D-Fusion: A Distinctive Fusion Calculus[*]

Michele Boreale[1], Maria Grazia Buscemi[2], and Ugo Montanari[2]

[1] Dipartimento di Sistemi e Informatica, Università di Firenze, Italy
[2] Dipartimento di Informatica, Università di Pisa, Italy
boreale@dsi.unifi.it, {buscemi,ugo}@di.unipi.it

Abstract. We study the relative expressive power of Fusion and pi-calculus. Fusion is commonly regarded as a generalisation of pi-calculus. Actually, we prove that there is no uniform fully abstract embedding of pi-calculus into Fusion. This fact motivates the introduction of a new calculus, D-Fusion, with two binders, λ and ν. We show that D-Fusion is strictly more expressive than both pi-calculus and Fusion. The expressiveness gap is further clarified by the existence of a fully abstract encoding of mixed guarded choice into the choice-free fragment of D-Fusion.

1 Introduction

The design of distributed applications based on XML, like Web services [22] or business-to-business systems [6], sees the emergence of a message-passing programming style. Languages like Highwire [5] provide, in a concurrent setting, sophisticated data structures, which allow programmers to describe and manipulate complex messages and interaction patterns. If one looks for 'foundational' counterparts of these programming languages the pi-calculus [9, 10] and the Fusion calculus [16] seem very promising candidates. Both of them, indeed, convey the idea of message-passing in a distilled form, and come equipped with a rich and elegant meta-theory.

The main novelty of Fusion when compared to the pi-calculus is the introduction of *fusions*. A fusion is a name equivalence that, when applied onto a term, has the effect of a (possibly non-injective) name substitution. Fusions are ideal for representing, e.g., forwarders for objects that migrate among locations [3], or forms of pattern matching between pairs of messages [5]. Computationally, a fusion is generated as a result of a synchronisation between two complementary actions, and it is atomically propagated to processes running in parallel with the active one. This happens in much the same way as, in logic programming, term substitutions resulting from a unification step on a subgoal can be forced on the other subgoals.

If compared to pi-calculus name-passing, fusion name-passing enables a more general binding mechanism. However, differently from the pi-calculus, the binding mechanism of Fusion ignores the issue of unicity of newly generated names. One of the goals of this paper is to show that this fact limits the expressiveness of Fusion. Overcoming

[*] Research partially supported by FET-Global Computing projects *PROFUNDIS* and *MIKADO*.

W.-N. Chin (Ed.): APLAS 2004, LNCS 3302, pp. 296–310, 2004.

this limitation calls for co-existence of two name binders, λ and ν: the former analogous to the only binder of Fusion, and the latter imposing unicity, like in pi-calculus. The resulting *distinctive* Fusion calculus, or *D-Fusion*, is at least as expressive as pi-calculus and Fusion separately, and in fact, we strongly argue, *more* expressive than both. However, the main motivation of the present study is not to push for adoption of yet another calculus, but rather to clarify the relationship between two fundamental concepts like binding and fusions, and propose a consistent and simple semantical framework to reason on them. A more precise account of our work follows.

The binding mechanism of pi-calculus generalises that of λ-calculus in several ways. Input prefix $a(x).$ binds like λx, and name passing takes place in pi-calculus in a way typical of functional programming, i.e., formal names are assigned their actual counterparts. The *restriction* binder ν, however, is very different from λ, as a restricted name can be exported (extruded), with the guarantee that it will never be identified to anything else.

Fusion calculus is presented in [16] as a more uniform and more expressive evolution of the pi-calculus. The main idea is to decompose input prefix $a(x).$ into a binder (x) and a prefix $a\langle x\rangle.$ In the polyadic case, matching between the input list and the output list of arguments induces name unification, i.e. a fusion. The latter is propagated across processes, but one or more binders can be used to control the scope (i.e. propagation) of the fusion. Thus one achieves both a perfect symmetry between input and output and a more general name passing mechanism.

At first sight, Fusion is more general than pi-calculus. And, indeed, the pi-calculus transition system can be embedded into Fusion's, provided that one identifies restriction (νx) with the (x) binder of Fusion [16].

Our first move is to argue that this embedding breaks down if comparing the two calculi on the basis of behavioural semantics. We prove that no 'uniform' encoding exists of pi-calculus into Fusion that preserves any 'reasonable' behavioural equivalence (at least as fine as trace equivalence). Here 'uniform' means homomorphic with respect to parallel composition and name substitution, mapping (νx) to (x) and preserving (a subset of) weak traces. As hinted before, the ultimate reason for this failure is that in Fusion all names are like logical variables, i.e., unification always succeeds, which is not true in the pi-calculus.

The above considerations motivate the introduction of a new calculus, D-Fusion, with two binders, λ and ν: the first generalises input prefix, and the second corresponds to restriction. Also, any issue of symmetry between input and output is preempted, since the calculus has just one kind of prefix (no polarisation); polarised prefixes can be easily encoded, though. In D-Fusion, while lambdas are used to control the propagation of fusions, restrictions are used to possibly inhibit fusions. In logical terms, this corresponds to consider unification not only among variables, but also among variables and dynamically generated constants (that is, ν-extruded names). As expected, unification fails whenever one tries to identify two distinct constants. We show that the additional expressive power achieved in this way is relevant. Both pi-calculus and Fusion are subcalculi of D-Fusion. Moreover, the combined mechanism of restriction and unification yields additional expressive power: it allows to express a form of pattern matching which cannot be expressed in the other two calculi. As a consequence, we prove, D-Fusion cannot be uniformly encoded neither into Fusion, nor into pi-calculus.

Next, the gap between D-Fusion and Fusion/pi-calculus is explored from a more concrete perspective. First, we exhibit a simple security protocol and a related *correlation* property that are readily translated into D-Fusion. The property breaks down if uniformly translating the protocol into Fusion. The failure is illuminating: in Fusion, one has no way of declaring unique fresh names to correlate different messages of the protocol.

Palamidessi has shown [13, 14] that nondeterministic *guarded choice* cannot be simulated in the choice $(+)$ -free pi-calculus in a fully abstract way, while preserving any 'reasonable' semantics. The reason is that it is not possible to atomically perform, in the absence of $+$, an external synchronisation and an internal exclusive choice among a number of alternatives. We prove that in D-Fusion, under mild typing assumptions, guarded choice can actually be simulated in a fully abstract way in the choice-free fragment. The encoding preserves a reasonable semantics, defined in terms of barbed equivalence ([11]). Informally, branches of a choice are represented as concurrent processes. Synchronisation is performed in the ordinary way, but it forces a fusion between a λ-name global to all branches and a ν-name local to the chosen branch. Excluded branches are atomically inhibited, since any progress would lead them to fusing two distinct ν-names.

In the present paper, we are mainly interested in assessing the expressive power of D-Fusion compared to other calculi. The principal tool for this study will be barbed bisimilarity and the induced equivalences, as they enjoy a uniform definition based only on a reduction relation, on an observation predicate and on context-closure. We defer the study of alternative, more tractable semantics for D-Fusion, like a form of 'labelled' bisimulation, to a forthcoming work.

The rest of the paper is organised as follows. Section 2 contains a proof that the pi-calculus cannot be uniformly encoded into Fusion. In Section 3 we introduce the D-Fusion calculus, its operational semantics and barbed congruence. In Section 4 we show that D-Fusion calculus is strictly more expressive than both pi-calculus and Fusion. We further explore this expressiveness gap in Section 5, by means of an example concerning a security protocol, and in Section 6, by encoding mixed guarded choice into the choice-free calculus. Section 7 contains a brief overview of some related work and a few concluding remarks.

2 Fusion and Pi

The aim of this section is to illustrate the difference between pi-calculus and Fusion, and to show that the former cannot be uniformly encoded in the latter.

The crucial difference between the pi-calculus and Fusion shows up in synchronisations: in Fusion, the effect of a synchronisation is not necessarily local, and is regulated by the scope of the binder (x). For example, an interaction between $\overline{u}v.P$ and $ux.Q$ will result in a fusion of v and x. This fusion will also affect any further process R running in parallel, as illustrated by the example below:

$$R\,|\,\overline{u}v.P\,|\,ux.Q \xrightarrow{\{x=v\}} R\,|\,P\,|\,Q.$$

The binding operator (x) can be used to limit the scope of the fusion, e.g.:

$$R\,|\,(x)\,\big(\overline{u}v.P\,|\,ux.Q\big) \xrightarrow{\tau} R\,|\,(P\,|\,Q)[{}^{v}\!/\!_{x}].$$

where τ denotes the identity fusion. For a full treatment of pi-calculus and Fusion we refer to [10] and to [16], respectively.

Below, we show that there is no 'reasonably simple' encoding of the pi-calculus Π into Fusion $\mathcal{F}$. We focus on encodings $[\![\cdot]\!]$ that have certain compositional properties and preserve (a subset of) weak traces. As to the latter, we shall only require that moves of P are reflected in $[\![P]\!]$, not vice-versa. We also implicitly require that the encoding preserves arity of I/O actions (the length of tuples carried on each channel). This is sometimes not the case for process calculi encodings; however, it is easy to relax this condition by allowing names of $[\![P]\!]$ to carry *longer* tuples. We stick to the simpler correspondence just for notational convenience. Note that the encoding presented in [16] does satisfy our criterion. We shall assume here the standard pi-calculus late operational semantics [10] and, for the purpose of our comparison, we shall identify the late input pi-action $a(\widetilde{x})$ with the Fusion input action $(\widetilde{x})\,a\widetilde{x}$.

Definition 1. *A translation* $[\![\cdot]\!] : \Pi \longrightarrow \mathcal{F}$ *is* uniform *iff for each* $P, Q \in \Pi$ *it holds that:*

- *for each trace of actions s not containing bound outputs,* $P \overset{s}{\Longrightarrow}$ *implies* $[\![P]\!] \overset{s}{\Longrightarrow}$;
- $[\![P|Q]\!] = [\![P]\!]\,|\,[\![Q]\!]$;
- *for each y,* $[\![(\nu y)\,P]\!] = (y)[\![P]\!]$;
- *for each substitution* σ, $[\![P\sigma]\!] = [\![P]\!]\sigma$.

Note that the above notion of uniform encoding is stronger than the one introduced in [14].

The next proposition generalises an example from [16]. Below, we fix an arbitrary Π-equivalence included in trace equivalence, $\sim_{\Pi}$, and an arbitrary $\mathcal{F}$-equivalence which is included in trace equivalence *and* is preserved by parallel composition, $\sim_{\mathcal{F}}$ (like, e.g., hyperequivalence of [16]).

Proposition 1. *There is no uniform translation* $[\![\cdot]\!] : \Pi \longrightarrow \mathcal{F}$ *such that for each* $P, Q \in \Pi$*:*

$$P \sim_{\Pi} Q \textit{ implies } [\![P]\!] \sim_{\mathcal{F}} [\![Q]\!]\,.$$

PROOF: Suppose that there exists such a translation $[\![\cdot]\!]$. Let P and Q be the following two pi-agents:

$$P \;=\; (\nu u,v)\;(\overline{a}\langle u,v\rangle\,|\,\overline{u}|v.\overline{w}) \qquad Q \;=\; (\nu u,v)\;\big(\overline{a}\langle u,v\rangle\,|\,(\overline{u}.(v.\overline{w})+v.(\overline{u}|\overline{w})\,)\big).$$

Obviously, $P \sim_{\Pi} Q$ (e.g. they are strongly late bisimilar). Suppose $[\![P]\!] \sim_{\mathcal{F}} [\![Q]\!]$. Let $R = a(x,y).(\overline{c}x|cy)$ and A and B be as follows:

$$\begin{aligned} A &= [\![P]\!]|R = (u,v)([\![\overline{a}\langle u,v\rangle]\!]\,|\,[\![\overline{u}]\!]\,|\,[\![v.\overline{w}]\!])\,|\,R \\ B &= [\![Q]\!]|R = (u,v)([\![\overline{a}\langle u,v\rangle]\!]\,|\,[\![\overline{u}.(v.\overline{w})+v.(\overline{u}|\overline{w})]\!])\,|\,R. \end{aligned}$$

Since $\sim_{\mathcal{F}}$ is preserved by $|$, A and B are $\sim_{\mathcal{F}}$-equivalent. By uniformity of the encoding, it is easy to check that $A \overset{\overline{w}}{\Longrightarrow}$. On the other hand, a careful case analysis shows that $B \overset{\overline{w}}{\not\Longrightarrow}$. This is a contradiction. □

3 The Distinctive Fusion Calculus, D-Fusion

Syntax. We consider a countable set of names $\mathcal{N}$ ranged over by $a,b,\ldots,u,v,\ldots,z$. We write $\tilde{x}$ for a finite tuple $x_1,\ldots,x_n$ of names. The set $\mathcal{DF}$ of D-Fusion *processes*, ranged over by $P,Q,\ldots$, is defined by the syntax:

$$P ::= \mathbf{0} \;\big|\; \alpha.P \;\big|\; P|P \;\big|\; P+P \;\big|\; [x=y]P \;\big|\; !P \;\big|\; \lambda x P \;\big|\; (\nu x)\,P$$

where *prefixes* α are defined as $\alpha ::= a\tilde{v}$. The occurrences of x in $\lambda x P$ and $(\nu x)\,P$ are *bound*, thus notions of *free names* and *bound names* of a process P arise as expected and are denoted by $\mathrm{fn}(P)$ and $\mathrm{bn}(P)$, respectively. The notion of *alpha-equivalence* also arises as expected. In the rest of the paper we will identify alpha-equivalent processes. A *context* $C[\cdot]$ is a process with a hole that can be filled with any process P, thus yielding a process $C[P]$.

Note that we consider one kind of prefix, thus ignoring polarities. However, a subcalculus with polarities can be easily retrieved, as shown at the end of this section.

The main difference from Fusion is the presence of two distinct binding constructs, λ and ν. The λ-abstraction operator corresponds to the binding construct of Fusion and generalises input binding of the pi-calculus. The restriction operator (ν) corresponds to the analogous operator of the pi-calculus: it allows a process to create a fresh, new name that will be kept distinct from other names.

Definition 2 (Structural Congruence). *The structural congruence $\equiv$ is the least congruence on processes satisfying the abelian monoid laws for Summation and Composition (associativity, commutativity and $\mathbf{0}$ as identity), the scope laws*

$$\begin{array}{ccc} (\nu x)\,\mathbf{0} \equiv \mathbf{0} & (\nu x)\,(\nu y)\,P \equiv (\nu y)\,(\nu x)\,P & (\nu x)\,(P+Q) \equiv (\nu x)\,P \,+\, (\nu x)\,Q \\ \lambda x\,\mathbf{0} \equiv \mathbf{0} & \lambda x \lambda y P \equiv \lambda y \lambda x P & \lambda x\,(P+Q) \equiv \lambda x P \,+\, \lambda x Q \end{array}$$

plus the scope extrusion *laws*

$$(\nu x)\,(P|Q) \equiv P|(\nu x)\,Q \textit{ where } x \notin \mathrm{fn}(P) \qquad \lambda x\,(P|Q) \equiv P|\lambda x Q \textit{ where } x \notin \mathrm{fn}(P)$$

and the swapping *law*

$$\lambda x\,(\nu y)\,P \;\equiv\; (\nu y)\,\lambda x P \;\; \textit{where} \;\; x \neq y.$$

Operational Semantics. For R a binary relation over $\mathcal{N}$, let $R^\star$ denote the reflexive, symmetric and transitive closure of R with respect to $\mathcal{N}$. We use σ, σ' to range over substitutions, i.e. finite partial functions from $\mathcal{N}$ onto $\mathcal{N}$. Domain and co-domain of σ, denoted $\mathrm{dom}(\sigma)$, $\mathrm{cod}(\sigma)$ are defined as expected. We denote by $t\sigma$ the result of applying σ onto a term t. Given a set/tuple of names $\tilde{x}$, we define $\sigma_{|\tilde{x}}$ as $\sigma \cap (\tilde{x} \times \mathcal{N})$, and $\sigma_{-\tilde{x}}$ as $\sigma - (\tilde{x} \times \mathcal{N})$.

Below, we define fusions, that is, name equivalences that arise as the result of equating two lists of names in a synchronisation.

Definition 3 (Fusions). *We let $\phi,\chi,\ldots$ range over* fusions, *that is total equivalence relations on $\mathcal{N}$ with only finitely many non-singleton equivalence classes. We let:*

- $\mathrm{n}(\phi)$ *denote* $\{x : x\phi y$ *for some* $y \neq x\}$*;*
- τ *denote the identity fusion (i.e.,* $\mathrm{n}(\tau) = \emptyset$*);*
- ϕ_{-z} *denote* $(\phi - (\{z\} \times \mathcal{N} \cup \mathcal{N} \times \{z\}))^{\star}$*;*
- $\{x = y\}$ *denote* $\{(x,y)\}^{\star}$*;*
- $\phi[x]$ *denote the equivalence class of* x *in* ϕ*.*

We now introduce a labelled transition system for D-Fusion. This will be useful in order to compare D-Fusion with other calculi and, in particular, to prove that D-Fusion cannot be encoded into pi-calculus (see Section 4). The reduction relation coincides with the identity fusion transition $\xrightarrow{\tau}$ of the labelled transition system.

Definition 4 (Labelled Transition System). *The transition relation* $P \xrightarrow{\mu} Q$*, for* μ *a label of the form* $(\nu\widetilde{x})\,\lambda\widetilde{y}\,a\widetilde{v}$ (action) *or of the form* $(\nu\widetilde{x})\,\phi$ (effect)*, is defined in Table 1.*

Some notations for actions and effects. The bound names of μ are written $\mathrm{bn}(\mu)$ and are defined as expected; when $\mu = (\nu\widetilde{x})\,\lambda\widetilde{y}\,a\widetilde{v}$ is an action we let $\mathrm{subj}(\mu) = a$ and $\mathrm{obj}(\mu) = \widetilde{v}$ denote the subject and object part of μ, otherwise they both denote a conventional value '$-$'. Moreover, $\mathrm{n}(\mu)$ denotes all names in μ. We use abbreviations such as $\mathrm{n}(\phi,\mu)$ to denote $\mathrm{n}(\phi) \cup \mathrm{n}(\mu)$ and $(\nu z)\,\mu$ for $(\nu\widetilde{x}z)\,\phi$, if $\mu = (\nu\widetilde{x})\,\phi$. Furthermore, we shall identify actions and effects up to reordering of the tuple $\widetilde{x}$ in $(\nu\widetilde{x})$ and $\lambda\widetilde{x}$.

The rules in Table 1 deserve some explanation. As mentioned, we have two kinds of labels, actions and effects. Apart from the absence of polarities, *actions* are governed by rules similar to those found in pi-calculus. The main difference is that on the same action one can find both ν- and λ-extruded names. On the other hand, *effects* are similar to those found in Fusion: an effect of the form ϕ can be created as a result of a communication (rule COM), and be propagated across parallel components, until a λ that binds a fused name z is encountered (rule λ-OPEN$_f$). At that point, a substitutive effect $[{}^{w}\!/z]$ is applied onto the target process and z is discarded from the fusion (the result is ϕ_{-z}). A major difference with respect to Fusion is that our effects can also ν-extrude names. The side condition '$\phi[z] \cap \widetilde{x} = \emptyset$' in rule ν-OPEN prevents effects from equating two distinct ν-extruded names. Note that the premises of rules COM and λ-OPEN only deal with binder-free actions and effects, while λ-OPEN$_a$ only deals with ν-binder-free actions. However, structural congruence permits freely moving λ's and ν's, so the general case is covered via STRUCT.

Let us illustrate the rules with some examples. We shall write $\{\widetilde{x} = \widetilde{y}\}.P$ for $(\nu c)\,(c\widetilde{x} | c\widetilde{y}.P)$ (for a fresh name c).

Example 1.

1. Let $P = (\nu c)\,((\nu x)\,cx.P_1 \,|\, cy.P_2)$. The interaction between $cx.P_1$ and $cy.P_2$ will result into a fusion $\{x = y\}$, that causes x to be extruded:

$$P \equiv (\nu x)\,(\nu c)\,(cx.P_1 \,|\, cy.P_2) \xrightarrow{(\nu x)\,\{x=y\}} (\nu c)\,(P_1 \,|\, P_2)\,.$$

Now consider $Q = \lambda y\,P$. The effect of λ-abstracting y in P is that of removing ϕ and getting the substitution $[{}^{x}\!/y]$ applied onto the continuation:

$$Q \xrightarrow{\tau} (\nu x)\,((\nu c)\,(P_1 \,|\, P_2)[{}^{x}\!/y])\,.$$

Table 1. Actions and effects transitions in D-Fusion

$$\text{(Pref)}\quad \alpha.P \xrightarrow{\alpha} P \qquad\qquad \text{(Sum)}\quad \frac{P_1 \xrightarrow{\mu} Q}{P_1 + P_2 \xrightarrow{\mu} Q}$$

$$\text{(Com)}\quad \frac{P_1 \xrightarrow{a\widetilde{u}} Q_1 \quad P_2 \xrightarrow{a\widetilde{v}} Q_2}{P_1|P_2 \xrightarrow{\{\widetilde{v}=\widetilde{u}\}} Q_1|Q_2}\quad |\widetilde{u}| = |\widetilde{v}| \qquad\qquad \text{(Par)}\quad \frac{P \xrightarrow{\mu} Q}{P|R \xrightarrow{\mu} Q|R}$$

$$(\nu\text{-Pass})\quad \frac{P \xrightarrow{\mu} Q}{(\nu z)\,P \xrightarrow{\mu} (\nu z)\,Q}\quad z \notin \mathrm{n}(\mu) \qquad\qquad (\lambda\text{-Pass})\quad \frac{P \xrightarrow{\mu} Q}{\lambda z P \xrightarrow{\mu} \lambda z Q}\quad z \notin \mathrm{n}(\mu)$$

$$(\nu\text{-Open})\quad \frac{P \xrightarrow{\mu} Q}{(\nu z)\,P \xrightarrow{(\nu z)\mu} Q}\quad \begin{cases} z \in \mathrm{n}(\mu) \\ \mu \text{ an action implies } z \neq \mathrm{subj}(\mu) \\ \mu = (\nu\widetilde{x})\,\phi \text{ implies } \phi[z] \cap \widetilde{x} = \emptyset \end{cases}$$

$$(\lambda\text{-Open}_a)\quad \frac{P \xrightarrow{\lambda\widetilde{y}a\widetilde{v}} Q}{\lambda z P \xrightarrow{\lambda\widetilde{y}za\widetilde{v}} Q}\quad z \in \widetilde{v} - (\{a\} \cup \widetilde{y}) \qquad\qquad (\lambda\text{-Open}_f)\quad \frac{P \xrightarrow{\phi} Q}{\lambda z P \xrightarrow{\phi_{-z}} Q[{}^{w}/z]}\quad z\,\phi\,w,\ w \neq z$$

$$\text{(Match)}\quad \frac{P \xrightarrow{\mu} Q}{[a=a]P \xrightarrow{\mu} Q} \qquad\qquad \text{(Struct)}\quad \frac{P_1 \equiv P \quad P \xrightarrow{\mu} Q \quad Q \equiv Q_1}{P_1 \xrightarrow{\mu} Q_1}$$

Symmetric rules for (Sum) and (Par) are not shown. Usual conventions about freshness of bound names apply.

2. Another example illustrating interplay between ν-bound, λ-bound and free names:

$$\lambda z\,(\nu w)\,\{z, a = w, b\}.P \xrightarrow{\{a=b\}} (\nu w)\,P[{}^{w}/z], \quad \text{but} \quad \lambda z\,(\nu w)\,\{z, z = w, b\}.P \xrightarrow{(\nu w)\,\{w=b\}} P[{}^{w}/z].$$

Encoding I/O Polarities. We can encode input and output polarities as follows:

$$\overline{c}\langle\widetilde{v}\rangle.P \stackrel{\triangle}{=} (\nu x)\,\lambda y\, c\widetilde{v}xy.P \qquad c\langle\widetilde{v}\rangle.P \stackrel{\triangle}{=} (\nu x)\,\lambda y\, c\widetilde{v}yx.P$$

for some chosen fresh x and y. The position of the ν-name x forbids fusions between actions with the same polarity and, hence, communication. For instance, the process $\overline{c}\langle\widetilde{v}\rangle.P|\overline{c}\langle\widetilde{u}\rangle.Q$ has no τ-transition, since the latter would force the fusion of two distinct ν-names, which is forbidden by the operational rules. We denote by $\mathcal{DF}^{\mathrm{p}}$, *polarised*

D-Fusion, the subset of $\mathcal{DF}$ in which every prefix can be interpreted as an input or output, in the above sense.

Barbed Congruence. We now define our main tools for assessing the expressive power of D-Fusion compared to other calculi, that is barbed bisimulation and barbed congruence.

Definition 5 (Barbs). *We write $P \downarrow a$ if and only if there exist an action μ and a process Q such that $P \xrightarrow{\mu} Q$ and* $\mathrm{subj}(\mu) = a$.

Definition 6 (Barbed Bisimulation). *A* barbed bisimulation *is a symmetric binary relation $\mathcal{R}$ between processes such that $P\,\mathcal{R}\,Q$ implies:*

1. *whenever $P \xrightarrow{\tau} P'$ then $Q \xrightarrow{\tau} Q'$ and $P'\,\mathcal{R}\,Q'$;*
2. *for each name a, if $P \downarrow a$ then $Q \downarrow a$.*

P is a barbed bisimilar to Q, written $P \dot{\sim} Q$, if $P\mathcal{R}Q$ for some barbed bisimulation $\mathcal{R}$.

Definition 7 (Barbed Congruence). *Two processes P and Q are* barbed congruent, *written $P \sim Q$, if for all contexts $C[\cdot]$, it holds that $C[P] \dot{\sim} C[Q]$.*

Example 2. 1. An example of 'expansion' for parallel composition is as follows:

$$(\nu k)\, ak.\mathbf{0}|av.\mathbf{0} \;\sim\; (\nu k)\, ak.av.\mathbf{0} \;+\; av.(\nu k)\, ak.\mathbf{0} \;+\; (\nu k)\,\{k = v\}.\mathbf{0}$$

On the other side,

$$((\nu k)\, ak.ak.\mathbf{0})|av.\mathbf{0} \;\not\sim\; (\nu k)\, ak.(ak.av.\mathbf{0} + av.ak.\mathbf{0}) \;+\; av.((\nu k)\, ak.ak.\mathbf{0}) \\ +\; (\nu k)\,\{k = \nu\}.ak.\mathbf{0},$$

since the two processes are not barbed bisimilar within a context $C[\cdot] = (\lambda x, v)([\cdot]\,|\,ax.\mathbf{0})$.

2. The following two examples show the effect of fusing a λ-abstracted name with a free name and with another λ-abstracted name, respectively:

$$\lambda v\,\{k = v\}.P \;\sim\; \tau.P[^k\!/\!_v] \qquad (\lambda k, v)\;\{k = v\}.P \;\sim\; \lambda k\,\tau.P[^k\!/\!_v].$$

On the contrary,

$$(\nu v)\,\{k = v\}.P \;\not\sim\; \tau.P[^k\!/\!_v],$$

since the two processes are not barbed bisimilar within a context $C[\cdot] = (\nu k)\,[\cdot]$.

4 Expressiveness of D-Fusion

Pi-calculus and Fusion are subcalculi of D-Fusion, because their respective labelled transition systems are embedded into polarised D-Fusion's, under the two obvious uniform translations from Π and $\mathcal{F}$ to $\mathcal{DF}^{\mathrm{p}}$ given below. The definition of uniformity can be extended to the case of encodings from $\mathcal{F}/\Pi$ into $\mathcal{DF}^{\mathrm{p}}$ in the obvious way: in particular, by requiring that (x) and (νx) be mapped to λx and (νx), respectively.

Definition 8. *The translations* $[\![\cdot]\!]_\pi : \Pi \to \mathcal{DF}^{\mathrm{p}}$ *and* $[\![\cdot]\!]_f : \mathcal{F} \to \mathcal{DF}^{\mathrm{p}}$ *are defined by extending in the expected homomorphic way the following clauses, respectively:*

$$[\![\overline{a}\langle x\rangle.P]\!]_\pi = \overline{a}\langle x\rangle.[\![P]\!]_\pi \quad [\![a(x).P]\!]_\pi = \lambda x\, a\langle x\rangle.[\![P]\!]_\pi \quad [\![(\nu x)\,P]\!]_\pi = (\nu x)\;[\![P]\!]_\pi$$
$$[\![\overline{a}\langle x\rangle.P]\!]_f = \overline{a}\langle x\rangle.[\![P]\!]_f \quad [\![a\langle x\rangle.P]\!]_f = a\langle x\rangle.[\![P]\!]_f \quad [\![(x)P]\!]_f = \lambda x\;[\![P]\!]_f$$

As expected, inclusion in term of labelled transition systems naturally lifts to bisimulation equivalences. Let $\sim^\pi$ and $\sim^{\mathrm{f}}$ denote barbed congruence, respectively, over Π ([18]) and over $\mathcal{F}$ (see [21]) (Note that, over image-finite processes, $\sim^\pi$ is pi-calculus early congruence [18], and $\sim^{\mathrm{f}}$ is Fusion hyper-equivalence [21]). Also, let $\sim^{[\![\pi]\!]}$ and $\sim^{[\![\mathrm{f}]\!]}$ be the equivalences on $\mathcal{DF}^{\mathrm{p}}$ obtained by closing barbed bisimulation $\dot{\sim}$ only under translated pi- and Fusion-contexts, respectively (e.g., $P \sim^{[\![\pi]\!]} Q$ iff for each Π-context $C[\cdot]$, $[\![C]\!]_\pi[P] \mathrel{\dot{\sim}} [\![C]\!]_\pi[Q]$).

Proposition 2.

1. *Let P and Q be two pi-calculus processes.* $P \sim^\pi Q$ *iff* $[\![P]\!]_\pi \sim^{[\![\pi]\!]} [\![Q]\!]_\pi$.
2. *Let P and Q be two Fusion processes.* $P \sim^{\mathrm{f}} Q$ *iff* $[\![P]\!]_f \sim^{[\![\mathrm{f}]\!]} [\![Q]\!]_f$.

More interesting, we now show that D-Fusion cannot be uniformly encoded into Π. The intuitive reason is that, in D-Fusion, the combined use of action prefix, fusions and restrictions allows one to express a form of pattern matching. This is not possible in Π, at least not atomically. To show this fact, we restrict our attention to polarised D-Fusion, $\mathcal{DF}^{\mathrm{p}}$. The reference semantics for Π is again the late operational semantics.

Given $P \in \Pi$ and a trace of actions s, let us write $P \overset{\hat{s}}{\Longrightarrow}$ if $P \overset{s'}{\Longrightarrow}$ for some trace s' that exhibits the same sequence of subject names, with the same polarity, as s (e.g., $s = a\langle\widetilde{x}\rangle \cdot \lambda\widetilde{y}\overline{b}\langle\widetilde{v}\rangle$ and $s' = a\langle\widetilde{z}\rangle \cdot \overline{b}\langle\widetilde{w}\rangle$).

Definition 9. *A translation* $[\![\cdot]\!] : \mathcal{DF}^{\mathrm{p}} \to \Pi$ *is* uniform *if for each* $P, Q \in \mathcal{DF}^{\mathrm{p}}$*:*

- *for each trace s,* $P \overset{s}{\Longrightarrow}$ *implies* $[\![P]\!] \overset{\hat{s}}{\Longrightarrow}$;
- $[\![P|Q]\!] = [\![P]\!]\,|\,[\![Q]\!]$;
- *for each y,* $[\![(\nu y)\,P]\!] = (\nu y)\,[\![P]\!]$;
- *for each substitution* σ, $[\![P\sigma]\!] = [\![P]\!]\sigma$.

Below, we denote by $\sim_{\mathcal{DF}^{\mathrm{p}}}$ any fixed equivalence over $\mathcal{DF}^{\mathrm{p}}$ which is contained in trace semantics (defined in the obvious way), and by $\sim_\Pi$ any fixed equivalence over Π which is contained in trace equivalence. Note that barbed congruence over $\mathcal{DF}^{\mathrm{p}}$, $\sim$, is contained in trace equivalence.

Proposition 3. *There is no uniform translation* $[\![\cdot]\!] : \mathcal{DF}^{\mathrm{p}} \to \Pi$ *such that* $\forall P, Q \in \mathcal{DF}^{\mathrm{p}}$*:*

$$P \sim_{\mathcal{DF}^{\mathrm{p}}} Q \Rightarrow [\![P]\!] \sim_\Pi [\![Q]\!].$$

PROOF: Suppose that there exists such a translation $[\![\cdot]\!]$. Let us consider the following two $\mathcal{DF}^{\mathrm{p}}$-processes P and Q:

$$P = (\nu c,k,h)\,(c\langle k\rangle.\overline{a}.\mathbf{0}|c\langle h\rangle.\overline{b}.\mathbf{0}|\overline{c}\langle k\rangle.\mathbf{0}) \qquad Q = \tau.\overline{a}.\mathbf{0}.$$

It holds that $P \sim Q$ in $\mathcal{DF}^{\mathrm{p}}$: the reason is that, in P, synchronisation between prefixes $c\langle h\rangle$ and $\overline{c}\langle k\rangle$, which carry different *restricted* names h and k, is forbidden (see rule ν-OPEN). Thus P can only make $c\langle k\rangle$ and $\overline{c}\langle k\rangle$ synchronise, and then perform $\overline{a}$. Thus, $P \sim_{\mathcal{DF}^{\mathrm{p}}} Q$ holds too.

On the other hand, by Definition 9, for any uniform encoding $[\![\cdot]\!]$, c and $\bar{c}$ in $[\![P]\!]$ can synchronise and, thus, $[\![P]\!] \xRightarrow{\overline{b}}$, while $[\![Q]\!] \not\xRightarrow{\overline{b}}$ (because of $b \notin \mathrm{fn}(Q)$ and of the uniformity with respect to substitutions). Thus $[\![P]\!] \not\sim_{\Pi} [\![Q]\!]$. □

Of course, it is also true that D-Fusion cannot be uniformly encoded into $\mathcal{F}$, as this would imply the existence of a uniform fully abstract encoding from Π to $\mathcal{F}$, which does not exist (Proposition 1).

The conclusion is that there is some expressiveness gap between D-Fusion on one side and the other two calculi on the other side, at least, as far as our simple notion of uniform encoding is concerned. This gap is further explored by means of more elaborate examples in the next two sections.

5 Example: Correlation

This example aims at illustrating the gap between D-Fusion and Fusion from a more concrete perspective. Consider the following simple protocol. An agent A asks a trusted server S for two keys, to be used to access two distinct services (e.g. A might be a proxy requiring remote connections on behalf of two different users). Communication between A and S takes place over an insecure public channel, controlled by an adversary, but it is protected by encryption and challenge-response nonces. Informally, the dialogue between A and S is as follows:

$$\begin{array}{l} 1.\ A \to S : n \\ 2.\ S \to A : \{n,k\}_{k_S} \\ 1'.\ A \to S : n' \\ 2'.\ S \to A : \{n',k'\}_{k_S} \end{array}$$

Here $\{\cdot\}_{(\cdot)}$ is symmetric encryption and k_S is a secret master key shared by A and S. A simple property of this protocol is that A should never receive k and k' in the wrong order (k' and then k), even in case S accepts new requests before completing old ones. Indeed, nonces n and n' are intended to avoid confusion of distinct sessions. In other words, nonces do *correlate* each request to S with the appropriate reply of S.

Below, we show that the above small protocol and the related ordering property can be readily translated and verified in D-Fusion. Next, we show that the property breaks down when (uniformly) translating the protocol into Fusion.

D-Fusion. Encryption is not a primitive operation in D-Fusion. However, in the present case, it is sensible to model an encrypted message $\{n,k\}_{k_S}$ as an output action $\overline{k_S}\langle n,k\rangle$: only knowing the master key k_S, and further specifying a session-specific nonce, it is possible to acquire the key k (similarly for $\{n',k'\}_{k_S}$, of course). Thus, assuming A concludes the protocol with a conventional 'commit' action and that p is the public channel, A, S and the whole protocol P might be specified as follows (below, we abbreviate $\lambda\widetilde{x}\,p\langle\widetilde{x}\rangle.X$ as $p(\widetilde{x}).X$):

$$\begin{aligned}
&A=(\nu n)\,\big(\overline{p}\langle n\rangle.\mathbf{0}|\lambda y\,k_S\langle n,y\rangle.(\nu n')\,(\overline{p}\langle n'\rangle.\mathbf{0}|\lambda y'\,k_S\langle n',y'\rangle.\overline{commit}\langle y,y'\rangle.\mathbf{0})\big)\\
&S=p(x).\big(\overline{k_S}\langle x,k\rangle.\mathbf{0}|p(x').\overline{k_S}\langle x',k'\rangle.\mathbf{0}\big)\\
&P=(\nu k_S)\,(A|S).
\end{aligned}$$

Let A_{spec} be the process defined like A, except that the $\overline{commit}\langle y,y'\rangle$ action is replaced by $\overline{commit}\langle k,k'\rangle$, and let $P_{\text{spec}}=(\nu k_S)\,(A_{\text{spec}}|S)$. The property that A should never receive k and k' in the wrong order is stated as: $P\sim P_{\text{spec}}$.

Informally, equivalence holds true because the second input action in A/A_{spec}, that is $\lambda y'\,k_S\langle n',y'\rangle$, can only get synchronised with the second output action in S, that is $\overline{k_S}\langle x',k'\rangle$. In fact, n' can be extruded only *after* x has been received, hence fusion of x and n' is forbidden. Note that the above protocol specification would not be easily translated in pi-calculus, because in A the input prefix $k_S\langle n,y\rangle$ has a ν-bound name n.

Fusion. Suppose P^f and P^f_{spec} are obtained by some uniform encoding of P and P_{spec} above into Fusion. It is not difficult to show that P^f can be 'attacked' by an adversary R that gets n and n' and fuse them together, $R=p(x).\big(\overline{p}\langle x\rangle.\mathbf{0}|p(y).\overline{p}\langle y\rangle.\mathbf{0}\big)$. Formally, for $\alpha=\overline{commit}\langle k',k\rangle$,

$$P^f|R\overset{\alpha}{\Longrightarrow}\text{ and, thus, }P^f|R\not\sim^{\text{he}}P^f_{\text{spec}}|R,$$

which proves that P^f and P^f_{spec} are not hyper-equivalent.

This example illustrates the difficulty of modelling fresh, indistinguishable quantities (nonces) in Fusion. This makes apparent that Fusion is not apt to express security properties based on correlation.

6 Encoding Guarded Choice

In this section we show how the combined mechanisms of fusions and restrictions can be used to encode different forms of guarded choice *via* parallel composition, in a clean and uniform way. Informally, different branches of a guarded choice will be represented as concurrent processes. The encodings add pairs of extra names to the object part of each action: these extra names are used as 'side-channels' for atomic coordination among the different branches. We start by looking at a simple example.

Example 3. Consider the guarded choice $A=\lambda x\,(\nu n)\,a\langle xn\rangle.P+\lambda x\,(\nu m)\,a\langle xm\rangle.Q$. Its intended 'parallel' implementation is the process:

$$B=\lambda x\Big(\,(\nu n)\,a\langle xn\rangle.P\,|\,(\nu m)\,a\langle xm\rangle.Q\,\Big)$$

(here, $x,n,m\notin\text{fn}(a,P,Q)$). Assume a channel discipline by which output actions on channel a must carry two identical names. In B, the parallel component that first consumes any such message, forces fusion of x either to n or to m, and consequently inhibits the other component. E.g.:

$$\lambda u\overline{a}\langle uu\rangle|B\xrightarrow{\tau}\sim\ (\nu n)\,(P\,|\,(\nu m)\,a\langle mn\rangle.Q)\quad\sim\quad P\,|(\nu n,m)\,a\langle mn\rangle.Q.$$

Under the mentioned assumption, $(\nu m,n)\,a\langle mn\rangle.Q$ is equivalent to $\mathbf{0}$, because there is no way of fusing m and n. Thus the process on the right of $\sim$ is equivalent to P. In other words, choice between P and Q has been resolved atomically.

The above line of reasoning can be formalised in two ways. One way is considering a new 'disciplined' equivalence $\sim^{\mathrm{d}}$, obtained by closing barbed bisimilarity only with respect to contexts $C[\cdot]$ obeying the mentioned channel discipline (i.e. for each $\overline{c}\langle\widetilde{v}\rangle$ in $C[\cdot]$ with $|\widetilde{v}| \geq 2$, $\widetilde{v} = \widetilde{w}uu$, for some $\widetilde{w}$ and u). The other way is keeping standard barbed congruence $\sim$, but inserting processes inside a 'firewall' that filters out $\overline{a}$-messages not respecting the given channel discipline. The latter can be easily defined in D-Fusion relying on 'non-linear' inputs:

$$\mathsf{F}_{a,a'}[\cdot] = (\nu a')\left(\lambda z\, azz.\overline{a'}\langle zz\rangle.\mathbf{0}|[\cdot]\right).$$

We state the result in both forms below.

Proposition 4. *Let A and B be as in Example 3.*

1. *$A \sim^{\mathrm{d}} B$;*
2. *Let A' and B' be the processes obtained from A and B, respectively, by replacing the outermost occurrences of a with a fresh a'. Then $\mathsf{F}_{a,a'}[A'] \sim \mathsf{F}_{a,a'}[B']$.*

Note that the result above exploits in a crucial way features of both Fusion (non-linearity of input actions, in the firewall, and sharing of input variable x, in B) and of D-Fusion (restricted input).

Proposition 4 can be generalised to fully abstract encodings of different forms of guarded choice. For the sake of simplicity, we will state the results in terms of 'disciplined' equivalences. We believe the results can also be stated in terms of ordinary barbed congruence, at the cost of breaking uniformity of the encoding and of introducing more sophisticated forms of 'firewalls'. We examine two cases, input-guarded choice and mixed choice.

Input-Guarded (ig) Choice. Let us fix, as a source language the fragment of polarised D-Fusion with guarded choice, $\mathcal{DF}^{\mathrm{p,ig}}$. In this language, input prefix and summation $+$ are replaced by input-guarded choice $\sum_{i\in i} a_i\langle\widetilde{x}_i\rangle.P_i$. The target language is the fragment of polarised D-Fusion with no form of summation. The relevant clauses of the encoding are:

$$[\![\sum_{i\in i} a_i\langle\widetilde{x}_i\rangle.P_i]\!]_{\mathrm{ig}} = \lambda z\,\Pi_{i\in I}(\nu n)\,\lambda\widetilde{x}_i\, a_i\langle\widetilde{x}_i zn\rangle.[\![P_i]\!]_{\mathrm{ig}} \qquad [\![\overline{a}\langle\widetilde{v}\rangle.P]\!]_{\mathrm{ig}} = \lambda z\,\overline{a}\langle\widetilde{v}zz\rangle.[\![P]\!]_{\mathrm{ig}}\,,$$

where $\Pi_{i\in I}X_i$ denotes the parallel composition of all X_i's. The encoding acts as a homomorphism over the remaining operators. Below, we denote by $\sim^{\mathrm{p,ig}}$ barbed congruence over $\mathcal{DF}^{\mathrm{p,ig}}$, and denote by $\sim^{[\![\mathrm{p,ig}]\!]}$ the equivalence over D-Fusion obtained by closing barbed bisimulation under translated contexts (i.e. $P \sim^{[\![\mathrm{p,ig}]\!]} Q$ iff for each $\mathcal{DF}^{\mathrm{p,ig}}$-context $C[\cdot]$, it holds $[\![C]\!]_{\mathrm{ig}}[P] \overset{\cdot}{\sim} [\![C]\!]_{\mathrm{ig}}[Q]$); note that both equivalences are *reasonable* semantics in the sense of [13]. The proof of the following theorem is straightforward, given that there is a 1-to-1 correspondence between reductions and barbs of R and of $[\![R]\!]_{\mathrm{ig}}$, for any R, and given that the encoding is compositional, in particular, for any context $C[\cdot]$, it holds $[\![C]\!]_{\mathrm{ig}}[[\![P]\!]_{\mathrm{ig}}] = [\![C[P]]\!]_{\mathrm{ig}}$.

Theorem 1 (Full Abstraction for ig Choice). *Let $P, Q \in \mathcal{DF}^{\mathrm{p,ig}}$. It holds that $P \sim^{\mathrm{p,ig}} Q$ if and only if $[\![P]\!]_{\mathrm{ig}} \sim^{[\![\mathrm{p,ig}]\!]} [\![Q]\!]_{\mathrm{ig}}$.*

Of course, the above theorem also yields a fully abstract encoding of input-guarded choice for pi-calculus, which may be viewed as a sub-calculus of $\mathcal{DF}^{\mathrm{p,ig}}$.

Mixed Choice in a Sorted pi-Calculus. As a source language we fix here a sorted version of polyadic pi-calculus [9] with 'mixed' choice, Π^{mix}. In this language, prefixes and $+$ are replaced by mixed summation, $\sum_{i\in I} a_i\,(\widetilde{x}_i).P_i + \sum_{j\in J} \overline{b_j}\langle\widetilde{v}_j\rangle.Q_j$. The target language is again the fragment of polarised D-Fusion with no summation at all. The encoding is a bit more complex than in the previous case, as it implies adding *two* pairs of extra names to coordinate different branches. The relevant clause is:

$$\begin{array}{l} [\![\sum_{i\in I} a_i\,(\widetilde{x}_i).P_i + \sum_{j\in J} \overline{b_j}\langle v_j\rangle.Q_j]\!]_{\mathrm{mix}} \quad = \\ (\lambda z, u)\,(\ \Pi_{i\in I}(\nu n)\,\lambda\widetilde{x}_i\, a_i\langle\widetilde{x}_i znuu\rangle.[\![P_i]\!]_{\mathrm{mix}} \ \mid\ \Pi_{j\in J}(\nu n)\,\overline{b_j}\langle\widetilde{v}_j uuzn\rangle.[\![Q_j]\!]_{\mathrm{mix}}\ ). \end{array}$$

Note that the relative positions of ν-names correctly forbid communication between branches of opposite polarities within the same choice (no 'incestuous' communication, according to the terminology of [12]). The encoding acts as a homomorphism over the remaining operators of Π^{mix}.

Below, $\sim^{\mathrm{mix}}$ denotes barbed congruence over Π^{mix}, and $\sim^{[\![\mathrm{mix}]\!]}$ the equivalence over D-Fusion obtained by closing barbed bisimulation under translated Π^{mix}-contexts. Both equivalences are reasonable semantics in the sense of [13]. The proof of the following theorem is again straightforward by correspondence on reductions and barbs, and by compositionality of the encoding.

Theorem 2 (Full Abstraction for Mixed Choice). *Let $P,Q\in\Pi^{\mathrm{mix}}$. It holds that $P\sim^{\mathrm{mix}} Q$ if and only if $[\![P]\!]_{\mathrm{mix}} \sim^{[\![\mathrm{mix}]\!]} [\![Q]\!]_{\mathrm{mix}}$.*

In a pi-calculus setting, it is well-known that mixed choice cannot be encoded into the choice-free fragment, if one requires the encoding be uniform and preserve a reasonable semantics [13, 14, 12]. The theorem above shows that pi-calculus mixed choice *can* be implemented into the choice-free fragment of D-Fusion. The encoding is uniform, deadlock- and divergence-free, and preserves a reasonable semantics. This is yet another evidence of the expressiveness gap between D-Fusion and pi-calculus.

7 Conclusions and Future Work

We have proposed the D-Fusion calculus, an extension of the fusion calculus where two distinct binders coexist, one analogous to the (x) binder in fusion, the other imposing name freshness. We have shown that D-Fusion is strictly more expressive than both Fusion and pi-calculus.

Our expressiveness results seem to suggest that the design of an efficient distributed implementation of D-Fusion might be nontrivial. This design would probably involve the introduction of a distributed model of the calculus, including, e.g., explicit fusions [4] for broadcasting fusions asynchronously and rollback mechanisms for handling fusion failures. We leave this task for future work. For the time being, we just note that distributed implementations of pi/fusion-like calculi do exist (e.g., the fusion machine of [3]) and may represent a good starting point for distributed implementations of D-Fusion.

Another point that deserves further study is characterization of D-Fusion barbed congruence in terms of a more tractable, labelled bisimulation, which would avoid universal quantification on all contexts. Preliminary results indicate that definition of this equivalence would require a (nontrivial) integration of *substitutive effects* à la fusion calculus [16], i.e. name substitutions resulting from fusions, with *distinctions* à la open pi-calculus [19].

We also plan to extend the D-Fusion calculus by generalising name fusions to substitutions over an arbitrary signature of terms. It would be interesting to compare the expressive power of this extended D-Fusion to systems of Concurrent Constraint or Logic Programming that allow creation of fresh names, such as lambda-Prolog [8], and CCP [17, 20].

In [7] Merro gives an encoding from asynchronous Chi calculus (a variant of Fusion, indipendently introduced by Fu, [2]) to (asynchronous) pi-calculus. However, no result on the other direction is proven. Here, we have proved that pi-calculus cannot be encoded into Fusion.

In [1] the synchronisation mechanism of the pi-calculus is extended to allow for polyadic synchronisation, where channels are vectors of names. The expressiveness of polyadic synchronisation, matching and mixed choice is compared.

Acknowledgments. The authors wish to thank Bjorn Victor for stimulating discussions. The anonymous referees provided valuable suggestions.

References

1. M. Carbone and S. Maffeis. On the Expressive Power of Polyadic Synchronisation in Pi-Calculus. To appear in *Nordic Journal of Computing*.
2. Y. Fu. A Proof Theoretical Approach to Communication. In *Proc. of ICALP '97*, LNCS 1256. Springer-Verlag, 1997.
3. P. Gardner, C. Laneve, and L. Wischik. The fusion machine (extended abstract). In *Proc. of CONCUR '02*, LNCS 2421. Springer-Verlag, 2002.
4. P. Gardner and L. Wischik. Explicit Fusions. *Theoretical Computer Science*. To appear.
5. L. G. Meredith, S. Bjorg, and D. Richter. Highwire Language Specification Version 1.0. Unpublished manuscript.
6. Microsoft Corp. Biztalk Server - `http://www.microsoft.com/biztalk`.
7. M. Merro. On the Expressiveness of Chi, Update, and Fusion calculi. In Proc. of EXPRESS '98, ENTCS 16(2), Elsevier Science, 1998.
8. D. Miller. Unification under a mixed prefix. *Journal of Symbolic Computation*,14(4):321–358, 1992.
9. R. Milner. The Polyadic pi-Calculus: a Tutorial. Technical Report, Computer Science Dept., University of Edinburgh, 1991.
10. R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes (parts I and II). *Information and Computation*, 100(1):1–77, 1992.
11. R. Milner and D. Sangiorgi. Barbed Bisimulation. In *Proc. of ICALP '92*, LNCS 623, Springer-Verlag, 1992.
12. U. Nestmann and B. C. Pierce. Decoding choice encodings. *Information and Computation*, 163(1):1–59, 2000.
13. C. Palamidessi. Comparing the Expressive Power of the Synchronous and the Asynchronous pi-calculus. In *Conf. Rec. of POPL'97*, 1997.
14. C. Palamidessi. Comparing the Expressive Power of the Synchronous and the Asynchronous pi-calculus. *Mathematical Structures in Computer Science*, 13(5):685–719, 2003.

15. J. Parrow and B. Victor. The Update Calculus. In *Proc. of AMAST'97*, LNCS 1349, Springer-Verlag, 1997.
16. J. Parrow and B. Victor. The Fusion Calculus: Expressiveness and Symmetry in Mobile Processes. In *Proc. of LICS'98*. IEEE Computer Society Press, 1998.
17. E. Shapiro. The Family of Concurrent Logic Programming Languages. *ACM Computing Surveys*, 21(3):413-510, 1989.
18. D. Sangiorgi. Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms. PhD thesis, Department of Computer Science, University of Edinburgh, 1992.
19. D. Sangiorgi. A Theory of Bisimulation for the pi-Calculus. *Acta Informatica*, 33(1): 69-97, 1996.
20. V. Saraswat. Concurrent Constraint Programming. The MIT Press, 1993.
21. B. Victor. The Fusion Calculus: Expressiveness and Symmetry in Mobile Processes. PhD thesis, Department of Computer Systems, Uppsala University, 1998.
22. World Wide Web Consortium (W3C) - `http://www.w3.org/TR/wsdl12`.