbf{0} & (x)(x{=}y) \equiv \mathbf{0} \\ x{=}y \mid P \equiv x{=}y \mid P\{y/x\} & & & \end{array}$$

The **reaction relation** is the smallest relation $\rightarrow$ satisfying the following axiom and closed with respect to $\equiv$ and contexts:

$$\overline{u}\,\widetilde{x}.P \mid u\,\widetilde{y}.Q \;\rightarrow\; \widetilde{x}{=}\widetilde{y} \mid P \mid Q$$

The explicit fusions in a term P generate an **equivalence relation** $\mathrm{Eq}(P)$ on names as follows. (Given two equivalence classes F and G we write $F \oplus G$ for their equivalence-closed union, and $F\backslash x$ for when x is in a singleton class and all other names are related as in F; and $\mathbf{I}$ for the identity relation.)

$$\begin{array}{ll} \mathrm{Eq}(x{=}y) = \{(x,y),(y,x)\} \cup \mathbf{I} & \mathrm{Eq}(\overline{u}\,\widetilde{x}.P) = \mathbf{I} \\ \mathrm{Eq}(P|Q) = \mathrm{Eq}(P) \oplus \mathrm{Eq}(Q) & \mathrm{Eq}(u\,\widetilde{x}.P) = \mathbf{I} \\ \mathrm{Eq}((x)P) = \mathrm{Eq}(P)\backslash x & \mathrm{Eq}(\mathbf{0}) = \mathbf{I} \\ \mathrm{Eq}(!P) = \mathrm{Eq}(P) & \end{array}$$

We write $P \vdash x{=}y$ as shorthand for $(x,y) \in \mathrm{Eq}(P)$. Note that $P \equiv Q$ implies $\mathrm{Eq}(P) = \mathrm{Eq}(Q)$, and that $P \vdash x{=}y$ if and only if $P \equiv x{=}y \mid P$.

Following Milner [20], we will assume *well-sorted* terms: the sorting prevents arity mismatches such as $\overline{u}\,x \mid u\,yz$. Moreover, fusions only fuse names of the same sort. We do not define the sorting system; it is sufficient to note just that each name x has a sort S, written $x{:}S$, which prevents mismatches.

3 Barbs and Labelled Transitions

The two standard ways to characterise the behaviour of a program are through observations (barbs) and labelled transitions. We define them here. We in fact give three definitions of the labelled transitions: the first is quotiented by $\equiv$ and is the same as for the pi calculus; the second introduces a new fusion labelled transition $\xrightarrow{?u=v}$; and the third uses the fusion transitions to provide a structurally

Table 2. Concretions

Concretions have the form $(\widetilde{x})(\widetilde{y} : P)$ where the names in $\widetilde{x}$ are distinct and contained in $\widetilde{y}$, and no $x \in \widetilde{x}$ is fused by $\mathrm{Eq}(P)$. Let C, D range over concretions. For a concretion $(\widetilde{x})(\widetilde{y} : P)$, we call the context $I = (\widetilde{x})(\widetilde{y} : \_)$ the *interface* of the concretion, and we write the concretion as $I : P$. The names $\widetilde{x}$ are bound in this concretion.

Structural congruence on concretions is defined by: $(\widetilde{x}_1)(\widetilde{y}_1 : P_1) \equiv (\widetilde{x}_2)(\widetilde{y}_2 : P_2)$ if and only if there exist fresh names $\widetilde{x}$ of the same size as $\widetilde{x}_1$ and $\widetilde{x}_2$, and permutations π_1, and π_2 and substitutions $\sigma_1 = \{\widetilde{x}/\pi_1\widetilde{x}\}$, $\sigma_2 = \{\widetilde{x}/\pi_2\widetilde{x}\}$ such that $P_1\sigma_1 \equiv P_2\sigma_2$ and $\widetilde{y}_1\sigma_1$ is identical to $\widetilde{y}_2\sigma_2$ up to $\mathrm{Eq}(P_1)$.

The **operators** of restriction, composition and application on concretions are as follows. Assume by alpha-renaming that $\widetilde{x}_1$ and $\widetilde{x}_2$ do not intersect, and $\widetilde{x}_1$ binds no name free in P_2 and $\widetilde{x}_2$ binds no names free in P_1.

$$(z)\ (\widetilde{x})(\widetilde{y} : P) \stackrel{\text{def}}{=} \begin{cases} (\widetilde{x})(\widetilde{y} : P) & \text{if } z \in \{\widetilde{x}\} \\ (z\widetilde{x})(\widetilde{y} : P) & \text{if } z \in \{\widetilde{y}\} \backslash \{\widetilde{x}\} \\ (\widetilde{x})(\widetilde{y} : (z)P) & \text{otherwise} \end{cases}$$

$$(\widetilde{x}_1)(\widetilde{y}_1{:}P_1) \mid (\widetilde{x}_2)(\widetilde{y}_2{:}P_2) \stackrel{\text{def}}{=} (\widetilde{x}_1\widetilde{x}_2)(\widetilde{y}_1\widetilde{y}_2 : P_1 \mid P_2)$$

$$(\widetilde{x}_1)(\widetilde{y}_1{:}P_1) \mathbin{@} (\widetilde{x}_2)(\widetilde{y}_2{:}P_2) \stackrel{\text{def}}{=} (\widetilde{x}_1\widetilde{x}_2)(\widetilde{y}_1{=}\widetilde{y}_2 \mid P_1 \mid P_2)$$

We sometimes write just P to stand for the empty concretion $()(\emptyset : P)$.

inductive characterisation, which is easier to use in proofs. All three are all equivalent (Theorem 5).

Formally, assume an infinite set $\mathcal{N}$ of names. Let μ range over $\{u, \overline{u} : u \in \mathcal{N}\}$, α over $\{\tau, u, \overline{u} : u \in \mathcal{N}\}$, and λ over $\{\tau, u, \overline{u}, ?u{=}v : u, v \in \mathcal{N}\}$. Write $x \notin \mu$ (or α or λ) when x does not occur in the label.

Definition 1 (Barbs) *The observation relation between terms and barbs μ, denoted $P \downarrow \mu$, is the smallest relation satisfying*

$$\mu\widetilde{x}.P \downarrow \mu \qquad \begin{aligned} P \mid Q \downarrow \mu \quad & \textit{if } P \downarrow \mu \\ (x)P \downarrow \mu \quad & \textit{if } P \downarrow \mu \textit{ and } x \notin \mu \\ Q \downarrow \mu \quad & \textit{if } Q \equiv P \downarrow \mu \end{aligned}$$

For labelled transitions, we use a symmetric generalisation of Milner's *concretions* [20]. As an example, we write $P \xrightarrow{\mu} I : P'$ to indicate that the term P can perform an input or output action μ, sending or receiving the data I, and end up in state P'. Concretions are defined in Table 2.

Definition 2 (Labelled transitions) *The labelled transition relation $P \xrightarrow{\alpha} I : P'$ is the smallest relation satisfying*

$$\mu\widetilde{x}.P \xrightarrow{\mu} \widetilde{x} : P \qquad P \mid Q \xrightarrow{\alpha} C \mid Q \quad \textit{if } P \xrightarrow{\alpha} C$$

$$\overline{u}\,\widetilde{x}.P \mid u\,\widetilde{y}.Q \xrightarrow{\tau} \widetilde{x}{=}\widetilde{y} \mid P \mid Q \qquad (x)P \xrightarrow{\alpha} (x)C \quad \text{if } P \xrightarrow{\alpha} C \text{ and } x \notin \alpha$$
$$Q \xrightarrow{\alpha} D \quad \text{if } Q \equiv P \xrightarrow{\alpha} C \equiv D$$

This definition of labelled transitions is the same as that of the pi calculus. The pi calculus also has an inductively defined alternative, but it does not carry over to the explicit fusion calculus. Instead, as a first step towards our own customised inductive characterisation, we extend our labelled transition system with *fusion transitions*. The fusion transitions are generated from the rule

$$\overline{u}\,x.P \mid v\,y.Q \quad \xrightarrow{?u=v}_e \quad x{=}y \mid P \mid Q$$

where $\xrightarrow{?u=v}$ indicates that the left-hand term contains an input and an output on the channels u and v. This means that the term has the potential for a tau transition in the context $u{=}v \mid \_$.

Definition 3 (Extended transitions) *The extended labelled transition relation $P \xrightarrow{\lambda}_e I : P$ is the smallest relation satisfying*

$$\mu\widetilde{x}.P \xrightarrow{\mu}_e \widetilde{x} : P \qquad P \mid Q \xrightarrow{\lambda}_e C \mid Q \text{ if } P \xrightarrow{\lambda}_e C$$
$$\overline{u}\,\widetilde{x}.P \mid u\,\widetilde{y}.Q \xrightarrow{\tau}_e \widetilde{x}{=}\widetilde{y} \mid P \mid Q \qquad (x)P \xrightarrow{\lambda}_e (x)C \text{ if } P \xrightarrow{\lambda}_e C \text{ and } x \notin \lambda$$
$$\overline{u}\,\widetilde{x}.P \mid v\,\widetilde{y}.Q \xrightarrow{?u=v}_e \widetilde{x}{=}\widetilde{y} \mid P \mid Q \qquad Q \xrightarrow{\lambda}_e D \text{ if } Q \equiv P \xrightarrow{\lambda}_e C \equiv D$$

We remark that the fusion calculus of Victor and Parrow uses a different fusion transition. Recall that the fusion calculus requires certain restrictions to be present before allowing reaction. It uses the transition $\overline{u}\,x.P \mid u\,y.Q \xrightarrow{!x=y}_{\text{fu}} P \mid Q$ to indicate that, if eventually x or y are restricted, then a tau transition will be possible. Another alternative fusion label has appeared recently in work by Milner [21]. Milner's fusion label $P \xrightarrow{u=v|\_} P'$ denotes that $u{=}v|\_$ is a minimal context necessary to allow reaction: that is to say, $P \xrightarrow{?u=v} P'$ and $(u, v) \notin \mathrm{Eq}(P)$. This is part of Milners' programme to treat transition labels as minimal contexts. It would be interesting to give such a labelled transition system for the explicit fusion calculus, and compare the resulting bisimulation with the bisimulations given here.

Our fusion transitions allow for an *inductive* characterisation of labelled transitions – that is, one where the left hand side is not quotiented by structural congruence. We need to introduce one more feature relating to explicit fusions. The point is that $u{=}v \mid \overline{u}\,x.P$ undergoes a $\xrightarrow{\overline{u}}$ transition, but it is also structurally congruent to $u{=}v \mid \overline{v}\,x.P$ which undergoes a $\xrightarrow{\overline{v}}$ transition. Thus, explicit fusions in a term can change the labels of the transitions it undergoes. We write $P \vdash \lambda = \lambda'$ when P contains sufficient explicit fusions to turn the label λ into λ'. The definition is given below, and generalises that of Table 1 which only applies to names.

Definition 4 (Structurally-derived transitions) *The labelled transition system $P \xrightarrow{\lambda}_s I : P'$ is the smallest relation satisfying*

$$\overline{u}\,\widetilde{x}.P \xrightarrow{\overline{u}}_s \widetilde{x} : P \qquad u\,\widetilde{x}.P \xrightarrow{u}_s \widetilde{x} : P$$

$$\frac{P \xrightarrow{\overline{u}}_s C \quad Q \xrightarrow{v}_s D}{P \mid Q \xrightarrow{?u=v}_s C@D} \qquad \frac{P \xrightarrow{u}_s C \quad Q \xrightarrow{\overline{v}}_s D}{P \mid Q \xrightarrow{?u=v}_s C@D} \qquad \frac{P \xrightarrow{?u=u}_s C}{P \xrightarrow{\tau}_s C}$$

$$\frac{P \xrightarrow{\lambda}_s C}{P \mid Q \xrightarrow{\lambda}_s C \mid Q} \qquad \frac{Q \xrightarrow{\lambda}_s D}{P \mid Q \xrightarrow{\lambda}_s P \mid D} \qquad \frac{P \xrightarrow{\lambda}_s C \quad P \vdash \lambda{=}\lambda'}{P \xrightarrow{\lambda'}_s C}*$$

$$\frac{P \mid !P \xrightarrow{\lambda}_s C}{!P \xrightarrow{\lambda}_s C} \qquad \frac{P \xrightarrow{\lambda}_s C \quad x \notin \lambda}{(x)P \xrightarrow{\lambda}_s (x)C} \qquad \frac{P \xrightarrow{\lambda}_s C \equiv D}{P \xrightarrow{\lambda}_s D}$$

The rule marked $*$ uses the judgement $P \vdash \lambda_1 = \lambda_2$ defined by

$$\begin{aligned}
P &\vdash ?u{=}v = ?v{=}u \\
P &\vdash x = y \quad \textit{if } (x,y) \in \mathrm{Eq}(P) \\
P &\vdash \overline{x} = \overline{y} \quad \textit{if } (x,y) \in \mathrm{Eq}(P) \\
P &\vdash ?x{=}y = ?u{=}v \quad \textit{if } (x,u) \in \mathrm{Eq}(P) \textit{ and } (y,v) \in \mathrm{Eq}(P).
\end{aligned}$$

We remark that the normal $\xrightarrow{\tau}$ transition is deduced from identity fusion transitions $\xrightarrow{?x=x}$. The following extended example illustrates both the ability of an explicit fusion to rename a label (marked * in the rules), and also the deduction of a $\xrightarrow{\tau}$ transition:

$$\frac{\dfrac{u{=}v \mid \overline{u}\,x.P \mid v\,y.Q \xrightarrow{?u=v}_s u{=}v \mid x{=}y \mid P \mid Q}{u{=}v \mid \overline{u}\,x.P \mid v\,y.Q \xrightarrow{?u=u}_s u{=}v \mid x{=}y \mid P \mid Q}}{u{=}v \mid \overline{u}\,x.P \mid v\,y.Q \xrightarrow{\tau}_s u{=}v \mid x{=}y \mid P \mid Q}\ (*)$$

Finally, we state the equivalence of these various kinds of observations and labels. Recall that μ ranges over $\{u, \overline{u}\}$, α ranges over $\{u, \overline{u}, \tau\}$, and λ ranges over $\{u, \overline{u}, \tau, ?u{=}v\}$.

Theorem 5 (Labelled transition systems)

1. *$P \downarrow \mu$ if and only if there exists C such that $P \xrightarrow{\mu} C$.*
2. *$P \xrightarrow{\alpha} C$ if and only if $P \xrightarrow{\alpha}_e C$.*
3. *$P \xrightarrow{\lambda}_e C$ if and only if $P \xrightarrow{\lambda}_s C$.*

Proof. Parts 1 and 2 follow from the definitions. Part 3 was given in [17]. □

4 Bisimulation Congruence

In this section we define the strong reduction-closed congruences for the explicit fusion calculus, and show that they are equivalent. The congruences we study are: (1) standard barbed and ground bisimulation congruences using labelled transitions; (2) the *inside-outside* bisimulation; and (3) a co-inductive efficient characterisation. The inside-outside bisimulation is the key to showing that all the congruences are equivalent. (The co-inductive efficient characterisation was the only one studied in [17].)

First we give the definitions of barbed and ground congruence. Barbed congruence is defined using the reaction relation and the observations (barbs); ground congruence depends on the labelled transitions.

Definition 6 (Barbed congruence) *A relation $\mathcal{S}$ on terms in the calculus is a (strong) barbed bisimulation iff whenever $P\ \mathcal{S}\ Q$ then*

1. *$P \downarrow \mu$ if and only if $Q \downarrow \mu$*
2. *if $P \to P'$ then $Q \to Q'$ such that $P'\ \mathcal{S}\ Q'$*
3. *if $Q \to Q'$ then $P \to P'$ such that $P'\ \mathcal{S}\ Q'$.*

We write $\dot{\sim}_b$ for the largest barbed bisimulation. A barbed bisimulation $\mathcal{S}$ is additionally a reduction-closed barbed congruence iff whenever $P\ \mathcal{S}\ Q$ then, for all contexts E, $E[P]\ \mathcal{S}\ E[Q]$. We write $\sim_b$ for the largest reduction-closed barbed congruence.

Definition 7 (Ground congruence) *A relation $\mathcal{S}$ is a (strong) ground bisimulation iff whenever $P\ \mathcal{S}\ Q$ then, assuming I binds no names free in P or Q,*

1. *$P \xrightarrow{\alpha} I : P'$ implies $Q \xrightarrow{\alpha} I : Q'$ and $P'\ \mathcal{S}\ Q'$ and*
2. *$Q \xrightarrow{\alpha} I : Q'$ implies $P \xrightarrow{\alpha} I : P'$ and $P'\ \mathcal{S}\ Q'$.*

We write $\dot{\sim}_g$ for the largest ground bisimulation. A ground bisimulation $\mathcal{S}$ is additionally a reduction-closed ground congruence iff whenever $P\ \mathcal{S}\ Q$ then $E[P]\ \mathcal{S}\ E[Q]$ for all contexts E. We write $\sim_g$ for the largest reduction-closed ground congruence.

We now define *inside-outside* bisimulation, which is the natural form of open bisimulation for the explicit fusion calculus. To motivate it, we first consider some example terms which are ground bisimilar but not ground congruent.

(1) The terms $u{=}v$ and $\mathbf{0}$ are ground bisimilar, since neither undergoes any transitions. But consider them inside the context $\_ \mid \overline{u} \mid v$. The first allows a reaction; the second does not. *Two congruent terms must necessarily contain the same explicit fusions inside.*

(2) We use the abbreviation $\tau.R = (u)(\overline{u} \mid u.R)$ with u fresh. Consider the programs

$$P \stackrel{\text{def}}{=} !\overline{y}.x.\tau.z \mid !x.\overline{y}.\tau.z$$
$$Q \stackrel{\text{def}}{=} !(w)(\overline{y}.\overline{w} \mid x.w.z)$$

The two are ground bisimilar. However, in a context $x{=}y \mid \_$, then Q has an action $\xrightarrow{z}$ after two steps, but P has one only after three. *Two congruent terms must necessarily behave in the same way under explicit fusion contexts.* Closure under fusion contexts is like closing under substitutions; the example programs P and Q were first given by Sangiorgi and Boreale [5] to show that ground bisimulation is not closed under substitution.

We will see that the two conditions given above are necessary and sufficient to prove congruence. The two conditions are formalised in the following bisimulation:

Definition 8 (Inside-outside bisimulation) *A ground bisimulation $\mathcal{S}$ is an inside-outside bisimulation iff whenever $P\ \mathcal{S}\ Q$ then*

1. $\mathrm{Eq}(P) = \mathrm{Eq}(Q)$,
2. *for all fusions $x{=}y$: $x{=}y|P\ \mathcal{S}\ x{=}y|Q$.*

We write $\overset{\mathrm{io}}{\sim}_g$ for the largest inside-outside bisimulation.

Our final bisimulation has a simple co-inductive definition that avoids all quantifications over any sort of context; it is standard to call such a bisimulation *efficient*. Recall that the fusion transition

$$\overline{u}\,x.P \mid v\,y.Q \quad \xrightarrow{?u=v} \quad x{=}y \mid P \mid Q$$

indicates that, in the presence of an explicit fusion $u{=}v$, the terms can react. This means that quantification over a term's fusion labels is as discriminating as quantifying over explicit fusion contexts.

Definition 9 (Efficient bisimulation) *An* efficient bisimulation *is a symmetric relation $\mathcal{S}$ such that if $P\ \mathcal{S}\ Q$ then, assuming I binds no names free in Q,*

1. $\mathrm{Eq}(P) = \mathrm{Eq}(Q)$,
2. *$P \xrightarrow{\alpha}_e I : P'$ implies $Q \xrightarrow{\alpha}_e I : Q'$ and $P'\ \mathcal{S}\ Q'$,*
3. *$P \xrightarrow{?u=v}_e P'$ implies $u{=}v|Q \xrightarrow{\tau}_e Q'$ and $u{=}v|P'\ \mathcal{S}\ Q'$.*

We write $\overset{\mathrm{e}}{\sim}_g$ for the largest efficient bisimulation.

We remark on why the third condition in the definition of $\overset{\mathrm{e}}{\sim}_g$ is as it is. The intuition is that $\overset{\mathrm{e}}{\sim}_g$ should coincide with $\overset{\mathrm{io}}{\sim}_g$. Now $\overset{\mathrm{io}}{\sim}_g$ says that, if P can react in a context $u{=}v$, then so can Q. The labelled transition $P \xrightarrow{?u=v}_e C$ also declares that P can react in the context, but it actually declares more information: not just that it can react in the context, but also that it actually contains input and output commands on u and v. Therefore, in the definition of efficient bisimulation, the consequent must remove this extra information about it containing u and v.

Theorem 10 $\sim_b\ =\ \sim_g\ =\ \overset{\mathrm{io}}{\sim}_g\ =\ \overset{\mathrm{e}}{\sim}_g$.

Proof. We first show that $\sim_b = \sim_g$. It is apparent from the definitions that $\sim_g \subseteq \sim_b$. The opposite direction, that $\sim_b \subseteq \sim_g$, requires more work. The issue is that, for a transition $P \xrightarrow{\mu} I{:}P'$, ground bisimulation $\sim_g$ records the data and resulting state $I{:}P'$, while barbed bisimulation $\sim_b$ discards it. Our task is to create a context R which can reconstruct it. In particular, given a transition $P \xrightarrow{\mu} I{:}P'$, we deduce that $P|R$ undergoes a tau transition, and hence (by barbed bisimilarity) so does $Q|R$, and we need an R sophisticated enough to deduce that $Q \xrightarrow{\mu} I{:}Q'$.

We use a family of contexts $R = u\,\widetilde{y}.\phi$ where all the names $\widetilde{y}$ are fresh, and ϕ is a fusion of two fresh names. This fusion ϕ acts as our litmus paper: supposing that $P \xrightarrow{\overline{u}} I{:}P'$, then P will react with R to liberate ϕ. By the assumption of barbed bisimilarity, $Q|R$ makes a matching transition which also liberates ϕ, and so the matching transition must have involved R, and there must be a transition $Q \xrightarrow{\overline{u}} J{:}Q'$. The remainder of the work is then to show that the interfaces I and J match. This amounts to picking apart the fusions involving $\widetilde{y}$, which is possible because each $y \in \widetilde{y}$ is fresh and distinct. More detail is given in Proposition 38 of [27].

For the proof that $\sim_g \subseteq \overset{\text{io}}{\sim}_g$, consider the three requirements in the definition of $P \overset{\text{io}}{\sim}_g Q$: first that $P \dot{\sim}_g Q$, which is also a requirement of $\sim_g$; second that $x{=}y|P \overset{\text{io}}{\sim}_g x{=}y|Q$, which is a special case of the more general context closure in $\sim_g$; third that $\mathrm{Eq}(P) = \mathrm{Eq}(Q)$. For the proof of this third point we start by supposing the contrary: that there exist two congruent terms $P \sim_g Q$ but with a pair of names u, v such that $(u, v) \in \mathrm{Eq}(P)$ but $(u, v) \notin \mathrm{Eq}(Q)$. Consider P in the example context $\overline{u}\,x \mid v\,y \mid \overline{x}$, where x and y are fresh names. This undergoes the transitions

$$P \mid \overline{u}\,x \mid v\,y \mid \overline{x} \quad \xrightarrow{\tau} \quad P \mid x{=}y \mid \overline{x} \quad \xrightarrow{\overline{y}} \quad P \mid x{=}y.$$

But no single transition in Q can result in $x{=}y$; therefore no $\xrightarrow{\overline{y}}$ can follow a single tau transition; therefore P and Q are not congruent. This contradiction proves that if $P \sim_g Q$ then $\mathrm{Eq}(P) = \mathrm{Eq}(Q)$.

Continuing in the forward direction, $\overset{\text{io}}{\sim}_g \subseteq \overset{\text{e}}{\sim}_g$ is apparent from the definitions.

Finally, we show that $\overset{\text{e}}{\sim}_g \subseteq \sim_g$. It is clear from the definitions that $\overset{\text{e}}{\sim}_g$ is a ground bisimulation; it remains to prove that it is a reduction-closed congruence. The proof for this was given in [17]. We sketch the proof briefly here: construct $\mathcal{S}$ as the smallest relation that contains $\overset{\text{e}}{\sim}_g$ and that is closed under

1. if $P \equiv P_1\ \mathcal{S}\ Q_1 \equiv Q$ then $P\ \mathcal{S}\ Q$;
2. if $P\ \mathcal{S}\ Q$ then $(x)P\ \mathcal{S}\ (x)Q$, $\mu\widetilde{x}.P\ \mathcal{S}\ \mu\widetilde{x}.Q$ and $!P\ \mathcal{S}\ !Q$;
3. if $P_1\ \mathcal{S}\ Q_1$ and $P_2\ \mathcal{S}\ Q_2$ then $P_1|P_2\ \mathcal{S}\ Q_1|Q_2$.

Clearly $\mathcal{S}$ is a reduction-closed congruence. It remains to prove that it is an efficient bisimulation, which is done by a lengthy induction on its construction. We only remark on why the third closure condition is as strong as it is. Imagine the weaker condition that if $P\ \mathcal{S}\ Q$ then $P|R\ \mathcal{S}\ Q|R$ and $R|P\ \mathcal{S}\ R|Q$. Now

consider the replication case, that $P \xrightarrow{\alpha} I : P'$ giving $!P \xrightarrow{\alpha} P'|!P$. We can deduce that $!Q \xrightarrow{\alpha} Q'|!Q$ and $P' \mathcal{S} Q'$. But now we need closure conditions on $\mathcal{S}$ that are strong enough to deduce that $P'|!P \mathcal{S} Q'|!Q$. The weaker conditions are not adequate. □

5 Shallow Congruence

In the previous section we considered 'reduction-closed congruences' – i.e. the largest congruences that *are also* bisimulations. These model the situation where environments change during execution (the norm in distributed computation). In this section we consider 'shallow congruences' – the largest congruences *contained in* bisimulation. These model the situation where a sub-program's context is fixed at compile-time. They are generally called just 'congruences' in the literature.

Intuitively, one might expect the reduction-closed and shallow congruences to generate the same relations, since one could presumably write an initial environment sophisticated enough to model a subsequently-changing environment. This result holds for ground congruences in the synchronous pi calculus. However, it is not true for the barbed congruences in the synchronous pi calculus. To prove the analogous result for the *weak* barbed congruences associated with the *asynchronous* pi calculus, Fournet and Gonthier [11] actually had to use a Universal Pi-calculus Machine for their initial environment, and they used it to simulate the execution of a Goedelised version of a program.

In the explicit fusion calculus, we prove that the shallow congruences coincide with the reduction-closed congruences. Our proof technique, like that of Fournet and Gonthier, involves creating an initial sophisticated environment. But because we are using ground rather than barbed bisimulation, and thanks to the inside-outside bisimulation for explicit fusions, our environment is much simpler.

Definition 11 (Shallow Congruence)

1. *Two terms P and Q are* shallow ground congruent *iff for all contexts E, $E[P] \dot{\sim}_g E[Q]$. We write $\sim_{gs}$ for the largest shallow ground congruence.*
2. *Two terms are* shallow barbed congruent *iff for all contexts E, $E[P] \dot{\sim}_b E[Q]$. We write $\sim_{bs}$ for the largest shallow barbed congruence.*

Theorem 12 $\sim_{gs} \subseteq \overset{\text{io}}{\sim}_g$.

Proof. We will construct a relation $\mathcal{S}$ such that clearly $\sim_{gs} \subseteq \mathcal{S}$, and prove that $\mathcal{S} \subseteq \overset{\text{io}}{\sim}_g$. The intuition is as follows. If two terms are related by $\sim_{gs}$, then they are congruent when placed in any initial context. We will design an particular context R_N parameterised by a set of sorted names N, and define

$$\mathcal{S} = \{(P, Q) : R_N|P \dot{\sim}_g R_N|Q \text{ for all } N \supseteq \text{fn}\, P|Q\}.$$

To prove that $\mathcal{S}$ is a $\overset{\text{io}}{\sim}_g$, we need R_N to be sophisticated enough to demonstrate the four properties of $\overset{\text{io}}{\sim}_g$ (Definition 8): whenever $P \mathcal{S} Q$ then

1. $\mathrm{Eq}(P) = \mathrm{Eq}(Q)$;
2. $x{=}y|P \;\mathcal{S}\; x{=}y|Q$ for any fusion $x{=}y$, for $x, y \in \mathrm{fn}\, P|Q$;
3. $P \xrightarrow{\tau} P'$ implies $Q \xrightarrow{\tau} Q'$ with $P' \;\mathcal{S}\; Q'$;
4. $P \xrightarrow{\mu} I{:}P'$ implies $Q \xrightarrow{\mu} I{:}Q'$ with $P' \;\mathcal{S}\; Q'$.

(Actually, Definition 8 combines the final two properties into a single line. We have separated them here because they involve very different proof techniques.)

We now define R_N. It uses a family of fresh channels r_S as part of a 'database' of names N: if $x{:}S \in N$ then R_N will contain $!\,\overline{r}_S x$. (The subscript S indicates the sort of x; a sortless proof is also possible). It also uses fresh channels to signal the tests of each of the four properties: t_{S1} for a test of property 1, t_{S2} for a test of property 2, and t_{S4a} and t_{S4b} to signal an input or output commitment for property 4. In the following, for the last three cases we write $S \in N$ to range over $\{S : (n{:}S \in N)\}$; and in the final case we write $\overline{r}\, x_i$ to stand for the correctly-sorted r (ie. indexed by the sort of x_i).

$$
\begin{aligned}
R_N = \quad & \prod_{n:S\in N} !\,\overline{r}_S n && \textit{database}\\
| \quad & \prod_{S\in N} !r_S(x).r_S(y).(\widetilde{a}\widetilde{b})\overline{t}_{S1}xy\widetilde{a}\widetilde{b}.(\overline{x}\,\widetilde{a}.\overline{a}_1 \mid y\,\widetilde{b}) && \textit{property 1}\\
| \quad & \prod_{S\in N} !r_S(x).r_S(y).(cd)\overline{t}_{S2}xycd.(\overline{c}\,x.\overline{d} \mid c\,y) && \textit{property 2}\\
| \quad & \prod_{S\in N} \left(\begin{array}{l} !r_S(u).u(\widetilde{x}).\overline{t}_{S4a}u\widetilde{x}.(!\overline{r}\,x_1 \mid \ldots \mid !\overline{r}\,x_n) \\ \mid !r_S(u).\overline{u}(\widetilde{x}).\overline{t}_{S4b}u\widetilde{x}.(!\overline{r}\,x_1 \mid \ldots \mid !\overline{r}\,x_n) \end{array} \right) && \textit{property 4}
\end{aligned}
$$

It remains to demonstrate how this initial context R_N can establish each of the properties. We start with property 4, because it will explain the database. Assume that $P \xrightarrow{\overline{u}} I{:}P'$ with $I = (\widetilde{z})(\widetilde{y} : \_)$. Then these transitions are possible:

$$
\begin{aligned}
R_N \mid P &\xrightarrow{\tau}\xrightarrow{\tau} (\widetilde{z}\widetilde{x})(\widetilde{y}{=}\widetilde{x} \mid R_N \mid P' \mid \overline{t}\,u\widetilde{x}.(!\overline{r}\,x_1|\ldots|!\overline{r}\,x_n))\\
&\equiv\ (\widetilde{z})(R_N \mid P' \mid \overline{t}\,u\widetilde{y}.(!\overline{r}\,y_1|\ldots|!\overline{r}\,y_n))\\
&\xrightarrow{\overline{t_{S4a}}}\ (\widetilde{z})(u\widetilde{y} : R_{N\widetilde{y}} \mid P').
\end{aligned}
$$

Hence also $R_N|Q$ makes the transition $R_N|Q \xrightarrow{\tau}\xrightarrow{\tau}\xrightarrow{\overline{t_{S4a}}} (\widetilde{z})(u\widetilde{y} : R_{N\widetilde{y}}|Q')$ with $R_{N\widetilde{y}}|P' \mathrel{\dot{\sim}_g} R_{N\widetilde{y}}|Q'$. This transition must have come from $Q \xrightarrow{\overline{u}} (\widetilde{x})(\widetilde{y} : Q')$, so proving the property. But there is an extra issue with property 4, which explains the database N and why definition of $\mathcal{S}$ was for N *equal or greater than* the free names of P and Q. If the transition $P \xrightarrow{\overline{u}} I{:}P'$ emits names $\widetilde{y}$ in the interface I, even as bound names, then subsequent tests of the properties must test over $\widetilde{y}$ as well as N. This is satisfied because, after the name $\widetilde{y}$ has been emitted, we obtain the larger testing context $R_{N\widetilde{y}}$.

For property 1, for any $x, y \in N$ we have $R_N|P \xrightarrow{\tau}\xrightarrow{\tau}\xrightarrow{\overline{t}_{S1}} (\widetilde{a}\widetilde{b})(xy\widetilde{a}\widetilde{b} : \overline{x}\,\widetilde{a}.a_1 \mid y\,\widetilde{b} \mid R_N \mid P)$. If $P \vdash x{=}y$ then this makes the further transitions $\xrightarrow{\tau}\xrightarrow{\overline{b}_1}$

for b_1 the first element of $\widetilde{b}$. By bisimilarity, $R_N|Q$ makes the same transitions – which can only have happened because $Q \vdash x{=}y$ as well. (In the case where x and y have empty sort, giving $\widetilde{a}$ and $\widetilde{b}$ with zero arity, then a_1 does not exist and this test would not work. The solution is to first use a transformation $\hat{\cdot}$ that converts all zero-arity channels into single-arity channels with a dummy argument. Then use $\mathcal{S}{=}\ \{(P,Q) : R_N|\hat{P} \dot{\sim}_g R_N|\hat{Q}\}$. We will ignore this additional complexity, since it does not substantially affect the proof.)

For property 2, for any $x, y \in N$ then $R_N|P \xrightarrow{\tau}\xrightarrow{\tau}\xrightarrow{\overline{t}_{S2}}\xrightarrow{\tau}\xrightarrow{\overline{d}} x{=}y \mid R_N \mid P$. By bisimilarity $R_N|Q$ makes the same transitions. Therefore $(x{=}y|P, x{=}y|Q) \in \mathcal{S}$.

For property 3, suppose $P \xrightarrow{\tau} P'$. Then $R_N|P \xrightarrow{\tau} R_N|P'$ and (through bisimilarity) $R_N|Q \xrightarrow{\tau} T'$ with $R_N|P' \dot{\sim}_g T'$. Where did this tau transition come from? Not from R_N interacting with Q, since they have no prefixes in common. Not from R_N on its own, since this could only have been an interaction between $!\overline{r}\,n$ and one of the four tests, in which case T' would admit either $\xrightarrow{\tau}\xrightarrow{\overline{t}_{S_1}}$ or $\xrightarrow{\tau}\xrightarrow{\overline{t}_{S_2}}$ or $\xrightarrow{u}\xrightarrow{\overline{t}_{S4a}}$ or $\xrightarrow{u}\xrightarrow{\overline{t}_{S4b}}$. But none of these is matched by $R|P'$, so the reaction must have come from Q alone, i.e. $R_N|Q \xrightarrow{\tau} R_N|Q'$ and hence $R_N|P' \dot{\sim}_g R_N|Q'$. □

Theorem 13 (Barbed congruence) $\sim_g \ = \ \sim_{gs} \ = \ \sim_{bs}$.

Proof. The first part is a trivial corollary of Theorem 12. The second part uses the same 'stratification of bisimilarities' technique as is used in the pi calculus ([23], Theorem 2.2.9). □

We remark that two extra operators, match and internal choice, are used in the proof by stratification. As discussed in [23], their use is easily encoded into a calculus which lacks them – such as the explicit fusion calculus studied here.

6 Conclusions and Future Work

One of Parrow's instincts behind the fusion calculus [22] was that it would yield a straightforward bisimulation theory: that the standard congruences would coincide, and that open bisimulation would be simpler. Parrow and Victor defined a congruence for the fusion calculus, *hyperequivalence*, but its definition is non-standard and perhaps a little complicated. This paper provides an in-depth account of strong bisimulation for the explicit fusion calculus. We have taken the standard congruence definitions from the pi calculus, shown that they can be applied directly to the explicit fusion calculus, and proved that they describe the same relation. Our proof techniques rely on the *inside-outside* bisimulation (Definition 8), which is analogous to (and simpler than) open bisimulation for the pi calculus. These results surprised us. The connections between fusion bisimulation and pi bisimulation have not previously been apparent, since each fusion calculus has used its own customised definitions of transitions and bisimulation. By contrast, we use standard definitions.

References

1. M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit substitutions. *Journal of Functional Programming*, 1(4):375–416, 1991.
2. R. M. Amadio, I. Castellani, and D. Sangiorgi. On bisimulations for the asynchronous pi calculus. *Theoretical Computer Science*, 195(2):291–324, 1998.
3. N. Benton, L. Cardelli, and C. Fournet. Modern concurrency abstractions for C#. In *ECOOP 2002*, LNCS 2374:415–440.
4. M. Boreale, M. Buscemi, and U. Montanari. D-fusion: a distinctive fusion calculus. Submitted for publication, 2003.
5. M. Boreale and D. Sangiorgi. Some congruence properties for pi-calculus bisimilarities. *Theoretical Computer Science*, 198(1–2):159–176, 1998.
6. Business Process Management Initiative. Business process modelling notation (BPML). http:// www.bpmi.org/.
7. S. Chaki, S. K. Rajamani, and J. Rehof. Types as models: Model checking message-passing programs. In *POPL'02*, pages 45–57. ACM Press.
8. S. Conchon and F. L. Fessant. Jocaml: Mobile agents for objective-caml. In *ASA'99 / MA'99*, pages 22–29. IEEE, Computer Society Press.
9. M. Corporation. Biztalk server. http:// www.microsoft.com/ biztalk/.
10. C. Fournet and G. Gonthier. The reflexive chemical abstract machine and the join-calculus. In *POPL'96*, pages 372–385. ACM Press.
11. C. Fournet and G. Gonthier. A hierarchy of equivalences for asynchronous calculi. In *ICALP'98*, LNCS 1443:844–855.
12. Y. Fu. The chi-calculus. In *Proceedings of ICAPDC '97*, pages 74–81. IEEE, Computer Society Press.
13. Y. Fu. Bisimulation lattice of chi processes. In *ASIAN 1998*, LNCS 1538:245–262.
14. Y. Fu. Open bisimulations on chi processes. In *CONCUR 1999*, LNCS 1664:304–319.
15. P. Gardner, C. Laneve, and L. Wischik. The fusion machine. In *CONCUR 2002*, LNCS 2421:418–433.
16. P. Gardner, C. Laneve, and L. Wischik. Linear forwarders. In *CONCUR 2003*, LNCS 2761:415–430.
17. P. Gardner and L. Wischik. Explicit fusions. In *MFCS 2000*, LNCS 1893:373–382.
18. K. Honda and N. Yoshida. On reduction-based process semantics. *Theoretical Computer Science*, 152(2):437–486, 1995.
19. M. Merro. On the expressiveness of chi, update and fusion calculi. In *EXPRESS 1998*, volume 16.2 of *Electronic Notes in Theoretical Computer Science*. Esevier.
20. R. Milner. *Communicating and Mobile Systems: the Pi-calculus*. Cambridge University Press, 1999.
21. R. Milner. Bigraphical reactive systems. In *CONCUR 2001*, LNCS 2154:16–35.
22. J. Parrow and B. Victor. The fusion calculus: Expressiveness and symmetry in mobile processes. In *LICS'98*, pages 176–185. IEEE, Computer Society Press.
23. D. Sangiorgi and D. Walker. *The Pi-Calculus: A Theory of Mobile Processes*. Cambridge University Press, 2001.
24. D. Sangiorgi. A theory of bisimulation for the pi-calculus. *Acta Informatica*, 33:69–97, 1996.
25. S. Thatte. XLANG: Web services for business process design. http:// www.gotdotnet.com/ team/ xml_wsspecs/ xlang-c/.
26. B. Victor and J. Parrow. Concurrent constraints in the fusion calculus. In *ICALP'98*, LNCS 1443:455–469.
27. L. Wischik. *Explicit Fusions: Theory and Implementation*. PhD thesis, University of Cambridge, 2001.