A behavioural theory for a π-calculus with preorders

Daniel Hirschkoff[a,*], Jean-Marie Madiot[a], Xu Xian[b]

[a] *ENS Lyon, Université de Lyon, CNRS, INRIA, France*
[b] *East China University of Science and Technology, China*

Abstract

We study the behavioural theory of $\pi\mathtt{P}$, a π-calculus in the tradition of Fusions and Chi calculi. In contrast with such calculi, reduction in $\pi\mathtt{P}$ generates a preorder on names rather than an equivalence relation. We present two characterisations of barbed congruence in $\pi\mathtt{P}$: the first is based on a compositional LTS, and the second is an axiomatisation. The results in this paper bring out basic properties of $\pi\mathtt{P}$, mostly related to the interplay between the restriction operator and the preorder on names.

Consequently, $\pi\mathtt{P}$ is a calculus in the tradition of Fusion calculi, in which both types and behavioural equivalences can be exploited in order to reason rigorously about concurrent and mobile systems.

Keywords: process calculi, behavioural equivalence, labelled transition system, pi-calculus, axiomatisation

1. Introduction

The π-calculus expresses mobility via name passing, and has two binders: the input prefix binds the value to be received, and restriction is used to delimit the scope of a private name. The study of Fusions [20], Chi [10], Explicit Fusions [25] and Solos [16] has shown that using restriction as the only binder is enough to express name passing. In such calculi (which, reusing a terminology from [13], we shall refer to as *fusion calculi*), the bound input prefix, $c(x).P$, is dropped in favour of free input, $cb.P$, and communication involving two prefixes cb and $\overline{c}a$ generates the *fusion* of names a and b. This yields a pleasing symmetry between input and output prefixes; moreover, one can encode bound input in terms of free input as $(\nu x)cx.P$. Fusion calculi therefore promote *minimality* (keep only restriction as a binder) and *symmetry* (input and output prefixes play similar roles). Moreover, and most importantly, fusions act on restricted names, in contrast with the π-calculus, where restricted names can only replace names bound by input (and are thus treated like constants).

The behavioural theory of existing fusion calculi is generally simpler than in the π-calculus (in particular, bisimilarity is a congruence). Fusion calculi have

[*] Corresponding author

notably been used to analyse concurrent constraints [24], to study distributed implementations of programming languages [6, 11] and to establish connections with proof theory [9].

Symmetry comes however at a price. It has indeed been shown in [13] that i/o-types (input/output types, [21]) cannot be adapted to a fusion calculus. Such types go beyond the simple discipline of sorting, and can be useful, in particular, to reason using *typed behavioural equivalences* [21, 23].

The intuitive reason of the incompatibility of i/o-types with fusions can be explained by considering the following structural congruence law in Explicit Fusions (but the point is essentially the same for other fusion calculi):

$$a(x).P \mid a{=}b \;\equiv\; b(x).P \mid a{=}b \enspace .$$

Process $a{=}b$ is an explicit fusion. The law says that in presence of $a{=}b$, an input on a can be viewed as the same input on b, *and vice-versa* (fusion processes are somehow akin to equators, in an asynchronous setting [15]). This shows that fusions define a symmetric relation on names; as proved in [13], this is incompatible with a nontrivial (i.e., asymmetric) subtyping relation, which is necessary for i/o-types to make sense.

This observation has led in [13] to the introduction of $\pi\mathtt{P}$, a π-calculus with name preorders. The most important difference between $\pi\mathtt{P}$ and existing name-passing calculi is that interaction does not have the effect of equating (or *fusing*) two names, but instead generates an *arc* process, as follows:

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

The arc a/b expresses the fact that anything that can be done using name b can be done using a as well (but not the opposite): we say that a is *above* b. Arcs induce a preorder relation on names, which can evolve along reductions.

Arcs can modify interaction possibilities: in presence of a/b, a is above b, hence a process emitting on b can also make an output transition along channel a. In general, an output on channel c can interact with an input on d provided c and d are *joinable*, written $c \curlyvee d$, which means that there is some name that is above both c and d according to the preorder relation. To formalise these observations, the operational semantics exploits *conditions* involving names, which are either of the form $b \prec a$ (a is above b), or $a \curlyvee b$ (a and b are joinable).

$\pi\mathtt{P}$ can be described as a variant of Explicit Fusions, in which arcs replace fusion processes. Beyond the possibility to define i/o-types and subtyping for $\pi\mathtt{P}$ [13], we would like to analyse the consequences of the novel aspects of $\pi\mathtt{P}$, whose behaviour does not seem to be reducible to existing calculi.

In particular, name preorders have an impact on how processes express behaviours. Barbed congruence for $\pi\mathtt{P}$, written $\simeq$, is defined in [13]. Some laws for $\simeq$ suggest that the behavioural theory of $\pi\mathtt{P}$ differs w.r.t. existing fusion calculi. As an illustration, consider the following *interleaving law*, which is valid in $\pi\mathtt{P}$ (and in π):

$$\overline{a}(x).\overline{b}(y).(\overline{x} \mid y) \;\simeq\; \overline{a}(x).\overline{b}(y).(\overline{x}.y + y.\overline{x}) \enspace .$$

$\overline{a}(x)$ is the emission of a fresh name x on a, and $\overline{x}$ (resp. y) stands for an output (resp. input) where the value being transmitted is irrelevant. In Fusions, unlike in the π-calculus, the process that creates successively two fresh names x and y cannot prevent the context from equating ("fusing") x and y. Hence, in order for the equivalence to hold, it is necessary to add a third summand on the right, $[x = y]\tau$. This example suggests that $\pi\mathtt{P}$ gives a better control on restricted names than existing fusion calculi. This issue also motivated the study of two variants of fusion calculi that have a refined notion of restriction [4, 5].

The main purpose of the present work is to deepen the study of the behavioural theory of $\pi\mathtt{P}$, in an untyped setting. We define a Labelled Transition System (LTS) for $\pi\mathtt{P}$, and show that the induced notion of bisimilarity, written $\sim$, characterises $\simeq$ (Section 3). It can be noted that [13] presents a characterisation of barbed congruence, using an LTS that is rather ad hoc, because it is based on the definition of the reduction relation. Unlike the latter, the LTS we present here is *structural*.

Defining a structural LTS which yields a characterisation of barbed congruence is rather standard in process calculi theory. The main advantage brought by bisimulation is to make behavioural equivalence proofs more tractable, by avoiding universal quantifications over contexts. Additionally, in our case, designing the LTS allows us to gain a better understanding of the operational aspects of $\pi\mathtt{P}$, bringing to light peculiarities that are specific to our calculus.

Indeed, the LTS reveals interesting aspects of interaction in $\pi\mathtt{P}$. An important observation is related to the interplay between arcs and the restriction operator. It is for instance possible for a process to react to an input offer on some channel, say c, without being actually able to perform an output on c. This is the case for $P_0 \triangleq (\nu a)(\overline{a}(x).\mathbf{0} \mid a/c)$, as $P_0 \mid cu$ can perform a reduction (note that P_0 could do an output on c if the arc a/c was replaced with c/a). This phenomenon leads to the addition of a new type of labels in the LTS, corresponding to what we call *protected actions*, that interact with a restricted set of other actions (accordingly, we introduce *protected names*, which correspond to (usages of) names where a protected action occurs: intuitively, name c is protected in P_0). As expected, protected actions correspond to observables in the reduction-based semantics supporting the definition of $\simeq$ (e.g. the protected action from P_0 can be observed using cu.)

Arc processes do not have transitions, but they induce relations between names, which in turn influence the behaviour of processes. Accordingly, strong bisimilarity, $\sim$, not only tests transitions, but also has a clause to guarantee that related processes entail the same conditions.

Finally, the LTS also includes a label $[\varphi]\tau$, expressing "conditional synchronisation". Intuitively, process $\overline{a} \mid b$ is not able to perform a τ transition by itself, but it should be when the environment entails $a \curlyvee b$. Hence, in order for our LTS to be compositional, we include labels of the form $[\varphi]\tau$, interpreted as "τ under the condition φ".

In Section 4, we provide a second characterisation of barbed congruence, by presenting a set of laws that define an axiomatisation of $\simeq$. Algebraic laws help analysing the behaviour of the constructs of the calculus and their interplay. We present a sample of behavioural equalities, and explain how they can be derived equationally, in Section 4.1.

The axiomatisation we give is less simple than, say, the one for Fusions in [20], for two reasons: first, we manipulate preorders between names rather than equivalence relations. Second, the preorder is explicitly represented in processes, so that some equational laws must describe the interplay between processes and the preorder relation. On the contrary, such aspects are dealt with implicitly in Fusions. We show how our ideas can be adapted to provide an axiomatisation of behavioural equivalence for Explicit Fusions in Section 4.3.

The axiomatisation exploits the idea that $\pi\mathtt{P}$ processes have a *state* component, corresponding to the preorder induced by arcs. Several laws in the axiomatisation express persistence of the state component (the state can only be extended along computation). Moreover, the restriction operator prevents the state from being globally shared in general: for instance, in process P_0 above, name a can be used instead of c, but is only known inside the scope of (νa). All in all, the handling of restriction in our axiomatisation requires more care than is usually the case, due to the necessity to express the "view" that subprocesses have on the preorder of names.

To present the axiomatisation, we renounce minimality. The syntax of the calculus in this paper differs from the one in [13]: we include bound prefixes and sums with conditions, as it is customary for axiomatisations for the π-calculus [19, 23][1]. We compare the calculus from [13] with ours in Remark 1 and Proposition 2. We show that the differences are unimportant: the calculus from [13] can be encoded into ours and the behavioural equivalence is unaffected. We discuss in Section 3.4 a presentation of transitions and bisimilarity based on free prefixes.

We focus in this paper on a finite calculus. This is sufficient to illuminate the main aspects of the behavioural theory of processes. We do not expect any particular difficulty to arise, in the definition of labelled transitions and bisimilarity, from the extension of $\pi\mathtt{P}$ with a replication operator.

Outline. After introducing $\pi\mathtt{P}$ and barbed congruence in Section 2, Section 3 is devoted to the characterisation based on the LTS, and Section 4 presents the axiomatisation. In passing, we discuss an LTS for a calculus featuring free prefixes (Section 3.4) and an axiomatisation for the calculus of Explicit Fusions (Section 4.3). Related work is discussed throughout the paper, where it is relevant.

This paper is an extended version of [14]. We provide here more detailed explanations about how congruence of bisimilarity is derived (in Section 3), for which the handling of name preorders involves technicalities which could not be

[1] It can be noted that the axiomatisation of Fusions given in [20] relies only on free input and output, and treats bound prefixes as derived operators. We think that, for $\pi\mathtt{P}$, handling prefixes for bound and protected actions as derived operators would introduce further technical complications that would make the axiomatisation unnecessarily verbose and obscure.

presented in [14], for lack of space. In Section 4, we provide several proofs that were omitted in [14]. The LTS based on free prefixes (Section 3.4) is also new w.r.t. [14]. Finally, we provide a complete axiomatisation for Explicit Fusions, which was only sketched in [14].

2. $\pi\mathsf{P}$: Reduction-Based Semantics

2.1. The Calculus: Preorders and Processes

We consider a countable set of names $a, b, c, \dots, x, y, \dots$, and define conditions (φ), extended names (α, β), prefixes (π) and processes (P, Q) as follows:

$$\text{conditions} \quad \varphi ::= a \prec b \;\big|\; a \curlyvee b \qquad\qquad \text{extended names} \quad \alpha, \beta ::= a \;\big|\; \{a\}$$

$$\text{prefixes} \quad \pi ::= \alpha(x) \;\big|\; \overline{\alpha}(x) \;\big|\; [\varphi]\tau$$

$$\text{processes} \quad P, Q ::= P \mid Q \;\big|\; (\nu a)P \;\big|\; b/a \;\big|\; \textstyle\sum_{i\in I} \pi_i.P_i$$

There are two forms of conditions, ranged over with φ: $\varphi = a \prec b$ is read "b is above a" and $\varphi = a \curlyvee b$ is read "a and b are joinable". In both cases, we have $\mathsf{n}(\varphi) = \{a, b\}$. We explain in Definition 2 how we extend relations $\prec$ and $\curlyvee$ to extended names. When $\mathsf{n}(\varphi) = \{a\}$, we say that φ is *reflexive* and we abbreviate prefix $[\varphi]\tau$ as τ. Condition $a \prec b$ is ensured by the arc process b/a.

In a prefix $\alpha(x)$ or $\overline{\alpha}(x)$, we say that extended name α is in subject position, while x is in object position. As discussed in Section 1, extended names include regular and *protected* names, of the form $\{a\}$, which can be used in subject position only. The intuition behind them is that there cannot be communications between two protected names, but all other combinations are allowed. We call *protected prefix* a prefix where the subject is a protected name. A prefix of the form $[\varphi]\tau$ is called a *conditional* τ, while other prefixes are called *visible*. Bound and free names for prefixes are given by: $\mathsf{bn}([\varphi]\tau) = \emptyset$ and $\mathsf{bn}(\alpha(x)) = \mathsf{bn}(\overline{\alpha}(x)) = \{x\}$, $\mathsf{fn}([\varphi]\tau) = \mathsf{n}(\varphi)$, $\mathsf{fn}(\alpha(x)) = \mathsf{fn}(\overline{\alpha}(x)) = \mathsf{n}(\alpha)$ with $\mathsf{n}(a) = \mathsf{n}(\{a\}) = \{a\}$.

In a sum process, we let I range over a finite set of integers. $\mathbf{0}$ is the inactive process, defined as the empty sum. We use S to range over sum processes of the form $\sum_{i\in I} \pi_i.P_i$, and write $\pi.P \in S$ if $\pi.P$ is a summand of S. We sometimes decompose sum processes using the binary sum operator, writing, e.g., $S_1 + S_2$ (in particular, $S + \mathbf{0} = S$). We abbreviate $\pi.\mathbf{0}$ as π, and write $\alpha(x).P$ simply as $\alpha.P$ when the transmitted name is not relevant, and similarly for $\overline{\alpha}$. In $(\nu a)P$, (νa) binds a in P, and prefixes $\alpha(x)$ and $\overline{\alpha}(x)$ bind x in the continuation process. The set of free names of P, $\mathsf{fn}(P)$, is defined in the usual way. $P\{b/a\}$ is the process obtained by substituting a with b in P, in a capture-avoiding way.

We use an overloaded notation, and define processes representing conditions:

$$a \curlyvee b \triangleq (\nu u)(u/a \mid u/b) \qquad\qquad a \prec b \triangleq b/a \ . \tag{1}$$

Below, Γ ranges over sets of conditions. We define $\Gamma \vdash \varphi$, meaning that Γ implies φ, and $P \rhd \Gamma$, meaning that P entails φ for all $\varphi \in \Gamma$:

$$\textsc{$\vdash$-refl}\ \frac{}{\Gamma \vdash a \prec a} \qquad \textsc{$\vdash$-in}\ \frac{\varphi \in \Gamma}{\Gamma \vdash \varphi} \qquad \textsc{$\vdash$-mirror}\ \frac{\Gamma \vdash b \curlyvee a}{\Gamma \vdash a \curlyvee b} \qquad \textsc{$\vdash$-trans}\ \frac{\Gamma \vdash a \prec b \quad \Gamma \vdash b \prec c}{\Gamma \vdash a \prec c} \qquad \textsc{$\vdash$-join}\ \frac{\Gamma \vdash a \prec b \quad \Gamma \vdash c \prec b}{\Gamma \vdash a \curlyvee c} \qquad \textsc{$\vdash$-extjoin}\ \frac{\Gamma \vdash a \prec b \quad \Gamma \vdash b \curlyvee c}{\Gamma \vdash a \curlyvee c}$$

$$\textsc{$\rhd$-arc}\ \frac{}{b/a \rhd a \prec b} \qquad \textsc{$\rhd$-combine}\ \frac{P \rhd \Gamma \quad \Gamma \vdash \varphi}{P \rhd \varphi} \qquad \textsc{$\rhd$-par-l}\ \frac{P \rhd \varphi}{P \mid Q \rhd \varphi} \qquad \textsc{$\rhd$-par-r}\ \frac{Q \rhd \varphi}{P \mid Q \rhd \varphi} \qquad \textsc{$\rhd$-res}\ \frac{P \rhd \varphi \quad a \notin \mathsf{n}(\varphi)}{(\nu a)P \rhd \varphi}$$

As an example, the reader might check that $(\nu u)(u/a \mid u/b) \mid b/c \rhd a \curlyvee c$. Note how $a \prec u$ and $b \prec u$ entail $a \curlyvee b$, which then extends outside the scope of (νu). We define $\Phi(P) = \{\varphi \mid P \rhd \varphi\}$, the set of all conditions entailed by P.

2.2. Reduction Semantics and Barbed Congruence

Definition 1. *Structural congruence, written $\equiv$, is the smallest congruence satisfying α-conversion for restriction, and the following axioms:*

$$P \mid \mathbf{0} \equiv P \qquad (P \mid Q) \mid R \equiv P \mid (Q \mid R) \qquad P \mid Q \equiv Q \mid P$$

$$(\nu a)\mathbf{0} \equiv \mathbf{0} \qquad (\nu c)(\nu d)P \equiv (\nu d)(\nu c)P \qquad (\nu a)(P \mid Q) \equiv (\nu a)P \mid Q \ \textit{if } a \notin \mathsf{fn}(Q)$$

$$\textstyle\sum_{i\in I}\pi_i.P_i \equiv \sum_{i\in I}\pi_{\sigma(i)}.P_{\sigma(i)} \quad \sigma \textit{ a permutation of } I$$

Relations $\equiv$ and $\rhd$ are used to define the reduction of processes. We rely on $\rhd$ and on Definition 2 to infer interactions of processes on extended names. This let us introduce reduction-closed barbed congruence, along the lines of [13].

Definition 2 (Conditions on extended names). *Notations $\alpha \curlyvee \beta$ and $\alpha \prec \beta$, for conditions relating extended names, are introduced as follows:*

$\curlyvee$	b	$\{b\}$
a	$a \curlyvee b$	$a \prec b$
$\{a\}$	$b \prec a$	*undefined*

$\prec$	b	$\{b\}$
a	$a \prec b$	$a \curlyvee b$
$\{a\}$	*undefined*	$b \prec a$

Undefined conditions cannot be entailed, e.g., $P \not\rhd \{a\}\curlyvee\{a\}$ and $P \not\rhd \{a\}\prec a$.

Definition 3 (Reduction). *Relation $\mapsto$ is defined by the following rules:*

$$\frac{R \rhd \alpha \curlyvee \beta \qquad x \neq y}{R \mid \overline{\alpha}(x).P + S_1 \mid \beta(y).Q + S_2 \ \mapsto\ R \mid (\nu xy)(x/y \mid P \mid Q)} \qquad \frac{P \equiv\ \mapsto\ \equiv P'}{P \mapsto P'}$$

$$\frac{R \rhd \varphi}{R \mid [\varphi]\tau.P + S \ \mapsto\ R \mid P} \qquad \frac{P \mapsto P'}{P \mid R \mapsto P' \mid R} \qquad \frac{P \mapsto P'}{(\nu a)P \mapsto (\nu a)P'}$$

Definition 4 (Barbs). *We write $P \downarrow_{\overline{a}}$ if $P \mid a(x).\omega \mapsto P'$, where P' is a process in which ω is unguarded, and ω is a special name that does not appear in P. We define similarly the barb $\downarrow_a$, using the tester $\overline{a}(x).\omega$.*

We can remark that $P_0 \downarrow_{\overline{c}}$, where $P_0 = (\nu a)(\overline{a}(x).\mathbf{0} \mid a/c)$ as in Section 1.

Definition 5. *Barbed congruence, $\simeq$, is the largest congruence that satisfies:*

- *if $P \downarrow_a$ and $P \simeq Q$ then $Q \downarrow_a$, and similarly for $\downarrow_{\overline{a}}$, and*
- *if $P \mapsto P'$ and $P \simeq Q$ then for some Q', $Q \mapsto Q'$ and $P' \simeq Q'$.*

The remainder of the paper is devoted to the presentation of two characterisations of $\simeq$. We first comment on the definition of $\pi\mathtt{P}$ given above.

One could consider an alternative version of reduction, called "eager", whereby arcs can rewrite prefixes in one step of computation, yielding, e.g., $d/c \mid c(x).P \mapsto d/c \mid d(x).P$. It appears in [13] that the present semantics is more compelling (e.g. $a(x).a(y)$ would not be equivalent to $a(x) \mid a(y)$ in the eager version).

Remark 1 (Encodability of free and protected prefixes). *In $\pi\mathtt{P}$, arcs act like "instantaneous forwarders". This allows us to define an encoding $[\cdot]_f$ from a calculus with free prefixes to a calculus with bound prefixes as follows (x is chosen fresh and $[\cdot]_f$ preserves other operators of the calculi.):*

$$[ab.P]_f \triangleq a(x).([P]_f \mid x/b) \qquad [\overline{a}b.P]_f \triangleq \overline{a}(x).([P]_f \mid b/x)\ .$$

Arcs x/b and b/x are installed in opposite directions, which reflects the asymmetry of $\pi\mathtt{P}$[2]. We return to this encoding below (Proposition 2), and show that it allows us to preserve and reflect behavioural equivalence in [13] into our calculus. We can also encode protected prefixes as follows (u is chosen fresh):

$$[\{a\}(x).P]_p \triangleq (\nu u)(u/a \mid u(x).[P]_p) \qquad [\{\overline{a}\}(x).P]_p \triangleq (\nu u)(u/a \mid \overline{u}(x).[P]_p)\ .$$

Although protected prefixes are in this sense redundant, we do not treat them as derived operators, to simplify the presentation (in particular in Section 4).

3. A Labelled Transition System for $\pi\mathtt{P}$

3.1. LTS and Bisimilarity

The LTS defines transitions $P \xrightarrow{\mu} P'$, where the grammar for the labels, μ, is the same as the one for prefixes π. We comment on the rules, given in Table 1.

The first two rules correspond to the firing of visible prefixes. The transition involves a fresh name x, upon which the participants in a communication "agree". Name y remains local, via the installation of an arc, according to the directionality of the prefix. (Adopting a rule with no arc installation would yield a more complex definition of $\sim$). The rule for the $[\varphi]\tau$ prefix is self explanatory.

[2]The encoding of the output prefix is the same as for πI in [22], with arcs replacing "links".

$$\overset{\textsc{$\rightarrowtail$-in}}{\frac{x \notin \mathsf{n}(\alpha) \cup \{y\} \cup \mathsf{fn}(P)}{\alpha(y).P \xrightarrow{\alpha(x)} (\nu y)(x/y \mid P)}} \qquad \overset{\textsc{$\rightarrowtail$-out}}{\frac{x \notin \mathsf{n}(\alpha) \cup \{y\} \cup \mathsf{fn}(P)}{\overline{\alpha}(y).P \xrightarrow{\overline{\alpha}(x)} (\nu y)(y/x \mid P)}} \qquad \overset{\textsc{$\rightarrowtail$-tau}}{\frac{}{[\varphi]\tau.P \xrightarrow{[\varphi]\tau} P}}$$

$$\overset{\textsc{$\rightarrowtail$-comm-l}}{\frac{P \xrightarrow{\overline{\alpha}(x)} P' \qquad Q \xrightarrow{\beta(x)} Q'}{P \mid Q \xrightarrow{[\alpha \curlyvee \beta]\tau} (\nu x)(P' \mid Q')}} \qquad \overset{\textsc{$\rightarrowtail$-tau-$\rhd$}}{\frac{P \xrightarrow{[\varphi_2]\tau} P' \qquad P \rhd \Gamma \qquad \Gamma, \varphi_1 \vdash \varphi_2}{P \xrightarrow{[\varphi_1]\tau} P'}}$$

$$\overset{\textsc{$\rightarrowtail$-in-$\rhd$}}{\frac{P \xrightarrow{\alpha(x)} P' \qquad P \rhd \alpha \prec \beta}{P \xrightarrow{\beta(x)} P'}} \qquad \overset{\textsc{$\rightarrowtail$-out-$\rhd$}}{\frac{P \xrightarrow{\overline{\alpha}(x)} P' \qquad P \rhd \alpha \prec \beta}{P \xrightarrow{\overline{\beta}(x)} P'}}$$

$$\overset{\textsc{$\rightarrowtail$-res}}{\frac{P \xrightarrow{\mu} P' \qquad a \notin \mathsf{fn}(\mu)}{(\nu a)P \xrightarrow{\mu} (\nu a)P'}} \qquad \overset{\textsc{$\rightarrowtail$-par-l}}{\frac{P \xrightarrow{\mu} P' \qquad \mathsf{bn}(\mu) \cap \mathsf{fn}(Q) = \emptyset}{P \mid Q \xrightarrow{\mu} P' \mid Q}} \qquad \overset{\textsc{$\rightarrowtail$-sum}}{\frac{\pi_i.P_i \xrightarrow{\mu} P'}{\sum_i \pi_i.P_i \xrightarrow{\mu} P'}}$$

Table 1: LTS for πP. Symmetric versions of the two rules involving $\mid$ are omitted.

The rule describing communication follows the lines of the corresponding rule for $\mapsto$; no arc is installed (but arcs are introduced in the prefix rules).

The three rules mentioning $\rhd$ are called *preorder rules*. The two preorder rules for visible actions exploit $\prec$ defined on extended names in Definition 2. Note that the condition involving $\rhd$ is *the same* in these two rules. For instance, if $P \xrightarrow{a(x)} P'$ and $P \rhd a \prec c$, we can derive $P \xrightarrow{c(x)} P'$. In this case, $P \rhd a \prec \{c\}$ is also derivable, hence $P \xrightarrow{\{c\}(x)} P'$. We can also check that P_0 from Section 1 satisfies $P_0 \xrightarrow{\overline{\{c\}}(y)}$, because $\overline{a}(x).\mathbf{0} \mid a/c \rhd c \curlyvee a$ (and hence $a \prec \{c\}$).

The other preorder rule can be used to modify conditional τs involved in a transition. As an example, let $P_1 \triangleq (\overline{a}(x).Q \mid n/u) \;\mid\; (u(y).R \mid n/a)$. Process P_1 can perform a τ transition: the two arcs can, intuitively, let the output at a and the input at u interact at name n. Technically, this can be derived by inferring a $\xrightarrow{[a \curlyvee u]\tau}$ transition (from the output on the left and the input on the right), which can then be turned into a τ transition, exploiting the fact that *the whole process* entails $a \curlyvee u$. Since arcs outside of the process could also contribute to this entailment, we add $a \curlyvee u$ to the label, rather than using a side condition.

Finally, the congruence rules, on the last line of Table 1, are as expected.

Definition 6 ($\sim$). *A symmetric relation $\mathcal{R}$ is a bisimulation if $P \; \mathcal{R} \; Q$ implies:*

- *If $P \rhd \varphi$ then $Q \rhd \varphi$.*
- *If $P \xrightarrow{\alpha(x)} P'$, with $x \notin \mathsf{fn}(Q)$, then there is Q' such that $Q \xrightarrow{\alpha(x)} Q'$ and $P' \; \mathcal{R} \; Q'$; we impose the same condition with $\overline{\alpha}$ instead of α.*

- *If* $P \xrightarrow{[\varphi]\tau} P'$ *then there is* Q' *such that* $Q \xrightarrow{[\varphi]\tau} Q'$ *and* $P' \mid \varphi \mathrel{\mathcal{R}} Q' \mid \varphi$.

Bisimilarity, written $\sim$*, is the greatest bisimulation.*

This definition can be related to the efficient bisimulation from [25]. In the last clause for $[\varphi]\tau$, we add φ in parallel; indeed, demanding $P' \mathrel{\mathcal{R}} Q'$ is too strong a requirement: arcs are permanent processes, so a context triggering the corresponding reduction necessarily entails φ (even after reductions).

Remark 2. *Our LTS does not have rules for opening and closing the scope of a restriction. Instead, we rely on arcs in* $\pi\mathtt{P}$ *to handle scope extrusion. To see this, consider the following* $\pi\mathtt{P}$ *transition where a private name* c *is emitted:*

$$\overline{a}(c).P \xrightarrow{\overline{a}(x)} (\nu c)(c/x \mid P) \ .$$

Name x *is visible in the label, and arc* c/x *is installed. Through* x*, the environment can affect* c*, so that* $\pi\mathtt{P}$ *actually* implements *scope extrusion via arcs, without the need to move restrictions. We have:*

$$\begin{aligned}\overline{a}(c).P \mid a(y).Q &\xrightarrow{\tau} (\nu x)((\nu c)(c/x \mid P) \mid (\nu y)(x/y \mid Q)) \\ &\simeq\ (\nu c)(\nu y)(P \mid c/y \mid Q) \ .\end{aligned}$$

In particular, in the encoding from the π*-calculus to* $\pi\mathtt{P}$ *given in [13], when a fresh name is emitted, it is extruded, and we have (*$[\![P]\!]$ *is the encoding of* P*):*

$$[\![a(y).Q \mid (\nu b)\overline{a}b.P]\!] \xrightarrow{\tau}\sim [\![(\nu b)(Q\{b/y\} \mid P)]\!] \ .$$

3.2. Towards the Characterisation Theorem

In order to present the proof that barbed congruence coincides with the bisimilarity induced by our LTS, we first need to establish several preliminary technical results, about relations $\vdash$, $\rhd$ and $\sqsubseteq$.

Mechanisation of some proofs. We have carefully separated proofs involving tedious case enumerations, which we have verified using the Coq proof system [8], from other proofs involving features that are harder to formalise, like binding mechanisms. Case analyses being both easy to deal with in Coq and error-prone in manual proofs, this provides us with a good trade-off between trust and bookkeeping proofs. The proof script files we mention throughout this paper are available from [18].

3.2.1. On the $\vdash$ relation.

To handle some case analyses, we decompose $\vdash$, using an additional relation, written $\Vdash$. The idea is that the transitive closure of $\Vdash$ is $\vdash$.

The predicate $\varphi_1, \varphi_2 \Vdash \varphi_3$ is defined as follows (note how the second line is the symmetric variant of the first):

$$\frac{}{a \prec b, b \prec c \Vdash a \prec c} \qquad \frac{}{a \prec c, b \prec c \Vdash a \curlyvee b} \qquad \frac{}{a \curlyvee b, c \prec a \Vdash c \curlyvee b} \qquad \frac{}{a \curlyvee b, c \prec b \Vdash a \curlyvee c}$$

$$\frac{}{b \prec c, a \prec b \Vdash a \prec c} \qquad \frac{}{b \prec c, a \prec c \Vdash a \curlyvee b} \qquad \frac{}{c \prec a, a \curlyvee b \Vdash c \curlyvee b} \qquad \frac{}{c \prec b, a \curlyvee b \Vdash a \curlyvee c}$$

We write $\varphi_1 \Vdash_{\varphi_2} \varphi_3$ whenever $\varphi_1, \varphi_2 \Vdash \varphi_3$ and we write $\Vdash_P$ for the union of all $\Vdash_\varphi$ for $P \rhd \varphi$. Intuitively, we use $\Vdash_P$ to decompose in smaller steps the implications between conditions that hold because of conditions entailed by P.

Lemma 1 shows that if φ can be decomposed, then so can $\Vdash_\varphi$.

Lemma 1. *If $\varphi_1, \varphi_2 \Vdash \varphi$ then $\Vdash_\varphi \subseteq (\Vdash_{\varphi_1}\Vdash_{\varphi_2} \cup \Vdash_{\varphi_2}\Vdash_{\varphi_1})$.*

Proof. By case analysis on the $\Vdash$ predicate (see proof script in [18].) $\square$

Relations $\vdash$ and $\Vdash$ relate the same conditions:

Lemma 2. *If $\varphi_1, \varphi_2 \Vdash \varphi_3$ then $\varphi_1, \varphi_2 \vdash \varphi_3$. Conversely, if $\Gamma \vdash \varphi$ then $\psi \Vdash_{\psi_1} \ldots \Vdash_{\psi_n} \varphi$ for some reflexive ψ and for some $\psi_1, \ldots, \psi_n \in \Gamma$.*

Proof. The first part is trivial: each rule for $\Vdash$ is simulated by (three) rules for $\vdash$. The second part follows from the fact that $(\Vdash_\varphi) \subseteq (\Vdash_{\psi_1} \ldots \Vdash_{\psi_n}\Vdash_\zeta)$ for some $\psi_1, \ldots, \psi_n \in \Gamma$ and some reflexive ζ, which we prove by induction on $\Gamma \vdash \varphi$ (Lemma 1 covers the rules $\vdash$-trans, $\vdash$-join, $\vdash$-extjoin). $\square$

Relation $\Vdash_P$ is semi-transitive (it can always be reduced to two iterations):

Lemma 3. *$(\Vdash_P)^* \subseteq (\Vdash_P\Vdash_P)$.*

Proof (Sketch). Suppose $\varphi_1 \Vdash_P \ldots \Vdash_P \varphi_3$. One obtains φ_3 from φ_1 by appending ψs such that $P \rhd \psi$ at the left or at the right of φ_1. We can in fact append all left ψs at first (an operation that can be done in one $\Vdash_P$ step), and only then, append all right ψs (the other $\Vdash_P$ step). (See proof script in [18].) $\square$

The proof can be adapted to refine Lemma 3 with a better control on names:

Lemma 4. *Suppose $\varphi_1 \Vdash_P^* \varphi_4$ and $\mathsf{n}(\varphi_i) = \{a_i, b_i\}$. There are φ_2, φ_3 such that $\varphi_1 \Vdash_P \varphi_2 \Vdash_P \varphi_4$, $\varphi_1 \Vdash_P \varphi_3 \Vdash_P \varphi_4$, with $\mathsf{n}(\varphi_2) = \{a_1, b_4\}$ and $\mathsf{n}(\varphi_3) = \{a_4, b_1\}$.*

Relation $\Vdash_{P|Q}$ can be decomposed into $\Vdash_P$ and $\Vdash_Q$.

Lemma 5. *$\Vdash_{P|Q} \subseteq (\Vdash_P \cup \Vdash_Q)^*$.*

Proof. We prove by induction on $P \mid Q \rhd \varphi_2$ that for all φ_1 and φ_3, if $\varphi_1, \varphi_2 \Vdash \varphi_3$ then $\varphi_1 (\Vdash_P \cup \Vdash_Q)^* \varphi_3$. There are three cases. For $\rhd$-par-l we know that $P \rhd \varphi_2$ so $\varphi_1 \Vdash_P \varphi_3$ and for $\rhd$-par-r we get the same way $\varphi_1 \Vdash_Q \varphi_3$. The case for the rule $\rhd$-combine is handled using Lemma 2. $\square$

3.2.2. On the $\rhd$ relation.

Lemmas 6-8 are proved using simple inductions on predicate $\rhd$. A context C is a process with one hole $[\cdot]$; we write $C[P]$ for C with $[\cdot]$ replaced with P.

Lemma 6. *If $P \rhd \varphi$ and $\mathsf{n}(\varphi) \setminus \mathsf{fn}(P) \neq \emptyset$, then φ is reflexive.*

Lemma 7. *If $\Phi(P) \subseteq \Phi(Q)$ then $\Phi(C[P]) \subseteq \Phi(C[Q])$.*

Lemma 8. *If $P \rhd \varphi$ then $\Phi(P \mid \varphi) = \Phi(P)$.*

Lemma 9. *If $P \equiv Q$ then $\Phi(P) = \Phi(Q)$.*

PROOF. By induction on the derivation of $P \equiv Q$. In most cases we show $\Phi(P) \subseteq \Phi(Q)$ and $\Phi(Q) \subseteq \Phi(P)$ separately, and by induction on $\rhd$ (we assume the last rule is not $\rhd$-COMBINE, since this case is immediate).

We focus on the following cases involving restrictions:

1. $(\nu a)P \equiv (\nu b)P\{b/a\}$ when $b \notin \mathsf{fn}(P)$: the last rule, and the one we apply, is $\rhd$-RES and we only need to prove $P \rhd \varphi \;\Rightarrow\; P\{b/a\} \rhd \varphi$ when $a, b \notin \mathsf{n}(\varphi)$. More generally $P \rhd \varphi \Rightarrow P\sigma \rhd \varphi\sigma$ for arbitrary substitutions σ.
2. $\nu a(P \mid Q) \equiv (\nu a P) \mid Q$ with $a \notin \mathsf{fn}(Q)$. Intuitively, we need to put together all contributions from P so that they are independent of Q. First, as before, from $\nu a(P \mid Q) \rhd \varphi$ we get $P \mid Q \rhd \varphi$.
   We treat the case where $\varphi = b_0 \curlyvee c_0$ (the other cases are simpler). The derivation of $P \mid Q \rhd \varphi$ is built using several $\rhd$-COMBINEs involving processes P and Q, which we summarise as follows: there exist $b_0, \ldots, b_n$ and $c_0, \ldots, c_m$ such that for each $\zeta \in \{b_n \curlyvee c_m, b_0 \prec b_1, \ldots, b_{n-1} \prec b_n, c_0 \prec c_1, \ldots, c_{m-1} \prec c_m\}$, we have either $P \rhd \zeta$ or $Q \rhd \zeta$. We eliminate occurrences of a in the b_is and the c_js. For this, we proceed as follows:
   (a) when $a \in \mathsf{n}(\zeta)$ we replace $Q \rhd \zeta$ with $P \rhd \zeta$ (using Lemma 6 as $a \notin \mathsf{fn}(Q)$).
   (b) Hence, if $b_i = a$ then $P \rhd b_{i-1} \prec a$ and $P \rhd a \prec b_{i+1}$, so using $\rhd$-COMBINE we replace them with $P \rhd b_{i-1} \prec b_{i+1}$.
   But if $i = n$, since $P \rhd b_n \curlyvee c_m$ and $a = b_n$, we deduce $P \rhd b_{n-1} \curlyvee c_m$ (the case $i = 0$ is not possible since $a \notin \mathsf{n}(\phi)$).
   (c) We continue as long as a appears in any of the b_i or c_j.

   We are left with a set of formulas ζ (entailing φ) such that $a \notin \mathsf{n}(\zeta)$ and either $P \rhd \zeta$ or $Q \rhd \zeta$, and hence $(\nu a)P \mid Q \rhd \zeta$. From this we get $(\nu a)P \mid Q \rhd \varphi$ by applying rule $\rhd$-COMBINE $n + m$ times. $\square$

3.2.3. On the $\sqsubseteq$ relation.

Definition 7. *We define a relation $\sqsubseteq^{\varphi}$ between labels as follows: (i) $\alpha_1(x) \sqsubseteq^{\varphi} \alpha_2(x)$ and $\overline{\alpha}_1(x) \sqsubseteq^{\varphi} \overline{\alpha}_2(x)$ when $\varphi = \alpha_2 \prec \alpha_1$, and (ii) $[\varphi_1]\tau \sqsubseteq^{\varphi} [\varphi_2]\tau$ when $\varphi_1, \varphi \vdash \varphi_2$. We write $\sqsubseteq_P$ for the smallest preorder containing all $\sqsubseteq^{\varphi}$ when $P \rhd \varphi$.*

Intuitively, $\eta \sqsubseteq_P \mu$ means that label μ is less general than η, given some condition (φ above) enforced by P. For instance, we have $\{a\}(x) \sqsubseteq_{\mathbf{0}} a(x)$. This notion is used in Lemma 17 to reason about transitions of processes.

Similarly as $\Vdash_{\varphi}$, when φ can be decomposed, so can $\sqsubseteq^{\varphi}$:

Lemma 10. *If $\varphi_1, \varphi_2 \vdash \varphi$ then $(\sqsubseteq^{\varphi}) \subseteq (\sqsubseteq^{\varphi_1} \sqsubseteq^{\varphi_2}) \cup (\sqsubseteq^{\varphi_2} \sqsubseteq^{\varphi_1})$.*

PROOF. We suppose $\mu \sqsubseteq^{\varphi} \eta$. If μ is a $[\varphi]\tau$, Lemma 1 is enough to conclude. Otherwise, we make an exhaustive case analysis on $\varphi_1, \varphi_2 \vdash \varphi$ and $\mu \sqsubseteq^{\varphi} \eta$, for instance: if $a \prec c, b \prec c \vdash a \curlyvee b$ and $\{a\}(x) \sqsubseteq^{a \curlyvee b} b(x)$ then $\{a\}(x) \sqsubseteq^{a \prec c} c(x)$ and $c(x) \sqsubseteq^{b \prec c} b(x)$. (The complete case analysis appears in the proof script [18].) $\square$

On extended names, $\prec$ is transitive and $\curlyvee$ can be extended on the left (or on the right), but cannot in general be derived from $\prec$, whereas it is the case for regular names. From this lemma, we get tools to manipulate the $\sqsubseteq_P$ relation.

Lemma 11 (Composing relations $\prec$ and $\curlyvee$). 1. *If $\alpha \prec \beta$ and $\beta \prec \gamma$ are defined, then so is $\alpha \prec \gamma$ and $\alpha \prec \beta, \beta \prec \gamma \vdash \alpha \prec \gamma$.*

2. *If $\alpha \prec \beta$ and $\beta \curlyvee \gamma$ are defined, then so is $\alpha \curlyvee \gamma$ and $\alpha \prec \beta, \beta \curlyvee \gamma \vdash \alpha \curlyvee \gamma$.*
3. *In general $\alpha \prec \beta$ and $\gamma \prec \beta$ do not imply $\alpha \curlyvee \gamma$.*

Proof. The first two items are trivial after a case analysis on α, β, γ, we give an example for each. For (1) $(\alpha, \beta, \gamma) = (a, \{b\}, \{c\})$ we need $a \curlyvee b, c \prec b \vdash a \curlyvee c$ (which holds by right extension of $\curlyvee$). For (2) $(\alpha, \beta, \gamma) = (\{a\}, \{b\}, c)$ we need $b \prec a, c \prec b \vdash c \prec a$ (which holds by transitivity of $\prec$). Finally a counterexample for (3) is for $\alpha = a$, $\beta = \{b\}$ and $\gamma = c$, since $\alpha \prec \beta = a \curlyvee b$ and $\gamma \prec \beta = c \curlyvee b$ do not entail $\alpha \curlyvee \gamma = a \curlyvee c$ ($\curlyvee$ is not transitive). (See proof script [18].) □

Corollary 1. *If $\alpha(x) \sqsubseteq_P \alpha_1(x)$ and $\alpha \curlyvee \beta$ is defined, then $\alpha_1 \curlyvee \beta$ is defined and $[\alpha \curlyvee \beta]\tau \sqsubseteq_P [\alpha_1 \curlyvee \beta]\tau$.*

Lemma 12. *$\sqsubseteq_{P|Q} \subseteq (\sqsubseteq_P \cup \sqsubseteq_Q)^*$.*

Proof. For visible prefixes, we conclude by transitivity of $\prec$ on extended names (Lemma 11). For conditional τ's, this is a consequence of Lemma 5.

Lemma 13. *If $\mu \sqsubseteq_{P|Q} \mu'$ and $a \notin \mathsf{fn}(\mu), \mathsf{fn}(\mu'), \mathsf{fn}(Q)$ then $\mu \sqsubseteq_{(\nu a P)|Q} \mu'$.*

Proof. We use Lemma 12 to obtain a sequence $\mu_0, \mu_1, \ldots, \mu_n$ with $\mu = \mu_0$ and $\mu' = \mu_n$ and $\mu_i \sqsubseteq^{\varphi_i} \mu_{i+1}$ with for each i, $P \rhd \varphi_i$ or $Q \rhd \varphi_i$. W.l.o.g. we can suppose when $a \in \mathsf{n}(\varphi_i)$ that $P \rhd \varphi_i$ (indeed, if instead we had $Q \rhd \varphi_i$, then φ_i would be trivial and thus $P \rhd \varphi_i$).

We then put together all subsequences $\varphi_i, \ldots, \varphi_j$ entailed by P, and do the same with Q, to obtain after some reindexing: $\mu_{j_1} \sqsubseteq_P \mu_{j_2} \sqsubseteq_Q \mu_{j_3} \sqsubseteq_P \ldots$ so that for all k, $a \notin \mathsf{fn}(\mu_{j_k})$. We can then write $\mu_{j_1} \sqsubseteq_{\nu a P} \mu_{j_2} \sqsubseteq_Q \mu_{j_3} \sqsubseteq_{\nu a P} \ldots$, and we are able to conclude that $\mu \sqsubseteq_{(\nu a P)|Q} \mu$. □

Lemma 14. *If $\mu \sqsubseteq_{P|Q} \mu'$ and $a \notin \mathsf{fn}(\mu), \mathsf{fn}(Q)$ then for some λ such that $a \notin \mathsf{fn}(\lambda)$, $\mu \sqsubseteq_{(\nu a P)|Q} \lambda \sqsubseteq_P \mu'$.*

Proof. As in the proof of Lemma 13, we obtain $\mu = \mu_{j_1} \sqsubseteq_P \mu_{j_2} \sqsubseteq_Q \cdots \sqsubseteq_P \mu_{j_n} = \mu'$, except that a may appear in $\mu' = \mu_{j_n}$, but not in any other $\mu = \mu_{j_k}$ if $k < n$. We choose $\lambda = \mu_{j_{n-1}}$, the rest of the proof is similar. □

Lemma 15. *If $P \xrightarrow{\mu} P'$ and $\eta \sqsubseteq_P \mu$ then $P \xrightarrow{\eta} P'$. Conversely, whenever $P \xrightarrow{\eta} P'$, there exists μ such that $\eta \sqsubseteq_P \mu$ and $P \xrightarrow{\mu} P'$, of which there is a proof, not bigger than the one for $P \xrightarrow{\eta} P'$, that does not end with a preorder rule.*

Proof. Both directions are proved by simple inductions. □

3.3. The Characterisation Theorem

We can now present the proof that barbed congruence and the bisimilarity induced by our LTS coincide, Theorem 1 below. We first present some auxiliary lemmas.

3.3.1. Results about transitions.

Lemma 16. *If $(\nu a)P \xrightarrow{\mu} P_1$ then $P_1 = (\nu a)P'$ for some P' such that $P \xrightarrow{\mu} P'$.*

Proof. Using Lemma 15 we know that $(\nu a)P \xrightarrow{\mu_1} P_1$ for some μ_1 such that $\mu \sqsubseteq_{\nu aP} \mu_1$, and that this transition is coming from $P \xrightarrow{\mu_1} P'$ with $P_1 = (\nu a)P'$. Since $\Phi(\nu aP) \subseteq \Phi(P)$, we know $\mu \sqsubseteq_P \mu_1$, so we can derive $P \xrightarrow{\mu} P'$. $\square$

The proof of the next lemma, stating that transitions commute with $\equiv$, follows from an analysis similar to the proof of Lemma 9.

Notation 1. *We adopt the following notation in writing derivations, to denote either an application of the first part of Lemma 15 (i.e. an application of several preorder rules), or a recursive case analysis on the rules deriving $P \xrightarrow{\mu} P'$ until a non-preorder rule is reached, by application of the second part of Lemma 15.*

$$\begin{array}{c} P \xrightarrow{\mu'} P' \\ \hline [\mu \sqsubseteq_P \mu'] \\ \hline P \xrightarrow{\mu} P' \end{array}$$

Lemma 17. *If $P \equiv Q$ and $P \xrightarrow{\mu} P'$ then $Q \xrightarrow{\mu}\equiv P'$.*

Proof. More generally, we prove by induction on the derivation of $P \equiv Q$ that for all μ, $((P \xrightarrow{\mu} P' \Rightarrow Q \xrightarrow{\mu}\equiv P')$ and $(Q \xrightarrow{\mu} Q' \Rightarrow P \xrightarrow{\mu}\equiv Q'))$. Thanks to this formulation, precongruence is handled by transition induction and equivalence properties are trivial. We give the remaining most complicated cases, one for associativity and one for extrusion. Then we perform an induction on $P \xrightarrow{\mu} P'$. We factor out the cases corresponding to preorder rules using Lemmas 15 and 9, and so we assume that the last rule is not a preorder rule.

1. $(P \mid Q) \mid R \equiv P \mid (Q \mid R)$: we assume the last rule is $\rightarrow$-PAR-L. We use Lemma 15 to get $\mu \sqsubseteq_{P|Q} \mu_1$, then a $\rightarrow$-PAR-R rule to get $Q \xrightarrow{\mu_1} Q'$. Getting back $P \mid (Q \mid R) \xrightarrow{\mu_1} P \mid (Q' \mid R)$ is easy, what is more tricky is to get right the label μ: for that we use again Lemma 15 thanks to the fact that $\mu \sqsubseteq_{P|(Q|R)} \mu_1$, in turn implied by $\mu \sqsubseteq_{P|Q} \mu_1$ (since $\Phi(P \mid Q) \subseteq \Phi((P \mid Q) \mid R) = \Phi(P \mid (Q \mid R))$ by rule $\triangleright$-PAR-L and Lemma 9).

$$\begin{array}{rc} & Q \xrightarrow{\mu_1} Q' \\ \text{PAR-R} & \hline P \mid Q \xrightarrow{\mu_1} P \mid Q' \\ & \hline [\mu \sqsubseteq_{P|Q} \mu_1] \\ & \hline P \mid Q \xrightarrow{\mu} P \mid Q' \\ \text{PAR-L} & \hline (P \mid Q) \mid R \xrightarrow{\mu} (P \mid Q') \mid R \end{array} \quad \rightsquigarrow \quad \begin{array}{cl} Q \xrightarrow{\mu_1} Q' & \\ \hline Q \mid R \xrightarrow{\mu_1} Q' \mid R & \text{PAR-L} \\ \hline P \mid (Q \mid R) \xrightarrow{\mu_1} P \mid (Q' \mid R) & \text{PAR-R} \\ \hline [\mu \sqsubseteq_{P|(Q|R)} \mu_1] & \\ \hline P \mid (Q \mid R) \xrightarrow{\mu} P \mid (Q' \mid R) & \end{array} .$$

2. $\nu a(P \mid Q) \equiv (\nu a P) \mid Q$: any transition from $\nu a(P \mid Q)$ must come from one from $P \mid Q$. After an application of Lemma 15 we have $\mu \sqsubseteq_{P|Q} \mu_1$ with a transition μ_1 that we can break into transitions from P and/or Q, depending on the last rule: we treat the case for $\rightarrow$-PAR-L and the case of $\rightarrow$-COM-* (the case for $\rightarrow$-PAR-R being simpler) and use Lemma 15 multiple times.
   (a) $\rightarrow$-PAR-L: action μ_1 (coming from P) may contain a. With Lemma 14 we obtain a transition along λ ($a \notin \mathsf{fn}(\lambda)$) to which we can apply Q's original role in the transformation of μ_1 into μ (combined to the role of $\nu a P$). The following derivations illustrate the reasoning exposed above.

$$\dfrac{\dfrac{\dfrac{\dfrac{P \xrightarrow{\mu_1} P'}{P \mid Q \xrightarrow{\mu_1} P' \mid Q}}{[\mu \sqsubseteq_{P|Q} \mu_1]}}{P \mid Q \xrightarrow{\mu} P' \mid Q} \quad a \notin \mathsf{n}(\mu)}{(\nu a)(P \mid Q) \xrightarrow{\mu} (\nu a)(P' \mid Q)} \quad \rightsquigarrow \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P \xrightarrow{\mu_1} P'}{[\lambda \sqsubseteq_P \mu_1]}}{P \xrightarrow{\lambda} P'} \quad a \notin \mathsf{n}(\lambda)}{\nu a P \xrightarrow{\lambda} \nu a P'}}{(\nu a P) \mid Q \xrightarrow{\lambda} (\nu a P') \mid Q}}{[\mu \sqsubseteq_{(\nu a P)|Q} \lambda]}}{(\nu a P) \mid Q \xrightarrow{\mu} (\nu a P') \mid Q}$$

   (b) $\rightarrow$-COM-L: ($\rightarrow$-COM-R is symmetric) transition $\mu = [\varphi]\tau$ is obtained from transitions $\overline{\alpha}(x)$ and $\gamma(x)$ such that $[\varphi]\tau \sqsubseteq_{P|Q} [\alpha \curlyvee \gamma]\tau$. Since $a \notin \mathsf{fn}(Q)$ we already know $a \neq \mathsf{n}(\gamma)$.
   Suppose first that $a \neq \mathsf{n}(\alpha)$. Then we can obtain the transition $\overline{\alpha}$ from $(\nu a)P$ as well, then the $[\alpha \curlyvee \gamma]\tau$ transition, and we conclude to get $[\varphi]\tau$ by Lemma 13.
   The interesting case is when $a = \mathsf{n}(\alpha)$. We defer to below the proof of the following implication:

$$[\varphi]\tau \sqsubseteq_{P|Q} [\alpha \curlyvee \gamma]\tau \quad \Longrightarrow \quad \exists \beta \quad a \neq \mathsf{n}(\beta) \ \wedge \ \begin{cases} [\varphi]\tau \sqsubseteq_{P|Q} [\beta \curlyvee \gamma]\tau \\ P \rhd \alpha \prec \beta \end{cases} \tag{2}$$

   and we remark that it implies $\overline{\beta}(x) \sqsubseteq_P \overline{\alpha}(x)$. We can now use Lemma 15 to easily build back the $[\varphi]\tau$ transition from $(\nu a P) \mid Q$, as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{P \xrightarrow{\overline{\alpha}(x)} P' \qquad Q \xrightarrow{\gamma(x)} Q'}{P \mid Q \xrightarrow{[\alpha \curlyvee \gamma]\tau} (\nu x)(P' \mid Q')}}{[[\varphi]\tau \sqsubseteq_{P|Q} [\alpha \curlyvee \gamma]\tau]}}{P \mid Q \xrightarrow{[\varphi]\tau} (\nu x)(P' \mid Q')} \quad a \notin \mathsf{n}(\varphi)}{(\nu a)(P \mid Q) \xrightarrow{[\varphi]\tau} (\nu a)(\nu x)(P' \mid Q')} \quad \rightsquigarrow \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P \xrightarrow{\overline{\alpha}(x)} P'}{[\overline{\beta}(x) \sqsubseteq_P \overline{\alpha}(x)]}}{P \xrightarrow{\overline{\beta}(x)} P'} \quad a \notin \mathsf{n}(\beta)}{\nu a P \xrightarrow{\overline{\beta}(x)} \nu a P'} \qquad Q \xrightarrow{\gamma(x)} Q'}{(\nu a P) \mid Q \xrightarrow{[\beta \curlyvee \gamma]\tau} (\nu x)((\nu a P') \mid Q')}}{[[\varphi]\tau \sqsubseteq_{(\nu a P)|Q} [\beta \curlyvee \gamma]\tau]}}{(\nu a P) \mid Q \xrightarrow{[\varphi]\tau} (\nu x)((\nu a P') \mid Q')}$$

   We prove now (2). Using Lemma 12 we decompose $[\varphi]\tau \sqsubseteq_{P|Q} [\alpha \curlyvee \gamma]\tau$ into a sequence of $\sqsubseteq^{\psi}$ with $P \rhd \psi \vee Q \rhd \psi$. We prove (2) by induction on this sequence, decomposing it "from the right" into $[\varphi]\tau \sqsubseteq_{P|Q} \eta \sqsubseteq^{\psi} [\alpha \curlyvee \gamma]\tau$: we assume (2) holds when $[\alpha \curlyvee \gamma]\tau$ is replaced with η and we decompose $\eta \sqsubseteq^{\psi} [\alpha \curlyvee \gamma]\tau$ into easy cases:

- $\eta = [\beta \curlyvee \gamma]\tau$ and $\psi = \alpha \prec \beta$ and $\mathsf{n}(\alpha) \neq \mathsf{n}(\beta)$:
  in this case, $P \rhd \psi$ and since $[\varphi]\tau \sqsubseteq_{P|Q} [\beta \curlyvee \gamma]\tau$ we are done.
- $\eta = [\beta \curlyvee \gamma]\tau$ and $\psi = \alpha \prec \beta$ and $\mathsf{n}(\alpha) = \mathsf{n}(\beta)$:
  in this case, $P \rhd \psi$ (even if $Q \rhd \psi$). We use the induction hypothesis to get β' such that $[\varphi]\tau \sqsubseteq_{P|Q} [\beta' \curlyvee \gamma]\tau$ and $\mathsf{n}(\beta') \neq \mathsf{n}(\beta)$ and $P \rhd \beta \prec \beta'$. This β' is enough, since $P \rhd \alpha \prec \beta'$.
- $\eta = [\alpha \curlyvee \delta]\tau$ with $\mathsf{n}(\delta) = a$:
  we interrupt the induction since we can derive a plain τ, hence any $[\varphi]\tau$ that we want.
- $\eta = [\alpha \curlyvee \delta]\tau$ and $\psi = \gamma \prec \delta$ and $P \rhd \psi$ or $Q \rhd \psi$:
  we use the induction hypothesis to get the wanted β and we replay ψ on top of it (the IH gives $[\varphi]\tau \sqsubseteq_{P|Q} [\beta \curlyvee \delta]\tau$, from which through $P, Q \rhd \gamma \prec \delta$ it is easy to get $[\varphi]\tau \sqsubseteq_{P|Q} [\beta \curlyvee \gamma]\tau$).

This concludes the case analysis. □

Lemma 18. *$\equiv$ is a bisimulation.*

PROOF. This follows from Lemmas 9 and 17. The correspondence between transitions is exact, hence the only non-trivial case is for the $[\varphi]\tau$ transition because one requires $P' \mid \varphi \equiv Q' \mid \varphi$ instead of just $P' \equiv Q'$. This holds because $\equiv$ is a congruence. □

Definition 8 (Bisimulation up to $\sim$). *A relation $\mathcal{R}$ is a bisimulation up to bisimilarity if it validates the clauses in the definition of $\sim$ (Definition 6), except that we require $P' \sim\mathcal{R}\sim Q'$ instead of $P' \mathrel{\mathcal{R}} Q'$ and $(P' \mid \varphi) \sim\mathcal{R}\sim (Q' \mid \varphi)$ instead of $(P' \mid \varphi) \mathrel{\mathcal{R}} (Q' \mid \varphi)$.*

Lemma 19. *If $\mathcal{R}$ is a bisimulation up to bisimilarity, then $\mathcal{R} \subseteq \sim$.*

PROOF. We prove $\sim\mathcal{R}\sim$ is a bisimulation. The only unusual transition μ is when $\mu = [\varphi]\tau$, but from $P \sim P_1 \mathrel{\mathcal{R}} Q_1 \sim Q$ and $P \xrightarrow{\mu} P'$ we know:

- that $P_1 \xrightarrow{\mu} P_1'$ and $(P' \mid \varphi) \sim (P_1' \mid \varphi)$ from the $\sim$ game,
- that $Q_1 \xrightarrow{\mu} Q_1'$ and $(P_1' \mid \varphi) \sim\mathcal{R}\sim (Q_1' \mid \varphi)$ from the $\mathcal{R}$ up-to game,
- that $P \xrightarrow{\mu} P'$ and $(Q_1' \mid \varphi) \sim (Q' \mid \varphi)$ from the second $\sim$ game.

We conclude by transitivity of $\sim$ since $(P' \mid \varphi) \sim\sim\mathcal{R}\sim\sim (Q' \mid \varphi)$. □

Lemma 20. *If $P \rhd \varphi$ then $P \mid \varphi \sim P$.*

PROOF. We prove that $\mathcal{R} = \{(P \mid \varphi, P) \mid P \rhd \varphi\}$ is a bisimulation up to $\equiv$ (and thus is included in $\sim$ by Lemmas 19 and 18). First, all transitions from P can be done by $P \mid \varphi$; sometimes we have to relate $P \mid \psi$ to $(P \mid \varphi) \mid \psi$ which we rewrite as $(P \mid \psi) \mid \varphi$ with $\equiv$ (and indeed $(P \mid \psi) \mid \varphi \mathrel{\mathcal{R}} (P \mid \psi)$).

Second, suppose $P \mid \varphi \xrightarrow{\mu} P_1$. Using Lemma 15 we get μ_1 such that $\mu \sqsubseteq_{P|\varphi} \mu_1$, and a proof of $P \mid \varphi \xrightarrow{\mu_1} P'$ with a non preorder rule as last rule. This rule must be a PAR rule coming from P so in fact $P_1 = P' \mid \varphi$ with $P \xrightarrow{\mu_1} P'$. Since $\Phi(P \mid \varphi) = \Phi(P)$ by Lemma 8, we know that $\mu \sqsubseteq_P \mu_1$. We can apply Lemma 15 again to deduce $P \xrightarrow{\mu} P'$, and indeed $P' \mid \varphi \mathrel{\mathcal{R}} P'$. □

Lemma 21. *$P \sim Q$ implies $(\nu a)P \sim (\nu a)Q$ for all a.*

PROOF. We prove $\mathcal{R} \triangleq \{(\nu a P, \nu a Q) \mid P \sim Q\}$ is a bisimulation up to bisimilarity (Lemma 19), which is relatively easy using Lemma 16. The only interesting case is for a conditional τ transition. If $\nu a P \xrightarrow{[\varphi]\tau} \nu a P'$ then $P \xrightarrow{[\varphi]\tau} P'$ and using $\sim$, $Q \xrightarrow{[\varphi]\tau} Q'$ with $(P' \mid \varphi) \sim (Q' \mid \varphi)$. Since we want to relate $(\nu a)P' \mid \varphi$ and $(\nu a)Q' \mid \varphi$ we $\sim$-rewrite them into $(\nu a)(P' \mid \varphi)$ and $(\nu a)(Q' \mid \varphi)$ (using Lemma 18) which are indeed related through $\mathcal{R}$. The condition about $\rhd$ is ensured by compositionality of $\rhd$ (Lemma 7). □

Definition 9 (Bisimulation up to $\sim$ and ν). *A relation $\mathcal{R}$ is a bisimulation up to restriction and bisimilarity if it validates the clauses of the usual bisimulation, except that the outcomes of the transitions, P_1 and Q_1, are requested to satisfy $P_1 \sim (\nu\tilde{a})P_2$ and $Q_1 \sim (\nu\tilde{a})Q_2$ with $P_2 \mathrel{\mathcal{R}} Q_2$, where $\tilde{a}$ stands for a (possibly empty) tuple of names.*

Lemma 22. *If $\mathcal{R}$ is a bisimulation up to restriction and bisimilarity then $\mathcal{R} \subseteq \sim$.*

PROOF. We write $\mathcal{R}^\nu$ for $\{(\nu\tilde{a})P, (\nu\tilde{a})Q) \mid P \mathrel{\mathcal{R}} Q\}$. We know that $\mathcal{R} \subseteq \mathcal{S} \triangleq {\sim}\mathcal{R}^\nu{\sim}$ so it is enough to prove that $\mathcal{S}$ is a bisimulation. Suppose $P \sim (\nu\tilde{a})P_1 \mathrel{\mathcal{R}^\nu} (\nu\tilde{a})Q_1 \sim Q$ with $P_1 \mathrel{\mathcal{R}} Q_1$. We start from a transition $P \xrightarrow{\mu} P'$, we use the bisimulation game on $\sim$ and Lemma 16 to deduce $(\nu\tilde{a})P_1 \xrightarrow{\mu} (\nu\tilde{a})P'_1$ with $P_1 \xrightarrow{\mu} P'_1$ and $(P' \mid A) \sim ((\nu\tilde{a})P'_1 \mid A)$ with $A = \varphi$ if $\mu = [\varphi]\tau$ or $A = \mathbf{0}$ otherwise. Using the bisimulation up-to on $\mathcal{R}$ we obtain $Q_1 \xrightarrow{\mu} Q'_1$ with $(P'_1 \mid A) \sim (\nu\tilde{c})P_2 \mathrel{\mathcal{R}^\nu} (\nu\tilde{c})Q_2 \sim (Q'_1 \mid A)$ for some $P_2 \mathrel{\mathcal{R}} Q_2$. We also get $(\nu\tilde{a})Q_1 \xrightarrow{\mu} (\nu\tilde{a})Q'_1$ from which using the game on the second $\sim$, $Q \xrightarrow{\mu} Q'$ with $((\nu\tilde{a})Q'_1 \mid A) \sim (Q' \mid A)$.

We now compose what we have: $(P' \mid A) \sim ((\nu\tilde{a})P'_1 \mid A)$ and $(P'_1 \mid A) \sim ((\nu\tilde{c})P_2)$. We can get from the latter $((\nu\tilde{a})P'_1 \mid A) \sim ((\nu\tilde{a}\tilde{c})P_2)$ using Lemmas 18 and 21, and then compose them using transitivity of $\sim$; similarly for Q: $(Q' \mid A) \sim ((\nu\tilde{a}\tilde{c})Q_2)$. Since $P_2 \mathrel{\mathcal{R}} Q_2$, the pair of $P' \mid A$ and $Q' \mid A$ is in $\mathcal{S}$. □

Definition 10 (Bisimulation up to $\sim$ and σ). *A relation $\mathcal{R}$ is a bisimulation up to bisimilarity and injective substitution if it validates the clauses of the usual bisimulation, except that the outcomes of the transitions, P_1 and Q_1, are requested to satisfy $P_1 \sim P_2\sigma$ and $Q_1 \sim Q_2\sigma$ with $P_2 \mathrel{\mathcal{R}} Q_2$, where σ stands for an injective name substitution.*

Proof. We prove that $\sim\mathcal{R}_\sigma\sim$ is a bisimulation where $\mathcal{R}_\sigma$ stands for $\{(P\sigma, Q\sigma) \mid P \mathrel{\mathcal{R}} Q \text{ and } \sigma \text{ is injective}\}$. $\square$

Lemma 23 is the most complex case in the proof of congruence of $\sim$, for which Lemmas 9 and 23 are needed.

Lemma 23. *$P \sim Q$ implies $P \mid R \sim Q \mid R$ for all R.*

Proof. We prove $\{(P \mid R, Q \mid R) \mid P \sim Q\}$ is a bisimulation up to restriction and bisimilarity, and (thanks to Lemmas 7 and 15) we only focus on transition $P \mid R \xrightarrow{[\varphi]\tau} P_1$. Lemma 15 gives φ_1 such that $[\varphi]\tau \sqsubseteq_{P|R} [\varphi_1]\tau$ and $\xrightarrow{[\varphi_1]\tau}$ can be derived from transitions of P and/or R, separately.

From that, getting a similar $Q \mid R \xrightarrow{[\varphi]\tau} Q_1$ is easy, but one must relate $P_1 \mid \varphi$ and $Q_1 \mid \varphi$ in $\mathcal{R}$ using three different assumptions:

1. $R \xrightarrow{[\varphi_1]\tau} R'$. Then $P \mid R' \mid \varphi \sim\mathcal{R}\sim Q \mid R' \mid \varphi$ (using "up to $\sim$").
2. $P \xrightarrow{[\varphi_1]\tau} P'$. Then $P' \mid \varphi_1 \sim Q' \mid \varphi_1$, which implies $P' \mid R \mid \varphi \sim\mathcal{R}\sim Q' \mid R \mid \varphi$. Indeed φ_1 is absorbed by φ since $P \mid Q \mid \varphi \rhd \varphi_1$ ("up to $\sim$").
3. P and R synchronised into $(\nu x)(P' \mid R')$. Then $P' \sim Q'$ and we can relate $(\nu x)(P' \mid R') \mid \varphi$ to $(\nu x)(Q' \mid R') \mid \varphi$ ("up to $\sim$ and ν"). $\square$

Proposition 1 (Congruence of $\sim$). *Bisimilarity is a congruence.*

Proof. Restriction and parallel composition are handled by Lemmas 21 and 23, so we focus on prefixing and sum. Suppose $P \sim Q$, we want to prove $\pi.P \sim \pi.Q$ ($\pi.P + S \sim \pi.Q + S$ follows easily). We show $\{(\pi.P, \pi.Q)\} \cup \sim$ is a bisimulation.

If $\pi = \alpha(x)$ and $\pi.P \xrightarrow{\beta(y)} (\nu x)(y/x \mid P)$ then $\pi.Q \xrightarrow{\beta(y)} (\nu x)(y/x \mid Q)$ and $(\nu x)(y/x \mid P) \sim (\nu x)(y/x \mid Q)$ by Lemmas 21 and 23. (Similarly for $\pi = \overline{\alpha}(x)$.)

If $\pi = [\varphi]\tau$ and $\pi.P \xrightarrow{[\varphi_1]\tau} P$ then $\pi.Q \xrightarrow{[\varphi_1]\tau} Q$ and we need to prove $P \mid \varphi \sim Q \mid \varphi$ which holds by Lemma 23. $\square$

The following lemma is useful for the proof of completeness of $\sim$ w.r.t. $\simeq$.

Lemma 24. *For any P, Q and φ, $P \xrightarrow{[\varphi]\tau} Q$ iff $P \mid \varphi \xrightarrow{\tau} Q \mid \varphi$.*

Proof. If $P \xrightarrow{[\varphi]\tau} P'$, then $P \mid \varphi \xrightarrow{[\varphi]\tau} P' \mid \varphi$ and $P \mid \varphi \xrightarrow{\tau} P' \mid \varphi$ by Lemma 15, since $\tau \sqsubseteq^{\varphi} [\varphi]\tau$ and $P \mid \varphi \rhd \varphi$.
From $P \mid \varphi \xrightarrow{\tau} P' \mid \varphi$ Lemma 15 provides a condition ζ such that $P \xrightarrow{[\zeta]\tau} P'$ and $\tau \sqsubseteq_{P|\varphi} [\zeta]\tau$, which implies in turn $P \mid \varphi \rhd \zeta$. We prove by induction on $P \mid \varphi \rhd \zeta$ that $[\varphi]\tau \sqsubseteq_P [\zeta]\tau$ (see proof script in [18]) and conclude with Lemma 15. $\square$

The following "Harmony lemma" relates reduction and $\xrightarrow{\tau}$ transitions.

Lemma 25. *If $P \mapsto P'$ then $P \xrightarrow{\tau}\sim P'$, and if $P \xrightarrow{\tau} P'$ then $P \mapsto\sim P'$.*

Proof. Follows from Lemma 17 ($\mapsto$ produces an arc a/b when $\xrightarrow{\tau}$ produces the bisimilar process $(\nu x)(a/x \mid x/b)$). $\square$

Theorem 1 (Characterisation). *$P \simeq Q$ iff $P \sim Q$.*

Proof. The proof follows a standard pattern. Soundness is a consequence of congruence (Proposition 1), Lemma 25, and the correspondence of barbs with visible transitions (which follows from Lemma 17).

For completeness, we have to show that contexts can express the conditions in the three clauses in Definition 6. We define accordingly tester processes. The first clause is handled using process $[\varphi]\tau.\overline{w}$. For visible transitions (second clause), the counterpart of, e.g., $\xrightarrow{\{a\}(x)}$, is given by tester process $\overline{a}(y).(z/y \mid \overline{w} \mid w)$. We use φ for the third clause (by Lemma 24). $\square$

As mentioned above, the calculus in [13] is a version of $\pi\mathtt{P}$ with prefixes for free input and output, and without the corresponding bound prefixes (as well as without sum and conditional τ). We call that calculus $\pi\mathtt{P}_1$, and write $\simeq_{\pi\mathtt{P}_1}$ for barbed congruence in $\pi\mathtt{P}_1$. The encoding $[\cdot]_\mathrm{f}$, which we introduced in Remark 1, allows us to embed $\pi\mathtt{P}_1$ into $\pi\mathtt{P}$ in a faithful way:

Proposition 2 (Correspondence with [13]). *$P \simeq_{\pi\mathtt{P}_1} Q$ iff $[P]_\mathrm{f} \simeq [Q]_\mathrm{f}$.*

Proof. $[P]_\mathrm{f} \simeq [Q]_\mathrm{f} \Rightarrow P \simeq_{\pi\mathtt{P}_1} Q$ follows from the fact that $[\cdot]_\mathrm{f}$ induces standard correspondences of reductions, barbs, and contexts, from $\pi\mathtt{P}_1$ to $\pi\mathtt{P}$:

- if $P \mapsto_{\pi\mathtt{P}_1} P'$ then $[P]_\mathrm{f} \mapsto\simeq [P']_\mathrm{f}$,
- if $[P]_\mathrm{f} \mapsto P_1$ then $P_1 \simeq [P']_\mathrm{f}$ and $P \mapsto_{\pi\mathtt{P}_1} P'$ for some P',
- $P \downarrow_a^{\pi\mathtt{P}_1}$ iff $[P]_\mathrm{f} \downarrow_a$
- if C is a $\pi\mathtt{P}_1$ context, for some $\pi\mathtt{P}$ context D, for all P, $[C[P]]_\mathrm{f} = D[[P]_\mathrm{f}]$.

Consider now $\pi\mathtt{P}'$, the subcalculus of $\pi\mathtt{P}$ with neither sums nor $[\varphi]\tau$ constructs, and $\simeq'$, the barbed congruence induced by $\pi\mathtt{P}'$ contexts. The proof of completeness of $\sim$ only uses $\pi\mathtt{P}'$ contexts, hence $\simeq' = \sim = \simeq$.

We prove the above correspondences between $\pi\mathtt{P}'$ and $\pi\mathtt{P}_1$ for the encoding $[P]_\mathrm{p}^\mathrm{b} \triangleq [[P]_\mathrm{p}]^\mathrm{b}$ where $[\cdot]^\mathrm{b}$ is the standard encoding of bound prefixes using free prefixes (homomorphic except $[a(x).P]^\mathrm{b} \triangleq (\nu x)ax.\,[P]^\mathrm{b}$ and $[\overline{a}(x).P]^\mathrm{b} \triangleq (\nu x)\overline{a}x.\,[P]^\mathrm{b}$). This implies that $[P]_\mathrm{p}^\mathrm{b} \simeq_{\pi\mathtt{P}_1} [Q]_\mathrm{p}^\mathrm{b} \Rightarrow P \simeq' Q$ for $\pi\mathtt{P}'$ processes.

Remarking that $[\cdot]_\mathrm{f} : \pi\mathtt{P}_1 \to \pi\mathtt{P}'$, we compose the encodings and we prove $P \simeq_{\pi\mathtt{P}_1} [[P]_\mathrm{f}]_\mathrm{p}^\mathrm{b}$ by induction on P using the following $\pi\mathtt{P}_1$ laws [13, Lemma 17]:

$$\overline{a}b.P \simeq_{\pi\mathtt{P}_1} (\nu x)\overline{a}x.(b/x \mid P) \qquad ab.P \simeq_{\pi\mathtt{P}_1} (\nu x)ax.(x/b \mid P) \;.$$

Finally, if $P \simeq_{\pi\mathtt{P}_1} Q$, then $[[P]_\mathrm{f}]_\mathrm{p}^\mathrm{b} \simeq_{\pi\mathtt{P}_1} [[Q]_\mathrm{f}]_\mathrm{p}^\mathrm{b}$, and thus $[P]_\mathrm{f} \simeq [Q]_\mathrm{f}$. $\square$

3.4. Labelled Transitions for Free Prefixes

We now consider $\pi\mathtt{P}_\mathrm{F}$, the variant of $\pi\mathtt{P}$ where only the grammar for prefixes is changed, as follows ($\pi\mathtt{P}_\mathrm{F}$ is along the lines of $\pi\mathtt{P}_1$ in Proposition 2), but also has sums and conditional τ): $\pi \;::=\; \alpha b \;\big|\; \overline{a}b \;\big|\; [\varphi]\tau$.

We adapt the LTS of Section 3.1 to $\pi\mathsf{P}_\mathrm{F}$. The only modification we need to make is to replace the rules for bound prefixes by the following:

$$\xrightarrow{}\text{-F-IN}\quad \frac{x \notin \mathsf{n}(\alpha) \cup \{b\} \cup \mathsf{fn}(P)}{\alpha b.P \xrightarrow{\alpha(x)} x/\!b \mid P} \qquad\qquad \xrightarrow{}\text{-F-OUT}\quad \frac{x \notin \mathsf{n}(\alpha) \cup \{b\} \cup \mathsf{fn}(P)}{\overline{\alpha} b.P \xrightarrow{\overline{\alpha}(x)} b/\!x \mid P}$$

The calculi $\pi\mathsf{P}_\mathrm{F}$ and $\pi\mathsf{P}$ are similar, and can be encoded into each other. The domain of $[\cdot]_\mathrm{f}$ can be trivially extended to $\pi\mathsf{P}_\mathrm{F}$ (extended names are handled like regular ones, e.g. $[\alpha b.P]_\mathrm{f} \triangleq \alpha(x).(x/\!b \mid [P]_\mathrm{f})$). In the other direction, we need to extend $[\cdot]^\mathrm{b}$ to $\pi\mathsf{P}$, with some care to handle binders in sums:

$$\left[\Sigma_i \pi_i.P_i\right]^\mathrm{b} \quad \triangleq \quad (\nu x)\Sigma_i \begin{cases} \overline{\alpha}x.P_i & \text{if} \quad \pi_i = \overline{\alpha}(x) \\ \alpha x.P_i & \text{if} \quad \pi_i = \alpha(x) \\ [\varphi]\tau.P_i & \text{if} \quad \pi_i = [\varphi]\tau \end{cases} \quad ,$$

where x is chosen fresh in the last clause. Note that we rely on α-conversion to have the same name x bound in all summands in the encoding of sums.

We can show that both encodings preserve transitions and bisimilarity, so, as in Proposition 2, they preserve barbed congruence.

4. Axiomatisation

4.1. Equational Laws for Strong Bisimilarity

4.1.1. Notations and Terminology.

We use A to range over processes that consist of compositions of φ processes, which we call *preorder processes*. We often view such processes as multisets of conditions. We use notation A, P to denote a process that can be written, *using the monoid laws for parallel composition*, as $A \mid P$, where P does not contain toplevel arcs. (Note that A might contain restrictions, corresponding to the definition of join processes given in (1), towards the end of Section 2.1.) For example, $\overline{a} \mid b \curlyvee c \mid d.e \mid f \prec g$ can be written $(b \curlyvee c \mid f \prec g), (\overline{a} \mid d.e)$ but there is no A, P notation for $(\nu a)(b \curlyvee c \mid d.e)$.

We write $\vdash P = Q$ whenever P and Q can be related by equational reasoning using the laws of Table 2.

Note that we omit the standard equations expressing that $\mid$ and $+$ obey the laws of commutative monoids, and that $+$ is idempotent. We also omit the laws for equational reasoning (equivalence (reflexivity, symmetry, transivity) and substitutivity (axioms can be applied within contexts)). We will reason up to these laws in the remainder.

4.1.2. Comments on the laws.

Before presenting the properties of the axiomatisation, we comment on the laws of our axiomatisation and illustrate them on some examples.

As usual, the expansion law allows us to rewrite the parallel composition of two sums into one, the third summand describing synchronisation in $\pi\mathsf{P}$.

Expansion law (we can suppose $x \neq y$, $\mathsf{bn}(\pi_i) \notin \mathsf{fn}(T)$, $\mathsf{bn}(\rho_j) \notin \mathsf{fn}(S)$.)

$$\underbrace{\textstyle\sum_i \pi_i.P_i}_{S} \mid \underbrace{\textstyle\sum_j \rho_j.R_j}_{T} = \textstyle\sum_i \pi_i.(P_i \mid T) + \sum_j \rho_j.(S \mid R_j) + \sum_{i,j} [\alpha \curlyvee \beta]\tau.(\nu xy)(x/y \mid P_i \mid R_j) \quad \text{when } \alpha \curlyvee \beta \text{ is defined and } \{\pi_i, \rho_j\} = \{\overline{\alpha}(x), \beta(y)\}$$

Laws for preorder processes

L1 $a \prec b \mid b \prec c = a \prec b \mid b \prec c \mid a \prec c$ L2 $a \prec b \mid c \prec b = a \prec b \mid c \prec b \mid a \curlyvee c$

L3 $a \prec b \mid b \curlyvee c = a \prec b \mid b \curlyvee c \mid a \curlyvee c$ L4 $a \prec a = \mathbf{0}$

Laws for prefixes (the counterparts of laws L11-L12 for output are omitted)

L5 $\varphi, S + \pi.P = \varphi, S + \pi.(\varphi \mid P)$ L6 $[\varphi]\tau.P = [\varphi]\tau.(\varphi \mid P)$

L7 $[a \prec a]\tau.P = [b \curlyvee b]\tau.P$

L8 $[a \curlyvee b]\tau.P = [a \curlyvee b]\tau.P + [a \prec b]\tau.P$

L9 $[a \curlyvee b]\tau.P = [a \curlyvee b]\tau.P + [b \curlyvee a]\tau.P$

L10 $a(x).P = a(x).P + \{a\}(x).P$

L11 $b/a, S + a(x).P = b/a, S + a(x).P + b(x).P$

L12 $a/b, S + \{a\}(x).P = a/b, S + \{a\}(x).P + \{b\}(x).P$

L13 $b/a, S + [a \prec c]\tau.P = b/a, S + [a \prec c]\tau.P + [b \prec c]\tau.P$

L14 $a/b, S + [c \prec a]\tau.P = a/b, S + [c \prec a]\tau.P + [c \prec b]\tau.P$

L15 $b/a, S + [a \curlyvee c]\tau.P = b/a, S + [a \curlyvee c]\tau.P + [b \curlyvee c]\tau.P$

L16 $b/a, S + [a \curlyvee c]\tau.P = b/a, S + [a \curlyvee c]\tau.P + [c \prec b]\tau.P$

L17 $\alpha(y).P = \alpha(x).(\nu y)(x/y \mid P)$ if $x \notin \mathsf{fn}(P)$

L18 $\overline{\alpha}(y).P = \overline{\alpha}(x).(\nu y)(y/x \mid P)$ if $x \notin \mathsf{fn}(P)$

Laws for restriction (the counterparts of laws L25 and L26 for output are omitted; $a \prec b \in A^{\neq}$ stands for $a \prec b \in A$ and $a \neq b$, and similarly for $a \curlyvee b$.)

L19 $(\nu b)P = (\nu a)(P\{a/b\})$ if $a \notin \mathsf{fn}(P)$ L20 $(\nu c)(\nu d)P = (\nu d)(\nu c)P$

L21 $P \mid (\nu a)Q = (\nu a)(P \mid Q)$ if $a \notin \mathsf{fn}(P)$ L22 $(\nu a)\mathbf{0} = \mathbf{0}$

L23 $$(\nu a)A = \{b \prec c \mid b \prec a,\ a \prec c \in A^{\neq}\} \uplus \{b \curlyvee c \mid b \prec a,\ c \prec a \in A^{\neq}\} \uplus \{b \curlyvee c \mid a \curlyvee c,\ b \prec a \in A^{\neq}\} \uplus \{\varphi \in A \mid a \notin \mathsf{n}(\varphi)\}$$

L24 $(\nu a)(A,\ S + \pi.P) = (\nu a)\big(A,\ S + \pi.(\nu a)(A \mid P)\ \big)$ $a \notin \mathsf{n}(\pi)$

L25 $$(\nu a)(A,\ S + a(x).P) = (\nu a)\big(A,\ S + \textstyle\sum_{a \prec b \in A^{\neq}} b(x).(\nu a)(A \mid P) + \sum_{\substack{b \prec a \in A^{\neq} \\ \vee a \curlyvee b \in A^{\neq}}} \{b\}(x).(\nu a)(A \mid P)\ \big)$$

L26 $(\nu a)(A,\ S + \{a\}(x).P) = (\nu a)\big(A,\ S + \sum_{b \prec a \in A^{\neq}} \{b\}(x).(\nu a)(A \mid P)\ \big)$

L27 $(\nu a)(A,\ S + [a \prec c]\tau.P) = (\nu a)\big(A,\ S + \sum_{a \prec b \in A^{\neq}} [b \prec c]\tau.(\nu a)(A \mid P)\ \big)$

L28 $(\nu a)(A,\ S + [c \prec a]\tau.P) = (\nu a)\big(A,\ S + \sum_{b \prec a \in A^{\neq}} [c \prec b]\tau.(\nu a)(A \mid P)\ \big)$

L29 $$(\nu a)(A,\ S + [a \curlyvee c]\tau.P) = (\nu a)\big(A,\ S + \textstyle\sum_{a \prec b \in A^{\neq}} [b \curlyvee c]\tau.(\nu a)(A \mid P) + \sum_{\substack{b \prec a \in A^{\neq} \\ \vee a \curlyvee b \in A^{\neq}}} [c \prec b]\tau.(\nu a)(A \mid P)\ \big)$$

($a \neq c$ in L27-L29)

Table 2: An axiomatisation of $\sim$

Preorders. Laws L1-L4 express basic properties of relations $\prec$ and $\curlyvee$, and actually provide an axiomatisation of $\sim$ for preorder processes.

Prefixes. Law L5 propagates φs in depth, expressing the persistence of condition processes (φ). Law L6 is the counterpart of the third clause of Definition 6, and describes the outcome of a $[\varphi]\tau$ transition. Similarly, laws L17-L18 correspond to the firing of visible transitions in the LTS (regarding these rules, see also the comments after Proposition 4).

We note that α-conversion for input prefixes follows from laws L17-L19, by deriving the following equalities (and similarly for the other visible prefixes):

$$a(y).P \overset{L17}{=} a(x).(\nu y)(x/y \mid P) \overset{L19}{=} a(x).(\nu y')(x/y' \mid P\{y'/y\}) \overset{L17}{=} a(y').P\{y'/y\}\,.$$

Laws L11-L16 can be used to expand process behaviours using the preorder: arcs can modify the subject of visible prefixes (L11-L12) and the condition in $[\varphi]\tau$ prefixes (L13-L16). Laws L8, L9 and L13-L16 rely on the defining properties of relations $\prec$ and $\curlyvee$. Finally, law L7 is used to equate all reflexive τ prefixes.

Restriction. Laws L19-L22 are standard. The other laws are used to "push" restrictions inside processes. Due to the necessity to handle the preorder component (A), they are rather complex.

Law L23 is used to eliminate a restriction on a name a in a preorder process, by propagating the information expressed by all φs that mention a.

Law L24 is rather self-explanatory, and shows how the A component prevents us from simply pushing the restriction downwards (under prefixes).

Laws L25-L29 describe a kind of "synchronous application" of the prefix laws seen above. For instance, the two summands in law L25 correspond to applications of laws L11-L12: as we push the restriction on a downwards, we make sure that all possible applications of these laws are taken into account.

Intuitively, L23 is used after laws L24-L29 have been used to erase all prefixes mentioning the restricted name a, pushing the restriction on a inwards.

All in all, the set of laws in Table 2 is rather lengthy. We make two comments on this. First, it can be remarked that axiomatisations often treat restriction separately, by first focusing on a restriction-free calculus. In $\pi\mathsf{P}$, because of preorder processes, we cannot in general push restrictions on top of sum processes, so the situation is more complex (see also the discussion about [17] in Section 5).

Second, we could have presented the laws in a more compact way, by writing *schemas*. A uniform presentation for laws L7-L16 and L25-L29 is as follows:

$$\frac{\eta \sqsubseteq_A \mu \quad \mu.P \in S}{A, S = A, S + \eta.P} \qquad \frac{a \in \mathsf{fn}(\mu) \quad \forall \eta \sqsubseteq_A \mu \quad a \in \mathsf{fn}(\eta) \vee \exists \rho\ \eta \sqsubseteq_A \rho \wedge \rho.P \in S}{(\nu a)(A, \mu.P + S) = (\nu a)(A, S)} \tag{3}$$

(To remove $\mu.P$ from $\mu.P + S$, the second rule requires that some $\rho.P$ are in S. The second rule can be used to add those summands to S.) We prefer however to write all rules explicitly, since this is how they are handled in proofs.

4.1.3. Examples of derivable equalities.

In the following examples, we sometimes switch silently to notation A, P to ease readability. We also allow ourselves to simplify some reasonings involving prefixes where the object is not important. We explain how the following equalities between $\pi\mathtt{P}$ processes can be derived:

$$\begin{array}{rclll} (\nu a)(b/a \mid a/c) & = & b/c & (\nu a)(S{+}a(x).P) = (\nu a)S & \\ (\nu a)(a/b \mid a(x).P) & = & \{b\}(x).(\nu a)P & \overline{a}(x).x = \overline{a}(x).\{x\} & a(x).\{x\} = a(x).\mathbf{0} \end{array}$$

The first equality above is established using law L23: before getting rid of the restriction on a, we compute all conditions not involving a that can be deduced from $b/a \mid a/c$. In this case, this is only b/c.

The second equality is a direct consequence of law L25.

Law L25 is also used for the third equality: only the second summand in the law is not empty, which gives $(\nu a)(a/b, a(x).P) = (\nu a)(a/b, \{b\}(x).(\nu a)P)$. Then, L21 allows us to restrict the scope of νa, and we can get rid of $(\nu a)a/b$ using law 23, which yields the result.

Another way to see the third equality is to observe that we can derive $a/b, a(x).P = a/b, a(x).P + \{a\}(x).P + \{b\}(x).P$ using laws L10 and L12. In the latter process, the sum is intuitively *expanded*, in the sense that all derivable toplevel summands have been made explicit. When considering the restricted version of both processes, it is sound to push the restriction on a downwards in the expanded process, to obtain the expected equality. In this sense, law L25 implements a "synchronous version" of this reasoning, so as to ensure that when pushing a restriction downwards, the behaviour of the process is fully expanded.

The next two equalities illustrate the meaning of protected names. We reason as follows: $\overline{a}(x).x \overset{L18}{=} \overline{a}(x').(\nu x)(x'/x, x) \overset{L25}{=} \overline{a}(x').(\nu x)(x'/x, \{x'\}.(\nu x)\mathbf{0})$. We then obtain the expected equality by getting rid of $(\nu x)\mathbf{0}$ and $(\nu x)x'/x$, using laws L21-L23. The reason why this equality holds is that fresh name x is emitted without the context having the ability to interact at x, since x will never be below another name in an arc. Therefore, the input at x is equivalent to a protected input.

In the last equality, because of the transition $a(x).\{x\} \xrightarrow{a(x')} (\nu x)(x'/x \mid \{x\})$, x will never be above another name, so that the prefix $\{x\}$ cannot be triggered, and is equivalent to $\mathbf{0}$. This equality is derived as follows:

$$a(x).\{x\} \overset{L17}{=} a(x').(\nu x)(x'/x \mid \{x\}) \overset{L26}{=} a(x').(\nu x)x'/x \overset{L23}{=} a(x').\mathbf{0}\ .$$

(we have explained above how $a(x').\mathbf{0} = a(x).\mathbf{0}$ can be derived).

We leave it to the reader to check that the law for interleaving, presented in Section 1, can be derived using the expansion law, followed by the rules for prefixes and restriction to get rid of the summand $[x \curlyvee y]\tau.(\nu t, u)(t/u)$.

4.2. Soundness and Completeness of the Axioms

Lemma 26 (Soundness). *The laws of Table 2 relate bisimilar processes.*

PROOF (SKETCH). Laws L7-L16 add smaller summands in the sense of $\sqsubseteq_A$ (they are instances of the first rule of (3) in Section 4.1.2), which hence yield transitions that can be performed by the left-hand side by Lemma 15 and transitivity of $\sqsubseteq_A$. When no new summand can be created, we say the sum is *complete*. In laws L24-L29 we assume that sums are complete, which means that:

1. if A, S can do a transition μ, then so can $\pi.P$ with $\pi.P \in S$,
2. all the new summands of the form $\rho.(\nu a)(A \mid P)$ are such that $\rho.P \in S$.

Using (1) we can ignore the extra summand on the left-hand sides, and using (2) we can ignore the extra sum on the right-hand sides ($\rho.P \in S$ and $\rho.(\nu a)(A \mid P)$ have transitions to $(\nu x)(B \mid P)$ and resp. $(\nu x)(B \mid (\nu a)(A \mid P))$, for some arc B, which are bisimilar within context $(\nu a)(A \mid -)$). The other laws are easy. □

4.2.1. Auxiliary Results: Preorder Processes, Prefixes, Restriction.

Before establishing completeness, we first need technical results, given by Propositions 3, 4 and 5. First, laws L1-L4 can be used to *saturate* preorder processes:

Proposition 3. *If $A_1, S_1 \sim A_2, S_2$, then there exists $A^\star$ such that $\vdash A_i, S_i = A^\star, S_i$ ($i = 1, 2$), and $A^\star = \prod\{\varphi \mid \varphi$ not reflexive and $A_1 \rhd \varphi\}$.*

(Note that we could have picked A_2 instead of A_1 above.)

Proof. We rely on a rewriting relation on preorder processes. We write $A \overset{\curlyvee}{\mapsto} A'$ whenever A' is obtained from A using one of the laws L1-L4, oriented from left to right, as a rewrite rule modulo associativity and commutativity of parallel composition. We furthermore impose that no reflexive condition is added in a rewrite step, nor a condition that is already contained in the preorder process.

We then prove the following three properties about $\overset{\curlyvee}{\mapsto}$:

1. If $A \overset{\curlyvee}{\mapsto} A'$, then $A \sim A'$: this is a consequence of Lemma 26 (or, alternatively, follows from Lemma 20). This is useful to be able to relate $\overset{\curlyvee}{\mapsto}$-normal forms through $\sim$ and hence say they entail the same conditions.
2. For any (finite) A, there is no infinite $\overset{\curlyvee}{\mapsto}$-chain emanating from A. Indeed, the rules defining $\overset{\curlyvee}{\mapsto}$ do not introduce any new name. Moreover, a new arc or join can only be added if it is not already present. Since there are finitely many conditions built on a finite set of names, $\overset{\curlyvee}{\mapsto}$ terminates.
3. Suppose A is a $\overset{\curlyvee}{\mapsto}$-normal form. Then, for any non-reflexive φ, if $A \rhd \varphi$, then φ appears in A (which we write $\varphi \in A$). This follows by induction on the derivation of $A \rhd \varphi$. The only interesting case is for the $\rhd$-COMBINE rule, i.e. we know that $A \rhd \Gamma$ and $\Gamma \vdash \varphi$. We conclude by associating to rules $\vdash$-TRANS, $\vdash$-JOIN and $\vdash$-EXTJOIN laws L1, L2 and L3 respectively.

The observations above entail the expected property. □

We say that A is a *saturated preorder process* whenever $A^\star \equiv A$. We use $A^\star$ to range over such processes. We can remark that even if A contains only arcs, $A^\star$ may contain restrictions, because of induced conditions involving $\curlyvee$.

The next lemma relates transitions of sums and the laws for prefixes.

Lemma 27. *If $A, S \xrightarrow{\mu} A, P$ then $\vdash A, S = A, S + \pi.Q$ for some π and Q such that μ and π only differ in their bound names and $\pi.Q \xrightarrow{\mu} P$.*

Proof. Suppose S is of the form $S_1 + \pi'.Q$ and that the transition $A, S \xrightarrow{\mu} A, P$ is in fact (Lemma 15) coming from the summand $\pi'.Q \xrightarrow{\mu'} P$, with $\mu \sqsubseteq_{A,S} \mu'$ and $\mu' =_\alpha \pi'$. Since $\Phi(A, S) = \Phi(A)$ we know also that $\mu \sqsubseteq_A \mu'$.

Then we have directly $\pi \sqsubseteq_A \pi'$. We prove by induction on $\pi \sqsubseteq_A \pi'$ that for all S, $\vdash A, S + \pi'.Q = A, S + \pi'.Q + \pi.Q$.

Reflexivity of $\sqsubseteq$ is handled by the fact that $+$ is idempotent.

Transitivity ($\pi_3 \sqsubseteq_A \pi_2 \sqsubseteq_A \pi_1$) is handled by monoid laws for $+$. We write Q_i for $\pi_i.Q$ below. We know by induction (1) and (2), from which follows (3):
(1) $\vdash A, S + Q_1 = A, S + Q_1 + Q_2$ (for all S)
(2) $\vdash A, S + Q_2 = A, S + Q_2 + Q_3$ (for all S)
(3) $\vdash A, S + Q_1 = A, S + Q_1 + Q_2 = A, (S + Q_1) + Q_2 + Q_3 = A, S + Q_1 + Q_3$

Base case. We now decompose $\sqsubseteq^\varphi$ when $A \rhd \varphi$. We know there are some $\varphi_1, \dots, \varphi_n \in A$ and a reflexive ψ such that $\sqsubseteq^\varphi = \sqsubseteq^\psi \sqsubseteq^{\varphi_1} \dots \sqsubseteq^{\varphi_n}$ so in fact we only need to prove the result when $\varphi \in A$ or when φ is reflexive:

- φ is reflexive: $\alpha(x) \sqsubseteq^{a \prec a} \alpha(x)$ follows from idempotence of $+$, $\{a\}(x) \sqsubseteq^{a \curlyvee a} a(x)$ is law L10 and $[\varphi_1]\tau \sqsubseteq^\varphi [\varphi_2]\tau$ is either L8 or idempotence of $+$.
- $\varphi \in A$ and $\alpha(x) \sqsubseteq^{\beta \prec \alpha} \beta(x)$: this yields several cases:
  - $a(x) \sqsubseteq^{b \prec a} b(x)$: law L11
  - $\{a\}(x) \sqsubseteq^{a \prec b} \{b\}(x)$: law L12
  - $\{a\}(x) \sqsubseteq^{a \curlyvee b} b(x)$: decompose $a \curlyvee b$ back into $u/a \mid u/b$ (law L23) then from $b(x)$ get $u(x)$ by L11, then $\{u\}(x)$ (L10) and then $\{a\}(x)$ (L12).
- $\varphi \in A$ and $[\varphi_1]\tau \sqsubseteq^\varphi [\varphi_2]\tau$ when $\varphi_1, \varphi \vdash \varphi_2$: this yields several cases again, we can decompose $\vdash$ into usages of $\Vdash$, reasoning up to transitivity. (We use instances of laws L16, L9, L15, L13.)

Laws L8-L16 can be used to "saturate" the topmost prefixes in sums. We express this using the equivalence below, and rely on Lemma 27 to prove Prop. 4:

Definition 11 (Head sum normal form, $\asymp_h$). *Given two sum processes S and T, we write $S \prec_h T$ whenever for any summand $\pi.P$ of S, there exists a summand $\pi.Q$ of T with $\pi.P \sim \pi.Q$. We let $S \asymp_h T$ stand for $S \prec_h T \wedge T \prec_h S$.*

Proposition 4. *Whenever $A^\star, S_1 \sim A^\star, S_2$, where S_1, S_2 are two sum processes, there are S_1', S_2' s.t. $\vdash A^\star, S_i = A^\star, S_i'$ (for $i = 1, 2$) and $S_1' \asymp_h S_2'$.*

Proof. We first use law L5 to replicate $A^\star$ under all prefixes in S_1 and S_2, which is useful later in the proof. We therefore suppose that for any summand $\pi.P$ of S_1 or S_2, $P = A^\star \mid P_0$ for some P_0.

We prove the following property:

$$\begin{array}{lcll} A^\star, S_1 \sim A^\star, S_2 & \Rightarrow & \exists Q & \vdash A^\star, S_2 = A^\star, S_2 + \pi.Q \\ \pi.P \in S_1 & & & \pi.P \sim \pi.Q \end{array} \tag{4}$$

by running the bisimulation game with a label μ such that π and μ differ only on their object (that should be fresh in μ): $\pi.P \xrightarrow{\mu} P'$, yielding $A^\star, S_1 \xrightarrow{\mu} A^\star \mid P'$; the game returns a transition $A^\star, S_2 \xrightarrow{\mu} A^\star \mid Q'$. Using Lemma 27 we get Q such that $A^\star, S_2 = A^\star, S_2 + \pi.Q$ and $\pi.Q \xrightarrow{\mu} Q'$. We now have to prove that $\pi.P \sim \pi.Q$. There are two cases:

1. if μ is a visible action, then, by definition of $\sim$, we have $A^\star \mid P' \sim A^\star \mid Q'$. We can now observe that $P' \sim A^\star \mid P'$ and $Q' \sim A^\star \mid Q'$, because $A^\star$ has been replicated under prefixes. We thus deduce $P' \sim Q'$, which, by congruence, gives $\mu.P' \sim \mu.Q'$. Using the appropriate law among L17-18, we deduce $\pi.P \sim \mu.P'$ and $\pi.Q \sim \mu.Q'$, which implies $\pi.P \sim \pi.Q$.
2. if $\mu = [\varphi]\tau$ then $\pi = \mu$ and $P' = P$, $Q' = Q$. The bisimulation game yields $\varphi \mid A^\star \mid P' \sim \varphi \mid A^\star \mid Q'$ and by the same reasoning as before, $\varphi \mid P' \sim \varphi \mid Q'$. By congruence $[\varphi]\tau.(\varphi \mid P') \sim [\varphi]\tau.(\varphi \mid Q')$, and by L6, $[\varphi]\tau.P' \sim [\varphi]\tau.Q'$ i.e. $\pi.P \sim \pi.Q$.

We have now (4). Equation (4) implies that $\vdash A^\star, S_2 = A^\star, S_2 + T_2$ with $S_1 \prec_h S_2 + T_2$ and $T_2 \prec_h S_1$.

By reasoning symmetrically about $S_2 + T_2$, we obtain T_1 such that $\vdash A^\star, S_1 = A^\star, S_1 + T_1$ with $S_2 + T_2 \prec_h S_1 + T_1$ and $T_1 \prec_h S_2 + T_2$. Since we also have $S_1 \prec_h S_2 + T_2$, we can conclude $S_1 + T_1 \prec_h S_2 + T_2$ and thus $S_1' \asymp_h S_2'$ with $\vdash S_i' = S_i + T_i$. □

Remark 3 (On the definition of $\asymp_h$). *In the definition of $\prec_h$, we impose $\pi.P \sim \pi.Q$, and not simply $P \sim Q$. The equivalence induced by the choice of the latter condition would indeed be too discriminating. To see why, consider $Q_1 = a(x).c/x$ and $Q_2 = a(x).\mathbf{0}$. Obviously, $c/x \not\sim \mathbf{0}$. On the other hand, we have $Q_1 \sim Q_2$: after a $\xrightarrow{a(y)}$ transition on both sides, we must compare $(\nu x)(c/x \mid y/x)$ and $(\nu x)(y/x)$, and both are bisimilar to $\mathbf{0}$. In order to derive $\vdash Q_1 = Q_2$, we rely on the following property, which explains the shape of laws L17, L18: $a(y).P \sim a(y).Q$ iff $(\nu y)(x/y \mid P) \sim (\nu y)(x/y \mid Q)$.*

Proposition 5 expresses that restrictions can be pushed inwards in processes. It refers to the following notion of measure on processes (which is useful to reason by induction on processes in the completeness proof):

Definition 12 (Measure on processes). *Given a $\pi\mathtt{P}$ process P, we define $|P|$ as the maximum number of prefixes in summands of P, i.e., $|\sum_i \pi_i.P_i| = \max_i(1 + |P_i|)$ (hence $|\mathbf{0}| = 0$), $|(\nu a)P| = |P|$, $|P \mid Q| = |P| + |Q|$, and $|a/b| = 0$.*

Proposition 5. *For any A, S, a, there exist A' and S' such that $\vdash (\nu a)(A, S) = A', S'$ and $|(\nu a)(A, S)| \geq |A', S'|$.*

Proof. Using name extrusion, we pull all toplevel restrictions of A, S in order to derive $\vdash A, S = (\nu \widetilde{a})(A_0, S_0)$, for some A_0, S_0 without toplevel restriction.

We then reason by induction over the number of names in $\widetilde{a}$. We apply laws L25-L29 from left to right, until name a does not appear free in any topmost prefix of the sum. At that point, since the restriction on a has been pushed under prefixes, a has no free occurrence in the sum. We can thus use name intrusion, so that law L23 can be applied to get rid of the restriction on a on the preorder part of the process.

This operation is iterated until restrictions are pushed under all prefixes, and law L22 can be used to get rid of the restriction. □

4.2.2. Establishing Completeness.

The grammar $P ::= A, \sum_i \pi_i.P_i \bigm| (\nu a)P$ defines what we call *$|$-free processes*: only arcs are composed, and the non-preorder part of processes is a sum.

Proposition 6 (Characterisation, without parallel composition).
For all $|$-free processes P and Q, $P \sim Q$ iff $\vdash P = Q$.

Proof. The 'if' part follows from Lemma 26 and congruence of $\sim$.

Suppose now $P \sim Q$. We reason by induction on $|P| + |Q|$.

By Proposition 5, there are sum-only processes P_0, Q_0 with no toplevel restriction such that $\vdash P = P_0$ and $\vdash Q = Q_0$.

We then reason up to associativity and commutativity of parallel composition to write $\vdash P_0 = A_1, S_1$ and $\vdash Q_0 = A_2, S_2$. We have $A_1, S_1 \sim A_2, S_2$, which gives, by Proposition 3, $\vdash A_i, S_i = A^\star, S_i$ for $i = 1, 2$, for some $A^\star$.

We can then apply Proposition 4 to deduce $\vdash A^\star, S_i = A^\star, S'_i$, for $i = 1, 2$, for some S'_1, S'_2 s.t. $S'_1 \asymp_h S'_2$.

To sum up, we have proved until now $\vdash P = A^\star, S'_1$, $\vdash Q = A^\star, S'_2$, and $S'_1 \asymp_h S'_2$.

We now prove, by induction over the number of summands in S'_1, that for any such summand $\pi.T_1$, there is a summand $\pi.T_2$ in S'_2 s.t. $\vdash \pi.T_1 = \pi.T_2$. Once this will be proved, we shall establish the same way the symmetrical property, which will allow us to deduce $\vdash S'_1 = S'_2$.

Suppose then $\pi.T_1$ is a summand of S'_1.

We reason by case analysis on the shape of π, and suppose $\pi = a(x)$. We know, since $S'_1 \asymp_h S'_2$, that there is a summand $a(x).T_2$ of S'_2 such that $a(x).T_1 \sim a(x).T_2$. We have $\vdash a(x).T_i = a(y).(\nu x)(y/x \mid T_i)$, for $i = 1, 2$, by law L17, induction hypothesis (on $(\nu x)(y/x \mid T_1) \sim (\nu x)(y/x \mid T_2)$) and congruence using the context $a(y).[\cdot]$.

Moreover, since $a(x).T_1 \sim a(x).T_2$, we know, by unfolding bisimilarity with the transition labelled by $a(y)$, that $(\nu x)(y/x \mid T_1) \sim (\nu x)(y/x \mid T_2)$. This allows us to rely on the induction hypothesis to show $\vdash (\nu x)(y/x \mid T_1) = (\nu x)(y/x \mid T_2)$ and hence $\vdash a(y).(\nu x)(y/x \mid T_1) = a(y).(\nu x)(y/x \mid T_2)$ which gives us, as announced, $\vdash \pi.T_1 = \pi.T_2$.

The other cases for the shape of π are treated similarly. □

We now move to the full calculus, by taking into account parallel composition. As is usually the case, this relies on a law for expansion.

The expansion law yields the following result, which then gives Theorem 2.

Lemma 28. *For any P, there exists a $|$-free process Q s.t. $\vdash P = Q$.*

PROOF. First $\vdash P = A, P_1$ using the monoid laws for parallel composition. Then by induction on P_1 we build S such that $\vdash P_1 = S$ and $|P_1| = |S|$. There are only two cases: sums, and parallel compositions of two sums (by induction hypothesis), on which we apply the expansion law. Both preserve $|\cdot|$. $\square$

We can then extend Proposition 6 to the whole $\pi\mathtt{P}$ calculus:

Theorem 2 (Axiomatisation of $\sim$). *For all P and Q, $P \sim Q$ iff $\vdash P = Q$.*

PROOF. The theorem follows from Proposition 6 and Lemma 28. $\square$

Remark 4 (Normal forms). *The proofs in this section suggest that we can define a strategy to apply the laws of our axiomatisation, in order to rewrite a $\pi\mathtt{P}$ process P to its* normal form, $\mathsf{nf}(P)$, *so that $P \sim Q$ iff $\mathsf{nf}(P) = \mathsf{nf}(Q)$.*

For preorder processes, the saturated form is a normal form for $\sim$: if $A_1 \sim A_2$, then, by Proposition 3, $\vdash A_1^\star = A_2^\star$. By contrast, the proof of Proposition 4 does not compute a canonical form for sum processes. For instance, from the equivalence

$$b/a \mid c/a \mid \overline{a}(x).\mathbf{0} \sim b/a \mid c/a \mid \overline{a}(x).\mathbf{0} + \overline{b}(x).\mathbf{0} \ ,$$

Proposition 4 rewrites these processes into $b/a \mid c/a \mid \overline{a}(x).\mathbf{0} + \overline{b}(x).\mathbf{0}$, but not into $b/a \mid c/a \mid \overline{a}(x).\mathbf{0} + \overline{b}(x).\mathbf{0} + \overline{c}(x).\mathbf{0}$, which could be seen as a normal form for $\sim$, obtained by saturating the sum. Actually, the normal form could even be

$$b/a \mid c/a \mid \overline{a}(x).\mathbf{0} + \overline{b}(x).\mathbf{0} + \overline{c}(x).\mathbf{0} + \overline{\{a\}}(x).\mathbf{0} + \overline{\{b\}}(x).\mathbf{0} + \overline{\{c\}}(x).\mathbf{0} \ ,$$

by virtue of several applications of (the counterpart for output of) law L10 with π_1 an output prefix and π_2 a protected output.

The full description of this normalisation procedure is left for future work.

4.3. Adapting our Axiomatisation to Explicit Fusions

We can reuse the ideas presented above to describe an axiomatisation for barbed congruence in Explicit Fusions (EF, [12, 25]). Accordingly, we adopt a presentation of the calculus that follows the lines of $\pi\mathtt{P}$ as we have introduced it, by having primitive bound prefixes. EF feature *fusion processes*, of the form $a{=}b$, which can equate names via $\equiv$: we have $a{=}b \mid P \equiv a{=}b \mid P\{b/a\}$.

The grammar of prefixes, conditions and processes is as follows:

$$\varphi ::= a{=}b \qquad \pi ::= \overline{a}(x) \mid a(x) \mid [a{=}b]\tau \qquad P ::= P|Q \mid \nu a P \mid \Sigma_i \pi_i.P_i \mid a{=}b$$

As in $\pi\mathtt{P}$, we adopt primitive bound prefixes. Free prefixes can be encoded: $[ab.P] = (\nu u)a(u).(u{=}b \mid P)$. Note in passing that fusions can be represented in $\pi\mathtt{P}$, encoding $a{=}b$ with $a/b \mid b/a$.

The following definition is adapted from the *efficient bisimulation* of [25]. The LTS is defined according to the approach in Table 1, except in their LTS $P \xrightarrow{\tau} P'$ does not necessarily imply that $P \xrightarrow{[a=b]\tau} P'$ for every a and b. So we have to change the third clause of the definition of bisimulation to take this into account. The way we handle objects does not matter, as the resulting bisimilarity is the same.

Definition 13 ($\sim_{\mathrm{EF}}$). *An* efficient bisimulation *is a symmetric relation $\mathcal{R}$ such that if $P\mathcal{R}Q$ then:*

1. *$P \rhd a{=}b$ iff $Q \rhd a{=}b$;*
2. *$P \xrightarrow{\mu} P'$ implies $Q \xrightarrow{\mu} Q'$ for some Q' s.t. $P'\mathcal{R}Q'$, for $\mu \neq [a{=}b]\tau$;*
3. *$P \xrightarrow{[a=b]\tau} P'$ implies $a{=}b \mid Q \xrightarrow{\tau} Q'$ and $a{=}b \mid P' \;\mathcal{R}\; Q'$.*

We write $\sim_{\mathrm{EF}}$ for the largest efficient bisimulation.

We write $\vdash_{\mathrm{EF}} P = Q$ if the equality can be derived using equational reasoning with the laws of Table 3 (we again omit the monoid laws for $\mid$ and $+$).

The axiomatisation is considerably simpler than in $\pi\mathsf{P}$. Local reasoning is possible because the stateful component of processes encodes an equivalence relation on names, while a form of global reasoning is necessary in the laws of Table 2 for $\pi\mathsf{P}$. The fact that fusions are symmetric yields Lemma 29:

Lemma 29. *For any P, a, b, we can derive $\vdash_{\mathrm{EF}} P \mid a{=}b = P\{a/b\} \mid a{=}b$.*

This renders useless protected names and simplifies greatly the handling of restriction: while the counterparts of Lemmas 26-28 and Propositions 3-6 hold, their proofs are much shorter, e.g. Proposition 5 boils down to two cases:

- either $A \rhd a = b$ with $b \neq a$, in which case $\vdash_{\mathrm{EF}} (\nu a)(A, S) = A\{b/a\}, S\{b/a\}$,
- or there is no such b and $\vdash_{\mathrm{EF}} (\nu a)(A, S) = A, (\nu a)S$,

which means there is no use for counterparts of laws L25-L29. We conclude:

Proposition 7. *For any P, Q, we have $P \sim_{\mathrm{EF}} Q$ iff $\vdash_{\mathrm{EF}} P = Q$.*

Restriction laws.

$$(\nu a)(P \mid Q) = P \mid (\nu a)Q \quad \text{and} \quad (\nu b)P = (\nu a)P\{a/b\} \text{ when } a \notin \mathsf{fn}(P)$$

$$(\nu a)\textstyle\sum_i \pi_i.P_i = \sum_{i \mid a \notin \mathsf{n}(\pi_i)} \pi_i.(\nu a)P_i$$

Laws for fusions.

$$a{=}a = \mathbf{0} \qquad a{=}b \mid a{=}c = a{=}b \mid a{=}c \mid b{=}c \qquad a{=}b = b{=}a \qquad (\nu a)a{=}b = \mathbf{0}$$

Laws for prefixes.

$$a{=}b \mid S + \pi.P = a{=}b \mid S + \pi.(a{=}b \mid P) \qquad [a{=}b]\tau.P = [a{=}b]\tau.(a{=}b \mid P)$$

$$\pi.P + \pi.P = \pi.P \qquad [a{=}b]\tau.P = [b{=}a]\tau.P \qquad a{=}b \mid [a{=}c]\tau.P = a{=}b \mid [b{=}c]\tau.P$$

$$a{=}b \mid S + a(x).P = a{=}b \mid S + b(x).P \qquad a{=}b \mid S + \overline{a}(x).P = a{=}b \mid S + \overline{b}(x).P$$

Expansion law.

$$\textstyle\sum_i \pi_i.P_i \mid \sum_j \rho_j.R_j = \sum_i \pi_i.(P_i \mid T) + \sum_j \rho_j.(S \mid R_j) + \sum_{i,j} [a{=}b]\tau.(\nu x)(P_i \mid R_j)$$

$$\text{where } \{\pi_i, \rho_j\} = \{\overline{a}(x), b(x)\}$$

Table 3: Axiomatisation for the Explicit Fusions calculus

5. Conclusions and Future Work

Working with a preorder on names has an influence on the behavioural theory of πP, notably through the interplay between arcs and restrictions. The preorder relation is represented explicitly in πP processes, using arcs. We do not see any natural "implicit version" of πP, mimicking the relation between Explicit Fusions and Fusions, whereby the extension of the preorder along a communication would not generate an arc process.

The stateful nature of the preorder component of πP processes can be related to *frames* in the applied π-calculus [1] and Psi-calculi [2]. Arcs in πP can be seen in some sense as substitutions, but they differ from the active substitutions of applied π. The latter map variables to terms, while, in the tradition of fusion calculi, we only have (channel) names in πP. Moreover, several arcs acting on the same name are allowed in πP, while a substitution acts on at most one variable in applied π. For these reasons, the behavioural theories of πP and applied π are rather different. Liu and Lin's proof system for applied π [17] departs from our axiomatisation for πP, but has in common the stateful component of processes.

The behavioural theory of πP is based on an operational account. An intriguing question is the construction of a denotational model for πP, and the comparison with known models for π and Fusions. Given the proximity of πP to Explicit Fusions, one way to proceed could be to adapt the approach of [3]. We would also like to study the weak version of behavioural equivalence.

The results of this work provide foundations for the behavioural theory of the πP calculus, which also has i/o-types (cf. [13]). As already mentioned, typed behavioural equivalence [7, 21] can be used to establish fine behavioural properties of concurrent systems. We would like to find out whether it can be helpful to refine untyped analyses of systems where Fusions have been used.

Acknowledgements. We thank Davide Sangiorgi and Fu Yuxi for useful discussions about this work, as well as the anonymous referees for their helpful remarks. This work has been supported by projects ANR 12IS02001 PACE, ANR 2010-BLAN-0305 PiCoq and NSF of China (61261130589).

[1] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proc. of POPL*, pages 104–115. ACM, 2001.

[2] J. Bengtson, M. Johansson, J. Parrow, and B. Victor. Psi-calculi: Mobile processes, nominal data, and logic. In *LICS*, page 39–48. IEEE, 2009.

[3] F. Bonchi, M. G. Buscemi, V. Ciancia, and F. Gadducci. A presheaf environment for the explicit fusion calculus. *J. Autom. Reasoning*, 49(2):161–183, 2012.

[4] M. Boreale, M. G. Buscemi, and U. Montanari. D-fusion: A distinctive fusion calculus. In *Proc. APLAS*, volume 3302 of *LNCS*, pages 296–310. Springer, 2004.

[5] M. Boreale, M. G. Buscemi, and U. Montanari. A general name binding mechanism. In *Proc. TGC*, volume 3705 of *LNCS*, pages 61–74. Springer, 2005.

[6] S. Carpineti, C. Laneve, and L. Padovani. PiDuce - A project for experimenting Web services technologies. *Sci. Comput. Program.*, 74(10):777–811, 2009.

[7] Y. Deng and D. Sangiorgi. Towards an algebraic theory of typed mobile processes. *Theor. Comput. Sci.*, 350(2-3):188–212, 2006.

[8] The Coq development team. *The Coq proof assistant reference manual.* LogiCal Project, 2004. Version 8.0.

[9] T. Ehrhard and O. Laurent. Acyclic solos and differential interaction nets. *Logical Methods in Computer Science*, 6(3), 2010.

[10] Y. Fu. The χ-calculus. In *APDC*, pages 74–81. IEEE Computer Society, 1997.

[11] P. Gardner, C. Laneve, and L. Wischik. The fusion machine. In *CONCUR*, volume 2421 of *Lecture Notes in Computer Science*, pages 418–433. Springer, 2002.

[12] P. Gardner and L. Wischik. Explicit fusions. In *MFCS*, volume 1893 of *LNCS*, pages 373–382. Springer, 2000.

[13] D. Hirschkoff, J.-M. Madiot, and D. Sangiorgi. Name-passing calculi: From fusions to preorders and types. In *LICS*, pages 378–387. IEEE, 2013.

[14] D. Hirschkoff, J.-M. Madiot, and X. Xu. A behavioural theory of a π-calculus with preorders (extended abstract). In *Proc. of FSEN*, LNCS. Springer, 2015.

[15] K. Honda and N. Yoshida. On reduction-based process semantics. *Theor. Comp. Sci.*, 152(2):437–486, 1995.

[16] C. Laneve and B. Victor. Solos in concert. *Mathematical Structures in Computer Science*, 13(5):657–683, 2003.

[17] J. Liu and H. Lin. Proof system for applied pi calculus. In *Proc. IFIP TCS*, volume 323 of *IFIP Advances in Inf. and Comm. Technol.*, pages 229–243. Springer, 2010.

[18] J.-M. Madiot. Coq proof scripts for some results of this paper, 2014. Available at http://madiot.org/pip-relations.tar.gz.

[19] J. Parrow and D. Sangiorgi. Algebraic theories for name-passing calculi. *Inf. Comput.*, 120(2):174–197, 1995.

[20] J. Parrow and B. Victor. The fusion calculus: expressiveness and symmetry in mobile processes. In *LICS*, pages 176 –185. IEEE, 1998.

[21] B. C. Pierce and D. Sangiorgi. Typing and subtyping for mobile processes. *Mathematical Structures in Computer Science*, 6(5):409–453, 1996.

[22] D. Sangiorgi. Pi-calculus, internal mobility, and agent-passing calculi. *Theor. Comput. Sci.*, 167(1&2):235–274, 1996.

[23] D. Sangiorgi and D. Walker. *The Pi-Calculus: a theory of mobile processes.* Cambridge University Press, 2001.

[24] B. Victor and J. Parrow. Concurrent constraints in the fusion calculus. In *Proc. of ICALP*, volume 1443, pages 455–469, 1998.

[25] L. Wischik and P. Gardner. Strong bisimulation for the explicit fusion calculus. In *Proc. of FoSSaCS*, volume 2987 of *LNCS*, pages 484–498, 2004.