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. Fusion calculus 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

A recent trend in the design of certain distributed applications like Web services [13] or business-to-business systems [5] based on XML sees the emergence of a message-passing programming style. Languages like Highwire [4] provide, in a concurrency setting, sophisticated data structures; these allow programmers to describe and manipulate complex messages and interaction patterns. If one looks for 'foundational' counterparts of these programming languages, both the pi-calculus [6, 7] and the Fusion calculus [11], 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 [2], or forms of pattern matching between pairs of messages [4]. 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, fusions enable a more general name matching mechanism during synchronisation. However, differently from the pi-calculus, the binding mechanism of Fusion just ignores the issue of unicity of newly generated names. One of the goals of this paper is to show that this fact severely limits the expressiveness of Fusion. Overcoming 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

Research supported in part by FET Global project *PROFUNDIS*.

imposing unicity. This implies combining, in a consistent way, the somehow conflicting mechanisms of fusions and *distinctions* à la open pi-calculus [12]. 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. A more precise account of our work follows.

The binding mechanism of pi-calculus generalises that of λ-calculus in several ways. Input prefix binds like λ, and name passing takes place in pi-calculus in a way typical of functional programming, i.e., formal names are assigned their actual counterpart. 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. Open pi-calculus [12] takes an important step forward over the original proposal, allowing for input actions whose formal parameters can be instantiated, hence 'fused', in a 'lazy' fashion – at any time when needed for synchronisation. In order to preserve the intended semantics of restriction, however, newly extruded names are kept distinct from 'old' names, using special relations called distinctions.

Fusion calculus is presented in [11] as a more uniform and more expressive version of 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 variables 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, in [11] it is stated that the pi-calculus transition system can be embedded into Fusion's, provided that one identifies restriction (νx) with the (x) binder of Fusion.

Our first move is to argue that this embedding breaks down if comparing the two calculi on the basis of abstract 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, and mapping (νx) to (x), and preserving (a subset of) weak traces. As hinted before, the ultimate reason is that in Fusion all names are like logical variables, i.e., their unification always succeeds.

The above failure motivates the introduction of a new calculus, D-Fusion, with two binders, λ and ν: the first generalises input prefix, and the second models restriction. Also, any issue of symmetry between input and output is preempted, since we have just one kind of prefix (no polarisation); polarised prefixes can be easily encoded, though. As in open pi-calculus, in D-Fusion distinctions are employed to constrain possible fusions. In logical terms, this corresponds to allow unification not only among variables, but also among variables and dynamically generated constants (that is, ν-extruded names). However, unification fails whenever one tries to identify two distinct constants, or to identify a 'recent' constant with an 'old' variable. We show that the additional expressive power achieved in this way is relevant. Both pi-calculus and Fusion can be uniformly encoded into D-Fusion. Moreover, the combined mechanism of restriction and unification yields additional expressive power: it allows to express a form of pat-

tern 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 [9, 10] that nondeterministic *guarded choice* cannot be simulated in the choice ($+$) -free pi-calculus in a fully abstract way, while preserving any 'reasonable' semantics. This is due to the impossibility of atomically performing, 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 in the sense of [9, 10]. Informally, branches of a choice are represented as concurrent processes. Synchronisation is performed in the ordinary way, but it forces a fusion between a global λ-name 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.

The rest of the paper is organised as follows. Section 2 contains the proof that the pi-calculus cannot be uniformly encoded into Fusion. In Section 3 we introduce the D-Fusion calculus and in Section 4 we give a notion of bisimulation for the calculus, and characterise it via a more handy symbolic semantics; we also introduce a notion of *sorting* and sorted bisimulation. In Section 5 we show that the D-Fusion calculus is more expressive than both pi-calculus and Fusion. We further explore this expressiveness gap in Sections 6 by means of an example and in Section 7 by encoding the mixed guarded choice into the choice-free calculus. Finally, Section 8 contains a brief overview of some related work and a few concluding remarks.

2 Fusion and Pi

The aim of this section is illustrating 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:

$$R \,|\, \overline{u}v.P \,|\, ux.Q \xrightarrow{\{x=v\}} (R \,|\, P \,|\, Q)[v/x].$$

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

$$R \,|\, (x)\left(\overline{u}v.P \,|\, ux.Q\right) \xrightarrow{\tau} R \,|\, (P \,|\, Q)[v/x].$$

where τ denotes the identity fusion. For a full treatment of pi-calculus and Fusion we refer to [7] and to [11], 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 [11] does satisfy our criterion. We shall assume here the standard pi-calculus late operational semantics [7] 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* $[\![\cdot]\!] : \Pi \to \mathcal{F}$ *is* uniform *if for each* $P, Q \in \Pi$ *it holds that:*

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

The next proposition generalises an example from [11]. 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 [11]).

Proposition 1. *There is no uniform translation* $[\![\cdot]\!] : \Pi \to \mathcal{F}$ *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:

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

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:

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

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 $A \stackrel{\overline{w}}{\Longrightarrow}$. On the other hand, a careful case analysis shows that $B \stackrel{\overline{w}}{\not\Longrightarrow}$. 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 $\widetilde{x}$ 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:

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

where *prefixes* α are defined as $\alpha ::= a\widetilde{v}$. 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.

Note that we consider one kind of prefix, thus ignoring polarities. However, a sub-calculus 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.

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 $\widetilde{x}$, we define $\sigma_{|\widetilde{x}}$ as $\sigma \cap (\widetilde{x} \times \mathcal{N})$, and $\sigma_{-\widetilde{x}}$ as $\sigma - (\widetilde{x} \times \mathcal{N})$.

Definition 2 (fusions). *We let $\phi, \chi, \ldots$ range over* fusions, *that is total equivalence relations on $\mathcal{N}$ with only finitely many non-singular equivalence classes. We let:*

- $\mathrm{n}(\phi)$ *denote* $\{x : x \phi y \text{ for some } y \neq x\}$*;*
- τ *denote the identity fusion (i.e.,* $\mathrm{n}(\tau) = \emptyset$*);*
- $\phi + \psi$ *denote the finest fusion which is coarser than ϕ and ψ, that is* $(\phi \cup \psi)^\star$*;*
- 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.*

A fusion ϕ arises as the result of equating two lists of names in a synchronisation. In general, names in the original lists might be either λ-abstracted or ν-extruded, or free. Informally, the effect of ϕ should be that of a substitution, mapping equivalent names onto representatives. However, in the presence of ν-, λ-abstracted and free ϕ-equivalent names, some care is needed to ensure that the representative be chosen appropriately. This concept is made precise in the next definition.

Definition 3 (induced substitutions). *Let $\widetilde{x}$ and $\widetilde{k}$ be two disjoint tuples of names and ϕ be a fusion such that for any two distinct $k_1, k_2 \in \widetilde{k}$, not $(k_1 \phi k_2)$. We say that a substitution σ is* induced by $\lambda\widetilde{x}$, $(\nu\widetilde{k})$ and ϕ, *if* $\mathrm{dom}(\sigma) = \mathrm{n}(\phi)$ *and for each equivalence class E of ϕ, σ maps all names in E onto one and the same name $y \in E$ such that:*

- *if* $\widetilde{k} \cap E = \{h\}$ *then* $y = h$;
- *if* $\widetilde{k} \cap E = \emptyset$ *and* $E - \widetilde{x} \neq \emptyset$ *then* $y \in E - \widetilde{x}$.

For example, suppose ϕ has only two non-singular equivalence classes $\{x, w, k\}$ and $\{y, h\}$, then a substitution σ induced by $\lambda xy.\ (\nu k)$ and ϕ is the one mapping x, w, k to k and y, h to h. It is important to stress that, by definition, no induced σ exists if for distinct $k_1, k_2 \in \widetilde{k}$ it holds $k_1 \,\phi\, k_2$. Also note that in general there may be more than one induced substitution. E.g. $\{a = b\}$ (with empty $\lambda\widetilde{x}$ and $(\nu\widetilde{k})$) induces both $[a/b]$ and $[b/a]$. We introduce the labelled transition system for D-Fusion.

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, while $\mathrm{subj}(\mu)$ and $\mathrm{obj}(\mu)$ denote the subject and object part of μ, if μ is an action, otherwise they both denote conventional value '$-$'. Moreover, $\mathrm{n}(\mu)$ denotes all the names of 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. A major difference is that our effects can also ν-extrude names. An effect $(\nu\widetilde{x})\,\phi$ 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, the corresponding substitution $\sigma_{|z}$ is applied onto the target process and z is discarded from the fusion (the result is ϕ_{-z}). Any ν-extruded name $w \in \widetilde{x}$ which is equivalent to z must be ν-closed back, provided that no free name is in that equivalence class. In rule Com, handling of λ- and ν-extruded names can be explained similarly, but note that those λ-extruded names $\widetilde{z}$ that are not instantiated (by $\sigma_{|\widetilde{y}'\widetilde{y}''}$) must be λ-closed (this happens when in some equivalence class all names are λ-abstracted). Finally, note that the side condition '$\phi[z] \cap \widetilde{x} = \emptyset$' in rule ν-Open prevent effects from equating two distinct ν-extruded names.

Let us illustrate the rules with some examples. We shall use $\{\widetilde{x} = \widetilde{y}\}.P$ as an abbreviation for $(\nu c)\,(c\widetilde{x}|c\widetilde{y}.P)$ (for a fresh name c) and we write $\tau.P$ when $\widetilde{x}$ and $\widetilde{y}$ are empty.

Example 1.

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

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

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

$$\lambda y\,(\nu x)\,(\nu c)\,(cx.P_1 \,|\, cy.P_2) \xrightarrow{\tau} (\nu x)\,((\nu c)\,(P_1 \,|\, P_2)[x/y]).$$

(ACT) $\alpha.P \xrightarrow{\alpha} P$ (MATCH) $\dfrac{P \xrightarrow{\mu} Q}{[a=a]P \xrightarrow{\mu} Q}$ (SUM) $\dfrac{P_1 \xrightarrow{\mu} Q}{P_1+P_2 \xrightarrow{\mu} Q}$

(ν-PASS) $\dfrac{P \xrightarrow{\mu} Q}{(\nu z)P \xrightarrow{\mu} (\nu z)Q} \quad z \notin \mathrm{n}(\mu)$ (λ-PASS) $\dfrac{P \xrightarrow{\mu} Q}{\lambda z P \xrightarrow{\mu} \lambda z Q} \quad z \notin \mathrm{n}(\mu)$

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

(λ-OPEN$_a$) $\dfrac{P \xrightarrow{(\nu \widetilde{x})\lambda \widetilde{y} a \widetilde{v}} Q}{\lambda z P \xrightarrow{(\nu \widetilde{x})\lambda \widetilde{y} z a \widetilde{v}} Q} \quad z \in \widetilde{v} - (\{a\} \cup \widetilde{x}\widetilde{y})$

(λ-OPEN$_f$) $\dfrac{P \xrightarrow{(\nu \widetilde{x})\phi} Q}{\lambda z P \xrightarrow{(\nu \widetilde{x}')\phi_{-z}} (\nu \hat{w})\,(Q\sigma_{|z})}$ $\begin{cases} z \in \mathrm{n}(\phi) \\ \sigma \text{ induced by } \lambda z,\ (\nu \widetilde{x}) \text{ and } \phi \\ \widetilde{x}' = \widetilde{x} \cap \mathrm{n}(\phi_{-z}) \\ \hat{w} = \widetilde{x} - \widetilde{x}' \end{cases}$

(COM) $\dfrac{P_1 \xrightarrow{(\nu \widetilde{x}')\lambda \widetilde{y}' a \widetilde{u}} Q_1 \quad P_2 \xrightarrow{(\nu \widetilde{x}'')\lambda \widetilde{y}'' a \widetilde{v}} Q_2}{P_1|P_2 \xrightarrow{(\nu \widetilde{x})\{\widetilde{v}=\widetilde{u}\}_{-\widetilde{y}'\widetilde{y}''}} (\nu \widetilde{w})\lambda \widetilde{z}((Q_1|Q_2)\sigma_{|\widetilde{y}'\widetilde{y}''})}$ $\begin{cases} |\widetilde{u}| = |\widetilde{v}| \\ \sigma \text{ induced by } \lambda \widetilde{y}'\widetilde{y}'',\ (\nu \widetilde{x}'\widetilde{x}'') \text{ and } \{\widetilde{v} = \widetilde{u}\} \\ \widetilde{z} = \widetilde{y}'\widetilde{y}'' \cap \mathrm{cod}(\sigma_{|\widetilde{y}'\widetilde{y}''}) \\ \widetilde{x} = \widetilde{x}'\widetilde{x}'' \cap \mathrm{n}(\{\widetilde{v} = \widetilde{u}\}_{-\widetilde{y}'\widetilde{y}''}) \\ \widetilde{w} = \widetilde{x}'\widetilde{x}'' - \widetilde{x} \end{cases}$

(PAR) $\dfrac{P \xrightarrow{\mu} Q}{P|R \xrightarrow{\mu} Q|R}$ (REP) $\dfrac{P|!P \xrightarrow{\mu} Q}{!P \xrightarrow{\mu} Q}$

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

Table 1. Actions and effects transitions in D-Fusion.

It is worth noting that our operational rules are preserved by a form of structural equivalence. For instance, let $R = (\nu x)\,\lambda y\,(\nu c)\,(cx.P_1 \,|\, cy.P_2)$. R has the same τ-transition as Q above.

2. Another example is as follows (P below is some continuation):

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

Encoding I/O polarities We can encode polarities as follows:

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

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.

4 Bisimulation

Like in the case of Fusion, a 'natural' semantics of D-Fusion is required to be closed under substitutions. However, one should be careful in keeping ν-extruded names distinct from others (λ-extruded and free). To this purpose, we introduce below a concept of *distinction*, akin to [12]. Distinctions keep track of ν-extruded names, for which fusions should be forbidden. Differently from [12], however, our distinctions keep track of λ-extruded names as well, and of the order of their extrusion relative to ν-extruded names. The idea is to treat ν-extruded names as constants, while allowing a fusion between two λ-extruded names, or a fusion of a 'recent' λ-extruded name to an 'old' ν-extruded name. Technically, we find it convenient to model distinctions as sequences.

Definition 5 (distinctions). *A* distinction *D is a sequence of labelled names of the form $x_1^{e_1}\cdots x_n^{e_n}$, with labels $e_i \in \{\nu,\lambda\}$ and x_i pairwise distinct, for $1 \le i \le n$. We let $\lambda(D)$ (resp. $\nu(D)$) denote the set of λ-labelled (resp. ν-labelled) names in D and let $\mathrm{n}(D) = \lambda(D) \cup \nu(D)$. Given a substitution σ and a distinction $D = x_1^{e_1}\cdots x_n^{e_n}$, we say that σ* respects *D, written $\sigma \vdash D$, if for each x_i^{ν} in D it holds that: $x_i\sigma = x_i$ and moreover for each x_j^{λ} in D, $x_j\sigma = x_i$ implies $j > i$. We let $D\sigma$ denote the result of applying σ to D and then erasing all duplicated (i.e. following the leftmost) occurrences of names in the sequence.*

Given a fusion ϕ and a distinction D, we say that ϕ and D induce *σ, written $\phi, D \rightsquigarrow \sigma$, if σ is a substitution induced by $\lambda\widetilde{x}$, $(\nu\widetilde{k})$ and ϕ, where $\widetilde{x} = \lambda(D)$ and $\widetilde{k} = \nu(D)$, and σ respects D.*

Example 2. Consider $D = x_0^{\lambda}\cdot x_1^{\nu}\cdot x_2^{\lambda}\cdot x_3^{\lambda}\cdot x_4{}^{\nu}$ and $\phi = \{x_1, x_2, x_4 = x_2, x_3, x_5\}$. We have that $\phi, D \rightsquigarrow \sigma$, where σ maps x_1, x_2, x_3 to x_1, and x_5 to x_4: in fact, we have that $\sigma \vdash D$.

Note that $D\sigma = x_0^{\lambda} \cdot x_1^{\nu} \cdot x_4^{\nu}$. On the other hand, if one considers D and $\{x_4 = x_0\}$ then there is no σ' induced by this effect and D: indeed, σ' should have to map x_0 to x_4, but this implies $\sigma' \not\models D$.

We use the following abbreviation: $P \xrightarrow{(\nu\tilde{x})\,\lambda\tilde{y}\,(\alpha,\phi)} P'$ means that either $P \xrightarrow{(\nu\tilde{x})\,\lambda\tilde{y}\,\alpha} P'$ and $\phi = -$, or $P \xrightarrow{(\nu\tilde{x})\,\phi} P'$ and $\alpha = -$ (here $-$ is just a conventional 'null' value). Moreover, we stipulate that $-, D \rightsquigarrow \mathtt{id}$, where $\mathtt{id}$ is the identity substitution.

Definition 6 (open bisimulation). *A set $\mathcal{R} = \{R_D\}_D$ of process relations indexed by distinctions is an* indexed simulation *if for each D, whenever $P\ R_D\ Q$ and $P \xrightarrow{(\nu\tilde{x})\,\lambda\tilde{y}\,(\alpha,\phi)} P'$ and $\phi, D \cdot \tilde{x}^{\nu} \rightsquigarrow \sigma$, then a transition $Q \xrightarrow{(\nu\tilde{x})\,\lambda\tilde{y}\,(\alpha,\phi)} Q'$ exists such that $P'\sigma\ R_{D'}\ Q'\sigma$, where $D' = D\sigma \cdot \tilde{x}^{\nu} \cdot \tilde{y}^{\lambda}$.*

$\mathcal{R}$ is an indexed bisimulation *if both $\mathcal{R}$ and $\mathcal{R}^{-1} = \{R_D^{-1}\}_D$ are indexed simulations.* Open bisimulation, $\sim$, *is the largest indexed bisimulation preserved by respectful substitutions, i.e.: for each σ and D, if $P \sim_D Q$ and σ respects D then $P\sigma \sim_{D\sigma} Q\sigma$.*

When no confusion arises, we write $P \sim Q$ for $P \sim_{\varepsilon} Q$, where ε is the empty distinction.

Proposition 2 (basic properties of $\sim_D$). *Let P and Q be two processes. Then:*

- *$P \sim_D Q$ and $x \notin \mathrm{n}(D)$ imply $\lambda x P \sim_D \lambda x Q$.*
- *$P \sim_D Q$ implies $(\nu x)P \sim_{D-x} (\nu x)Q$.*

Prefix, parallel composition, matching, sum and replication operators preserve $\sim_D$.

Example 3. 1. An example of 'expansion' for parallel composition is as follows:

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

Note that, after the extrusion of k, free name v can *still* be fused to k. On the contrary, if we consider a distinction D and let $v \in \mathrm{n}(D)$, no fusion at all is possible:

$$((\nu k)\,ak.ak)|av \;\sim_D\; (\nu k)\,ak.(ak.av + av.ak) \;+\; av.((\nu k)\,ak.ak).$$

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:

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

Symbolic bisimulation The definition of open bisimulation contains an implicit universal quantification over substitutions, which makes it not handy. The next definition and the subsequent theorem provide us with an alternative formulation of this equivalence akin to the 'efficient characterisation' of [12]. The new formulation is based on symbolic transitions of the form $P \xrightarrow{\mu}_{\chi} Q$, where χ is a logical condition necessary for the transition to take place. For technical convenience, we shall represent conditions as fusions. The defining clauses for $\xrightarrow{\mu}_{\chi}$ are obtained from those in Table 1 by simply

attaching the appropriate χ. In particular, action prefix is defined as $\alpha.P \xrightarrow{\alpha}_{\tau} P$, while communication and matching rules are defined as follows:

$$(\textsc{Match}_S)\ \frac{P \xrightarrow{\mu}_{\chi} Q}{[a=b]P \xrightarrow{\mu}_{\chi+\{a=b\}} Q} \qquad (\textsc{Com}_S)\ \frac{P_1 \xrightarrow{(\nu\tilde{x})\lambda\tilde{y}\, a\tilde{u}}_{\chi_1} Q_1 \quad P_2 \xrightarrow{(\nu\tilde{x}')\lambda\tilde{y}'\, b\tilde{v}}_{\chi_2} Q_2}{P_1|P_2 \xrightarrow{(\nu\tilde{x})\{\tilde{v}=\tilde{u}\}_{-\tilde{y}'\tilde{y}''}}_{\chi_1+\chi_2+\{a=b\}} (\nu\tilde{w})\lambda\tilde{z}((Q_1|Q_2)\sigma_{|\tilde{y}'\tilde{y}''})},$$

where the side condition of $\textsc{Com}_S$ is the same as in rule $\textsc{Com}$. The other rules just propagate χ, with the proviso that λ- and ν- extruded names do not occur in χ. Given fusions ϕ and ψ, we write $\phi \vDash \psi$ if $\psi \subseteq \phi$.

Definition 7 (symbolic bisimulation). *A set $\mathcal{R} = \{R_D\}_D$ of process relations indexed by distinctions is an* indexed symbolic simulation *if whenever $P\ R_D\ Q$ and $P \xrightarrow{(\nu\tilde{x})\lambda\tilde{y}(\alpha,\phi)}_{\chi} P'$ and $(\phi+\chi), D\cdot\tilde{x}^{\nu} \rightsquigarrow \sigma$, then a transition $Q \xrightarrow{(\nu\tilde{x})\lambda\tilde{y}(\beta,\psi)}_{\chi'} Q'$ exists such that:*

1. $\chi \vDash \chi'$,
2. $\chi \vDash \{\mathrm{subj}(\alpha) = \mathrm{subj}(\beta)\}$ *and* $\chi \vDash \{\mathrm{obj}(\alpha) = \mathrm{obj}(\beta)\}$,
3. $\phi+\chi = \psi+\chi$,
4. $P'\sigma\ R_{D'}\ Q'\sigma$, *where* $D' = D\sigma\cdot\tilde{x}^{\nu}\cdot\tilde{y}^{\lambda}$.

$\mathcal{R}$ is a indexed symbolic bisimulation *if both $\mathcal{R}$ and $\mathcal{R}^{-1} = \{R_D^{-1}\}_D$ are indexed open simulations. The largest indexed symbolic bisimulation is written $\sim^{\mathrm{symb}}$.*

Theorem 1. *For each P, Q and D, $P \sim_D Q$ if and only if $P \sim_D^{\mathrm{symb}} Q$.*

PROOF: For any ϕ and σ, let $\phi\sigma$ denote the fusion $\{(x\sigma, y\sigma) \mid x\,\phi\,y\}^{\star}$, and let the *fusion induced by* σ (seen as a binary relation) be defined as $\sigma^{\star}$. The proof of the theorem is based on the following operational correspondences:

1. $P \xrightarrow{(\nu\tilde{x})\lambda\tilde{y}(\alpha,\phi)}_{\chi} P'$ implies there is a transition $P\sigma \xrightarrow{(\nu\tilde{x})\lambda\tilde{y}(\alpha\sigma,\phi\sigma)} P'\sigma$, for each substitution σ induced by χ.
2. $P\sigma \xrightarrow{(\nu\tilde{x})\lambda\tilde{y}(\beta,\psi)} P_1$ implies there is a fusion χ induced by σ and a transition $P \xrightarrow{(\nu\tilde{x})\lambda\tilde{y}(\alpha,\phi)}_{\chi'} P'$ s.t. $\chi \vDash \chi'$, $\beta = \alpha\sigma$ and $\psi = \phi\sigma$ and $P_1 = P'\sigma$.

□

Example 4. Fusions and conditions play pretty different roles in bisimulation semantics. As an example, let $P = [a=b](ca \mid \bar{c}b)$ and $Q = ca \mid \bar{c}b$. Both in P and Q the interaction between ca and $\bar{c}b$ results in a fusion $\{a=b\}$, but $P \not\sim Q$ since P can perform an action only under the condition $\chi = \{a=b\}$, i.e. $P \xrightarrow{\{a=b\}}_{\{a=b\}} \mathbf{0}$ while $Q \xrightarrow{\{a=b\}}_{\tau} \mathbf{0}$.

Sorting Sorting systems are used in polyadic pi-calculus to discipline the use of channels [6]. We adapt the concept of sorting from pi-calculus, and slightly extend it in order to also control 'patterns' that can arise in the object part of actions. In a sorting system **S**, the set of names is partitioned into a family of countable *sorts*, denoted as $S, S', \ldots$. Moreover, there is a sorting function that maps each sort S to a *pattern*

$(x_1 : S_1, \ldots, x_n : S_n)$, written also $\mathbf{S} : \ S \mapsto \widetilde{x} : \widetilde{S}$, with the x_i's not necessarily distinct: the intended meaning is that names of sort S should only carry tuples $\widetilde{v}$ of type $\widetilde{S}$ s.t. $x_i = x_j$ implies $v_i = v_j$.

Like in polyadic pi-calculus, one can devise typing systems which guarantee that well-typed processes will never perform 'illegal' (in the above sense) actions. However, here we are primarily interested in defining the open bisimilarity $\sim^{\mathrm{S}}$ naturally induced by a sorting system $\mathbf{S}$. Let us say that a substitution σ is $\mathbf{S}$-*respectful* if, for each x, $x\sigma$ and x belong to the same sort. Similarly, we say a fusion ϕ is $\mathbf{S}$-respectful if each equivalence class is entirely included in one sort (hence any substitution induced by ϕ is respectful). In a $\mathbf{S}$-sorted D-fusion, for a given sorting system $\mathbf{S}$, we assume that in every prefix $a\widetilde{v}$ each v_i belongs to the appropriate sort, and we *only consider respectful substitutions and fusions*. For $\alpha = a\widetilde{v}$ an action, where $a \in S \mapsto \widetilde{x} : \widetilde{S'}$, the fusion induced by α, denoted ϕ_α, is defined as $\sum_{x_i,\, x_j \in \widetilde{x},\, x_i = x_j} \{v_i = v_j\}$ (here 'Σ' denotes sum of fusions defined in Def. 2). Furthermore, we shall denote by $\equiv$ equivalence over actions induced by the law: $\lambda x\, \mu \equiv \mu$ if $x \notin \mathrm{n}(\mu)$.

With the above conventions and notations, the definition of $\sim^{\mathrm{S}}$ is obtained from Definition 6 by making the clauses of actions and effects explicitly distinct, and changing the clause for *actions* as follows:

> if $P \xrightarrow{(\nu\widetilde{x})\,\lambda\widetilde{y}\,a\widetilde{v}} P'$ and $\phi_{a\widetilde{v}}, D \cdot \widetilde{x}^{\nu} \cdot \widetilde{y}^{\lambda} \rightsquigarrow \sigma$ then a transition $Q \xrightarrow{(\nu\widetilde{x})\,\lambda\widetilde{y}'\,a\widetilde{v}'} Q'$ exists s.t. $\phi_{a\widetilde{v}'}, D \cdot \widetilde{x}^{\nu} \cdot \widetilde{y}'^{\lambda} \rightsquigarrow \sigma'$ and $\lambda\widetilde{y}\,(a\widetilde{v}\sigma) \equiv \lambda\widetilde{y}'\,(a\widetilde{v}'\sigma')$, $\sigma_{-\widetilde{y}} = \sigma'_{-\widetilde{y}'}$ and $P'\sigma\, R_{D'}\, Q'\sigma'$, where $D' = (D \cdot \widetilde{x}^{\nu} \cdot \widetilde{y}^{\lambda})\sigma$.

Note that $\sim^{\mathrm{S}}$ is in general more generous than ordinary open bisimilarity $\sim$. For example, assuming $a \in S \mapsto (x : S', x : S')$, then $(\nu n)\,\lambda x\, axn.P \sim^{\mathrm{S}} (\nu n)\,\lambda x\, anx.P \sim^{\mathrm{S}} (\nu n)\, ann.P[^{n}/_{x}]$ and $(\nu n)\,(\nu m)\, amn.P \sim^{\mathrm{S}} \mathbf{0}$. We will see applications of sorted bisimilarity in Section 7.

5 Expressiveness of D-Fusion

Below, we provide the two obvious uniform and fully abstract translations from Π and $\mathcal{F}$ to $\mathcal{DF}$. The definition of uniformity can be extended to the case of encodings from $\mathcal{F}/\Pi$ into $\mathcal{DF}$ in the obvious way: in particular, by requiring that (x) and (νx) be mapped to λx and (νx), respectively.

Definition 8 ($[\![\cdot]\!]_\pi$). *The translation $[\![\cdot]\!]_\pi : \Pi \to \mathcal{DF}$ is defined by extending in the expected homomorphic way the following clauses:*

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

For any distinction $D = x_1^{e_1} \cdots x_n^{e_n}$, we can generate an equivalent Π-distinction D_π (that is, an irreflexive, symmetric relation on $\mathcal{N}$, [12]) by setting $x_i\ D_\pi\ x_j$ iff: either ($e_i = e_j = \nu$ and $i \neq j$) or ($e_i = \nu$ and $e_j = \lambda$ and $j < i$) or ($e_j = \nu$ and $e_i = \lambda$ and $i < j$). It is reasonable to consider here only Π-distinctions that can be generated in this way. For any such Π-distinction D_π, let us denote by $[\![D_\pi]\!]$ a chosen $\mathcal{DF}$-distinction that generates D_π. Let $\sim^{\mathrm{o}}_{D_\pi}$ denote open D_π-bisimilarity over Π ([12]).

Proposition 3. *$P \sim^{\mathrm{o}}_{D_\pi} Q$ iff $[\![P]\!]_\pi \sim_{[\![D_\pi]\!]} [\![Q]\!]_\pi$.*

We turn now our attention to Fusion.

Definition 9 ($[\![\cdot]\!]_f$). *The translation $[\![\cdot]\!]_f : \mathcal{F} \to \mathcal{DF}$ is defined by extending in the expected homomorphic way the following clauses:*

$$[\![\overline{a}\langle x\rangle.P]\!]_f = \overline{a}\langle x\rangle.[\![P]\!]_f \qquad [\![a\langle x\rangle.P]\!]_f = a\langle x\rangle.[\![P]\!]_f \qquad [\![(x)P]\!]_f = \lambda x\,[\![P]\!]_f$$

Let $\sim^{\mathrm{he}}$ denote hyper-equivalence over $\mathcal{F}$ (see [11]).

Proposition 4. *$P \sim^{\mathrm{he}} Q$ iff $[\![P]\!]_f \sim [\![Q]\!]_f$.*

Most 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 fusions and restrictions allows one to express a form of input with pattern matching. This is not possible in Π without breaking atomicity of certain actions (e.g. choice). To show this, we restrict our attention to polarised D-Fusion, $\mathcal{DF}^{\mathrm{p}}$.

Given $P \in \Pi$ and a trace of actions s, let us write $P \stackrel{\hat{s}}{\Longrightarrow}$ if $P \stackrel{s'}{\Longrightarrow}$ for some trace s' that has 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 $s' = a\langle\widetilde{z}\rangle \cdot \overline{b}\langle\widetilde{w}\rangle$). The reference semantics for Π is again the late operational semantics.

Definition 10. *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, $P \stackrel{s}{\Longrightarrow}$ implies $[\![P]\!] \stackrel{\hat{s}}{\Longrightarrow}$;*
- *$[\![P|Q]\!] = [\![P]\!]|[\![Q]\!]$;*
- *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.

Proposition 5. *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:

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

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, due to uniformity, it is easily checked that $[\![P]\!] \stackrel{\overline{b}}{\Longrightarrow}$, while $[\![Q]\!] \stackrel{\overline{b}}{\not\Longrightarrow}$ (the proof of the latter relies on $b \notin \mathrm{fn}(Q)$ and on the uniformity with respect to substitutions). Thus $[\![P]\!] \not\asymp_{\Pi} [\![Q]\!]$. □

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.

6 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:

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

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 help A to contrast attackers replying old, compromised keys or trying to get distinct sessions confused. 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 $p(\widetilde{x}).X$):

$$A = (\nu n)\left(\overline{p}\langle n\rangle|\lambda y\, k_S\langle n,y\rangle.(\nu n')\,(\overline{p}\langle n'\rangle|\lambda y'\, k_S\langle n',y'\rangle.\overline{commit}\langle y,y'\rangle)\right)$$
$$S = p(x).\left(\overline{k_S}\langle x,k\rangle|p(x').\overline{k_S}\langle x',k'\rangle\right)$$
$$P = (\nu k_S)\,(A|S).$$

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 P, that is $\overline{k_S}\langle x',k'\rangle$. In fact, n' can be extruded only *after* x has been received, hence the appropriate distinction will forbid fusion of x and n'. The equivalence is formally shown by exhibiting the appropriate symbolic indexed bisimulation, composed basically by all pairs of derivatives P' and Q' s.t. $P \stackrel{s}{\Longrightarrow} P'$ and $Q'$is obtained from P' by replacing $\overline{commit}\langle y,y'\rangle$ with $\overline{commit}\langle k,k'\rangle$, with distinctions as given by the bound names in s.

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, $R = p(x).\left(\overline{p}\langle x\rangle|p(y).\overline{p}\langle y\rangle\right)$. Formally, for $\alpha = \overline{commit}\langle k',k\rangle$,

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

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.

7 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 adds 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 5. 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:

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

(here, $x,n,m \notin \text{fn}(a,P,Q)$). Assume now a sorting discipline by which messages travelling on channel a must carry two identical names (i.e. $a \in S \mapsto (x:S', x:S')$, according

to Section 4). 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 sorting 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 bisimilarity induced by the above sorting, say $\sim^{\mathrm{S}}$, which only takes into account transitions that obey the sorting discipline (the rightmost two names in the object part of $a/\overline{a}$-actions are the same). The other way is keeping standard bisimilarity $\sim$, while inserting processes inside a 'firewall' that filters out a-messages not respecting the given sorting. The latter can be easily defined in D-Fusion relying on 'non-linear' inputs:

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

We state the result in both forms below.

Proposition 6. *Let A and B be as in Example 5.*

1. *$A \sim^{\mathrm{S}} B$;*
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 6 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 sorted bisimilarity. We believe the results can also be stated in terms of untyped bisimilarity, 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 $\sum_{i\in i} a_i\langle\widetilde{x_i}\rangle.P_i$. We also assume a sorting system on this source language. 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, **S1** indicates the sorting that extends the sorting system **S** of the source language by imposing that the two extra object names introduced by the encoding be the same (i.e., if **S**: $S \mapsto \widetilde{x} : \widetilde{S'}$, then **S1**:

$S \mapsto (\widetilde{x} : \widetilde{S'}, y : S_0, y : S_0)$, for a fresh y and a new sort S_0) and $\sim^{\mathrm{S1}}$ is the corresponding open bisimilarity. The proof of the following theorem is straightforward, given that there is a 1-to-1 correspondence between moves of P and moves of $[\![P]\!]_{\mathrm{ig}}$, under the sortings **S** and **S1**.

Theorem 2 (full abstraction for ig choice). *Let $P, Q \in \mathcal{DF}^{\mathrm{p,ig}}$. It holds that $P \sim^{\mathrm{S}} Q$ if and only if $[\![P]\!]_{\mathrm{ig}} \sim^{\mathrm{S1}} [\![Q]\!]_{\mathrm{ig}}$.*

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 [6] 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:

$$[\![\textstyle\sum_{i\in I} a_i(\widetilde{x_i}).P_i + \sum_{j\in J} \overline{b_j}\langle v_j\rangle.Q_j]\!]_{\mathrm{mix}} \quad =$$
$$(\lambda z,u)\,(\ \textstyle\Pi_{i\in I}(\nu n)\,\lambda\widetilde{x_i}\, a_i\langle \widetilde{x_i} z n u u\rangle.[\![P_i]\!]_{\mathrm{mix}} \ \mid\ \Pi_{j\in J}(\nu n)\,\overline{b_j}\langle \widetilde{v_j} u u z n\rangle.[\![Q_j]\!]_{\mathrm{mix}}\ ).$$

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 [8]). The encoding acts as a homomorphism over the remaining operators of Π^{mix}.

Below, $\sim^{\mathrm{S}}$ indicates sorted open bisimilarity in Π^{mix}. We denote by **S2** the D-Fusion sorting induced by the above translation of the source sorting (i.e., if **S**: $S \mapsto \widetilde{x} : \widetilde{S'}$, then **S2**: $S \mapsto (\widetilde{x} : \widetilde{S'}, y : S_0, y : S_0, z : S_0, z : S_0)$, where S_0 is a new sort and y and z are fresh). The corresponding bisimilarity is written $\sim^{\mathrm{S2}}$.

Theorem 3 (full abstraction for mixed choice). *Let $P, Q \in \Pi^{\mathrm{mix}}$. It holds that $P \sim^{\mathrm{S}} Q$ if and only if $[\![P]\!]_{\mathrm{mix}} \sim^{\mathrm{S2}} [\![Q]\!]_{\mathrm{mix}}$.*

In a pi-calculus setting, it is known that mixed choice cannot be encoded into the choice-free fragment, if one requires the encoding be uniform and preserve a 'reasonable semantics' [9, 10, 8]. The theorem above shows that, under mild typing conditions, 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.

8 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 more expressive than both Fusion and pi-calculus.

Our expressiveness results seem to suggest that an efficient distributed implementation of D-Fusion might be nontrivial to design. This design would certainly involve the introduction of a distributed model of the calculus, including, e.g., explicit fusions [3] for broadcasting fusions asynchronously, and primitives for handling explicit localities. 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 [2]. One may expect that, starting from these, building a distributed implementation of D-Fusion should not be much harder.

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 and it is shown how the degree of synchronisation of a calculus increases its expressive power.

We plan to extend the D-Fusion calculus by generalising name fusions to arbitrary substitutions over a signature of terms. We believe that the extended D-Fusion would be strictly more expressive that Logic Programming, the intuition being that restriction (creation of new fresh names) cannot be modelled in LP.

Finally, it would also be interesting to study a coalgebraic model for D-Fusion. A coalgebraic framework would present several advantages. For instance, morphisms between coalgebras enjoy the property of "reflecting behaviours" and thus they allow to characterise bisimulation equivalences in more abstract terms as kernels of morphisms.

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. P. Gardner, C. Laneve, and L. Wischik. The fusion machine (extended abstract). In *Proc. of CONCUR '02*, LNCS 2421. Springer-Verlag, 2002.
3. P. Gardner and L. Wischik. Explicit Fusions. *Theoretical Computer Science*. To appear.
4. L. G. Meredith, S. Bjorg, and D. Richter. Highwire Language Specification Version 1.0. Unpublished manuscript.
5. Microsoft Corp. Biztalk Server - `http://www.microsoft.com/biztalk`.
6. R. Milner. The Polyadic pi-Calculus: a Tutorial. Technical Report, Computer Science Dept., University of Edinburgh, 1991.
7. R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes (parts I and II). *Information and Computation*, 100(1):1–77, 1992.
8. U. Nestmann and B. C. Pierce. Decoding choice encodings. *Information and Computation*, 163(1):1–59, 2000.
9. C. Palamidessi. Comparing the Expressive Power of the Synchronous and the Asynchronous pi-calculus. In *Conf. Rec. of POPL'97*, 1997.
10. C. Palamidessi. Comparing the Expressive Power of the Synchronous and the Asynchronous pi-calculus. *Mathematical Structures in Computer Science*, 13(5):685–719, 2003.
11. J. Parrow and B. Victor. The Fusion Calculus: Expressiveness and Symmetry in Mobile Processes. In *Proc. of LICS'98*. IEEE Computer Society Press, 1998.
12. D. Sangiorgi. A Theory of Bisimulation for the pi-Calculus. *Acta Informatica*, 33(1): 69-97, 1996.
13. World Wide Web Consortium (W3C) - `http://www.w3.org/TR/wsdl12`.