Information and Computation 251 (2016) 335–360

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Name-passing calculi: From fusions to preorders and types

Daniel Hirschkoff [a,c,*], Jean-Marie Madiot [a], Davide Sangiorgi [b]

[a] Université de Lyon, ENS Lyon, CNRS, INRIA, France
[b] INRIA and Università di Bologna, Italy
[c] LIP, ENS Lyon, site Monod, 46, allée d'Italie, 69364 Lyon Cedex 7, France

ARTICLE INFO

Article history:
Received 6 October 2014
Received in revised form 20 May 2016
Available online 19 October 2016

Keywords:
Process calculus
Type system
Subtyping
Behavioural equivalence
Expressiveness

ABSTRACT

The fusion calculi are a simplification of the pi-calculus in which input and output are symmetric and restriction is the only binder. We highlight a major difference between these calculi and the pi-calculus from the point of view of types, proving some impossibility results for subtyping in fusion calculi. We propose a modification of fusion calculi in which the name equivalences produced by fusions are replaced by name preorders, and with a distinction between positive and negative occurrences of names. The resulting calculus allows us to import subtype systems, and related results, from the pi-calculus. We examine the consequences of the modification on behavioural equivalence (e.g., context-free characterisations of barbed congruence) and expressiveness (e.g., full abstraction of the embedding of the asynchronous pi-calculus).

1. Introduction

The π-calculus is the paradigmatical name-passing calculus, that is, a calculus where names (a synonym for "channels") may be passed around. Key aspects for the success of the π-calculus are the minimality of its syntax and its expressiveness. The π-calculus (often simply referred to as 'π') is notably able to represent functions, objects, and various kinds of distributed systems and protocols. This expressiveness comes at a price: often, desirable behavioural properties, or algebraic laws, fail. The reason is that, when employing π-calculus to describe a system, one normally follows a discipline that governs how names can be used. The discipline can be made explicit by means of *types*. Types bring in other benefits, notably the possibility of statically detecting many programming errors. Types are indeed a fundamental aspect of the π-calculus theory, and one of the most important differences between name-passing calculi and process calculi such as CCS in which names may not be passed.

One of the basic elements in type systems for name-passing calculi is the possibility of separating the capabilities for actions associated to a name, e.g., the capability of using a name in input or in output. The control of capabilities has behavioural consequences, because it allows one to express constraints on the use of names.

For a simple example, consider a process P that implements two distinct services `FACT` and `EXP`, to compute the factorial and the self-exponentiation ($n \mapsto n^n$) of an integer. The two services are accessible using channels `fact` and `exp`, that must be communicated to clients of the services. We assume here only two clients, that receive the channels via c_1 and c_2:

$$P \triangleq (\nu\, \mathtt{fact}, \mathtt{exp})\left(\overline{c_1}\langle \mathtt{fact}, \mathtt{exp}\rangle.\overline{c_2}\langle \mathtt{fact}, \mathtt{exp}\rangle.(\mathtt{FACT} \mid \mathtt{EXP})\right) \quad (1)$$

* Corresponding author at: LIP, ENS Lyon, site Monod, 46, allée d'Italie, 69364 Lyon Cedex 7, France.
E-mail address: Daniel.Hirschkoff@ens-lyon.fr (D. Hirschkoff).

http://dx.doi.org/10.1016/j.ic.2016.10.003
0890-5401/

We expect that outputs at `fact` or `exp` from the clients are eventually received and processed by the appropriate service. This is however not necessarily the case: a malign client can disrupt the expected protocol by simply offering an input at `fact` or `exp` and then throwing away the values received, or forwarding the values to the wrong service. These misbehaviours are ruled out by a capability type system imposing that the clients only obtain the output capability on the names `fact` and `exp`, when receiving them from c_1 and c_2. The typing rules for capabilities are straightforward [31], and mimic those for the typing of references in imperative languages with subtyping.

The control of capabilities is at the basis of more complex type systems, with a finer control on names. For instance, type systems imposing constraints on successive usages of the names like usage-based type systems and deadlock-detection systems, session types, and so on [21,22,17].

Capabilities are closely related to subtyping. In the example (1), P creates names `fact` and `exp`, and possesses both the input and the output capabilities on them; it however transmits to the clients only a subset of the capabilities (namely the output capability alone). The subset relation on capabilities gives rise to a subtype relation on types. All forms of subtyping for the π-calculus or related calculi in the literature require a discipline on capabilities. Subtyping can also be used to recover well-known forms of subtyping in other computational paradigms, e.g., functional languages or object-oriented languages, when an encoding of terms into processes is enhanced with an encoding of types [33].

Numerous extensions or variants of the π-calculus have been proposed, for various reasons: tailoring the calculus to specific application domains, like security, systems modelling, or distributed programming, adding orthogonal features such as locations, facilitating implementations. An interesting family of variants of the π-calculus are — what we call here — the *fusion calculi*: Fusions [29], Update [28], Explicit Fusions [34], Chi [9], Solos [24]. The beauty of fusion calculi is the simplification achieved by removing binding from the input construct. Thus input prefixing becomes symmetric to output prefixing, and restriction remains as the only binder.

In fusion calculi, the effect of a synchronisation between an output $\overline{a}b.P$ and an input $ac.Q$ is to fuse the two object names b and c, which become interchangeable. Thus communications produce, step-by-step, an equivalence relation on names. Existing fusion calculi differ in the way the name equivalence is handled. The operational theories of these calculi (behavioural equivalences and algebraic theory), and their implementation, have been widely studied, see e.g. [29,11,30,7,2]. Expressiveness is not affected when moving from π to fusion calculi. In certain cases, actually, the free input brings even extra power. For instance it can yield a simple representation of delayed input, and of strong reduction strategies of the λ-calculus. Such behaviours can be encoded in π only under certain syntactical constraints, and in a rather indirect way [26].

Like in the case of the π-calculus, however, the expressiveness of fusion calculi makes desirable behavioural properties fail. For instance, the problems of misbehaving clients in (1) remain. Actually, in fusion calculi additional problems arise; for example a client receiving the two channels `fact` and `exp` along c_i could fuse them using an input $c_i(n,n).R$. Now `fact` and `exp` are indistinguishable, and a request to one of these services can reach any of the two (moreover, if the definition of a service involves recursion, a recursive call could be redirected towards the other service).

In this work, we study the addition of types to fusion calculi; more generally, to single-binder calculi, where input is not binding (fusion calculi are single-binder, and, in addition, reductions fuse names). We begin by highlighting a striking difference between the π-calculus and fusion calculi. We present impossibility results for subtyping (and hence for general capability-based type systems, implicitly or explicitly involving subtyping) in fusion calculi. In the statement of these results, we assume a few basic properties of type systems for name-passing calculi, such as strengthening, weakening and type soundness, and the validity of the ordinary typing rules for the base operators of parallel composition and restriction.

These results do not rule out completely the possibility of having subtyping or capabilities in fusion calculi, because of the few basic assumptions we make. They do show, however, that such type systems with subtyping and capabilities would have to be more complex than those for ordinary name-passing calculi such as the π-calculus, or require modifications or constraints in the syntax of the calculi. We actually present one of such systems in Section 7.2, for a subset of the calculus of Fusions.

The main goal of the paper is to understand how type systems from the π-calculus can be adapted to a single-binder calculus. Intuitively, the impossibility of adapting capability types to fusion calculi arises because at the heart of the operational semantics for fusion calculi is an equivalence relation on names, generated through name fusions. In contrast, subtyping and capability systems are built on a preorder relation (be it subtyping, or set inclusion among subsets of capabilities). The equivalence on names forces one to have an equivalence also on types, instead of a preorder. The crux of our solution is to replace the equivalence on names with a preorder, and a distinction on occurrences of names, between 'positive' and 'negative'.

In the resulting single-binder calculus, $\pi\,\mathtt{P}$ ('π with Preorder'), reductions generate a preorder. The basic reduction rule is

$$\overline{c}a.P \mid cb.Q \longrightarrow P \mid Q \mid a/b \ .$$

The particle a/b, called an *arc*, sets a to be above b in the name preorder. Such a process may redirect a prefix at b to become a prefix at a.

As a by-product of our analysis, we introduce a notion of *polarity* in the usage of names in $\pi\,\mathtt{P}$. Negative usages of names intuitively correspond to situations in which a name can be replaced by another. In particular, b occurs in negative position in a/b. This is also the case when b occurs in input object position, like in prefix cb above. Other occurrences of names,

including subject of prefixes, are positive. Note that the binding input construct of the π-calculus can be represented in our calculus, and in doing so, $a(x).P$ is translated into a $\pi\mathtt{P}$ process where x occurs negatively.

We show that the I/O (input/output) capability systems of the π-calculus can be adapted to $\pi\mathtt{P}$. The adaptation is the result of a generalisation of the typing rules of the π-calculus, that takes into account the negative and positive occurrences of names. A better understanding of type systems with subtyping in name-passing calculi is a by-product of this study. For instance, the study suggests that it is essential for subtyping that substitutions produced by communications (in $\pi\mathtt{P}$, the substitutions produced by arcs) only affect the positive occurrences of names.

The modification from fusion calculi to $\pi\mathtt{P}$ also brings in behavioural differences. For instance, both in the π-calculus and in $\pi\mathtt{P}$, a process that creates a new name a can have the guarantee that a will remain different from all other known names, even if a is communicated to other processes (only the creator of a can break this, by using a in negative position). This is not true in fusion calculi, where the emission of a may produce fusions between a and other names.

We present two possible semantics for $\pi\mathtt{P}$. They differ on the moment arcs enable substitutions. In the *eager* semantics, arcs may freely act on prefixes; in the *by-need* semantics, arcs act on prefixes only when interactions occur. The eager semantics mainly serves as a decomposition of the by-need semantics into basic rewriting steps, yielding simpler proofs. We provide a characterisation of the reference contextual behavioural equivalence (barbed congruence) as a context-free labelled bisimilarity for the by-need semantics. We also compare and contrast the semantics, both between them and with semantics based on name fusions.

We then analyse the expressiveness of name preorders, by comparing $\pi\mathtt{P}$ with other name passing calculi. To demonstrate the proximity with the π-calculus, we show that the embedding of the asynchronous π-calculus into $\pi\mathtt{P}$ is fully abstract (full abstraction of the encoding of the π-calculus into fusion calculi fails). We also exhibit an encoding of Explicit Fusions into $\pi\mathtt{P}$, where fusions are translated into pairs of arcs.

We conclude by examining the following syntactic constraint in single-binder calculi: each name, say b, may occur at most once in negative position (this corresponds to input object, as in $ab.P$, or to the source of an arc, as in a/b). Under this constraint, the two semantics for $\pi\mathtt{P}$, eager and by-need, coincide. In fusion calculi, the constraint allows us to import the I/O type systems from π. The constraint is however rather strong, and, in fusion calculi, breaks the semantic duality between inputs and outputs, whereby input and output prefixes can be swapped.

Further related work. Capability types and subtyping for processes have been introduced in [31], and used in many type systems, like, e.g., in [13], notably yielding typed behavioural equivalences. We do not address richer forms of types in our paper, like, e.g., behavioural types [17] or semantic subtyping [5]. Central to $\pi\mathtt{P}$ is the preorder on names, that breaks the symmetry of name equivalence in fusion-like calculi. In Update [28] and (asymmetric versions of) Chi [9], reductions produce ordinary substitutions on names. In practice, however, substitutions are not much different from fusions: a substitution $\{a/b\}$ fuses a with b *and* makes a the representative of the equivalence class. Still, substitutions are directed, and in this sense Update and Chi look closer to $\pi\mathtt{P}$ than the other fusion calculi. Update was refined to the fusion calculus [29] because of difficulties in the extension with polyadicity. Another major difference for Update and Chi with respect to $\pi\mathtt{P}$ is that in the former calculi substitutions replace all occurrences of names, whereas $\pi\mathtt{P}$ takes into account the distinction between positive and negative occurrences.

The question of controlling the fusion of private names has been addressed in the U-calculus [3]. This calculus makes no distinction between input and output, and relies on two forms of binding to achieve a better control of scope extrusion, leading to a sensible behavioural theory that encompasses Fusions and π. Thus the calculus is not single-binder. It is unclear how capability types could be defined in the U-calculus, as it does not have primitive constructs for input and output.

This paper is an extended version of [14]. With respect to that work, we provide here more detailed definitions and proofs, especially in Sections 5 and 6. Additionally, some results in Section 6 (about the relationship between $\pi\mathtt{P}$ and some fusion calculi) were not given in [14].

Paper outline. Section 2 gives some background. In Section 3, we present our impossibility results on type systems for fusion-like calculi. Section 4 defines $\pi\mathtt{P}$ and its type system. The behavioural theory of $\pi\mathtt{P}$ is explored in Section 5, and we give some expressiveness results in Section 6. Section 7 studies the syntactical restriction mentioned above (unique negative occurrence of names), which can be applied to $\pi\mathtt{P}$ and Fusions. We discuss future work in Section 8.

2. Background on name-passing calculi

In this section we group terminology and notation that are common to all the calculi discussed in the paper. For simplicity of presentation, all calculi in the paper are finite. The addition of operators like replication for writing infinite behaviours goes as expected. The results in the paper would not be affected.

We informally call *name-passing* the calculi in the π-calculus tradition, which have the usual constructs of parallel composition and restriction, and in which computation is interaction between input and output constructs. *Names* identify the pairs of matching inputs/outputs, and the values transmitted may themselves be names. Restriction is a binder for the names; in some cases the input may be a binder too. Examples of these calculi are the π-calculus, the asynchronous π-calculus [16,4], the Join calculus [8], the Distributed π-calculus [12], the fusion calculus [29], and so on. Binders support the usual alpha-conversion mechanism, and give rise to the usual definitions of free and bound names.

We use $a, b, \ldots$ to range over names. In a free input $ab.P$, bound input $a(b).P$, output $\overline{a}b.P$, we call a the *subject* of the prefix, and b the *object*. We sometimes abbreviate prefixes as $a.P$ and $\overline{a}.P$ when the object carried is not important. We omit trailing $\mathbf{0}$, for instance writing $\overline{a}b$ in place of $\overline{a}b.\mathbf{0}$. We write $P\{a/b\}$ for the result of applying the substitution of b with a in P.

Calculi having fusions. When restriction is the only binder (hence the input construct is not binding), we say that the calculus *has a single binder*.

If in addition interaction involves fusion between names, so that we have

$$(\boldsymbol{\nu} c)\,(\overline{a}b.P \mid ac.Q \mid R) \quad \Longrightarrow \quad (P \mid Q \mid R)\{b/c\}\ , \tag{2}$$

we say that the calculus *has name-fusions*, or, more briefly, has *fusions*. In (2), we suppose that the calculus comes with a reduction relation (which usually is closed under structural congruence), and $\Longrightarrow$ stands for an arbitrary number of reduction steps. Moreover, when the calculus has no prefixes, as it is the case in Solos, we omit P and Q.

(We are not requiring that (2) is among the rules of the operational semantics of the calculus, just that (2) holds. The shape of (2) has been chosen so to capture the existing calculi; the presence of R allows us to capture also the calculus of Solos.) It is intended that calculi with name-fusions have a single binder.

In this sense, all single-binder calculi in the literature (Update [28], Chi [9], Fusions [29], Explicit Fusions [11], Solos [24]) have fusions. In Section 4 we will introduce a single-binder calculus without fusions. We shall add another element to this family in Section 4, in which, in contrast with existing calculi in the family, interaction does not involve fusion of names.

In all calculi in the paper, (reduction-closed) barbed congruence will be our reference behavioural equivalence. Its definition only requires a reduction relation, $\longrightarrow$, and a notion of barb on names, $\downarrow_a$. Intuitively, a barb at a holds for a process if that process can accept an offer of interaction at a from its environment.

Definition 1 *(Reduction-closed barbed congruence).* Let $\mathcal{L}$ be a process calculus, in which a reduction relation $\longrightarrow_{\mathcal{L}}$ and barb predicates $\downarrow_a^{\mathcal{L}}$, for each a in a given set of names, have been defined. A relation $\mathcal{R}$ on the processes of $\mathcal{L}$ is:

- *context-closed* if $P\,\mathcal{R}\,Q$ implies $C[P]\,\mathcal{R}\,C[Q]$ for each context C of $\mathcal{L}$;
- *barb-preserving* if $P\,\mathcal{R}\,Q$ implies that for any name a, $P\downarrow_a^{\mathcal{L}}$ implies $Q\downarrow_a^{\mathcal{L}}$;
- *reduction-closed* if $P\,\mathcal{R}\,Q$ and $P\longrightarrow_{\mathcal{L}} P'$ imply there is Q' such that $Q\longrightarrow_{\mathcal{L}} Q'$ and $P'\,\mathcal{R}\,Q'$.

Then *reduction-closed barbed congruence in* $\mathcal{L}$, written $\simeq_{\mathcal{L}}$, is the largest symmetric relation on the processes of $\mathcal{L}$ that is context-closed, reduction-closed, and barb-preserving.

The weak version of the equivalence, *weak reduction-closed barbed congruence*, written $\approx_{\mathcal{L}}$, is defined in the usual way, replacing the relation $\longrightarrow_{\mathcal{L}}$ with its reflexive and transitive closure $\Longrightarrow_{\mathcal{L}}$, and the barbs $\downarrow_a^{\mathcal{L}}$ with the weak barbs $\Downarrow_a^{\mathcal{L}}$, where $\Downarrow_a^{\mathcal{L}}$ is the composition of the relations $\Longrightarrow_{\mathcal{L}}$ and $\downarrow_a^{\mathcal{L}}$ (i.e., the barb is visible after some internal actions).

3. Typing and subtyping with fusions

We consider typed versions of languages with fusions. We show that in such languages it is impossible to have a non-trivial subtyping, assuming a few simple and standard typing properties of name-passing calculi.

We use T, U to range over types, and Γ, Δ to range over type environments, i.e., partial functions from names to types. We write $\mathrm{dom}(\Gamma)$ for the domain of Γ, that is, the set of names on which Γ is defined. In name-passing calculi, a type system assigns a type to each name. Typing judgements are of the form $\Gamma \vdash P$ (process P respects the type assignments in Γ), and $\Gamma \vdash a : T$ (name a can be assigned type T in Γ).[1]

The following are the standard typing rules for the operators of parallel composition and restriction in name-passing calculi:

$$\frac{\Gamma \vdash P_1 \qquad \Gamma \vdash P_2}{\Gamma \vdash P_1 \mid P_2} \qquad \frac{\Gamma, x : T \vdash P}{\Gamma \vdash (\boldsymbol{\nu} x)\ P} \tag{3}$$

The first rule says that any two processes typed in the same type environment can be composed in parallel. The second rule handles name restriction.

In resource-sensitive type systems, i.e., those for linearity [23,20] and receptiveness [33], where one counts certain occurrences of names, the rule for parallel composition has to be modified. As mentioned earlier, in this paper we stick to basic type systems, ignoring resource consumption.

[1] We consider in this paper basic type systems and basic properties for them; more sophisticated type systems exist where processes have a type too, e.g., behavioural type systems.

In name-passing calculi, the basic type construct is the channel (or connection) type $\sharp\, T$. This is the type of a name that may carry, in an input or an output, values of type T. Consequently, we also assume that the following rule for prefixes $ab.P$ and $\overline{a}b.P$ is *admissible*:

$$\frac{\Gamma(a) = \sharp\, T \qquad \Gamma(b) = T \qquad \Gamma \vdash P}{\Gamma \vdash \alpha.P}\ \alpha \in \{ab, \overline{a}b\} \tag{4}$$

(Here we consider input and output prefixes with a continuation; in calculi in which prefixes may not have a continuation, e.g., the asynchronous π-calculus or Solos, P would be missing from the rules.) In the rule, the type of the subject and of the object of the prefix are compatible.

Again, these need not be the typing rules for prefixes; we are just assuming that the rules are valid in the type system. We can observe that the standard rule for prefix would have, as hypotheses,

$$\Gamma \vdash a : \sharp\, T \qquad \Gamma \vdash b : T\ .$$

These imply, but are not equivalent to, the hypotheses in (4), for instance in presence of subtyping.

Fundamental properties of type systems are:

- Subject Reduction (or Type Soundness): if $\Gamma \vdash P$ and $P \to P'$, then $\Gamma \vdash P'$;
- Weakening: if $\Gamma \vdash P$ and a is fresh, then $\Gamma, a : T \vdash P$;
- Strengthening: whenever $\Gamma, a : T \vdash P$ and a is fresh for P, then $\Gamma \vdash P$;
- Closure under injective substitutions: if $\Gamma, a : T \vdash P$ and b is fresh, then $\Gamma, b : T \vdash P\{b/a\}$.

Definition 2. A typed calculus with single binder is *plain* if it satisfies Subject Reduction, Weakening, Strengthening, Closure under injective substitutions, and the typing rules (3) and (4) are admissible.

If the type system admits subtyping, then another fundamental property is narrowing, which authorises, in a typing environment, the specialisation of types:

- Narrowing: if $\Gamma, a : T \vdash P$ and $U \leq T$ then also $\Gamma, a : U \vdash P$.

When narrowing holds, we say that the calculus *supports narrowing*.

A typed calculus *has trivial subtyping* if, whenever $T \leq U$, we have $\Gamma, a : T \vdash P$ iff $\Gamma, a : U \vdash P$. When this is not the case (i.e., there are T, U with $T \leq U$, and T, U are not interchangeable in all typing judgements) we say that the calculus has *meaningful* subtyping.

Standard i/o-types for the π-calculus provide an example of meaningful subtyping: we have for instance $\sharp\, T \leq \circ T$, and these types are not interchangeable in all typing derivations.

We are now ready to prove that, under the assumptions of Definition 2, a calculus with fusions may only have trivial subtyping.

Theorem 3. *A typed calculus with fusions that is plain and supports narrowing has trivial subtyping.*

Proof. We define the following context:

$$E \triangleq (\boldsymbol{\nu} c, b)(\overline{u}b \mid uc \mid \overline{v}a \mid vc \mid [\cdot])\ .$$

Note that in E we only use b as an output object (in $\overline{u}b$). The intention is that, given some fresh names, u, v, c, and some process P, $E[P]$ should reduce to $P\{a/b\}$. Indeed, by applying hypothesis (2) twice, we have

$$\begin{aligned} E[P] = (\boldsymbol{\nu} b, c)(\overline{u}b \mid uc \mid \overline{v}a \mid vc \mid P) &\Longrightarrow (\boldsymbol{\nu} b)(\overline{v}a \mid vb \mid P\{b/c\}) \\ &= (\boldsymbol{\nu} b)(\overline{v}a \mid vb \mid P) \\ &\Longrightarrow P\{a/b\}\ . \end{aligned}$$

Suppose $U \leq T$, we show $\Gamma, a : T \vdash P$ iff $\Gamma, a : U \vdash P$, which means that the type system has trivial subtyping. The implication from left to right is narrowing.

To prove the implication from right to left, suppose $\Gamma, a : U \vdash P$, and prove $\Gamma, a : T \vdash P$. We can pick some name b fresh for P, and deduce, by Closure under injective name substitution, $\Gamma, b : U \vdash P\{b/a\}$.

In the typing environment $\Gamma, b{:}U, u{:}\sharp\, T, v{:}\sharp\, T, c{:}T, a{:}T$ the process $\overline{u}b$ is well-typed thanks to Narrowing and Weakening, hence so is $(\overline{u}b \mid uc \mid \overline{v}a \mid vc \mid P\{b/a\})$. By the restriction rule we get $\Gamma, a{:}T, u{:}\sharp\, T, v{:}\sharp\, T \vdash E[P\{b/a\}]$, the latter reducing to $P\{b/a\}\{a/b\}$ by (2). Since b has been taken fresh, $P\{b/a\}\{a/b\} = P$. Hence, by Subject Reduction, $\Gamma, a{:}T, u{:}\sharp\, T, v{:}\sharp\, T \vdash P$. We finally deduce $\Gamma, a : T \vdash P$ by Strengthening. □

One may wonder whether, in Theorem 3, more limited forms of narrowing, or a narrowing in the opposite direction, would permit some meaningful subtyping. Narrowing is interesting when it is used to modify the type of the values exchanged along a name, that is, the type of the object of a prefix. (In process calculi, communication is the analog of application for functional languages, and changing the type of a prefix object is similar to changing the type of a function or of its argument.) In other words, disallowing narrowing on objects would make subtyping useless. We now show that allowing *any* form of narrowing, on one prefix object, would force subtyping to be trivial.

Theorem 4. *We consider a typed calculus with fusions which is plain. We suppose that there is at least one prefix α with object b, where b is different from the subject of α. We further suppose that there are two types S and T such that $S \leq T$ and one of the following forms of narrowing holds for all Γ:*

(1) *whenever $\Gamma, b : T \vdash \alpha.\mathbf{0}$, we also have $\Gamma, b : S \vdash \alpha.\mathbf{0}$;*
(2) *whenever $\Gamma, b : S \vdash \alpha.\mathbf{0}$, we also have $\Gamma, b : T \vdash \alpha.\mathbf{0}$.*

Then S and T are interchangeable in all typing judgements.

As a consequence, authorising one of the above forms of narrowing for all S and T such that $S \leq T$ implies that the calculus has trivial subtyping.

Proof. For all Δ we prove that $\Delta, x : T \vdash P$ iff $\Delta, x : S \vdash P$. Let x_1, x_2, a_1 and a_2 be fresh names, we define

$$\Delta_i \triangleq \Delta,\ x_i : T,\ x_{3-i} : S\ .$$

We prove that $\Delta_i \vdash P\{x_1/x\}$ implies $\Delta_i \vdash P\{x_2/x\}$ for all $i \in \{1, 2\}$. This is enough to conclude, using Weakening, Strengthening and Closure under injective substitutions. We rely on process $D \triangleq \overline{a_1}x_1 \mid \overline{a_2}x_2 \mid a_1 y \mid a_2 y$ to simulate the substitution of x_1 with x_2:

$$(\boldsymbol{\nu} x_1, y)(D \mid P\{x_1/x\}) \implies P\{x_2/x\}\ . \tag{5}$$

This way, it is enough to find some types T_{a_1} T_{a_2}, T_y such that $\Delta' \vdash D$, with $\Delta' = \Delta_i, a_1 : T_{a_1}, a_2 : T_{a_2}, y : T_y$. We suppose that the subject of α is either a_1 or a_2, picking the most suitable choice, without loss of generality. We suppose as well that the object b is the most suitable among x_1, x_2 or y.

To illustrate how we choose such types, we provide an example. Suppose: that $i = 1$ (i.e., the type of x_1 is T and the type of x_2 is S), that narrowing of type (1) holds, and that α is an output. Then we take α to be $\overline{a_2}x_2$ and we choose $T_{a_1} = T_{a_2} = \sharp\, T$ and $T_y = T$. We then have $\Delta' \vdash D$ quite easily: the interesting part is to obtain $a_2 : \sharp\, T,\ x_2 : S \vdash \overline{a_2}x_2$ from the composition of the trivial $a_2 : \sharp\, T,\ x_2 : T \vdash \overline{a_2}x_2$ and from narrowing of type (1). Then, by Subject Reduction, with (5) and Strengthening, we get that $\Delta_1 \vdash P\{x_2/x\}$. This particular case corresponds to the first line of the following table.

The other choices for T_{a_1} T_{a_2}, and T_y are listed in the table, following three parameters: is i equal to 1 or to 2? Which form of narrowing, between (1) and (2), holds? Is α an output or an input?

i	form	α	T_{a_1}	T_{a_2}	T_y
1	(1)	$\overline{a_2}x_2$	$\sharp\, T$	$\sharp\, T$	T
1	(1)	$a_1 y$	$\sharp\, T$	$\sharp\, S$	S
1	(2)	$\overline{a_1}x_1$	$\sharp\, S$	$\sharp\, S$	S
1	(2)	$a_2 y$	$\sharp\, T$	$\sharp\, S$	T
2	(1)	$\overline{a_2}x_2$	$\sharp\, T$	$\sharp\, T$	T
2	(1)	$a_2 y$	$\sharp\, S$	$\sharp\, T$	S
2	(2)	$\overline{a_1}x_1$	$\sharp\, S$	$\sharp\, S$	S
2	(2)	$a_1 y$	$\sharp\, S$	$\sharp\, T$	T

Using the hypothesis on α, we can prove in all these cases that $\Delta' \vdash D$, relying on the fact that the type system is plain. The latter hypothesis also gives us $\Delta' \vdash P\{x_1/x\}$. We use rules from (3) and Subject Reduction to deduce that $\Delta' \vdash P\{x_2/x\}$. From this, Strengthening is enough to conclude. □

Remark 5. Theorems 3 and 4 apply to all fusion calculi: Fusions, Explicit Fusions, Update, Chi, Solos.

In calculi of mobile processes, the basis for having subtyping is a type system for *capabilities*, usually the input/output (I/O) capabilities [31,13]. Another consequence of Theorems 3 and 4 is that it is impossible, in plain calculi with fusions, to have an I/O type system; more generally, it is impossible to have any capability-based type system that supports meaningful subtyping.

Actually, to apply the theorems, it is not even necessary for the capability type system to have an explicit notion of subtyping. For Theorem 3, it is sufficient to have sets of capabilities with a non-trivial ordering under inclusion, meaning

that we can find two capability types T and U such that whenever $\Gamma, a : U \vdash P$ holds then also $\Gamma, a : T \vdash P$ holds, but not the converse (e.g., T provides more capabilities than U). We could then impose a subtype relation $\leq$ on types, as the least preorder satisfying $T \leq U$. Theorem 3 then tells us that type soundness and the other properties of Definition 2 would require also $U \leq T$ to hold, i.e., T and U are interchangeable in all typing judgements. In other words, the difference between the capabilities in T and U has no consequence on typing. Similarly, to apply Theorem 4 it is sufficient to find two capability types T and U and a single prefix in whose typing the type U can replace T.

4. A calculus with name preorders

4.1. Preorders, positive and negative occurrences

We now refine the fusion calculi by replacing the equivalence relation on names generated through communication by a preorder, yielding calculus $\pi\,\mathbb{P}$ ('π with Preorder'). As the preorder on types given by subtyping enables promotions between related types, so the preorder on names of $\pi\,\mathbb{P}$ enables promotions between related names. Precisely, if a is below a name b in the preorder, then a prefix at a may be promoted to a prefix at b and then interact with another prefix at b. Thus an input $av.P$ may interact with an output $\overline{b}w.Q$; and, if also c is below b, then $av.P$ may as well interact with an output $\overline{c}z.R$.

The ordering on names is introduced by means of the *arc* construct, a/b, that declares the *source* b to be below the *target* a. The remaining operators are as for fusion calculi (i.e., those of the π-calculus with bound input replaced by free input). Restriction is the only binder.

$$P ::= \mathbf{0} \mid P \mid P \mid \overline{a}b.P \mid ab.P \mid (\boldsymbol{\nu} a)P \mid a/b \ .$$

The semantics of the calculus is given by a reduction relation, which is based on structural congruence:

Definition 6 *(Structural congruence).* Structural congruence on $\pi\,\mathbb{P}$, written $\equiv$, is the smallest congruence satisfying associativity and commutativity of $\mid$ and the following rules:

$$P \mid \mathbf{0} \equiv P \qquad (\boldsymbol{\nu} a)\mathbf{0} \equiv \mathbf{0} \qquad (\boldsymbol{\nu} a)(\boldsymbol{\nu} b)P \equiv (\boldsymbol{\nu} b)(\boldsymbol{\nu} a)P \qquad (\boldsymbol{\nu} a)(P \mid Q) \equiv ((\boldsymbol{\nu} a)P) \mid Q \ \text{ if } a \notin \mathrm{fn}(Q) \ .$$

We explain the effect of reduction by means of contexts, rather than separate rules for each operator. Contexts provide a more succinct presentation, and a simpler comparison with an alternative semantics (Section 5). An *active context* is a context in which the hole may reduce. Thus the only difference between active contexts and ordinary contexts is that in active contexts, the hole may not occur underneath a prefix. We use C to range over (ordinary) contexts, and E for active contexts.

We now define the reduction relation (the subscript in $\longrightarrow_{\mathrm{ea}}$, for "eager", will distinguish this from the alternative semantics discussed in Section 5.1):

Definition 7 *(Eager reduction).* Eager reduction, written $\longrightarrow_{\mathrm{ea}}$, is defined by the following rules:

$$\text{R-SCon}: \ \frac{P \equiv E[Q] \qquad Q \longrightarrow_{\mathrm{ea}} Q' \qquad E[Q'] \equiv P'}{P \longrightarrow_{\mathrm{ea}} P'} \qquad \text{R-SubOut}: \ a/b \mid \overline{b}c.Q \longrightarrow_{\mathrm{ea}} a/b \mid \overline{a}c.Q$$

$$\text{R-Inter}: \ \overline{a}b.P \mid ac.Q \longrightarrow_{\mathrm{ea}} P \mid Q \mid b/c \qquad \text{R-SubInp}: \ a/b \mid bc.Q \longrightarrow_{\mathrm{ea}} a/b \mid ac.Q$$

We write $\Longrightarrow_{\mathrm{ea}}$ for the reflexive and transitive closure of $\longrightarrow_{\mathrm{ea}}$.

Rule R-Inter shows that communication generates an arc. Rules R-SubOut and R-SubInp show that arcs only act on the subject of prefixes; moreover, they only act on *unguarded* prefixes (i.e., prefixes that are not underneath another prefix). The rules also show that arcs are persistent processes. Acting only on prefix subjects, arcs can be thought of as particles that "redirect prefixes": an arc a/b redirects a prefix at b towards a higher name a. These design choices lead to natural commitments in the eager version, and are amenable to a more efficient implementation — a similar approach is chosen to implement efficiently the fusion calculi using the Fusion Machine [10].

Arcs remind us of special π-calculus processes, called forwarders or wires [18], which under certain hypotheses allow one to model substitutions; as for arcs, so the effect of forwarders is to replace the subject of prefixes. Note also that arcs are different from explicit substitutions: only the latter are binding, and there can be several arcs acting on the same name.

We present below some examples of reduction. We sometimes omit the object part of prefixes, when it is not important, writing e.g. $e.P$.

$$
\begin{array}{rl}
 & \overline{a}c.\overline{c}a.e.P \mid ad.de.\overline{a}.Q \\
\longrightarrow_{\mathrm{ea}} & \overline{c}a.e.P \mid de.\overline{a}.Q \mid c/d \\
\longrightarrow_{\mathrm{ea}} & \overline{c}a.e.P \mid ce.\overline{a}.Q \mid c/d \\
\longrightarrow_{\mathrm{ea}} & e.P \mid \overline{a}.Q \mid c/d \mid a/e \\
\longrightarrow_{\mathrm{ea}} & a.P \mid \overline{a}.Q \mid c/d \mid a/e \\
\longrightarrow_{\mathrm{ea}} & P \mid Q \mid c/d \mid a/e
\end{array}
$$

Reductions can produce multiple arcs that act on the same name. This may be used to represent certain forms of choice, as in the following processes:

$$
\begin{array}{l}
(\boldsymbol{\nu} h, k)\ (bu.cu.\overline{u} \mid \overline{b}h.h.P \mid \overline{c}k.k.Q) \\
\Longrightarrow_{\mathrm{ea}}\ (\boldsymbol{\nu} h, k)\ (\overline{u} \mid h/u \mid k/u \mid h.P \mid k.Q)\ .
\end{array}
$$

Arcs h/u and k/u may act on $\overline{u}$, and are therefore in competition with each other. The outcome of the competition determines which process between P and Q is activated. For instance, reduction may continue as follows:

$$
\begin{array}{ll}
\longrightarrow_{\mathrm{ea}} & (\boldsymbol{\nu} h, k)\ (\overline{k} \mid h/u \mid k/u \mid h.P \mid k.Q) \\
\longrightarrow_{\mathrm{ea}} & (\boldsymbol{\nu} h, k)\ (h/u \mid k/u \mid h.P \mid Q)\ .
\end{array}
$$

The semantics above formalises an *eager* behaviour for arcs: arcs act on prefixes, substituting their subjects, regardless of the possible consequences on future interactions. For instance, in the example above, the arc k/u could have acted on $\overline{u}$ even in the absence of the matching prefix $k.Q$. Under the eager semantics the substitution produced by an arc is a commitment — replacing a with b in a prefix commits that prefix to go along a channel that is at least as high as b in the preorder. In Section 5.1, we formalise a *by-need* behaviour of arcs, in which substitutions are performed only when reduction takes place. In by-need semantics, there is no separate action of commitment.

Definition 8 *(Positive and negative occurrences).* In an input $ab.P$ and an arc a/b, the name b has a *negative occurrence*. All other occurrences of names in input, output and arcs are *positive occurrences*.

An occurrence in a restriction $(\boldsymbol{\nu} a)$ is neither negative nor positive, intuitively because restriction acts only as a binder, and does not stand for an usage of the name (in particular, it does not take part in a substitution).

Negative occurrences are particularly important, as by properly tuning them, different usages of names may be obtained. For instance, a name with zero negative occurrence is a constant (i.e., it is a channel, and may not be substituted); and a name that has a single negative occurrence is like a π-calculus name bound by an input (we investigate this situation in Section 7.1). Names having two or more negative occurrences are specific to $\pi\mathtt{P}$.

The number of negative occurrences of a name is invariant under reduction. Note also that the set of positive occurrences is non-increasing.

Lemma 9. *If $P \longrightarrow_{\mathrm{ea}} P'$ then for each b, the number of negative occurrences of b in P and P' is the same.*

4.2. Types

We now show that the I/O capability type system and its subtyping can be transplanted from π to $\pi\mathtt{P}$.

In the typing rules for I/O-types in the (monadic) π-calculus [31], two additional types are introduced: $\mathtt{o}\,T$, the type of a name that can be used only in output and that carries values of type T; and $\mathtt{i}\,T$, the type of a name that can be used only in input and that carries values of type T. The subtyping rules stipulate that $\mathtt{i}$ is covariant, $\mathtt{o}$ is contravariant, and $\sharp$ is invariant. Subtyping is brought up into the typing rules through the subsumption rule. The most important typing rules are those for input and output prefixes; for input we have (in [31]):

$$
\text{T-InpBound}: \quad \frac{\Gamma \vdash a : \mathtt{i}\,T \qquad \Gamma, b : T \vdash P}{\Gamma \vdash a(b).P}\ .
$$

The π-calculus supports narrowing, and this is essential in the proof of subject reduction.

The type system for $\pi\mathtt{P}$ is presented in Table 1. It relies on the subtyping relation, written $T_1 \leq T_2$, which is used to define the typing judgement for names, written $\Gamma \vdash a : T$. The typing judgement for processes, written $\Gamma \vdash P$, states that process P is well-typed according to the typing hypotheses expressed in Γ. In the two typing judgements, Γ ranges over a finite map from names to types; we write $\Gamma(a)$ for the type associated to a in Γ. With respect to the π-calculus, only the rule for input needs an adjustment, as $\pi\mathtt{P}$ uses free, rather than bound, input. The idea in rule T-InpFree of $\pi\mathtt{P}$ is however the same as in rule T-InpBound of π: we look up the type of the object of the prefix, say T, and we require $\mathtt{i}\,T$ as the type for the subject of the prefix. To understand the typing of an arc a/b, recall that such an arc allows one to replace b with a. Rule T-Arc essentially ensures that a has at least as many capabilities as b, in line with the intuition for subtyping in capability type systems.

Table 1
The type system of $\pi\mathtt{P}$.

Types (**1** is the unit type): $T ::= \mathtt{i}\,T \mid \mathtt{o}\,T \mid \sharp\,T \mid \mathbf{1}$

Subtyping rules:

$$\frac{}{\sharp\,T \leq \mathtt{i}\,T} \qquad \frac{}{\sharp\,T \leq \mathtt{o}\,T} \qquad \frac{S \leq T}{\mathtt{i}\,S \leq \mathtt{i}\,T} \qquad \frac{S \leq T}{\mathtt{o}\,T \leq \mathtt{o}\,S} \qquad \frac{}{T \leq T} \qquad \frac{S \leq T \quad T \leq U}{S \leq U}$$

Typing rules:

$$\text{T-Name}\ \frac{}{\Gamma, a : T \vdash a : T} \qquad \text{Subsumption}\ \frac{\Gamma \vdash a : S \quad S \leq T}{\Gamma \vdash a : T} \qquad \text{T-Res}\ \frac{\Gamma, a : T \vdash P}{\Gamma \vdash (\nu a)P} \qquad \text{T-Par}\ \frac{\Gamma \vdash P \quad \Gamma \vdash Q}{\Gamma \vdash P \mid Q} \qquad \text{T-Nil}\ \frac{}{\Gamma \vdash \mathbf{0}}$$

$$\text{T-Out}\ \frac{\Gamma \vdash a : \mathtt{o}\,T \quad \Gamma \vdash b : T \quad \Gamma \vdash P}{\Gamma \vdash \overline{a}b.P} \qquad \text{T-InpFree}\ \frac{\Gamma \vdash a : \mathtt{i}\,\Gamma(b) \quad \Gamma \vdash P}{\Gamma \vdash ab.P} \qquad \text{T-Arc}\ \frac{\Gamma \vdash a : \Gamma(b)}{\Gamma \vdash a/b}$$

Common to all premises of T-InpBound, T-InpFree and T-Arc is the look up of the type of names that occur negatively (the source of an arc and the object of an input prefix): the type that appears for b in the hypothesis is precisely the type found in the conclusion (within the process or in Γ). In contrast, the types for positive occurrences may be different (e.g., because of subsumption, $\Gamma \vdash a : \mathtt{i}\,T$ may hold even if $\Gamma(a) \neq \mathtt{i}\,T$).

Note moreover that we cannot type inputs like outputs: consider

$$\text{T-InpFree2-Wrong}: \quad \frac{\Gamma \vdash a : \mathtt{i}\,T \qquad \Gamma \vdash b : T}{\Gamma \vdash ab} \ .$$

Rule T-InpFree2-Wrong would accept, for instance, an input ab in an environment Γ where $a : \mathtt{i}\,\mathtt{i}\,\mathbf{1}$ and $b : \sharp\,\mathbf{1}$. Using subtyping and subsumption, we could then derive $\Gamma \vdash b : \mathtt{i}\,\mathbf{1}$. In contrast, rule T-InpFree, following the input rule of the π-calculus, makes sure that the object of the input does not have too many capabilities with respect to what is expected in the type of the subject of the input. Enforcing this constraint is necessary for subject reduction. As a counterexample, consider $\Gamma = a : \sharp\,\mathtt{i}\,\mathbf{1}$, $b : \sharp\,\mathbf{1}$, $c : \mathtt{i}\,\mathbf{1}$, and assume we are using rule T-InpFree2-Wrong. We would have $\Gamma \vdash P$, for $P \triangleq ab \mid \overline{a}c \mid \overline{b}$. However, $P \longrightarrow_{\mathrm{ea}} c/b \mid \overline{b} \longrightarrow_{\mathrm{ea}} c/b \mid \overline{c}$, and the final derivative is not typable under Γ (as Γ only authorises inputs at c).

In $\pi\mathtt{P}$, narrowing depends on the negative or positive occurrences of a name.

Theorem 10 *(Polarised narrowing). Let T_1 and T_2 be two types such that $T_1 \leq T_2$.*

1. *If a occurs only positively in P, then $\Gamma, a : T_2 \vdash P$ implies $\Gamma, a : T_1 \vdash P$.*
2. *If a occurs only negatively in P, then $\Gamma, a : T_1 \vdash P$ implies $\Gamma, a : T_2 \vdash P$.*
3. *If a occurs both positively and negatively in P, then it is in general unsound to replace, in a typing $\Gamma \vdash P$, the type of a in Γ with a subtype or supertype.*

Proof.

1. All premises involving a are of the form $\Gamma(a) \leq T$ for some T. Applying the transitivity rule for the subtyping relation suffices to conclude.
2. When a appears only negatively, premises are of the form $T \leq \Gamma(a)$. Indeed, the premise $\Gamma \vdash a : \Gamma(b)$ (in rule T-Arc) is equivalent to $\Gamma(a) \leq \Gamma(b)$ and the premise $\Gamma \vdash a : \mathtt{i}\,\Gamma(b)$ (in rule T-InpFree) is equivalent to $\exists T\ \Gamma(a) \leq \mathtt{i}\,T \wedge T \leq \Gamma(b)$, by covariance of $\mathtt{i}$.
3. $a : T, b : T, c : T \vdash a/b \mid b/c$ when for example $T = \mathtt{i}\,\sharp\,\mathbf{1}$, but replacing the type of b alone with $\sharp\,\sharp\,\mathbf{1}$ or $\mathtt{i}\,\mathtt{i}\,\mathbf{1}$ is impossible, even if $\sharp\,\sharp\,\mathbf{1} \leq \mathtt{i}\,\sharp\,\mathbf{1}$ and $\mathtt{i}\,\sharp\,\mathbf{1} \leq \mathtt{i}\,\mathtt{i}\,\mathbf{1}$. □

Theorem 10 (specialised to prefixes) does not contradict Theorem 4, because in $\pi\mathtt{P}$, reduction does not satisfy relation (2) presented in Section 2, as the resulting process would not be $(P \mid Q \mid R)\{b/c\}$, but rather $(\nu c)(P \mid Q \mid R \mid b/c)$. In the latter term, names b and c are related by an arc instead of being equated in the whole process, as it is the case after a substitution.

Remark 11. Theorem 10 may be seen as a refinement of the standard narrowing result for name-passing calculi. In the π-calculus, for instance, a free name only has positive occurrences. Hence the usual statement of narrowing corresponds to Theorem 10(1). In an input $a(b).P$, the binder for b represents a negative occurrence, so that if b is free in P then b has both positive and negative occurrences, which means that the type of b may not be modified, as by Theorem 10(3). In contrast, Theorem 10(2) is vacuous in π, as a name b with only negative occurrences is found in an input $a(b).P$ where b is not free in P.

In general, in a name-passing calculus, if a name has only positive occurrences, then its type (be it declared in the typing environment, or in the binding occurrence of that name within the process) may be replaced with a subtype, and

conversely for names with only negative occurrences. The type of names with both positive and negative occurrences may not be changed.

Our system enjoys subject reduction:

Theorem 12 *(Subject reduction). If* $\Gamma \vdash P$ *and* $P \longrightarrow_{\mathrm{ea}} P'$ *then also* $\Gamma \vdash P'$.

The proof of Theorem 12 is straightforward, given Theorem 10 (1) combined with the variance properties of $\mathtt{i}$ and $\mathtt{o}$. The theorem also holds for the by-need version of reduction (Section 5.1).

Remark 13. The subject reduction theorem for $\pi\mathtt{P}$ remains valid if an arc a/b is allowed to act not only on the subject of prefixes, but, more generally, on all positive occurrences of b (for instance occurrences of b as the object of an output prefix). Also, subject reduction does not require that the substitutions generated by an arc are performed one at a time: several substitutions could be performed simultaneously, like in Fusions.

The subject reduction theorem would however break in general if arcs acted also on negative occurrences of b, as the arc would then behave as a fusion. In Section 7, we show that additional constraints allow for a refined version of subject reduction in this case.

5. Behaviours

In this section, we analyse the behavioural theory of $\pi\mathtt{P}$. We first motivate and define an alternative reduction relation. We then study the properties of the induced notion of barbed congruence.

5.1. An alternative semantics: by-need reduction

The operational semantics given to $\pi\mathtt{P}$ in Section 4 is close to a simple rewriting-based implementation, by allowing arcs to act locally, at any time. The effect of an arc is irreversible: the application of an arc a/b to a prefix at b commits that prefix to interact along a name that is greater than, or equal to, a in the preorder among names. A commitment may disable certain interactions, even block a prefix for ever. Consider, e.g.,

$$(\boldsymbol{\nu} a, c)(b v.P \mid \overline{c} w.Q \mid a/b \mid c/b) \ . \tag{6}$$

There is a competition between the two arcs; if the first wins, the process is deadlocked:

$$\longrightarrow_{\mathrm{ea}} \quad (\boldsymbol{\nu} a, c)\,(a v.P \mid \overline{c} w.Q \mid a/b \mid c/b)$$

since a and c are unrelated in the preorder.

We consider here an alternative semantics, in which the action of arcs is not a commitment: arcs come about only when interaction occurs. For this reason we call the new semantics *by-need* (arcs act only when 'needed'), whereas we call *eager* the semantics of Section 4.1 (arcs act regardless of matching prefixes). In this semantics, as in the π-calculus, an interaction involves both a synchronisation and a substitution; however unlike in the π-calculus where the substitution is propagated to the whole term, here substitution only replaces the subject of the interacting prefixes.

Relations on names: preorder, joinability. The formalisation of the new semantics makes use of a preorder on names induced by arcs. An arc is *active* if it is unguarded, i.e., it is not underneath a prefix. The preorder induced by P is the least preorder $\leq$ that includes $b \leq a$ for each active arc a/b in P, similarly for the preorder induced by a context C.

We write $P \triangleright a \curlyvee b$ (this judgement is defined formally below) if $\{a, b\}$ has an upper bound in the preorder induced by P, that is, there is a name that is above both a and b; in this case we also say that a and b are *joinable*. Similarly we write $E \triangleright a \curlyvee b$ for active contexts. For instance, we have $(\boldsymbol{\nu} u)(u/a \mid u/b \mid Q) \triangleright a \curlyvee b$, and $(\boldsymbol{\nu} v)(\overline{v}t \mid (\boldsymbol{\nu} w)(w/v \mid a/w \mid [\cdot])) \triangleright a \curlyvee v$.

Given an active context E, the set of *captured names* of E, $\mathrm{cn}(E)$, is defined as follows: $c \in \mathrm{cn}(E)$ iff the hole occurs in the scope of a restriction on c in E ($\mathrm{cn}(E)$ is included in the set of names that are bound in E, but might be distinct from it):

$$\mathrm{cn}([\cdot]) = \emptyset \qquad \mathrm{cn}(P \mid E) = \mathrm{cn}(E \mid P) = \mathrm{cn}(E) \qquad \mathrm{cn}((\boldsymbol{\nu} a)E) = \{a\} \cup \mathrm{cn}(E)$$

Definition 14 *(Reachability/joinability of names).* We let conditions φ range over $a \leqslant b$, read "b is reachable from a", and $a \curlyvee b$, read "a and b are joinable". In both cases, we have $\mathrm{n}(\varphi) = \{a, b\}$.

We first define a judgement $\Gamma \vdash \varphi$, meaning that the set of conditions Γ implies condition φ, as follows:

$$\frac{}{\Gamma \vdash a \leq a} \qquad \frac{\varphi \in \Gamma}{\Gamma \vdash \varphi} \qquad \frac{\Gamma \vdash b \curlyvee a}{\Gamma \vdash a \curlyvee b} \qquad \frac{\Gamma \vdash a \leq b \quad \Gamma \vdash b \leq c}{\Gamma \vdash a \leq c} \qquad \frac{\Gamma \vdash a \leq b \quad \Gamma \vdash c \leq b}{\Gamma \vdash a \curlyvee c} \qquad \frac{\Gamma \vdash a \leq b \quad \Gamma \vdash b \curlyvee c}{\Gamma \vdash a \curlyvee c}$$

We then define a judgement $P \rhd \Gamma$ on processes, meaning that P entails φ for all $\varphi \in \Gamma$:

$$\frac{}{a/b \rhd b \leq a} \qquad \frac{P \rhd \Gamma \qquad \Gamma \vdash \varphi}{P \rhd \varphi} \qquad \frac{P \rhd \varphi}{P \mid Q \rhd \varphi} \qquad \frac{P \rhd \varphi \quad a \notin \mathrm{n}(\varphi)}{(\nu a)P \rhd \varphi} \ ,$$

and a similar judgement $E \rhd \Gamma$ on active contexts, meaning that E entails all $\varphi \in \Gamma$ at the occurrence of the hole in E:

$$\frac{E \rhd \Gamma \qquad \Gamma \vdash \varphi}{E \rhd \varphi} \qquad \frac{E \rhd \varphi}{P \mid E \rhd \varphi} \qquad \frac{P \rhd \varphi \quad \mathrm{n}(\varphi) \cap \mathrm{cn}(E) = \emptyset}{P \mid E \rhd \varphi} \qquad \frac{E \rhd \varphi}{(\nu a)E \rhd \varphi}$$

(we have omitted symmetrical rules for $Q \mid P$ and $E \mid P$).

One can check that the relation induced by the judgement $P \rhd a \leqslant b$ is a preorder on names.

Example 15 *(Mediators).* A process $M_{fg} = (\boldsymbol{\nu} c)(c/f \mid c/g)$ acts like a *mediator*: it joins names f and g (we have $M_{fg} \rhd f \curlyvee g$). Mediators remind us of equators in the π-calculus [18], or of fusions in the explicit fusion calculus, but lack the transitivity property (e.g., $M_{fg} \mid M_{gh} \rhd f \curlyvee h$ does not hold).

Definition 16 *(By-need reduction).* *By-need* reduction, $P \longrightarrow_{\mathrm{bn}} P'$, is defined by the following rules:

$$\text{BN-SCon}: \ \frac{P \equiv Q \quad Q \longrightarrow_{\mathrm{bn}} Q' \quad Q' \equiv P'}{P \longrightarrow_{\mathrm{bn}} P'} \qquad \text{BN-Red}: \ \frac{E \rhd a \curlyvee b}{E[ac.P \mid \overline{b}d.Q] \longrightarrow_{\mathrm{bn}} E[P \mid d/c \mid Q]}$$

Relation $\Longrightarrow_{\mathrm{bn}}$ is the reflexive transitive closure of $\longrightarrow_{\mathrm{bn}}$.

While the eager semantics has simpler rules, the by-need semantics avoids 'too early commitments' on prefixes. For instance, the only immediate reduction of the process in (6) is

$$\longrightarrow_{\mathrm{bn}} (\boldsymbol{\nu} a, c)\ (P \mid w/v \mid Q \mid a/b \mid c/b)$$

where prefixes $bv.P$ and $\overline{c}w.Q$ interact because their subjects are joinable in the preorder generated by the two arcs.

Lemma 17 *(Eager and by-need).* *$P \longrightarrow_{\mathrm{bn}} P'$ (by-need semantics) implies $P \Longrightarrow_{\mathrm{ea}} P'$ (eager semantics).*

Proof. Suppose the transition involves processes $ac.P$ and $\overline{b}d.Q$, thanks to a derivation of $E \rhd a \curlyvee b$. The latter means that u is an upper bound of $\{a, b\}$, via some active arcs in E. Then, according to the eager semantics, we are able to rewrite the prefixed processes into $uc.P$ and $\overline{u}d.Q$. □

Corollary 18 *(Subject reduction, by-need semantics).* *Theorem 12 holds for the by-need semantics.*

We now move to a comparison of the two semantics.

Behavioural equivalence. We contrast barbed congruence in $\pi\mathbb{P}$ under the two semantics we have given, eager and by-need. In order to define both versions of barbed congruence, we need to define *barbs*. This requires some care, as in $\pi\mathbb{P}$ the interaction of a process with its environment may be mediated by arcs. To have a uniform definition of barbs under the eager and by-need semantics, we follow the definition of success in testing equivalence [6], using a special signal ω that may not appear in processes.

Definition 19. Given process P and name a, we write $P \downarrow_a^{\mathrm{ea}}$ (resp. $P \downarrow_a^{\mathrm{bn}}$) if there is a prefix α with subject a such that $P \mid \alpha.\omega \longrightarrow_{\mathrm{ea}} P'$ (resp. $P \mid \alpha.\omega \longrightarrow_{\mathrm{bn}} P'$) where ω is not free in P and ω is unguarded in P'.

Hence $P \downarrow_a$ if the offer of the environment of an action at a may be accepted by P. Remark that $P \downarrow_a^{\mathrm{bn}}$ holds if $P = E[\alpha.Q]$ for some active context E that does not capture a, and some prefix $\alpha.Q$, in which the subject b of α satisfies $E \rhd a \curlyvee b$. With the same notations, $P \downarrow_a^{\mathrm{ea}}$ if $b = a$.

Weak barbs and barbed congruence are then defined in the standard way, as outlined in Section 2. We write $\simeq_{\mathrm{ea}}$ and $\approx_{\mathrm{ea}}$ (resp. $\simeq_{\mathrm{bn}}$ and $\approx_{\mathrm{bn}}$) for the strong and weak versions of eager (resp. by-need) barbed congruence (Definition 1).

Example 20 *(Contrasting eager and by-need semantics).* The eager and by-need semantics of $\pi\mathbb{P}$ yield incomparable equivalences. The two following laws are valid in the by-need case, and fail in the eager case:

$$(\boldsymbol{\nu} a)a/c = \mathbf{0} \qquad a \mid a = a.a \quad .$$

To see the failure of the first law in the eager semantics, consider a context $C \triangleq [\cdot] \mid (\boldsymbol{\nu} b)(b/c) \mid c \mid \overline{c}.\overline{w}$; then $C[(\boldsymbol{\nu} a)(a/c)]$ can lose the possibility of emitting at w, by reducing in two steps to $(\boldsymbol{\nu} a)(a/c \mid a) \mid (\boldsymbol{\nu} b)(b/c \mid \overline{b}.\overline{w})$, because of a commitment determined by arcs; this cannot happen for $C[\mathbf{0}]$. In the by-need semantics, there are no early commitments, and the two processes are hence equal.

Similarly, in the eager semantics, it is possible to put $a \mid a$ in a context where two arcs rewrite each a prefix differently, while one can only rewrite the topmost prefix in $a.a$. This scenario cannot be played in the by-need semantics.

On the other hand, the following law is valid for strong (and weak) eager equivalence, but fails to hold in the by-need case:

$$(\boldsymbol{\nu} a, b, u)(a/u \mid b/u \mid \overline{u} \mid a.\overline{w}) = (\boldsymbol{\nu} v)(\overline{v} \mid v.\tau.\overline{w} \mid v.\mathbf{0}) \ .$$

($\tau.\overline{w}$ stands for $(\boldsymbol{\nu} c)(c \mid \overline{c}.\overline{w})$ with $c \neq w$). The intuition is that concurrent arcs are used on the left-hand side to implement internal choice. As a consequence of the law $(\boldsymbol{\nu} a)a/c = \mathbf{0}$, in the by-need case, process b/u can be disregarded on the left, so that the process on the left necessarily does the output on w.

Example 20 shows that the two semantics are incomparable. Lemma 17 suggests that the eager reduction is a decomposition of the by-need reduction. We can moreover remark that the addition of dynamic operators like guarded choice is rather natural under the by-need semantics. It is delicate in the eager semantics; for instance it would be unclear whether, in a process like $c/a \mid (a.P + b.Q)$, the arc should be allowed to trigger the choice.

We have introduced $\pi\mathtt{P}$ with the eager semantics because it is simpler, and closer to an implementation, but we find the by-need semantics more compelling. Below, unless otherwise stated, we work under by-need, though we also indicate what we know under eager.

5.2. Context-free characterisations of barbed congruence

When it comes to proving behavioural equalities, the definition of barbed congruence is troublesome, as it involves a heavy quantification on contexts. One therefore looks for context-free coinductive characterisations, as labelled bisimilarities that take into account not only reductions within a process, but also the potential interactions between the process and its environment (e.g., input and output actions). We present such a characterisation for the by-need equivalence.

5.2.1. An LTS for by-need semantics

Labelled transitions. As actions for the by-need labelled bisimilarity, we use, besides τ-actions, only free input and free output. Thus the grammar for actions is:

$$\mu \quad ::= \quad \tau \mid ab \mid \overline{a}b \ .$$

Labelled transitions are written $P \xrightarrow{\mu}_{\mathrm{bn}} P'$. Input and output transitions are given by the following rules:

$$\text{BN-Inp}: \ \frac{E \rhd a \curlyvee b \qquad \{b,d\} \cap \mathrm{cn}(E) = \emptyset}{E[ac.P] \xrightarrow{bd}_{\mathrm{bn}} E[d/c \mid P]} \qquad \text{BN-Out}: \ \frac{E \rhd a \curlyvee b \qquad \{b,d\} \cap \mathrm{cn}(E) = \emptyset}{E[\overline{a}c.P] \xrightarrow{\overline{b}d}_{\mathrm{bn}} E[c/d \mid P]}$$

The purpose of the two rules is to define the input and output transitions, keeping labels as simple as possible. The two rules are not supposed to be composed together to derive τ-actions. Internal transitions have already been defined, in the reduction semantics: we take relation $\xrightarrow{\tau}_{\mathrm{bn}}$ to coincide with the reduction relation $\longrightarrow_{\mathrm{bn}}$.

A compositional semantics, which does not refer to structural congruence and the reduction relation, is presented in separate work [15], together with an axiomatisation of behavioural equivalence in $\pi\mathtt{P}$.

To understand rules BN-Inp and BN-Out, suppose the environment is offering an action at b. Since a and b are joinable, there is a name, say e, that is above both a and b in the preorder; hence the prefix at a in the process and the prefix at b in the environment can be transformed into prefixes at e, and can interact. The need for the preorder explains why we found it convenient to express actions via active contexts.

In the action, the use of a fresh object d allows us to ignore name extrusion and allows us to work with a simpler set of transitions labels.

We show an example transition derived using BN-Out: we have, for a fresh d (similar observations can be made for BN-Inp):

$$(\boldsymbol{\nu} u)\left(u/b \mid (\boldsymbol{\nu} a, c)(u/a \mid \overline{a}c.P)\right) \xrightarrow{\overline{b}d}_{\mathrm{bn}} (\boldsymbol{\nu} u)\left(u/b \mid (\boldsymbol{\nu} a, c)(u/a \mid c/d \mid P)\right) \ .$$

Here the process can interact with the environment at b (and hence perform a transition where b is the subject), because a and b are joinable. Name c is not extruded; instead the arc c/d redirects interactions on d to c.

Another way to understand visible transitions is given by the following implications (which follow by a simple case analysis on the rules used to derive the transitions):

Lemma 21. *We have, for any name d,*

- *if* $P \xrightarrow{\overline{b}d}_{\mathrm{bn}} P'$ *then* $(P \mid bd) \longrightarrow_{\mathrm{bn}} P'$;
- *if* $P \xrightarrow{bd}_{\mathrm{bn}} P'$ *then* $(P \mid \overline{b}d) \longrightarrow_{\mathrm{bn}} P'$.

Behavioural equivalence. In $\pi\mathtt{P}$, bisimulation requires, besides the usual condition on transitions, invariance under the addition of arcs (clause 1 below). Moreover, we require two bisimilar processes to entail the same joinability conditions between names (clause 2).

Definition 22 *(By-need bisimulation).* A *by-need* bisimulation $\mathcal{R}$ is a set of pairs (P, Q) s.t. $P\,\mathcal{R}\,Q$ implies:

1. $(P \mid a/b)\,\mathcal{R}\,(Q \mid a/b)$, for all names a, b (invariance under arcs);
2. $P \triangleright a \curlyvee b$ implies $Q \triangleright a \curlyvee b$, for all names a, b;
3. if $P \xrightarrow{\mu}_{\mathrm{bn}} P'$, then $Q \xrightarrow{\mu}_{\mathrm{bn}} Q'$ and $P'\,\mathcal{R}\,Q'$ (where the object part of μ is fresh);
4. the converse of clauses (2) and (3).

(Strong) bisimilarity, written $\sim_{\mathrm{bn}}$, is the largest by-need bisimulation.

We shall sometimes use $\sim_{\mathrm{bn}}$*-bisimulation* as a synonym for *by-need bisimulation* below.

In clause 3, for actions, no extrusion or binding on names is involved. Further, the objects of the actions are *fresh names*. In this sense, the bisimulation reminds us of the *ground bisimulation* in the π-calculus [33], in which a single instantiation of each bound name is considered (in contrast with the ordinary bisimulation of the π-calculus, where names bound in an input must be instantiated with all names free in the tested processes plus a new fresh name). In the π-calculus, however, ground bisimulation coincides with barbed congruence only in asynchronous variants of the calculus. In contrast, $\pi\mathtt{P}$ is synchronous.

Behavioural laws. We now present some examples and $\sim_{\mathrm{bn}}$ (in)equalities that are established using the coinductive proof method of bisimulation. The fact that we can actually rely on this method to derive $\simeq_{\mathrm{bn}}$ laws is justified below (Section 5.2.2). All equalities and inequalities also hold under the eager semantics, though for some equalities this is true only in the weak case (e.g., Lemma 26).

Any input and output of $\pi\mathtt{P}$ can be transformed into a bound prefix, by introducing a new restricted name:

Lemma 23 *(From free to bound prefixes). We have, for fresh* x' *and* y', $ax.P \sim_{\mathrm{bn}} (\boldsymbol{\nu} x')ax'.(x'/x \mid P)$ *and* $\overline{b}y.Q \sim_{\mathrm{bn}} (\boldsymbol{\nu} y')\overline{b}y'.(y/y' \mid Q)$.

Note the presence of x'/x in the former equivalence, and of y/y' in the latter, which in particular preserves polarities. If these laws are applied to all inputs and outputs of a process P, then the result is a behaviourally equivalent process P', in which all names exchanged in an interaction are fresh. Thus P' reminds us of a variant of π that achieves symmetry between input and output constructs, namely πI, the π-calculus with internal mobility [32].

Lemma 24. *We have* $(\boldsymbol{\nu} b, c)\overline{a}c.\overline{a}b.\mathbf{0} \not\sim_{\mathrm{bn}} (\boldsymbol{\nu} c)\overline{a}c.\overline{a}c.\mathbf{0}$, *and* $(\boldsymbol{\nu} b, c)ac.ab.\mathbf{0} \sim_{\mathrm{bn}} (\boldsymbol{\nu} c)ac.ac.\mathbf{0}$.

These laws show a difference between input and output in behavioural equalities. The reason for the inequality is that the first process can produce two transitions with objects e, f yielding $P \triangleq \boldsymbol{\nu} c\,(c/f \mid c/e)$, and then $P \triangleright e \curlyvee f$.

In fusion calculi, processes $(\boldsymbol{\nu} b, c)\overline{a}c.\overline{a}b.\mathbf{0}$ and $(\boldsymbol{\nu} c)\overline{a}c.\overline{a}c.\mathbf{0}$ are not bisimilar (like in $\pi\mathtt{P}$), but the duality in Fusions implies that $(\boldsymbol{\nu} b, c)ac.ab.\mathbf{0}$ and $(\boldsymbol{\nu} c)ac.ac.\mathbf{0}$ are also not bisimilar (unlike in $\pi\mathtt{P}$).

We now move to behavioural properties of arcs in $\pi\mathtt{P}$.

Lemma 25 *(*$\simeq_{\mathrm{bn}}$ *laws about arcs). When* $a \neq b$ *and* $b \neq c$, $(\boldsymbol{\nu} b)(a/b \mid b/c) \simeq_{\mathrm{bn}} a/c$ *and* $a/b \mid a/b \simeq_{\mathrm{bn}} a/b$.

The following laws involve arcs in relation with negative or positive occurrences of names. These are reminiscent of standard results about substitutions, forwarders and equators in the asynchronous π-calculus (see, e.g., [33]).

Lemma 26 *(Substitution and polarities).*

1. *If name a has only positive occurrences in P, then* $(\boldsymbol{\nu} a)(P \mid b/a) \sim_{\mathrm{bn}} P\{b/a\}$;
2. *if name a has only negative occurrences in P, then* $(\boldsymbol{\nu} a)(P \mid a/b) \sim_{\mathrm{bn}} P\{b/a\}$;
3. $(\boldsymbol{\nu} a)(P \mid b/a \mid a/b) \sim_{\mathrm{bn}} P\{b/a\}$.

5.2.2. Congruence

For the comparison between labelled bisimilarity ($\sim_{\mathrm{bn}}$) and barbed congruence ($\simeq_{\mathrm{bn}}$), the most delicate part is the proof of congruence for bisimilarity. This is due: (*i*) to the shape of visible transitions, where an arc is introduced and the object part in the label is always a fresh name, and (*ii*) to the use of $\equiv$ in the definition of transitions.

To prove congruence under restriction and parallel composition, we rely on a standard up-to technique for bisimulation, 'bisimulation up to bisimilarity', obtained by replacing any mention of $\mathcal{R}$ with $\sim_{\mathrm{bn}}\mathcal{R}\sim_{\mathrm{bn}}$ in clauses (1) and (3) of Definition 22:

Definition 27 *(By-need bisimulation up to $\sim_{\mathrm{bn}}$).* A relation $\mathcal{R}$ is a *by-need bisimulation up to* $\sim_{\mathrm{bn}}$ if $P\ \mathcal{R}\ Q$ implies:

1. $(P \mid a/b) \sim_{\mathrm{bn}}\mathcal{R}\sim_{\mathrm{bn}} (Q \mid a/b)$, for all a, b;
2. $P \triangleright a \curlyvee b$ implies $Q \triangleright a \curlyvee b$, for all a, b;
3. if $P \xrightarrow{\mu}_{\mathrm{bn}} P'$, then $Q \xrightarrow{\mu}_{\mathrm{bn}} Q'$ and $P' \sim_{\mathrm{bn}}\mathcal{R}\sim_{\mathrm{bn}} Q'$ (where the object part of μ is fresh);
4. the converse of clauses (2) and (3).

Lemma 28. *If $\mathcal{R}$ is a by-need bisimulation up to $\sim_{\mathrm{bn}}$ then $\mathcal{R} \subseteq \sim_{\mathrm{bn}}$.*

The proof is standard (we show that $\sim_{\mathrm{bn}}\mathcal{R}\sim_{\mathrm{bn}}$ is a bisimulation, using the fact that $\sim_{\mathrm{bn}}$ itself is one, and is transitive).

Lemma 29. *If $P \equiv Q$ and $P \triangleright a \curlyvee b$ then $Q \triangleright a \curlyvee b$.*

The proof is an induction on the derivation of $P \equiv Q$.

Lemma 30. *If $P \simeq_{\mathrm{bn}} Q$ and $P \triangleright a \curlyvee b$, then $Q \triangleright a \curlyvee b$.*

Proof. Given a and b, we introduce the context $E = (- \mid \overline{a}.f \mid b.g)$, where f and g are fresh. We observe that $R \triangleright a \curlyvee b$ iff $E[R] \longrightarrow_{\mathrm{bn}} R_1$ for some R_1 such that $R_1 \downarrow^{\mathrm{bn}}_f$ and $R_1 \downarrow^{\mathrm{bn}}_g$. By definition of $\simeq_{\mathrm{bn}}$ we know that $E[P] \simeq_{\mathrm{bn}} E[Q]$, which implies the expected result. □

Lemma 31. *If $\mathcal{R}$ is invariant under arcs and preserves $\curlyvee$, then, whenever $P\ \mathcal{R}\ Q$, we have $P \triangleright a \leqslant b$ iff $Q \triangleright a \leqslant b$.*

Proof. Let P and Q be processes and f be a fresh name. Then $P \triangleright a \leqslant b$ iff $(P \mid f/b) \triangleright a \curlyvee f$, and similarly for Q. Thanks to the first hypothesis on $\mathcal{R}$ we have $(P \mid f/b)\,\mathcal{R}\,(Q \mid f/b)$ and we conclude with the second hypothesis. □

By definition of by-need bisimulation (Definition 22), Lemma 31 holds in particular if $\mathcal{R}$ is a by-need bisimulation.

Lemma 32. *If $P \equiv Q$ then $P \sim_{\mathrm{bn}} Q$.*

Proof. We show that $\equiv$ is a by-need bisimulation. The clauses 1), 2), 4) are dealt with using respectively the fact that $\equiv$ is a congruence, Lemma 29, and symmetry of $\equiv$.

For clause 3), we first observe that when $\mu = \tau$, we can conclude easily, since $\xrightarrow{\tau}_{\mathrm{bn}} = \longrightarrow_{\mathrm{bn}}$ is stable by $\equiv$ (i.e., $\equiv\longrightarrow_{\mathrm{bn}}\equiv$ is included in $\longrightarrow_{\mathrm{bn}}$). For the remaining labels, we examine the case where $\mu = bd$, the other case being similar. We know that $P = E[ac.P_1]$ with $E \triangleright a \curlyvee b$ and $P' = E[d/c \mid P_1]$. And since $E[ac.P_1] \equiv Q$, we know that $Q = E'[ac.P'_1]$ which entails $Q \xrightarrow{bd} E'[d/c \mid P'_1] \equiv P'$. □

We say that $\mathcal{R}$ is a *by-need bisimulation up to* $\equiv$ if it satisfies Definition 22 with $\equiv\mathcal{R}\equiv$ instead of $\mathcal{R}$ in clauses (1) and (3). Of course, using Lemmas 28 and 32, if $\mathcal{R}$ is a by-need bisimulation up to $\equiv$ then $\mathcal{R} \subseteq \sim_{\mathrm{bn}}$.

Lemma 33 *(Congruence for restriction). If $P \sim_{\mathrm{bn}} Q$ then for all c, $(\boldsymbol{\nu} c)P \sim_{\mathrm{bn}} (\boldsymbol{\nu} c)Q$.*

Proof. We show that $\mathcal{R} = \{((\boldsymbol{\nu} c)P, (\boldsymbol{\nu} c)Q),\ P \sim_{\mathrm{bn}} Q\}$ is a bisimulation up to $\equiv$. Invariance under arcs is handled by reasoning up to $\equiv$. Clause (2) about preservation of joinability is immediate because derivability $(\boldsymbol{\nu} x)P \triangleright \varphi$ only depends on whether $P \triangleright \varphi$ is derivable.

Clause (3) is split into two cases:

- Silent transitions: a τ transition $(\boldsymbol{\nu} x)P \longrightarrow_{\mathrm{bn}} P_1$ cannot involve the external binder: if $(\boldsymbol{\nu} x)P \equiv E[\overline{a}c.R_1 \mid bd.R_2]$ and $P_1 \equiv E[c/d \mid R_1 \mid R_2]$ then P_1 can be written $P_1 \equiv (\boldsymbol{\nu} x)P'$ where $P \longrightarrow_{\mathrm{bn}} P'$ (indeed $E_1 \triangleright a \curlyvee b$ iff $(\boldsymbol{\nu} x)E_1 \triangleright a \curlyvee b$). Then we use the fact that $P \sim_{\mathrm{bn}} Q$ to infer the same transition from $Q \longrightarrow_{\mathrm{bn}} Q'$ with $P' \sim_{\mathrm{bn}} Q'$ and hence $(\boldsymbol{\nu} x)Q \longrightarrow_{\mathrm{bn}} (\boldsymbol{\nu} x)Q'$, which gives $P_1 \equiv (\boldsymbol{\nu} x)P'\ \mathcal{R}\ (\boldsymbol{\nu} x)Q'$.

- Visible transitions: suppose $(\boldsymbol{\nu} x)P \xrightarrow{bd}_{\mathrm{bn}} P_1$, then $P = E[ac.R]$ and $P_1 = (\boldsymbol{\nu} x)E[d/c.R]$. This gives $P \xrightarrow{bd}_{\mathrm{bn}} P'$ and $P_1 = (\boldsymbol{\nu} x)P'$ for some P', where $E \triangleright a \curlyvee b$ and $(\boldsymbol{\nu} x)E \triangleright a \curlyvee b$. Since we also know that $(\boldsymbol{\nu} x)E$ does not capture b, we can infer the same transition bd from Q and conclude. □

The following behavioural law on arcs will be useful below.

Lemma 34 *(Transitivity of arcs). For all active contexts E we have: $E[a/c] \sim_{\mathrm{bn}} E[(\boldsymbol{\nu} b)(a/b \mid b/c)]$.*

Proof. Let $\mathcal{R}$ be the relation corresponding to the above law. We show that $\mathcal{R}$ is a $\sim_{\mathrm{bn}}$-bisimulation up to $\equiv$. By definition, $\mathcal{R}$ is stable by parallel composition of arcs, since E can be an arbitrary active context.

Concerning the condition about joinability of names, the left-to-right implication is rather clear. From right to left, we must prove that we cannot get more from $(\boldsymbol{\nu} b)(a/b \mid b/c)$ than from a/c, which is ensured by the restriction $(\boldsymbol{\nu} b)$.

Now concerning the condition on transitions, we know from the condition about joinability that the preorders on names induced by related processes coincide. After any visible transition, the contexts are changed but the resulting processes are still related through $\mathcal{R}$. After a silent transition, since the contexts may be changed using $\equiv$ we use clause (3) up to $\equiv$, the resulting processes being related through $\equiv\mathcal{R}\equiv$. □

It remains to establish that $\sim_{\mathrm{bn}}$ is preserved by parallel composition. For this we introduce *communication contexts*. These are, intuitively, the composition of two active contexts, one hosting an input, the other an output. Intuitively, the input and output processes inserted in the holes may produce a τ-action. Communication contexts, ranged over by G, have two holes, each occurring exactly once.

$$G ::= P \mid G \mid \_ \mid \_ \mid P \mid G \mid \boldsymbol{\nu} a\, G \mid E_1 \mid E_2 \ .$$

By convention the leftmost hole is the first one, the other is the second one. We write $P = G[ac.Q][\overline{b}d.R]$ if P is obtained from G by filling the first hole with $ac.Q$, and the second hole with $\overline{b}d.R$.

Communication contexts can be used to decompose a $\xrightarrow{\tau}_{\mathrm{bn}}$ transition:

Lemma 35. *Suppose $P \xrightarrow{\tau}_{\mathrm{bn}} P'$ (that is, $P \longrightarrow_{\mathrm{bn}} P'$). Then one of the following statements holds:*

- *either $P = G[\overline{a}b.Q][cd.R]$ and $P' \sim_{\mathrm{bn}} \boldsymbol{\nu} f\ (G[b/f \mid Q][f/d \mid R])$,*
- *or $P = G[cd.R][\overline{a}b.Q]$ and $P' \sim_{\mathrm{bn}} \boldsymbol{\nu} f\ (G[f/d \mid R][b/f \mid Q])$,*

where $G \triangleright a \curlyvee c$ and f is fresh.

Proof. The two cases are similar, the main technical difficulty is to keep track of the usages of structural congruence. If $P \longrightarrow_{\mathrm{bn}} P'$, it means that $P \equiv E[\overline{a}b.Q_1 \mid ac.R_1]$ and $P' \equiv E[b/c \mid Q_1 \mid R_1]$. From the first relation we can get G such that $P = G[\overline{a}b.Q][cd.R]$ (with $G \triangleright a \curlyvee c$, $Q \equiv Q_1$ and $R \equiv R_1$ — we leave out the symmetric case for which the output is the second argument of G).

Using the laws of structural congruence for restriction, we pull upwards restrictions that occur *unguarded* (i.e., not under a prefix), and write $G \equiv (\boldsymbol{\nu} \hat{b}, \hat{d})G'$, where $\hat{b} = \emptyset$ if b is not bound and $\hat{b} = \{b\}$ if b is captured by G.

We then reason as follows, relying on Lemma 34 and using structural congruence:

$$\begin{aligned}
P &\equiv (\boldsymbol{\nu} \hat{b}, \hat{d})G'[\overline{a}b.Q][cd.R] \\
&\longrightarrow_{\mathrm{bn}} (\boldsymbol{\nu} \hat{b}, \hat{d})(b/d \mid G'[Q][R]) \\
&\sim_{\mathrm{bn}} (\boldsymbol{\nu} \hat{b}, \hat{d})((\boldsymbol{\nu} f)(b/f \mid f/d) \mid G'[Q][R]) \\
&\equiv (\boldsymbol{\nu} f)(\boldsymbol{\nu} \hat{b}, \hat{d})(G'[b/f \mid Q][f/d \mid R]) \\
&\equiv (\boldsymbol{\nu} f)G[b/f \mid Q][f/d \mid R] \ .
\end{aligned}$$

Lemma 32 allows us deduce the expected result. □

The proof of congruence for parallel composition uses the following property, which allows us to decompose the arc which is introduced along an interaction (rule E-Red) into two arcs, associated to the two visible prefixes taking part in the interaction. This corresponds to the fact that in rules BN-Inp and BN-Out, transitions introduce an arc.

Lemma 36. *Suppose $Q \xrightarrow{bf}_{\mathrm{bn}} Q'$. Then for any context E, for any name b that is not captured by E, and for any fresh name f, we have $Q \mid E[\overline{b}d.R_1] \xrightarrow{\tau}_{\mathrm{bn}} \sim_{\mathrm{bn}} \boldsymbol{\nu} f\ (Q' \mid E[d/f \mid R_1])$.*

We are ready to present the proof of congruence under parallel composition. For this, we rely on *bisimulations up to* $\sim_{\mathrm{bn}}$ *and restriction*. We let $\mathcal{S}^{\boldsymbol{\nu}}$ stand for the smallest relation such that $(\boldsymbol{\nu}\tilde{x})P\,\mathcal{S}^{\boldsymbol{\nu}}\,(\boldsymbol{\nu}\tilde{x})Q$ whenever $P\,\mathcal{S}\,Q$. In a bisimulation up to $\sim_{\mathrm{bn}}$ and restriction, $\sim_{\mathrm{bn}}\mathcal{R}^{\boldsymbol{\nu}}\sim_{\mathrm{bn}}$ replaces $\mathcal{R}$ in clauses (1) and (3) of Definition 22. In particular, clause (3) becomes (whenever $\mu \neq \tau$, we suppose that the object part of μ is fresh):

(3) if $P \xrightarrow{\mu}_{\mathrm{bn}} P'$, then $Q \xrightarrow{\mu}_{\mathrm{bn}} Q'$ for some P'', Q'', $\tilde{x}$ s.t. $P' \sim_{\mathrm{bn}} \boldsymbol{\nu}\tilde{x}\,P''$, $Q' \sim_{\mathrm{bn}} \boldsymbol{\nu}\tilde{x}\,Q''$, and $P''\,\mathcal{R}\,Q''$.

Lemma 37. *If $\mathcal{R}$ is a by-need bisimulation up to $\sim_{\mathrm{bn}}$ and restriction then $\mathcal{R} \subseteq \sim_{\mathrm{bn}}$.*

The proof is standard (we use Lemma 32 and ${\sim_{\mathrm{bn}}}^{\boldsymbol{\nu}} \subseteq \sim_{\mathrm{bn}}$, i.e., Lemma 33, to show that $\sim_{\mathrm{bn}}\mathcal{R}^{\boldsymbol{\nu}}\sim_{\mathrm{bn}}$ is a by-need bisimulation).

Proposition 38 *(Congruence for parallel composition). If $P \sim_{\mathrm{bn}} Q$ then $P \mid R \sim_{\mathrm{bn}} Q \mid R$.*

Proof (Sketch). We first establish the property using an additional hypothesis ("Special case"), and then move to the general case.

- Special case: we first suppose that all arcs in R occur under at least one prefix. We show that

  $$\{(P \mid R,\ Q \mid R),\ P \sim_{\mathrm{bn}} Q \text{ and } R \text{ does not contain active arcs}\}$$

  is a bisimulation up to restriction and up to bisimilarity.
  Suppose then $P \mid R \xrightarrow{\tau}_{\mathrm{bn}} U$, and suppose both P and R contribute to the transition (the other possibilities are easier). We also suppose that P makes the input (the case of output is symmetric). We have, using Lemma 35:

  $$P = E[ac.P_1] \qquad R = F[\overline{b}d.R_1]$$

  where $E \triangleright a \curlyvee b$ (since no arc is active in R), and, for some fresh f, $P' = E[f/c \mid P_1]$ and $R' = F[d/f \mid R_1]$:

  $$U \sim_{\mathrm{bn}} (\boldsymbol{\nu} f)\,(P' \mid R')\ .$$

  Using rule EN-INP, we also have $P \xrightarrow{bf}_{\mathrm{bn}} P'$. Hence, since $P \sim_{\mathrm{bn}} Q$, $Q \xrightarrow{bf}_{\mathrm{bn}} Q'$ and $P' \sim_{\mathrm{bn}} Q'$ for some Q', which gives $Q' = E'[a'c'.Q_1]$ for some a' s.t. $E' \triangleright a' \curlyvee b$, and $Q' = E'[f/c' \mid Q_1]$. From this, Lemma 36 gives us directly:

  $$Q \mid R \xrightarrow{\tau}_{\mathrm{bn}} \sim_{\mathrm{bn}} (\boldsymbol{\nu} f)\,(Q' \mid R')\ .$$

  We write R' as follows:

  $$R' \ \equiv\ (\boldsymbol{\nu}\tilde{n})\left(R'' \mid \sigma\right)\ ,$$

  where σ is a parallel composition of arcs and R'' contains no active arc. We then have

  $$P' \mid R' \ \equiv\ (\boldsymbol{\nu}\tilde{n})\left(P' \mid \sigma \mid R''\right)\ ,$$

  and similarly for $Q' \mid R'$. We can conclude by remarking that $P' \sim_{\mathrm{bn}} Q'$ entails $P' \mid \sigma \sim_{\mathrm{bn}} Q' \mid \sigma$, and using up to restriction to remove the topmost restrictions.
- General case: consider now the case where R is an arbitrary process. We reason by induction on R, to show that for all P and Q, $P \sim_{\mathrm{bn}} Q$ implies $P \mid R \sim_{\mathrm{bn}} Q \mid R$. The cases where R is a prefixed process or $R = \mathbf{0}$ are treated by the special case above.
  The case where $R = u/v$ holds by definition of $\sim_{\mathrm{bn}}$: $P \sim_{\mathrm{bn}} Q$ implies $P \mid u/v \ \sim_{\mathrm{bn}}\ Q \mid u/v$.
  If $R = R_1 \mid R_2$, then by induction $P \mid R_1 \ \sim_{\mathrm{bn}}\ Q \mid R_1$, which gives, by induction again, $(P \mid R_1) \mid R_2 \ \sim_{\mathrm{bn}}\ (Q \mid R_1) \mid R_2$, hence the result by associativity of $\mid$.
  Suppose now $R = (\boldsymbol{\nu}c)R'$. We can suppose w.l.o.g. $c \notin \mathrm{fn}(P) \cup \mathrm{fn}(Q)$. Then by induction $P \mid R' \ \sim_{\mathrm{bn}}\ Q \mid R'$, which gives, by Lemma 33, $(\boldsymbol{\nu}c)(P \mid R') \ \sim_{\mathrm{bn}}\ (\boldsymbol{\nu}c)(Q \mid R')$. Lemma 32 gives $(\boldsymbol{\nu}c)(P \mid R') \ \sim_{\mathrm{bn}}\ P \mid (\boldsymbol{\nu}c)R'$, and similarly for Q, hence $P \mid R \ \sim_{\mathrm{bn}}\ Q \mid R$. This concludes the proof. □

Theorem 39. *Bisimilarity is a congruence.*

Proof. Follows from Lemma 33 and Proposition 38, closure of $\sim_{\mathrm{bn}}$ under prefixes being immediate. □

Theorem 40 *(Soundness). If $P \sim_{\mathrm{bn}} Q$ then $P \simeq_{\mathrm{bn}} Q$.*

Proof. We inspect the definition of $\simeq_{\mathrm{bn}}$.

Preservation of barbs: in the case where the barb is fresh, i.e., f does not appear in any arc, $P \downarrow_f^{\mathrm{bn}}$ is equivalent to $P \xrightarrow{\alpha}$ where α is an input or output label with subject f. In the case of general barbs: $P \downarrow_a^{\mathrm{bn}}$ is equivalent to $(P \mid \alpha.f) \xrightarrow{\tau} {}_{\mathrm{bn}}\downarrow_f^{\mathrm{bn}}$ for some α whose subject is a.

Closure under reduction holds trivially since $\longrightarrow_{\mathrm{bn}}$ coincides with $\xrightarrow{\tau}_{\mathrm{bn}}$. Finally, Theorem 39 guarantees closure by contexts. □

5.2.3. Completeness of $\sim_{\mathrm{bn}}$

Completeness is proved along the lines of similar completeness proofs in the literature [27,33].

Given a prefix α, we write $\overline{\alpha}$ for the dual prefix, i.e. $\overline{\overline{a}b} = ab$ and $\overline{ab} = \overline{a}b$.

Lemma 41. *Let P and P' be processes and f a name fresh w.r.t. P and such that $P' \not\downarrow_f^{\mathrm{bn}}$. We have that $P \xrightarrow{\alpha}_{\mathrm{bn}} \equiv P'$ if and only if there exists a process P_1 such that $P_1 \downarrow_f^{\mathrm{bn}}$ and*

$$P \mid \overline{\alpha}.(\overline{f} \mid f) \longrightarrow_{\mathrm{bn}} P_1 \longrightarrow_{\mathrm{bn}} P' \ .$$

Proof. Let us consider the case where α is an input prefix bd, the output case being similar. We prove the two implications.

Left to right: since $\longrightarrow_{\mathrm{bn}}$ is stable by $\equiv$ we directly suppose that $P \xrightarrow{\alpha}_{\mathrm{bn}} P'$. Then $P = E[ac.Q]$ with $E \triangleright a \curlyvee b$ and $P' = E[d/c \mid Q]$. We reason as follows:

$$\begin{aligned} P_\alpha &\triangleq P \mid \overline{\alpha}.(\overline{f} \mid f) \\ &\equiv E[ac.Q \mid \overline{b}d.(\overline{f} \mid f)] \\ &\longrightarrow_{\mathrm{bn}} E[d/c \mid Q \mid \overline{f} \mid f] \triangleq P_1 \\ &\longrightarrow_{\mathrm{bn}} E[d/c \mid Q] = P' \ . \end{aligned}$$

Right to left: since $P_1 \downarrow_f^{\mathrm{bn}}$ and f is fresh for P, we know that $\overline{\alpha}$ has been triggered, that is, $P_\alpha \equiv E[ac.Q \mid \overline{b}d.(\overline{f} \mid f)]$ with $E \triangleright a \curlyvee b$ and $P' \equiv E[d/c \mid Q]$ since P' has no barb at f. This means that P is of the form $P \equiv E[ac.Q]$. Hence $P \xrightarrow{\alpha}_{\mathrm{bn}} \equiv P'$. □

Theorem 42 *(Completeness). If $P \simeq_{\mathrm{bn}} Q$ then $P \sim_{\mathrm{bn}} Q$.*

Proof. We show that $\simeq_{\mathrm{bn}}$ is a $\sim_{\mathrm{bn}}$-bisimulation up to $\equiv$. The clause for preservation of $\curlyvee$ is treated using Lemma 30. The clause about parallel composition of arcs is trivial, as well as the symmetry and the clause for the τ-transition. We are left with the clause involving input and output transitions.

Suppose $P \xrightarrow{\alpha}_{\mathrm{bn}} P'$, and let f be a name fresh with respect to P, P' and Q. Lemma 41 yields P_1 such that $P_1 \downarrow_f^{\mathrm{bn}}$ and a sequence of reductions which we can transport to Q:

$$Q \mid \overline{\alpha}.(\overline{f} \mid f) \longrightarrow_{\mathrm{bn}} Q_1 \longrightarrow_{\mathrm{bn}} Q_2 \ .$$

We know that $P_1 \simeq_{\mathrm{bn}} Q_1$ and $P' \simeq_{\mathrm{bn}} Q_2$, hence $Q_1 \downarrow_f^{\mathrm{bn}}$ and $Q_2 \not\downarrow_f^{\mathrm{bn}}$ (since f is fresh for P'). Another application of Lemma 41 yields $Q \xrightarrow{\alpha}_{\mathrm{bn}} \equiv Q_2$, hence the result. □

Theorem 43 *(Characterisation of barbed congruence). In $\pi\mathrm{P}$, relations $\sim_{\mathrm{bn}}$ and $\simeq_{\mathrm{bn}}$ coincide.*

Proof. Consequence of Theorems 42 and 40. □

Hence all the laws stated above for $\sim_{\mathrm{bn}}$ hold for $\simeq_{\mathrm{bn}}$.

6. Expressiveness of $\pi\mathrm{P}$

We compare $\pi\mathrm{P}$ with other calculi, both as examples of the use of the calculus and as a test for its expressiveness. We study the (explicit) fusion calculus (Section 6.1) and two versions of the π-calculus (Section 6.2).

When useful, we work in a *polyadic* version of $\pi\mathrm{P}$; the addition of polyadicity goes as for other name-passing calculi in the literature. Polyadic $\pi\mathrm{P}$ can be encoded into the monadic calculus along the lines of the analogous encoding for the π-calculus [33, Definition 3.1.4].

All results in this section use the by-need semantics; we do not know their status under the eager semantics.

6.1. Explicit Fusions

Bi-directional arcs, e.g., $a/b \mid b/a$, work as name fusions (cf., Lemma 26(3)). We thus can encode calculi based on name fusions into $\pi\mathtt{P}$. As an example, we consider the explicit fusion calculus [34]. Its syntax extends the fusion calculus with a fusion construct $a{=}b$, yielding the following grammar:

$$P ::= \mathbf{0} \mid P \mid P \mid \overline{a}b.P \mid ab.P \mid (\boldsymbol{\nu} a)P \mid a{=}b \ .$$

Since the construct of name fusion is symmetric, its effect can be conveniently expressed in terms of structural congruence $\equiv$, which is defined as in Definition 6, in addition to the following axioms:

$$a{=}a \equiv \mathbf{0} \qquad (\boldsymbol{\nu} a)a{=}b \equiv \mathbf{0} \qquad a{=}b \mid P \equiv a{=}b \mid P\{a/b\} \ .$$

Note that laws about symmetry ($a{=}b \equiv b{=}a$), transitivity ($a{=}b \mid b{=}c \equiv a{=}c \mid b{=}c$), and 'selective rewriting' ($a{=}b \mid P\{a/x\} \equiv a{=}b \mid P\{b/x\}$) are admissible.

The reduction relation $\longrightarrow_{\mathrm{EF}}$ builds on $\equiv$, and is defined as follows:

$$\frac{}{\overline{a}b.P \mid ac.Q \longrightarrow_{\mathrm{EF}} b{=}c \mid P \mid Q} \qquad \frac{P \longrightarrow_{\mathrm{EF}} P'}{P \mid R \longrightarrow_{\mathrm{EF}} P' \mid R} \qquad \frac{P \longrightarrow_{\mathrm{EF}} P'}{(\boldsymbol{\nu} a)P \longrightarrow_{\mathrm{EF}} (\boldsymbol{\nu} a)P'} \qquad \frac{P \equiv \longrightarrow_{\mathrm{EF}} \equiv P'}{P \longrightarrow_{\mathrm{EF}} P'} \ .$$

The encoding from Explicit Fusions to $\pi\mathtt{P}$ is defined as follows for prefixes and explicit fusions, the other constructs being encoded homomorphically (names w and y are chosen fresh in the encodings of the output and, respectively, input prefixes below):

$$\begin{array}{rcl} [\![\overline{a}\langle v\rangle.P]\!] & = & (\boldsymbol{\nu} w)\overline{a}\langle v, w\rangle.w\langle v\rangle.[\![P]\!] \\ [\![a\langle x\rangle.Q]\!] & = & (\boldsymbol{\nu} y)a\langle x, y\rangle.\overline{y}\langle x\rangle.[\![Q]\!] \\ [\![a{=}b]\!] & = & a/b \mid b/a \end{array}$$

In Explicit Fusions, a reduction step introduces a name fusion, say $a{=}b$. In the $\pi\mathtt{P}$ encoding, this is mimicked in two steps, so to be able to produce, correspondingly, two bidirectional arcs, a/b and b/a. The first step installs the first arc. The second step is a communication on a private name, and has the effect of installing the reverse arc. We do not know whether this encoding is fully abstract. We present an operational correspondence result, given by Theorem 49 below.

In order to derive Theorem 49, we present a series of technical results. We let $P \triangleright a = b$ stand for $P \triangleright a \leqslant b$ and $P \triangleright b \leqslant a$. We write $P =_{a,b} Q$ iff $P\{b/a\} = Q\{b/a\}$, i.e., P and Q only differ in some occurrences of names a and b. We also write $\Phi(P) \triangleq \{\varphi \mid P \rhd \varphi\}$.

Lemma 44. *If $P =_{a,b} Q$, then $\Phi(P \mid [\![a{=}b]\!]) = \Phi(Q \mid [\![a{=}b]\!])$.*

Proof. First, as a consequence of the definition of the judgement $P \rhd \varphi$, we can remark that Φ is monotonic (in the sense that $\Phi(P) \subseteq \Phi(P \mid Q)$), and that there is a function f such that $\Phi(P_1 \mid P_2) = f(\Phi(P_1), \Phi(P_2))$ (where f is monotonic). Let $A \triangleq [\![a{=}b]\!]$.

We prove the lemma by induction on P. The case where P does not contain any prefix is proved by establishing the simple laws $\Phi(a/c \mid A) = \Phi(b/c \mid A)$ and $\Phi(c/a \mid A) = \Phi(c/b \mid A)$. For example the first inclusion can be obtained by remarking that $\Phi(a/c) = \Phi((\boldsymbol{\nu} b)(b/c \mid a/b)) \subseteq \Phi(b/c \mid a/b)$ (by Lemma 34, assuming $a \neq b$ and $b \neq c$ without loss of generality) and then that $\Phi(b/c \mid a/b \mid A) \subseteq \Phi(b/c \mid A \mid A) = \Phi(b/c \mid A)$ by monotonicity and idempotence (since $\Phi(P \mid P) = \Phi(P)$ for all P).

Using idempotence and Lemma 29, we deduce that $\Phi(P_1 \mid P_2 \mid A) = f(\Phi(P_1 \mid A), \Phi(P_2 \mid A))$. This observation allows us to treat all other cases of the induction. □

We extend the definition of $=_{a,b}$ to conditions: $\varphi =_{a,b} \psi$ iff $\varphi\{b/a\} = \psi\{b/a\}$. Lemma 44 can be slightly generalised:

Lemma 45. *If $P =_{a,b} Q$ and $\varphi =_{a,b} \psi$ then $P \mid [\![a{=}b]\!] \rhd \varphi$ iff $Q \mid [\![a{=}b]\!] \rhd \psi$.*

Proof. By Lemma 44 we only have to prove that if $R = S \mid [\![a{=}b]\!]$ then $R \triangleright \varphi$ implies $R \triangleright \psi$, which is easy, since for each case there is a rule from Definition 14 whereby a/b (resp. b/a) is used to replace an occurrence of a with b (resp. vice versa). □

Lemma 46. *If $P =_{a,b} Q$ then $(P \mid [\![a{=}b]\!]) \sim_{\mathrm{bn}} (Q \mid [\![a{=}b]\!])$.*

Proof. Let $\mathcal{R} \triangleq \{(P \mid [\![a{=}b]\!],\ Q \mid [\![a{=}b]\!]),\ \text{with } P =_{a,b} Q\}$. We prove that $\mathcal{R}$ is a $\sim_{\mathrm{bn}}$-bisimulation, by symmetry of $=_{a,b}$ there are three properties to check (cf. Definition 22):

1. invariance under arcs is immediate;
2. follows by Lemma 44;

3. we use Lemma 45 to ensure that the communication is possible (when $\mu = \tau$), or that the subject of μ can be related to the subject of the prefix being triggered (when $\mu \neq \tau$). The resulting processes are still related through $=_{a,b}$ since this relation commutes with $\equiv$ and contexts. □

Lemma 47. *If P and Q are prefix-free and $\Phi(P) = \Phi(Q)$ then $P \sim_{\mathrm{bn}} Q$.*

Proof. The corresponding relation is a $\sim_{\mathrm{bn}}$-bisimulation: all condition checks are straightforward, even when we add arcs since Definition 14 is compositional: $\Phi(P \mid Q)$ only depends on $\Phi(P)$ and $\Phi(Q)$. □

Lemma 48. *For every process P of Explicit Fusions, if either $[\![P]\!] \triangleright a \leqslant b$ or $[\![P]\!] \triangleright a \curlyvee b$, then $[\![P]\!] \triangleright a = b$ and $P \equiv P \mid a{=}b$ (i.e. a and b are related through the fusions in P).*

Proof. First we prove that $[\![P]\!] \triangleright a \leqslant b$ implies $[\![P]\!] \triangleright b \leqslant a$ by induction on the derivation of the first judgement. The only interesting case is when we use an arc b/a: by definition of the encoding, we know that the arc a/b is present as soon as b/a is, which gives the expected property.

We then exploit this property in the case where $[\![P]\!] \triangleright a \curlyvee b$. We know that there is a name u such that $a \leqslant u$ and $b \leqslant u$ and we use the first part of the proof to deduce $u \leqslant a$ and $u \leqslant b$, which yields $a \leqslant b$ and $b \leqslant a$. □

Theorem 49 *(Explicit Fusions, operational correspondence). Let P, Q be processes of the explicit fusion calculus, and $\longrightarrow_{\mathrm{EF}}$ the reduction relation in the calculus.*

1. *If $P \equiv Q$ then $[\![P]\!] \simeq_{\mathrm{bn}} [\![Q]\!]$;*
2. *if $P \longrightarrow_{\mathrm{EF}} P'$ then $[\![P]\!] \longrightarrow_{\mathrm{bn}} \approx_{\mathrm{bn}} [\![P']\!]$;*
3. *conversely, if $[\![P]\!] \longrightarrow_{\mathrm{bn}} Q$, then $Q \approx_{\mathrm{bn}} [\![P']\!]$ for some P' such that $P \longrightarrow_{\mathrm{EF}} P'$.*

A similar result holds for the fusion calculus [29]. The present statement is simpler, because in Explicit Fusions the triggering of a reduction step does not depend on the presence of a restriction.

Proof. 1) Thanks to Theorem 43, it is enough to prove $[\![P]\!] \sim_{\mathrm{bn}} [\![Q]\!]$, which we do by induction on the derivation of $P \equiv Q$. The standard base cases like associativity are treated easily, because the translation yields structurally congruent processes.

The other base cases are those dealing with fusion processes:

- law $[\![a{=}b \mid P]\!] \sim_{\mathrm{bn}} [\![a{=}b \mid P\{a/b\}]\!]$ follows by Lemma 46,
- laws $[\![a{=}a]\!] \sim_{\mathrm{bn}} [\![0]\!]$ and $[\![(\nu a)a{=}b]\!] \sim_{\mathrm{bn}} [\![0]\!]$ follow by Lemma 47.

We conclude thanks to the fact that $\sim_{\mathrm{bn}}$ is a congruence and an equivalence relation.

2) Thanks to 1) and the fact that relation $\longrightarrow_{\mathrm{bn}}$ is preserved by active contexts, we only have to consider the base case of the reduction relation: $R \triangleq \overline{a}b.P \mid ac.Q \longrightarrow_{\mathrm{EF}} b{=}c \mid P \mid Q \triangleq R'$.

Moreover, since $\approx_{\mathrm{bn}}$ is stable by $\equiv$ and active contexts, we can restrict ourselves to considering the reduction $[\![R]\!] \longrightarrow_{\mathrm{bn}} (\nu w, y)(b/c \mid w/y \mid wb.[\![P]\!] \mid \overline{y}\langle c\rangle.[\![Q]\!])$. The latter process can only make a deterministic reduction to process $[\![R']\!] \mid (\nu w, y)(w/y)$, which is strongly bisimilar to $[\![R']\!]$ by Lemma 47.

3) The reduction $[\![P]\!] \longrightarrow_{\mathrm{bn}} P_1$ comes from a communication between two prefixes $[\![\overline{a}\langle v\rangle.Q]\!]$ and $[\![b\langle x\rangle.R]\!]$ where P must be of the form $P \equiv (\nu\tilde{c})(S \mid \overline{a}\langle v\rangle.Q \mid b\langle x\rangle.R)$ and $[\![S]\!] \triangleright a \curlyvee b$. Lemma 48 gives $S \equiv S \mid a{=}b$ to get finally $P \equiv (\nu\tilde{c})(S \mid \overline{a}\langle v\rangle.Q \mid a\langle x\rangle.R)$ and $P \longrightarrow_{\mathrm{EF}} (\nu\tilde{c})(S \mid v{=}x \mid Q \mid R) \triangleq P'$. We relate $[\![P']\!]$ to P_1 through $\approx_{\mathrm{bn}}$ by the same reasoning as in 2. □

6.2. π-calculus

The embedding of the π-calculus into any fusion calculus is defined by translating the bound input construct as follows:

$$[\![a(x).P]\!] = (\nu x)\, ax.[\![P]\!]$$

(the other constructs being translated homomorphically). The same encoding can be used for $\pi\mathbb{P}$.

The encoding of π-calculus into Fusions is not fully abstract for barbed congruence. For instance, in the π-calculus, a freshly created channel is guaranteed to remain different from all other existing channels. Thus in a process $\nu a\,(\overline{b}a.(a.P \mid \overline{c}.Q))$, the two prefixes $a.P$ and $\overline{c}.Q$ may never interact with each other in the π-calculus. This property does not hold after translation in the fusion calculus, since a recipient of the newly created name a can equate it with another name (e.g., with c, using the context $bc.\mathbf{0} \mid [\cdot]$).

As is the case for the encoding of Fusions in Section 6.1, we do not know whether the encoding of the full π-calculus into $\pi\mathbb{P}$ is fully abstract; we only present an operational correspondence result (Section 6.2.1). In Section 6.2.2, we establish full abstraction for the encoding restricted to Aπ, the asynchronous subset of the π-calculus where no continuation is allowed after the output prefix. For Aπ, full abstraction holds with respect to barbed congruence in the π-calculus.

6.2.1. Operational correspondence for synchronous processes

We use the following properties of the encoding, where $\longrightarrow_\pi$ stands for reduction in the π-calculus. Barbs in the π-calculus are defined in the standard way: $P \downarrow_a$ iff $P \equiv (\nu \tilde{c})(\alpha.P \mid R)$ where α is a prefix whose subject is a (this is equivalent to $P = E[\alpha.P_1]$ for some active context E).

Lemma 50 *(Operational correspondence). Let P, Q be π-calculus processes.*

1. *$P \equiv Q$ iff $[\![P]\!] \equiv [\![Q]\!]$;*
2. *if $P \longrightarrow_\pi P'$ then $[\![P]\!] \longrightarrow_{\mathrm{bn}} \simeq_{\mathrm{bn}} [\![P']\!]$;*
3. *conversely, if $[\![P]\!] \longrightarrow_{\mathrm{bn}} P_1$ then there is P' such that $P \longrightarrow_\pi P'$ and $P_1 \simeq_{\mathrm{bn}} [\![P']\!]$;*
4. *$P \downarrow_a$ iff $[\![P]\!] \downarrow_a$.*

Proof.

1. The direction from left to right is straightforward. For the converse direction, suppose $[\![P]\!] \equiv R_1$, then there exists R such that $P \equiv R$ and we can obtain R_1 from $[\![R]\!]$ only by moving restrictions on input objects. Hence if $R_1 = [\![Q]\!]$, we have necessarily $R = Q$.
2. By induction on $P \longrightarrow_\pi P'$.
   Base case: if $\overline{a}b.P \mid a(x).Q \longrightarrow_\pi P \mid Q\{b/x\}$ we have $[\![\overline{a}b.P \mid a(x).Q]\!] \longrightarrow_{\mathrm{bn}} (\nu x)([\![P]\!] \mid b/x \mid [\![Q]\!]) \simeq_{\mathrm{bn}} [\![P]\!] \mid [\![Q]\!]\{b/x\}$ by Lemma 26 since x has no negative occurrence in $[\![Q]\!]$.
   Inductive case: $E[P] \longrightarrow_\pi E[P']$ with $P \longrightarrow_\pi P'$ and E is an active context. It is enough to observe that $[\![E]\!]$ is an active context, since $\longrightarrow_{\mathrm{bn}}$ and $\simeq_{\mathrm{bn}}$ are both preserved by active contexts.
   Inductive case: $P \equiv P_1 \longrightarrow_\pi P_1' \equiv P'$ with, by induction, $[\![P_1]\!] \longrightarrow_{\mathrm{bn}} \simeq_{\mathrm{bn}} [\![P_1']\!]$. Using the property established above, $[\![P]\!] \equiv \longrightarrow_{\mathrm{bn}} \simeq_{\mathrm{bn}} \equiv [\![P']\!]$. We conclude by remarking that $(\equiv \longrightarrow_{\mathrm{bn}} \simeq_{\mathrm{bn}} \equiv) \subseteq (\longrightarrow_{\mathrm{bn}} \simeq_{\mathrm{bn}})$, by definition of $\longrightarrow_{\mathrm{bn}}$ and $\simeq_{\mathrm{bn}}$.
3. Suppose $[\![P]\!] \longrightarrow_{\mathrm{bn}} P_1$. Since $[\![P]\!]$ does not contain any arc process, the reduction comes from a communication between two prefixes on the same name a: $[\![P]\!] \equiv E_1[\overline{a}b.[\![Q]\!] \mid ax.[\![R]\!]]$ with E binding x, and then, keeping track of all modifications brought by $\equiv$, we know that P_1 is of the form $P_1 \equiv E_1[[\![Q]\!] \mid b/x \mid [\![R]\!]]$. We can recover $P \equiv E[\overline{a}b.Q \mid a(x).R] \longrightarrow_\pi E[Q \mid R\{b/x\}] \triangleq P'$. Then $[\![P']\!] = [\![E]\!][[\![Q]\!] \mid [\![R]\!]\{b/x\}] \equiv E_1[[\![Q]\!] \mid [\![R]\!]\{b/x\}] \simeq_{\mathrm{bn}} P_1$.
   In the reasoning above, we rely on the following decomposition: if $[\![P]\!] \equiv E_1[\overline{a}b.Q_1 \mid ax.R_1]$ then $Q_1 \equiv [\![Q]\!]$, $R_1 \equiv [\![R]\!]$ and $P \equiv E[\overline{a}b.Q \mid a(x).R]$ with $[\![E]\!][(\nu x)[\cdot]] \equiv E_1[\cdot]$. We prove this decomposition by combining techniques we have used in the first item, to deduce that Q_1 and R_1 are structurally congruent to the encodings of some processes, with the fact that structural congruence acts independently on either sides of the prefixes $\overline{a}b$ and ax.
4. The implication from left to right is straightforward by induction. Remark in passing that to detect the input barb, a synchronous "tester process" $\overline{a}b.\omega$ is needed (note that the *asynchronous* version of behavioural equivalence does not take input barbs into account). The other implication follows from the fact that there is no arc in $[\![P]\!]$ so $[\![P]\!] \downarrow_a$ if and only if $[\![P]\!]$ contains a prefix whose subject is a (which is equivalent to P satisfying the same property). □

One inclusion of the full abstraction result actually holds for the whole π-calculus:

Lemma 51. *Let P and Q be π terms. Then $[\![P]\!] \simeq_{\mathrm{bn}} [\![Q]\!]$ implies $P \simeq_\pi Q$.*

Proof. The relation $\{(P, Q) \mid [\![P]\!] \simeq_{\mathrm{bn}} [\![Q]\!]\}$ is reduction-closed (consequence of Lemma 50), barb-preserving (consequence of Lemma 50), and context-closed: if C is a π context then there exists a $\pi\mathtt{P}$ context C_1 such that $[\![C[P]]\!] = C_1[[\![P]\!]]$, and similarly for Q. Hence $[\![P]\!] \simeq_{\mathrm{bn}} [\![Q]\!]$ implies $[\![C[P]]\!] \simeq_{\mathrm{bn}} [\![C[Q]]\!]$. □

6.2.2. Full abstraction for asynchronous processes

We now establish full abstraction for the encoding of Aπ, the asynchronous π-calculus. We say that $P \in \pi\mathtt{P}$ is *asynchronous* if the continuation of all outputs in P is $\mathbf{0}$. We can remark that the encoding of a process in Aπ is an asynchronous $\pi\mathtt{P}$ process.

We first establish correspondence results for labelled transitions. Labelled transitions in the π-calculus are written $\xrightarrow{\mu}_\pi$ (in particular, $\mu = a(x)$ is a *late* input). The following lemma relates visible labels in π to visible labels in $\pi\mathtt{P}$.

Lemma 52 *(Label correspondences). Let P be any π process and f a fresh name.*

1. *If $P \xrightarrow{\overline{a}c}_\pi P'$ then $[\![P]\!] \xrightarrow{\overline{a}f}_{\mathrm{bn}} \equiv c/f \mid [\![P']\!]$.*
2. *If $P \xrightarrow{\overline{a}(c)}_\pi P'$ then $[\![P]\!] \xrightarrow{\overline{a}f}_{\mathrm{bn}} \equiv (\nu c)(c/f \mid [\![P']\!])$.*
3. *If $P \xrightarrow{a(x)}_\pi P'$ then $[\![P]\!] \xrightarrow{af}_{\mathrm{bn}} \equiv (\nu x)(f/x \mid [\![P']\!])$.*
4. *If $[\![P]\!] \xrightarrow{\overline{a}f}_{\mathrm{bn}} P_1$ then either $P \xrightarrow{\overline{a}c}_\pi P'$ with $P_1 \equiv c/f \mid [\![P']\!]$, or $P \xrightarrow{\overline{a}(c)}_\pi P'$ with $P_1 \equiv (\nu c)(c/f \mid [\![P']\!])$.*
5. *If $[\![P]\!] \xrightarrow{af}_{\mathrm{bn}} P_1$ then $P \xrightarrow{a(x)}_\pi P'$ with $P_1 \equiv (\nu x).(f/x \mid [\![P']\!])$.*

The asynchronous version of $\pi\mathtt{P}$ is defined as a syntactic restriction of $\pi\mathtt{P}$, by forbidding continuations after output prefixes, along the lines of Aπ.

Lemma 53 *(Decomposition of transitions, asynchronous $\pi\mathtt{P}$). We say that an arc a/b is* visible *in a process P if either a or b (or both) occurs free in P.*

Let P be an asynchronous $\pi\mathtt{P}$ term with no visible arc, σ a parallel composition of arcs, and f, g some fresh names.

1. *If $P \mid \sigma \xrightarrow{\tau}_{\mathrm{bn}} P_t$ then*
   - *either $P \xrightarrow{\tau}_{\mathrm{bn}} P_1$ with $P_t \equiv P_1 \mid \sigma$,*
   - *or $P \xrightarrow{\overline{a}f}_{\mathrm{bn}} P_1 \xrightarrow{bg}_{\mathrm{bn}} P_2$ with $P_t \sim_{\mathrm{bn}} (\boldsymbol{\nu} f, g)(P_2 \mid f/g) \mid \sigma$ and $\sigma \triangleright a \curlyvee b$.*
2. *Suppose $P \xrightarrow{\overline{a}f}_{\mathrm{bn}} P_1 \xrightarrow{bg}_{\mathrm{bn}} P_2$ and $\sigma \triangleright a \curlyvee b$. Then $P \mid \sigma \xrightarrow{\tau}_{\mathrm{bn}} \sim_{\mathrm{bn}} (\boldsymbol{\nu} f, g)(P_2 \mid f/g) \mid \sigma$.*

This result is the $\pi\mathtt{P}$ counterpart of a similar property in Aπ. In the proof, we use $\sim_{\mathrm{bn}}$ for renaming and concatenating arcs involving fresh names using Lemma 34.

Lemma 54. *Let P and Q be Aπ processes. Then $P \simeq_\pi Q$ implies $[\![P]\!] \simeq_{\mathrm{bn}} [\![Q]\!]$.*

Proof. Thanks to Theorem 40 and to the characterisation of barbed congruence by ground bisimilarity in the asynchronous π-calculus [33], we only have to prove that $P \sim_{\mathrm{g}} Q$ implies $[\![P]\!] \sim_{\mathrm{bn}} [\![Q]\!]$. We do so by showing that the following relation is a $\sim_{\mathrm{bn}}$-bisimulation up to restriction and $\sim_{\mathrm{bn}}$:

$$\mathcal{R} \triangleq \{([\![P]\!] \mid \sigma, [\![Q]\!] \mid \sigma) \;\mid\; P \sim_{\mathrm{g}} Q\} \ ,$$

where σ stands for any parallel composition of arcs.

In order to establish this property, we remark that $[\![P]\!]$ is an arc-free process. We have to check the correspondences on transitions between π and $\pi\mathtt{P}$, using Lemmas 50, 52 and 53. We analyse all possible transitions from $[\![P]\!] \mid \sigma$:

1. In the case of an input transition, we have $[\![P]\!] \mid \sigma \xrightarrow{af}_{\mathrm{bn}} \sim_{\mathrm{bn}} (\boldsymbol{\nu} x)(f/x \mid [\![P']\!] \mid \sigma)$ with $P \xrightarrow{b(x)}_\pi P'$ for some b such that $\sigma \triangleright a \curlyvee b$. We deduce from $P \sim_{\mathrm{g}} Q$ that $[\![Q]\!] \xrightarrow{bf}_{\mathrm{bn}} \sim_{\mathrm{bn}} (\boldsymbol{\nu} x)(f/x \mid [\![Q']\!])$. We use σ to derive a transition along the original label af to a process bisimilar to $(\boldsymbol{\nu} x)(f/x \mid [\![Q']\!] \mid \sigma)$, and the resulting processes are related by $\sim_{\mathrm{bn}} \mathcal{R}^{\boldsymbol{\nu}} \sim_{\mathrm{bn}}$.
2. Similarly for an output transition $[\![P]\!] \mid \sigma \xrightarrow{\overline{a}f}_{\mathrm{bn}} \sim_{\mathrm{bn}} (\boldsymbol{\nu} \hat{c})(c/f \mid [\![P']\!])$, we have $P \xrightarrow{\boldsymbol{\nu}\hat{c}bc}_\pi P'$ with $\hat{c} \in \{\emptyset, \{c\}\}$ and $\sigma \triangleright a \curlyvee b$. The reasoning is similar to the previous case.
3. The case of a τ transition $[\![P]\!] \mid \sigma \xrightarrow{\tau}_{\mathrm{bn}} P_t$ yields two cases by Lemma 53:
   (a) either $P_t \equiv P_1 \mid \sigma$ and $[\![P]\!] \xrightarrow{\tau}_{\mathrm{bn}} P_1$. This is handled using Lemma 50;
   (b) or, in two steps:

$$[\![P]\!] \xrightarrow{\overline{a}f}_{\mathrm{bn}} \xrightarrow{bg}_{\mathrm{bn}} (\boldsymbol{\nu} \hat{c}, x)(c/f \mid g/x \mid [\![P'']\!]) \triangleq P_2$$
$$P \xrightarrow{\boldsymbol{\nu}\hat{c}\overline{a}c}_\pi \xrightarrow{b(x)}_\pi P''$$

   such that $\sigma \triangleright a \curlyvee b$ and $P_t \sim_{\mathrm{bn}} (\boldsymbol{\nu} f, g)(P_2 \mid f/g)$.
   We can again play the ground bisimilarity game, first with transition $\boldsymbol{\nu}\hat{c}\overline{a}c$ and then with transition $b(x)$ to get $Q \xrightarrow{\boldsymbol{\nu}\hat{c}\overline{a}c}_\pi \xrightarrow{b(x)}_\pi Q''$ with $P'' \sim_{\mathrm{g}} Q''$. Using Lemma 52 we obtain two transitions along $\overline{a}f$ and bg from $[\![Q]\!]$, which we compose with Lemma 53 to get $[\![Q]\!] \mid \sigma \xrightarrow{\tau}_{\mathrm{bn}} (\boldsymbol{\nu} f, g)((\boldsymbol{\nu}\hat{c}, x)(g/x \mid c/f \mid [\![Q'']\!]) \mid f/g)$. Bisimilarity can be used to finally get processes related through $\mathcal{R}^{\boldsymbol{\nu}}$ (see the definition of $\mathcal{R}^{\boldsymbol{\nu}}$ before Lemma 37):

$$(\boldsymbol{\nu} f, g, \hat{c}, x)([\![P'']\!] \mid \sigma') \;\; \mathcal{R}^{\boldsymbol{\nu}} \;\; (\boldsymbol{\nu} f, g, \hat{c}, x)([\![Q'']\!] \mid \sigma')$$

   with $\sigma' = \sigma \mid c/f \mid f/g \mid g/x$.

Relation $\mathcal{R}$ is symmetric, and clearly satisfies the clause about joinability and the clause about the addition of arcs. Thus $\mathcal{R}$ is a $\sim_{\mathrm{bn}}$-bisimulation up to restriction and $\sim_{\mathrm{bn}}$. □

Theorem 55 *(Full abstraction for the asynchronous π-calculus). Suppose P, Q are Aπ processes. Then $P \simeq_\pi Q$ iff $[\![P]\!] \simeq_{\mathrm{bn}} [\![Q]\!]$.*

Proof. Follows from Lemmas 51 and 54. □

In the theorem, $\simeq_\pi$ is barbed congruence in the full π-calculus, as opposed to *asynchronous* barbed congruence, where output barbs are observable but input barbs are not. We conjecture that in a π-calculus with neither replication nor choice,

asynchronous and synchronous barbed congruences coincide, and hence the theorem holds also under asynchronous barbed congruence.

Theorem 55 (and the results on types) shows that $\pi\mathtt{P}$, although syntactically similar to Fusions, is closer to the π-calculus in the management of names.

Discussion. As stated above, we do not know whether full abstraction holds for the encoding of the full π-calculus. More precisely, if the source calculus features a replication or choice operator, full abstraction would break at least for ground, early and late bisimilarities. The situation is less clear for open bisimilarity. It is also unknown whether fusion calculi can be encoded in a fully abstract way. We leave these two questions for future work.

If they exist, translations in the opposite direction (from $\pi\mathtt{P}$ to π) would likely be more convoluted: under some conditions, an arc a/b can be encoded as the π-calculus process $!b(x).\overline{a}x$, called a *forwarder* from b to a. For this encoding to work, two constraints on the source calculus are necessary: *asynchrony* (all outputs have $\mathbf{0}$ as continuation) and *locality* (a name that is used in input subject in the process may not have negative occurrences; in the π-calculus, this may be rephrased by saying that a name bound by an input may not occur as subject of an input). It might be possible to encode a version of $\pi\mathtt{P}$ satisfying these two constraints into $AL\pi$, the version of π that satisfies the same constraints. This encoding would probably bear some resemblance with M. Merro's encoding of the asynchronous monadic fusion calculus into Aπ [25].

7. Unique negative occurrences of names

In this section we consider a constrained version of the calculi discussed in the paper, where each name may have at most one negative occurrence in a process. In the fusion calculus [29] the constraint means that each name appears at most once as the object of an input. In $\pi\mathtt{P}$, the constraint affects also arcs, as their source is a negative occurrence.

We call $\pi\mathtt{P}1$ and Fu1 the constrained versions of $\pi\mathtt{P}$ and Fusions. (Note that in the case of a calculus with replications, if a free name of P appears negatively in P, then $!P$ does not qualify as a constrained process.)

It is also possible to adapt this constraint to Explicit Fusions. This requires, given a process P of Explicit Fusions, to assign a "polarisation", for each (occurrence of an) explicit fusion $a{=}b$ occurring in P, whereby either a or b is considered as negative | for prefixes, polarity is defined as above. Therefore, a process P belongs to the constrained Explicit Fusions calculus if it is possible to assign a polarity to all names occurring in explicit fusions in such a way that every name occurs at most once in negative position (either as an input object or in the negative side of an explicit fusion construct).

In order for this restriction to make sense, we need the constraint to be preserved by structural congruence. This is the case for all structural congruence laws of Explicit Fusions, but one. In particular, the law $P \mid a{=}b \equiv P\{b/a\} \mid a{=}b$ preserves the constraint, even if moving from the left hand side to the right hand side might involve changing the polarisation of the fusion $a{=}b$ (in order to deem b as positive in $a{=}b$). The law that causes a problem is the reflexivity law $a{=}a \equiv \mathbf{0}$, as including it would entail that structural congruence would not preserve the constraint (consider, for instance, $ca \mid \mathbf{0} \equiv ca \mid a{=}a$). We therefore do not include the reflexivity law in the definition of $\equiv$. This has no effect on the induced equivalence, because symmetry and transitivity are still derivable without it.

We call the resulting calculus EF1. The constraint is rather draconian, bringing the calculi closer to the π-calculus, where the constraint is enforced by having binding input. It may be remarked that when encoding the π-calculus in $\pi\mathtt{P}$ (see Section 6.2), the image is included in $\pi\mathtt{P}1$. Still, the constraint is more generous than tying the input to a binder as in π. For instance, we have more complex forms of causality involving input, as in the process $(\boldsymbol{\nu}x)(ax.\overline{w}t \mid \overline{b}x)$, where the input at a blocks the output at w, and can be triggered before or after the output at b takes place. Delayed inputs [25] are thus simply encodable in $\pi\mathtt{P}1$ as $(\boldsymbol{\nu}x)\,(ax \mid P)$, whereas in π they represent a non-trivial extension of the calculus.

In the remainder of this section, we show the consequences of imposing unique negative occurrences of names for behavioural equivalence (Section 7.1) and types (Section 7.2).

7.1. Behavioural properties of constrained calculi

By Lemma 9, a $\pi\mathtt{P}1$ process can only reduce to a $\pi\mathtt{P}1$ process. This allows us to adapt (weak) behavioural equivalence ($\approx_{\mathrm{bn}}$, Definition 1) to $\pi\mathtt{P}1$, yielding $\approx_{\pi\mathtt{P}1\mathrm{bn}}$, where in the last clause, we only allow contexts which, when applied to the processes being compared, yield $\pi\mathtt{P}1$ processes. Similarly, $\approx_{\mathrm{ea}}$, the eager equivalence, yields $\approx_{\pi\mathtt{P}1\mathrm{ea}}$.

We show that the constraint makes certain differences between calculi or semantics disappear. First, in the eager semantics for $\pi\mathtt{P}1$, reduction steps where arcs are used to rewrite prefixes are deterministic.

Lemma 56 *(Eager reduction in $\pi\mathtt{P}1$). Consider $P \in \pi\mathtt{P}1$, and write $P \rightarrow_{\mathrm{rew}} P'$ whenever $P \longrightarrow_{\mathrm{ea}} P'$ where the reduction is a rewrite step involving an arc. Then $P \rightarrow_{\mathrm{rew}} P'$ implies $P \approx_{\pi\mathtt{P}1\mathrm{ea}} P'$.*

Proof. In $\pi\mathtt{P}1$, every name has at most one father in the partial order (a is a father of b if a is above b and no other name stands between a and b in the order), by definition of $\pi\mathtt{P}1$, which has two important consequences. The first consequence is that P and P' have the same barbs when $P \rightarrow_{\mathrm{rew}} P'$. The second is that $\rightarrow_{\mathrm{rew}}$ satisfies the diamond property (that is, that any two $\rightarrow_{\mathrm{rew}}$ transitions sharing the same origin commute) in $\pi\mathtt{P}1$.

This allows us to show that $\mathcal{R} = \{(P, P').\ P \to_{\mathsf{rew}} P'\} \cup \{(P, P)\} \subseteq \approx_{\pi\,\mathbb{P}1\mathrm{ea}}$. The interesting cases arise from the reductions of P. If $P \to_{\mathsf{rew}} P_1$, then the diamond property allows us to conclude. If $P \longrightarrow_{\mathsf{ea}} P_1$ through a communication, then either $P' \to_{\mathsf{rew}} \longrightarrow_{\mathsf{ea}} P_1$ (in the case where the communication involves the prefix that has been rewritten using $\to_{\mathsf{rew}}$), or $P' \longrightarrow_{\mathsf{ea}} P'_1$ with $P_1 \to_{\mathsf{rew}} P'_1$. In both cases, we close the diagram. □

In addition to Lemma 56, we can also observe that in $\pi\,\mathbb{P}1$, we have the law $(\nu a)a/b \approx_{\pi\,\mathbb{P}1\mathrm{ea}} 0$, which follows from Theorem 57 and Example 20.

In fact, since reduction steps involving rewritings in the eager semantics do not correspond to commitments, the two versions of weak equivalence in $\pi\,\mathbb{P}1$ coincide:

Theorem 57 *(Coincidence of eager and by-need semantics in $\pi\,\mathbb{P}1$).* $\approx_{\pi\,\mathbb{P}1\mathrm{ea}} = \approx_{\pi\,\mathbb{P}1\mathrm{bn}}$.

Proof. We first prove two commutation properties, where we write, like above, $P \to_{\mathsf{rew}} P'$ when $P \to_{\mathsf{ea}} P'$ follows from a rewriting step involving an arc, and $P \to_{\mathsf{com}} P'$ when $P \to_{\mathsf{ea}} P'$ follows from a communication step. We have the following:

1. $(\leftarrow_{\mathsf{rew}} \to_{\mathsf{bn}}) \subseteq (\to_{\mathsf{bn}} \leftarrow_{\mathsf{rew}}^{=})$ (which implies $(\Leftarrow_{\mathsf{rew}} \Rightarrow_{\mathsf{bn}}) \subseteq (\Rightarrow_{\mathsf{bn}} \Leftarrow_{\mathsf{rew}})$). (Here we write $\mathcal{R}^{=}$ for the extension of $\mathcal{R}$ with the reflexive relation.) The interesting case is when $\to_{\mathsf{rew}}$ rewrites a name a to b, and a is involved in the communication taking place in the $\to_{\mathsf{bn}}$ step (which exploits $a \curlyvee c$). In this case, the $\to_{\mathsf{bn}}$ step can simulate the rewriting step (since $b \curlyvee c$), because $\to_{\mathsf{rew}}$ makes no commitment.
2. $(\to_{\mathsf{rew}} \to_{\mathsf{bn}}) \subseteq (\to_{\mathsf{bn}} \to_{\mathsf{rew}}^{=})$ (which implies $(\Rightarrow_{\mathsf{rew}} \to_{\mathsf{bn}}) \subseteq (\to_{\mathsf{bn}} \Rightarrow_{\mathsf{rew}})$). Using the same kind of reasoning, if $\to_{\mathsf{rew}}$ rewrites a name a into b that is used afterwards in the communication step of $\to_{\mathsf{bn}}$ (using $b \curlyvee c$), then $\to_{\mathsf{bn}}$ can use the property $a \curlyvee c$.

To establish $\approx_{\pi\,\mathbb{P}1\mathrm{bn}} \subseteq \approx_{\pi\,\mathbb{P}1\mathrm{ea}}$, we prove that $\mathcal{R} = (\Leftarrow_{\mathsf{rew}} \approx_{\pi\,\mathbb{P}1\mathrm{bn}} \Rightarrow_{\mathsf{rew}})$ satisfies the following stability property under eager reductions: $(\leftarrow_{\mathsf{ea}} \mathcal{R}) \subseteq (\mathcal{R} \Leftarrow_{\mathsf{ea}})$. Consider P, P_1, Q_1, Q and P' such that $P \to_{\mathsf{ea}} P'$ and $P \Leftarrow_{\mathsf{rew}} P_1 \approx_{\pi\,\mathbb{P}1\mathrm{bn}} Q_1 \Rightarrow_{\mathsf{rew}} Q$. We must show that we can find some Q' to close the diagram.

- The case $P \to_{\mathsf{rew}} P'$ is straightforward (we choose $Q' = Q$).
- Suppose then $P \to_{\mathsf{com}} P'$, which implies $P \to_{\mathsf{bn}} P'$. By property 2 above, we get $P_1 \to_{\mathsf{bn}} P'_1$ and $P'_1 \Rightarrow_{\mathsf{rew}} P'$. This allows us to play the $\approx_{\pi\,\mathbb{P}1\mathrm{bn}}$ game, and obtain $P'_1 \approx_{\pi\,\mathbb{P}1\mathrm{bn}} Q'_1$ such that $Q_1 \Rightarrow_{\mathsf{bn}} Q'_1$, which, thanks to property 1, gives $Q'_1 \Rightarrow_{\mathsf{rew}} Q'$ with $Q \Rightarrow_{\mathsf{bn}} Q'$ (hence $Q \Rightarrow_{\mathsf{ea}} Q'$ by Lemma 17). The reasoning is depicted on the following diagram:

$$\begin{array}{ccccccc}
P & \xLeftarrow[\mathsf{rew}]{} & P_1 & \overset{\approx_{\pi\,\mathbb{P}1\mathrm{bn}}}{\cdots} & Q_1 & \xRightarrow[\mathsf{rew}]{} & Q \\
{\scriptstyle\mathsf{com}}\downarrow{\scriptstyle\mathsf{bn}} & & \downarrow{\scriptstyle\mathsf{bn}} & & \Downarrow{\scriptstyle\mathsf{bn}} & & \Downarrow{\scriptstyle\mathsf{bn}\,\mathsf{ea}} \\
P' & \xLeftarrow[\mathsf{rew}]{} & P'_1 & \overset{\approx_{\pi\,\mathbb{P}1\mathrm{bn}}}{\cdots} & Q'_1 & \xRightarrow[\mathsf{rew}]{} & Q'
\end{array}$$

To prove $\approx_{\pi\,\mathbb{P}1\mathrm{ea}} \subseteq \approx_{\pi\,\mathbb{P}1\mathrm{bn}}$ we show that $\approx_{\pi\,\mathbb{P}1\mathrm{ea}}$ enjoys the following property of stability under by-need reductions: $(\leftarrow_{\mathsf{bn}} \approx_{\pi\,\mathbb{P}1\mathrm{ea}}) \subseteq (\approx_{\pi\,\mathbb{P}1\mathrm{ea}} \Leftarrow_{\mathsf{bn}})$. Suppose that $P \approx_{\pi\,\mathbb{P}1\mathrm{ea}} Q$ and $P \to_{\mathsf{bn}} P'$ which implies by Lemma 17 that $P \Rightarrow_{\mathsf{ea}} P'$. This allows us to draw the $\approx_{\pi\,\mathbb{P}1\mathrm{ea}}$ diagram, to get $Q \Rightarrow_{\mathsf{ea}} Q'$ such that $P' \approx_{\pi\,\mathbb{P}1\mathrm{ea}} Q'$. Since $\to_{\mathsf{ea}} = (\to_{\mathsf{com}} \cup \to_{\mathsf{rew}})$ and $\to_{\mathsf{com}} \subseteq \to_{\mathsf{bn}}$, we deduce $\Rightarrow_{\mathsf{ea}} \subseteq (\to_{\mathsf{bn}} \cup \to_{\mathsf{rew}})^*$ and from property 2 above we have $\Rightarrow_{\mathsf{ea}} \subseteq \Rightarrow_{\mathsf{bn}} \Rightarrow_{\mathsf{rew}}$, i.e., $Q \Rightarrow_{\mathsf{bn}} Q'' \Rightarrow_{\mathsf{rew}} Q'$ for some Q''. We are able to close the diagram with Q'' since $\Rightarrow_{\mathsf{rew}} \subseteq \approx_{\pi\,\mathbb{P}1\mathrm{ea}}$ by Lemma 56. Finally, since a name has a negative occurrence in Q' if and only if it has one in Q'', we have that $P' \approx_{\pi\,\mathbb{P}1\mathrm{ea}} Q' \approx_{\pi\,\mathbb{P}1\mathrm{ea}} Q''$ implies $P' \approx_{\pi\,\mathbb{P}1\mathrm{ea}} Q''$.

$$\begin{array}{ccccccc}
P & = & P & \overset{\approx_{\pi\,\mathbb{P}1\mathrm{ea}}}{\cdots} & Q & = & Q \\
{\scriptstyle\mathsf{bn}}\downarrow & & \Downarrow{\scriptstyle\mathsf{ea}} & & \Downarrow{\scriptstyle\mathsf{ea}} & & \Downarrow{\scriptstyle\mathsf{bn}} \\
P' & = & P' & \overset{\approx_{\pi\,\mathbb{P}1\mathrm{ea}}}{\cdots} & Q' & \underset{\mathsf{rew}}{\overset{\approx_{\pi\,\mathbb{P}1\mathrm{ea}}}{\cdots}} & Q''
\end{array}$$

We thus proved that the reduction diagrams for $\approx_{\pi\,\mathbb{P}1\mathrm{ea}}$ and $\approx_{\pi\,\mathbb{P}1\mathrm{bn}}$ can be related. Properties involving context closure and barbs are common in the eager and by-need cases, so that $\approx_{\pi\,\mathbb{P}1\mathrm{ea}} = \approx_{\pi\,\mathbb{P}1\mathrm{bn}}$. □

It also appears that the calculi $\pi\,\mathbb{P}1$ and Fu1, defined by the constraint on unique negative occurrence of names, are behaviourally close. For instance, in $\pi\,\mathbb{P}1$ the directionality of arcs is irrelevant (that is, arcs behave like explicit fusions), as shown by the following law (where we omit the subscripts 'ea' and 'bn' in the light of Theorem 57).

Lemma 58. $a/b \approx_{\pi\,\mathbb{P}1} b/a$.

Proof. The relation relating $C[\pi.E[a/b]\{a/x\}]$ with $C[\pi.E[b/a]\{b/x\}]$ (for any context C, any active context E, and where x can only appear in subject position in E) is reduction-closed and barb preserving. □

Note that due to the constraint, the processes in Lemma 58 must be compared in contexts where a and b only occur positively.

We now prove that contextual equivalences in EF1 and $\pi\,\mathbb{P}1$ coincide. For this, when $P \in \pi\,\mathbb{P}1$, we write $\overline{P} \in$ EF1 for the canonical projection in EF1. We moreover refine the behavioural equivalences as follows: we write $P \approx^N_{\pi\,\mathbb{P}1} Q$ iff $P \approx_{\pi\,\mathbb{P}1} Q$ and all names which occur free in negative position in P and Q belong to the set of names N. Moreover, when closing by contexts in checking $\approx^N_{\pi\,\mathbb{P}1}$, we impose that contexts cannot use names in N in negative position.

Relation $\cong^N_{\text{EF1}}$ is defined along the same lines; for context closure, we impose that there exists a polarisation of each fusion construct such that the constraints about N and about uniqueness of negative occurrence are satisfied.

Lemma 59. *For all N, if all names occurring free in negative position in P and Q belong to N, we have $P \approx^N_{\pi\,\mathbb{P}1} Q$ iff $\overline{P} \cong^N_{\text{EF1}} \overline{Q}$.*

For example Lemma 58 equivalently states that $a/b \approx^{\{a,b\}}_{\pi\,\mathbb{P}1} b/a$.

Proof. We prove the two implications:

- $\{(P, Q) \mid \overline{P} \cong^N_{\text{EF1}} \overline{Q}\} \subseteq \approx^N_{\pi\,\mathbb{P}1}$ for all N:
  - context closure: if C is a $\pi\,\mathbb{P}1$ context without negative name in N, there exists a EF1 context $\overline{C}$ such that $\overline{C}[\overline{R}] = \overline{C[R]}$ and there is a trivial polarisation of $\overline{C}$ such that no negative name in $\overline{C}$ is in N. Then if we write N_C for the names occurring free in negative position in C, we have that $\overline{C}[\overline{P}] \cong^{N \cup N_C}_{\text{EF1}} \overline{C}[\overline{Q}]$, which means $C[P]$ and $C[Q]$ are in the relation;
  - reduction closure follows from the following operational correspondence:
    - if $P \to_{\pi\,\mathbb{P}} P'$ then $\overline{P} \to_{EF} \overline{P'}$ (indeed if $P \rhd a \curlyvee b$ then $\overline{P} \rhd a = b$)
    - if $\overline{P} \to_{EF} P_1$ then for some P', $P \to_{\pi\,P} P'$ and $P_1 = \overline{P'}$ (indeed if $\overline{P} \rhd a = b$ then $P \rhd a \curlyvee b$ since $P \in \pi\,\mathbb{P}1$)
  - the above two results allow us to deduce barb preservation.
- $\{(P_1, Q_1) \mid P_1 \overset{\leftrightarrow}{=} \overline{P} \wedge Q_1 \overset{\leftrightarrow}{=} \overline{Q} \wedge P \approx^N_{\pi\,\mathbb{P}1} Q\} \subseteq \cong^N_{\text{EF1}}$ for all N, where $\overset{\leftrightarrow}{=}$ is the smallest congruence on EF1 terms satisfying $a{=}b \overset{\leftrightarrow}{=} b{=}a$ for all a, b:
  - context closure: if there some EF1 context C without negative name in N, then for some $\pi\,\mathbb{P}1$ context D we know that: for all $R \in \pi\,\mathbb{P}1$ such that name negatively occurring in R is in N, $\overline{D[R]} \overset{\leftrightarrow}{=} C[\overline{R}]$. By context closure of $\approx^N_{\pi\,\mathbb{P}1}$, we know that $D[P] \approx^{N \cup N_D}_{\pi\,\mathbb{P}1} D[Q]$ and hence $(C[P_1], C[Q_1])$ is in the relation;
  - reduction closure and barb preservation are handled in the same way as above (note that $\overset{\leftrightarrow}{=}$ is a strong bisimulation). □

7.2. Capabilities and subtypes in constrained fusions

Another difference that disappears under the constraint of unique negative occurrences of names is the one concerning capabilities and subtyping in fusion calculi with respect to π and $\pi\,\mathbb{P}$, exposed in Sections 3 and 4. Indeed, it is possible to equip FU1 with an I/O type system and subtyping. To achieve this, we can use exactly the rules of $\pi\,\mathbb{P}$ in Section 4.2 — with the exception of T-ARC as FU1 does not have arcs.

To understand how I/O types work in FU1, let us analyse how the constraint on unique occurrences of negative names evolves along reductions. We first consider the following reduction step in FU1 ($\longrightarrow_{\text{FU1}}$ stands for reduction in FU1):

$$(\nu c)(\overline{a}b.P \mid ac.Q \mid R) \longrightarrow_{\text{FU1}} (P \mid Q \mid R)\{b/c\} \ . \tag{7}$$

In this reduction, only positive occurrences of c are replaced with b, since the only negative occurrence of c is in ac in the initial process. Hence, no negative occurrence of b is introduced.

The other kind of reduction in FU1 is as follows (note that the restriction is now on b, and that the substitution intuitively goes in the opposite direction w.r.t. $\pi\,\mathbb{P}$):

$$(\nu b)(\overline{a}b.P \mid ac.Q \mid R) \longrightarrow_{\text{FU1}} (P \mid Q \mid R)\{c/b\} \ . \tag{8}$$

Along reduction, the negative occurrence of c in prefix ac disappears. In the original process, the only visible usage of b is in $\overline{a}b$, which is a positive occurrence. Therefore, there is at most one negative occurrence of b in P, Q, R, hence $(P \mid Q \mid R)\{c/b\}$ contains at most one negative occurrence of c. The resulting process is hence a FU1 process. Finally, note that the following reduction from FU1 (the special case, when $b = c$) only makes disappear prefixes, and needs no special treatment:

$$\overline{a}b.P \mid ab.Q \longrightarrow_{\text{FU1}} P \mid Q \ .$$

Polarised narrowing (Theorem 10) holds in FU1 because every process of FU1 is also a process of $\pi\,\mathbb{P}$. Regarding the Subject Reduction property, we observe that typechecking the process obtained after reduction in (8) may involve changing the type of name c into a smaller type: after the reduction, name c is used at type T_b, which is a smaller type than the type of c. Accordingly, the statement of Subject Reduction is refined:

Theorem 60. *Let P be a* Fu1 *process. If $\Gamma \vdash P$ and $P \longrightarrow_{\text{Fu1}} P'$, then $\Gamma' \vdash P'$, where for at most one name c, $\Gamma'(c) \leq \Gamma(c)$; for any other name $b \neq c$, $\Gamma'(b) = \Gamma(b)$.*

Proof (Sketch). By induction on $\longrightarrow_{\text{Fu1}}$, only the base case is interesting (closure by contexts and structural congruence are handled by compositionality of the type system). There are two subcases, depending on the kind of whether the input object or the output object is being replaced.

Suppose we have the following typings of each case, which we write with additional type annotations:

1. $\Gamma, b : T_b \vdash (\boldsymbol{\nu} c : T_c)(\overline{a}b.P \mid ac.Q \mid R)$ (using (7) the process reduces to $(P \mid Q \mid R)\{b/c\}$),
2. $\Gamma, c : T_c \vdash (\boldsymbol{\nu} b : T_b)(\overline{a}b.P \mid ac.Q \mid R)$ (using (8) the process reduces to $(P \mid Q \mid R)\{c/b\}$).

By definition of Fu1, P, Q, and R have no negative occurrence of c. Moreover, because of the typing rules, we have $T_b \leq T_c$.

In the first case, since c does not appear in negative position in $P \mid Q \mid R$, applying positive narrowing on $T_b \leq T_c$ is enough to conclude without changing the typing environment.

In the second case, we replace b with c, and assign type T_b to c in the typing environment for the resulting process. This allows us to typecheck the newly created occurrences of c in the resulting process. The previously existing ones are still well-typed using positive narrowing. □

Remark 61. Theorem 60 does not contradict Theorems 3 and 4: the calculus obeys a syntactical constraint, which restricts the way processes can be composed. Indeed, it is not always the case that if P and Q are in Fu1, then $P \mid Q$ is also in Fu1. Therefore, in the proof of Theorem 3, it would be impossible in general to replace a with b using contexts like $(\overline{u}b \mid ua \mid -)$ or $(\overline{u}b \mid \overline{v}a \mid uc \mid vc \mid -)$, since the latter is not a Fu1 context.

8. Concluding remarks and future work

Here we discuss some lines for future work, in addition to those already mentioned in the main text.

At the cost of renouncing to the duality between input and output prefixes, $\pi\mathtt{P}$ brings the advantages of both fusion calculi (simple syntax, single binder) and the π-calculus (rich type systems and behavioural theory). We can in particular envisage to study *typed barbed equivalence* in $\pi\mathtt{P}$, which is an important tool to reason about the modelisation of systems.

We have presented and compared two semantics for $\pi\mathtt{P}$, eager and by-need. The behavioural equalities determined by the two semantics are incomparable (Example 20). However they do coincide under certain syntactic constraints (Theorem 57). While we tend to consider the advantages so far uncovered for the by-need superior, more work is needed to draw more definite conclusions.

A possible advantage of by-need is a smoother extension with dynamic operators like guarded choice, in which an action may discard a component. (In the eager case it is unclear what the effect of an arc that acts on one of the summands of a choice should be.) Choice would be useful for axiomatisations. In by-need, we would have for instance

$$(\boldsymbol{\nu} b, c)\overline{a}b.\overline{a}c.(\overline{b}|c) \sim (\boldsymbol{\nu} b, c)\overline{a}b.\overline{a}c.(\overline{b}.c + c.\overline{b}).$$

The law, valid in both $\pi\mathtt{P}$ and π, illustrates the possibility of generating fresh names that cannot be identified with other names, even if they are exported. The law fails in fusion calculi as a recipient might decide to equate b and c (cf. Section 6.2).

The coinductive characterisation of behavioural equivalence in $\pi\mathtt{P}$ has been presented in the strong case, and should be extended to the weak case. A compositional operational semantics for the eager semantics is not difficult to define. For the by-need semantics, the situation is less simple. A major advantage of such semantics would be a simpler proof of the congruence for parallel composition. (In the paper we have adopted an approach based on the reduction framework for both semantics, for uniformity and to ease comparison.) By the time the current version of the paper has been completed, a compositional presentation of the by-need operational semantics has been proposed [15].

The Psi calculi [1] provide a general framework for the (fully mechanised) behavioural theory of a generalisation of several name-passing calculi, including Fusions and π. The Psi calculi rely on an equivalence relation on channels to specify whether two channels can communicate. It could be interesting to study the representation of $\pi\mathtt{P}$ into Psi calculi [1]. Because of the generality of Psi, and in view of Section 3, adapting our results on types and behavioural equivalences for $\pi\mathtt{P}$ seems delicate.

Works by Hüttel [19,20] have shown how types can be adapted from π to Psi calculi, giving a sorting system for Explicit Fusions, in the case of [19] (Ref. [20] focuses on linear type systems). To our knowledge, no published work exists at the moment concerning capability types for Psi calculi.

Another question we would like to study is the distributed implementation of $\pi\mathtt{P}$. We believe that the ideas of the Fusion Machine [10] can be adapted to execute $\pi\mathtt{P}$ processes in a distributed manner, for the eager semantics. While the Fusion Machine uses (distributed) forwarders to provide an efficient implementation of equivalence classes between names, in $\pi\mathtt{P}$ a forwarder $a \leqslant b$ would be run to implement b/a. Implementing the by-need semantics should intuitively involve a complex protocol, to test for $a \curlyvee b$.

For the execution of $\pi\mathtt{P}1$ processes, we could furthermore exploit the property that the (distributed) agent implementing a channel can host at most one forwarder pointing outwards. Theorem 57 tells us that such an implementation would actually provide the by-need semantics for $\pi\mathtt{P}1$ processes.

Moreover, the implementation could benefit from the fact that $\pi\mathtt{P}$ is a typed calculus. Much like types are useful for modelling purposes, by making it possible to validate more equivalences, they also allow optimisations in programs, such as inlining or tail-call optimisations.

Acknowledgments

The authors acknowledge support from the ANR projects 2010-BLAN-0305 PiCoq, ANR-14-CE25-0005 Elica and 12IS02001 PACE.

References

[1] J. Bengtson, M. Johansson, J. Parrow, B. Victor, Psi-calculi: mobile processes, nominal data, and logic, in: Proceedings of the 24th Annual IEEE Symposium on Logic in Computer Science, LICS 2009, IEEE Computer Society, 2009, pp. 39–48.
[2] F. Bonchi, M.G. Buscemi, V. Ciancia, F. Gadducci, A presheaf environment for the explicit fusion calculus, J. Autom. Reason. 49 (2) (2012) 161–183.
[3] M. Boreale, M.G. Buscemi, U. Montanari, A general name binding mechanism, in: TGC, in: Lect. Notes Comput. Sci., vol. 3705, Springer, 2005, pp. 61–74.
[4] G. Boudol, Asynchrony and the Pi-Calculus, Technical report 1702, INRIA, 1992.
[5] G. Castagna, R. De Nicola, D. Varacca, Semantic subtyping for the pi-calculus, Theor. Comput. Sci. 398 (1–3) (2008) 217–242.
[6] R. De Nicola, M. Hennessy, Testing equivalences for processes, Theor. Comput. Sci. 34 (1984) 83–133.
[7] G.L. Ferrari, U. Montanari, E. Tuosto, B. Victor, K. Yemane, Modelling fusion calculus using HD-automata, in: Algebra and Coalgebra in Computer Science: First International Conference, CALCO 2005, Proceedings, in: Lect. Notes Comput. Sci., vol. 3629, Springer, 2005, pp. 142–156.
[8] C. Fournet, G. Gonthier, The reflexive CHAM and the join-calculus, in: Conference Record of POPL'96: The 23rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, ACM Press, 1996, pp. 372–385.
[9] Y. Fu, The χ-calculus, in: Proceedings of the 1997 Advances in Parallel and Distributed Computing Conference, APDC '97, IEEE Computer Society, 1997, pp. 74–81.
[10] P. Gardner, C. Laneve, L. Wischik, The fusion machine, in: CONCUR 2002 – Concurrency Theory, 13th International Conference, Proceedings, in: Lect. Notes Comput. Sci., vol. 2421, Springer, 2002, pp. 418–433.
[11] P. Gardner, L. Wischik, Explicit fusions, in: Mathematical Foundations of Computer Science 2000, 25th International Symposium, MFCS 2000, Proceedings, in: Lect. Notes Comput. Sci., vol. 1893, Springer, 2000, pp. 373–382.
[12] M. Hennessy, A Distributed Pi-Calculus, Cambridge University Press, 2007.
[13] M. Hennessy, J. Riely, Resource access control in systems of mobile agents, Inf. Comput. 173 (2002) 82–120.
[14] D. Hirschkoff, J.-M. Madiot, D. Sangiorgi, Name-passing calculi: from fusions to preorders and types, in: 28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, IEEE Computer Society, 2013, pp. 378–387.
[15] D. Hirschkoff, J.-M. Madiot, X. Xu, A behavioural theory for a π-calculus with preorders, J. Log. Algebraic Methods Program. 84 (6) (2015) 806–825.
[16] K. Honda, M. Tokoro, An object calculus for asynchronous communication, in: ECOOP'91 European Conference on Object-Oriented Programming, Proceedings, in: Lect. Notes Comput. Sci., vol. 512, Springer, 1991, pp. 133–147.
[17] K. Honda, V.T. Vasconcelos, M. Kubo, Language primitives and type discipline for structured communication-based programming, in: Programming Languages and Systems – ESOP'98, 7th European Symposium on Programming, Held as Part of ETAPS'98, Proceedings, in: Lect. Notes Comput. Sci., vol. 1381, Springer, 1998, pp. 122–138.
[18] K. Honda, N. Yoshida, On reduction-based process semantics, Theor. Comput. Sci. 152 (2) (1995) 437–486.
[19] H. Hüttel, Typed Ψ-calculi, in: CONCUR 2011 – Concurrency Theory – 22nd International Conference, Proceedings, in: Lect. Notes Comput. Sci., vol. 6901, 2011, pp. 265–279.
[20] H. Hüttel, Types for resources in Ψ-calculi, in: Trustworthy Global Computing – 8th International Symposium, TGC 2013, Revised Selected Papers, in: Lect. Notes Comput. Sci., vol. 8358, 2013, pp. 83–102.
[21] N. Kobayashi, Type systems for concurrent programs, in: 10th Anniversary Colloquium of UNU/IIST, in: Lect. Notes Comput. Sci., vol. 2757, Springer, 2003, pp. 439–453.
[22] N. Kobayashi, A new type system for deadlock-free processes, in: CONCUR 2006 – Concurrency Theory, 17th International Conference, Proceedings, in: Lect. Notes Comput. Sci., vol. 4137, Springer, 2006, pp. 233–247.
[23] N. Kobayashi, B.C. Pierce, D.N. Turner, Linearity and the pi-calculus, ACM Trans. Program. Lang. Syst. 21 (5) (1999) 914–947. Preliminary summary appeared in Proceedings of POPL'96.
[24] C. Laneve, B. Victor, Solos in concert, Math. Struct. Comput. Sci. 13 (5) (2003) 657–683.
[25] M. Merro, Locality in the Pi-Calculus and Applications to Distributed Objects, PhD thesis, École des Mines, France, 2000.
[26] M. Merro, D. Sangiorgi, On asynchrony in name-passing calculi, Math. Struct. Comput. Sci. 14 (5) (2004) 715–767.
[27] R. Milner, D. Sangiorgi, Barbed bisimulation, in: Automata, Languages and Programming, 19th International Colloquium, ICALP92, Proceedings, in: Lect. Notes Comput. Sci., vol. 623, Springer, 1992, pp. 685–695.
[28] J. Parrow, B. Victor, The update calculus (extended abstract), in: Algebraic Methodology and Software Technology, 6th International Conference, AMAST '97, Proceedings, in: Lect. Notes Comput. Sci., vol. 1349, Springer, 1997, pp. 409–423.
[29] J. Parrow, B. Victor, The fusion calculus: expressiveness and symmetry in mobile processes, in: Thirteenth Annual IEEE Symposium on Logic in Computer Science, IEEE Computer Society, 1998, pp. 176–185.
[30] J. Parrow, B. Victor, The tau-laws of fusion, in: CONCUR '98: Concurrency Theory, 9th International Conference, Proceedings, in: Lect. Notes Comput. Sci., vol. 1466, Springer, 1998, pp. 99–114.
[31] B.C. Pierce, D. Sangiorgi, Typing and subtyping for mobile processes, Math. Struct. Comput. Sci. 6 (5) (1996) 409–453.
[32] D. Sangiorgi, Pi-calculus, internal mobility, and agent-passing calculi, Theor. Comput. Sci. 167 (1–2) (1996) 235–274.
[33] D. Sangiorgi, D. Walker, The Pi-Calculus: a Theory of Mobile Processes, Cambridge University Press, 2001.
[34] L. Wischik, P. Gardner, Explicit fusions, Theor. Comput. Sci. 340 (3) (2005) 606–630.