Concurrent Nets

A Study of Prefixing in Process Calculi

Emmanuel Beffara, François Maurel

Équipe PPS, Université Paris 7 & CNRS

Abstract

We introduce the calculus of concurrent nets as an extension of the fusion calculus in which usual prefixing is replaced by arbitrary monotonic guards. Then we use this formalism to describe the prefixing policy of standard calculi as a particular form of communication. By developing a graphical syntax, we sharpen the geometric intuition and finally we provide an encoding of these guards as causality in the prefix-free fragment, in the spirit of the encoding of the fusion calculus into solos by Laneve and Victor, proving that communication by fusion is expressive enough to implement arbitrary monotonic guards.

1 Introduction

The π-calculus [10] has generated a wide range of calculi on the search for both a simplification of the syntax and a widening of the expressiveness. Fu's χ-calculus [2], Parrow and Victor's fusion calculus [12] and Gardner and Wischik's explicit fusions [3] are important examples where name substitution is replaced by unification, which makes the calculus simpler, more symmetric and yet more expressive. Most models for concurrent and mobile computation are geometric in nature, and even term calculi have a strong spatial intuition. Indeed, every process calculus comes with a handful of structural rules for commutation and scoping that define appropriate notions of locality. This geometric flavour of term calculi led to the proposal of several graphical syntaxes for existing calculi, like π-nets [9] or solo diagrams [7], and to the introduction of new purely graphical calculi, like Milner's recent work on bigraphs [5].

In a sense, the evolution from π-calculus to fusion corresponds to the evolution from syntactical communication to a more geometric one; however the sequentiality imposed by prefixing remains very syntactical. Motivated by the search for a more general form of prefixing, we introduce and study the calculus of concurrent nets as a similar evolution towards a geometrical formulation of sequentiality constraints. Each action in a process gets associated with a semaphore that indicates when the action has been performed, and subsequently prefixing is replaced by the use of guards that are monotonic

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

functions of those semaphores. We define a graphical syntax for this calculus, in which guards appear as a new form of communication between actions.

The first contribution of this paper is to show how concurrent nets extend existing calculi, and notably the π-calculus and the fusion calculus. The characterisation of these sub-calculi yields a classification of the forms of sequentiality in use in process calculi using natural geometric arguments.

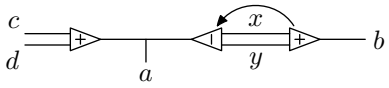
With the solos calculus [8], Laneve and Victor simplify the fusion calculus by removing prefixing and they show by means of an encoding that no expressive power is lost. Our second contribution extends their theorem by encoding the arbitrary monotonic guards of concurrent nets into pure communications, using a new and more geometric approach suggested by our graphical syntax.

The authors would like to thank Claudia Faggian, Vincent Danos and Pierre-Louis Curien for their helpful remarks on this work.

2 Definitions

2.1 Introductory Examples

Informally, a concurrent net is a web of input and output actions related by channels. Each action has a principal port (the subject) and a set of auxiliary ports (the objects). For instance, the process $\bar{a}(cd) \mid a(xy).\bar{b}(xy)$ in π-calculus is represented as



Formally, the set of channels here is $\mathcal{C} = \{a, b, c, d, x, y\}$, but only a, b, c, d are considered public, which is represented by the fact that they have dangling edges while x and y have none. The actions $\bar{a}(cd)$ on the left and $a(xy)$ in the middle form a redex. The reduction of this redex will remove both actions and connect c with x and d with y. The arrow means that the third action $\bar{b}(xy)$ is prefixed by $a(xy)$, i.e. it will not be reduced as long as $a(xy)$ is present.

We generalise this prefixing in two ways. Consider the following examples:

In both examples, we have two receptions $a(x)$ and $b(y)$ and one emission $\bar{c}(xy)$. In the process on the left, the two-headed arrow means that the emission is prefixed by both receptions, i.e. $\bar{c}(xy)$ will be blocked until both $a(x)$ and $b(y)$ have been reduced, but these may happen in any order. This cannot be expressed directly in π-calculus, but some calculi (*e.g.* the join calculus [1]) do provide this kind of synchronisation. In the process on the right, the two disjoint arrows mean that $\bar{c}(xy)$ will be able to act as soon as either $a(x)$ or $b(y)$ is consumed. To our knowledge, this too cannot be expressed directly in other calculi. Note that, if $a(x)$ is consumed, y may be communicated before anyone writes on b. This phenomenon is typical of fusion calculi.

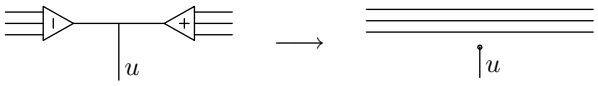


Fig. 1. Reduction of a redex.

2.2 Syntax and Semantics

We assume a countable set $\mathcal{C}$ of channel names and a disjoint set $\mathcal{L}$ of location names. The elements of $\mathcal{L}$ are used to name occurrences of actions, as detailed below. We write $\tilde{x}$ to represent a finite sequence of channel names $x_1 \dots x_{|\tilde{x}|}$.

Definition 2.1 $\mathcal{C}$-terms are defined by the following grammar:

$$\begin{array}{ll} \text{actions} & \alpha := \bar{u}(\tilde{x}) \mid u(\tilde{x}) \\ \text{prefixes} & \pi := 0 \mid 1 \mid \ell \mid \pi + \pi \mid \pi\pi \\ \text{terms} & P := \mathbf{0} \mid (P \mid P) \mid (x)P \mid (\ell)P \mid \ell : \alpha \mid \langle \pi \rangle\, P \end{array}$$

where u, x, x_i range over channels and ℓ ranges over locations.

- The α-*equivalence* on $\mathcal{C}$-terms is generated by the renaming of bound names. A channel or location name x is bound by the closest surrounding (x).
- A location ℓ is *defined* in P if some $\ell : \alpha$ occurs in P with ℓ unbound. We denote by $\mathrm{loc}(P)$, the set of locations defined in P.
- We require that each location be defined at most once in any subterm (this condition on terms will be preserved by reduction).

The actions $\bar{u}(\tilde{x})$ and $u(\tilde{x})$ correspond to the emission and reception of some sequence of channels $\tilde{x}$ on a channel u. Prefixes are either blocked (0), enabled (1), simple (a location ℓ), disjunctive ($\pi_1 + \pi_2$) or conjunctive ($\pi_1\pi_2$). A prefixing $\langle \ell \rangle\, P$ means that P is blocked until the action $\ell : \alpha$ is performed. When this happens, the name ℓ is replaced by 1 in every prefix. Subsequently, a process $\langle \pi \rangle\, P$ is blocked until the prefix π is reduced to 1 by the replacement of some ℓ by 1 and the application of structural rules. The other constructions are standard: $\mathbf{0}$ is the inactive process, $P_1 \mid P_2$ represents two processes in parallel and $(x)P$ represents the process P with a local name x.

We use the notation α for $(\ell)(\ell : \alpha)$, i.e. when the location ℓ is not used in any prefix. The notation $u^\varepsilon(\tilde{x})$ refers to an action with arbitrary polarity, where ε is $+$ for the action $\bar{u}(\tilde{x})$ and $-$ for $u(\tilde{x})$.

Example 2.2 The process

$$\mathit{shared\_continuation}(a, b, P) := (\ell_1)(\ell_2)\big(\ell_1 : a() \mid \ell_2 : b() \mid \langle \ell_1 + \ell_2 \rangle\, P\big)$$

is a typical process that does not exist primitively in usual calculi such as fusion calculus. The process P in $\mathit{shared\_continuation}(a, b, P)$ is enabled by the unblocking of ℓ_1 or ℓ_2 or both (the location ℓ_1 is unblocked when the occurrence of $a()$ in $\ell_1 : a()$ is used in reduction and similarly for ℓ_2 and $b()$). Even if it may be encoded in fusion calculus, as shown in Sec. 4, such a mechanism

is not primitive. For instance, a similar process in fusion calculus would be $Q = (f)(a().\bar{f}() \mid b().\bar{f}() \mid f().P)$ which has the intended meaning: when a or b has been performed, the flag f is released and P becomes available. The methodological difference is quite subtle: in this example, the encoding works because there is only one $\bar{f}()$ that can interact (and the other possible $\bar{f}()$ is not used and can be garbage collected) whereas in $\mathit{shared\_continuation}(a, b, P)$, both ℓ_1 and ℓ_2 enable P and no garbage collection rule is necessary.

Example 2.3 Another more involved example is the process

$$P = (\ell_a \ell_b \ell_c)\big(\ \ell_a : a_1() \mid \ell_b : b_1() \mid \ell_c : c_1() \\ \mid\ \langle \ell_a \rangle\, \bar{a}_2() \mid \langle \ell_b \rangle\, \bar{b}_2() \mid \langle \ell_c \rangle\, \bar{c}_2() \mid \langle \ell_a \ell_b \rangle\, \bar{d}() \mid \langle \ell_a + \ell_b \ell_c \rangle\, \bar{e}()\ \big)$$

The process P listens on three ports a_1, b_1 and c_1. For each of them it answers respectively on ports a_2, b_2 and c_2. Furthermore, when both $a_1()$ and $b_1()$ have been fired, P sends $\bar{d}()$, and when $a_1()$ or both $b_1()$ and $c_1()$ have been fired $\bar{e}()$ is sent. In fusion calculus, a similar process could be

$$Q = (f_a f_{b1} f_{b2} f_c g)\big(\ a_1().\big(\bar{f}_a() \mid \bar{g}() \mid \bar{a}_2()\big) \\ \mid b_1().\big(\bar{f}_{b1}() \mid \bar{f}_{b2}() \mid \bar{b}_2()\big) \mid c_1().\big(\bar{f}_c() \mid \bar{c}_2()\big) \\ \mid f_a().f_{b1}().\bar{d}() \mid f_{b2}().f_c().\bar{g}() \mid g().\bar{e}()\ \big)$$

but this is more complex: one must be cautious when programming such a process to check that the flags f_a, f_{b1}, f_{b2}, f_c and g are sufficient for the intended purpose. Moreover, the encoding is too sequential: in the encoding, we write $f_a().f_{b1}().\bar{d}()$ but could as well write $f_{b1}().f_a().\bar{d}()$ in this case. This asymmetry prevents an easy and natural understanding of such a process.

These processes exemplify a situation where one has simple processes in $\mathcal{C}$-terms and tricky encodings in fusion. This situation is typical as shown by encodings (in solos) developed in Sec. 4. Therefore, we see $\mathcal{C}$-terms as a plain process calculus with important properties but also as some kind of macro language giving new *design patterns* for process calculi such as π or solos.

Definition 2.4 The structural equivalence on terms is the smallest congruence $\equiv$ containing α-equivalence and such that

- the set of terms is a commutative monoid with $\mid$ as the composition and $\mathbf{0}$ as the neutral element,
- let Π be the set of prefixes, then $(\Pi, 0, +)$ and $(\Pi, 1, .)$ are commutative monoids, mutually distributive, with $1 + x \equiv 1$ and $0\,x \equiv 0$,
- prefixing obeys the following rules, where ℓ does not occur in π:

$$\langle 0 \rangle\, P \equiv \mathbf{0} \qquad \langle 1 \rangle\, P \equiv P \qquad \langle \pi_1 \rangle\, \langle \pi_2 \rangle\, P \equiv \langle \pi_1 \pi_2 \rangle\, P$$
$$\langle \pi \rangle\, (P_1 \mid P_2) \equiv \langle \pi \rangle\, P_1 \mid \langle \pi \rangle\, P_2 \quad \langle \pi \rangle\, (x)P \equiv (x)\, \langle \pi \rangle\, P \quad \langle \pi \rangle\, (\ell)P \equiv (\ell)\, \langle \pi \rangle\, P$$

- channel and location scoping obeys the following standard equivalences, where z is not free in P, and n is neither defined nor used in P:

$$\begin{array}{lll} (x)(y)P \equiv (y)(x)P & (z)P \equiv P & P \mid (z)Q \equiv (z)(P \mid Q) \\ (\ell)(m)P \equiv (m)(\ell)P & (n)P \equiv P & P \mid (n)Q \equiv (n)(P \mid Q) \end{array}$$

The operational semantics of $\mathcal{C}$-terms is defined as a labelled transition system (LTS). The choice of an LTS instead of a simple reduction system comes from fusion effects and prefix updates. When a process P reduces into P', the process $P \mid Q$ reduces into $P' \mid Q'$ where Q' is Q with some unified variables and some reduced prefixes (the reduction in P may have unblocked some locations that appear as prefixes in Q). Hence, for compositionality, we use an LTS which can properly deal with parallel composition.

Transitions are labelled either (φ, L) or $((\tilde{x})\alpha, L)$ where φ is an equivalence over $\mathcal{C}$, L is a subset of $\mathcal{L}$, $\tilde{x}$ is a subset of $\mathcal{C}$ and α is an action. (φ, L) means that a unification φ is performed, and $((\tilde{x})\alpha, L)$ means that the action α is fired and some variables $\tilde{x}$ are extruded. In both cases, some locations L are unblocked (the actions in the locations L have been fired).

Definition 2.5 We write $\{\tilde{x} = \tilde{y}\}$ to denote the smallest equivalence that unifies $\tilde{x}$ with $\tilde{y}$, $x \notin \varphi$ if the equivalence class of x is $\{x\}$, $\varphi \setminus x$ for $\varphi \cap (\mathcal{C} \setminus \{x\})^2 \cup \{(x,x)\}$, and $[1/L]$ for the substitution of each name in L by 1 in prefixes. A substitution σ *implements* a relation φ if $x \ \varphi \ y$ iff $\sigma(x) = \sigma(y)$. The transition rules for $\mathcal{C}$-terms, up to structural equivalence, are

$$\frac{}{\ell : \alpha \xrightarrow{\alpha,\{\ell\}} \mathbf{0}} \qquad \frac{P_1 \xrightarrow{(\tilde{x}_1)\bar{a}(\tilde{y}_1),L_1} P_1' \qquad P_2 \xrightarrow{(\tilde{x}_2)a(\tilde{y}_2),L_2} P_2' \qquad |\tilde{y}_1| = |\tilde{y}_2|}{P_1 \mid P_2 \xrightarrow{\{\tilde{y}_1=\tilde{y}_2\}\setminus\tilde{x}_1\tilde{x}_2,L_1\cup L_2} (\tilde{x}_1\tilde{x}_2)(P_1' \mid P_2')\sigma[1/L_1, L_2]}$$

where σ implements $\{\tilde{y}_1 = \tilde{y}_2\}$ and $z \notin \tilde{x}_1\tilde{x}_2 \Rightarrow \sigma(z) \notin \tilde{x}_1\tilde{x}_2$, and

$$\frac{P \xrightarrow{(\tilde{x})a^\varepsilon(\tilde{y}),L} P' \qquad z \notin a\tilde{y}}{(z)P \xrightarrow{(\tilde{x})a^\varepsilon(\tilde{y}),L} (z)P'} \qquad \frac{P \xrightarrow{\varphi,L} P' \qquad z \notin \varphi}{(z)P \xrightarrow{\varphi,L} (z)P'}$$

$$\frac{P \xrightarrow{(\tilde{x})a^\varepsilon(\tilde{y}),L} P' \qquad z \in \tilde{y},\ z \neq a}{(z)P \xrightarrow{(z\tilde{x})a^\varepsilon(\tilde{y}),L} P'} \qquad \frac{P \xrightarrow{\varphi,L} P' \qquad z \ \varphi \ y,\ z \neq y}{(z)P \xrightarrow{\varphi\setminus\{z\},L} P'[y/z]}$$

$$\frac{P \xrightarrow{(\tilde{x})a^\varepsilon(\tilde{y}),L} P'}{(\ell)P \xrightarrow{(\tilde{x})a^\varepsilon(\tilde{y}),L\setminus\{\ell\}} (\ell)P'} \qquad \frac{P \xrightarrow{\varphi,L} P'}{(\ell)P \xrightarrow{\varphi,L\setminus\{\ell\}} (\ell)P'}$$

$$\frac{P \xrightarrow{(\tilde{x})\alpha,L} P'}{P \mid Q \xrightarrow{(\tilde{x})\alpha,L} P' \mid Q[1/L]} \qquad \frac{P \xrightarrow{\varphi,L} P'}{P \mid Q \xrightarrow{\varphi,L} P' \mid Q[1/L]}$$

When an action $\ell : \alpha$ is performed, the name ℓ is replaced by 1 in the rest of the process. The name ℓ can be interpreted as that of a global variable (or

a semaphore) with a monotonic value, initially set to 0, which gets the value 1 on activation of α. Then a prefix is a monotonic combination of semaphores.

Definition 2.6 Bisimilarity on $\mathcal{C}$-terms is defined as follows:

- A process P has a *barb* on (u, L) if there is a transition $P \xrightarrow{(\tilde{x})u^{\varepsilon}(\tilde{y}),L} P'$ for some $\tilde{x}$, $\tilde{y}$, ε, and P'. We denote it $P \downarrow (u, L)$.
- A bisimulation is a symmetric binary relation $\mathcal{S}$ such that $P \mathcal{S} Q$ implies
  - for all u, L, if $P \downarrow (u, L)$ then $Q \downarrow (u, L)$,
  - for any transition $P \xrightarrow{\varphi,L} P'$ there is a Q' such that $Q \xrightarrow{\varphi,L} Q'$ and $P'\sigma \mathcal{S} Q'\sigma$ for some substitution σ that implements φ.
- A bisimulation $\mathcal{S}$ is *stable* if it is closed under arbitrary name substitution and if $P \mathcal{S} Q$ implies $P[1/L] \mathcal{S} Q[1/L]$ for all $L \subset \mathcal{L} \setminus (\mathrm{loc}(P) \cup \mathrm{loc}(Q))$.

Two processes are bisimilar if they are related by a bisimulation.

Without the last clause, our definition is a standard barbed bisimulation, except that the observation relation $\downarrow$ makes location names observable. Stable bisimilarity is more pertinent, because bisimilarity is not preserved under the effects of the context. For instance, let $P = \ell_1 : \bar{a}()$, $Q = \ell_1 : \bar{a}() \mid \langle m \rangle\, \ell_2 : a()$ and $R = m : \bar{c}() \mid c()$. P and Q are bisimilar since they have the same barbs and have no transition. However, $P \mid R$ and $Q \mid R$ are not: they have the same barbs and both have one transition labelled $(\mathrm{id}, \{m\})$, which leads to P in the case of $P \mid R$ and to $\ell_1 : \bar{a}() \mid \ell_2 : a()$ in the case of $Q \mid R$, and these reducts cannot be bisimilar since the former has no transition while the latter has one. So a stable bisimulation is a bisimulation that is preserved under the effects that the context may produce. Note that the condition above restricts L to be composed of locations that are *not defined* in P or Q: these locations may occur in the guards in P and Q, and they may be substituted by 1 because of transitions in the context, as illustrated in the previous example.

It also makes sense to define an associated notion of weak bisimulation, in which only observable transitions are considered. Observability here refers both to name fusion and location freeing, i.e. a transition $\xrightarrow{\varphi,L}$ is observable as soon as φ is not the identity or L is not empty.

Definition 2.7 Weak bisimilarity on $\mathcal{C}$-terms is defined as follows:

- The τ-reduction relation $\to$ is defined as $P \to P'$ iff $P \xrightarrow{\mathrm{id},\emptyset} P'$ where id stands for the identity relation $\{(x, x) \mid x \in \mathcal{C}\}$. The relation $\to^*$ is the reflexive transitive closure of $\to$.
- A weak bisimulation is a symmetric relation $\mathcal{S}$ such that $P \mathcal{S} Q$ implies
  - for all u, L, if $P \downarrow (u, L)$ then $Q \to^* \downarrow (u, L)$,
  - for any $P \xrightarrow{\varphi,L} P'$ with $(\varphi, L) \neq (\mathrm{id}, \emptyset)$, there is a Q' such that $Q \to^* \xrightarrow{\varphi,L} Q'$ and $P'\sigma \mathcal{S} Q'\sigma$ for some substitution σ that implements φ.

Two processes are weakly bisimilar if they are related by a weak bisimulation.

2.3 Graphical Syntax

The calculus of $\mathcal{C}$-terms has a large number of structural rules to define appropriate notions of locality and scope in processes. The following canonical form property yields a graphical formulation that avoids the need for such rules.

Proposition 2.8 *Any $\mathcal{C}$-term P is structurally equivalent to a term with the following shape, where the product stands for parallel composition:*

$$P \equiv (\tilde{w})(\tilde{m}) \prod_{i=1}^{n} \langle \pi_i \rangle \, \ell_i : u_i^{\varepsilon_i}(\tilde{x}_i) \quad \textit{with} \quad \pi_i \equiv \sum_{j=1}^{p_i} \ell_{i,j,1} \cdots \ell_{i,j,q_{i,j}}$$

for some $n \geqslant 0$, $p_i \geqslant 0$ and $q_{i,j} \geqslant 0$, where the $\ell_{i,j,k}$ are elements of $\{\ell_i \mid 1 \leqslant i \leqslant n\}$. The sets $\tilde{w}$ and $\tilde{m}$ represent respectively private channels and locations. Such a formulation is called an enumeration *of P.*

Proof. By scope extrusion, all binders may be moved to the top level of the syntax tree, and the distribution and composition rules for prefixes lead to the expression of P as a composition of elementary guarded actions. The standard form of guards is obtained by distributivity of conjunction over disjunction.□

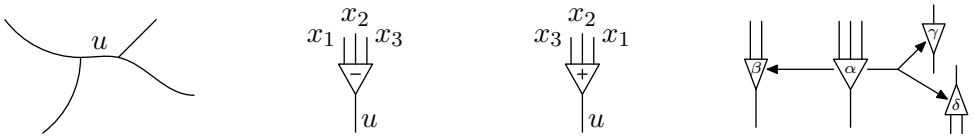
Hence, a process can be described as a set of locations, each with an associated action and prefix. Actions are built on a set of channel names, some of which are bound. Unbound channels form a set called the interface. The prefix of an action is either 1 or a disjunction of non-empty sets of locations. Prefixes define a relation: the enabling relation between non empty sets of locations (i.e. occurrences of actions) and actions, in the spirit of event structures. This leads to the following algebraic definition, where $\mathcal{C}^*$ stands for the set of finite sequences over $\mathcal{C}$ and $\mathcal{P}_0(\mathcal{A})$ is the set of non-empty subsets of $\mathcal{A}$ (the non-emptiness condition is justified after Defintion 2.10).

Definition 2.9 A concurrent net consists of

- a set $\mathcal{C}$ of *channels*,
- a subset $\mathcal{I}$ of $\mathcal{C}$ called the *interface*,
- a set $\mathcal{A}$ of *actions* labelled by elements of $\{+,-\} \times \mathcal{C} \times \mathcal{C}^*$,
- an *enabling* relation $\vdash$ between $\mathcal{P}_0(\mathcal{A})$ and $\mathcal{A}$.

In the sequel, channels are ranged over by Latin letters and actions are ranged over by Greek letters. A positive action $(+, u, \tilde{x})$ is written $\bar{u}(\tilde{x})$ and a negative action $(-, u, \tilde{x})$ is written $u(\tilde{x})$. In such an action, u is the *principal* channel and the elements of $\tilde{x}$ are the *auxiliary* channels.

Figure 2 shows the graphical conventions we use to represent concurrent nets: the channels are the edges in a hypergraph over actions, the channels in the interface are those with dangling edges. Actions are represented by triangles with the polarity in the middle, the principal channel (the subject) is connected to a vertex of the triangle and the auxiliary channels are con-



channel u | input $u(x_1x_2x_3)$ | output $\bar{u}(x_1x_2x_3)$ | enabling arrows $\beta \vdash \alpha\,;\, \gamma, \delta \vdash \alpha$

Fig. 2. Graphical syntax for concurrent nets.

nected to the opposite side. By convention, the auxiliary ports of negative actions are ordered from left to right (when looking from the principal channel) while those of positive actions are ordered from right to left, which leads to cleaner figures. The enabling relation is represented by arrows: for each element $\beta_1, \dots, \beta_n \vdash \alpha$, we draw a multi-headed arrow from α to each of the β_i. Different elements of $\vdash$ are represented by disjoint arrows. The arrows represent guards, so an action is enabled when there is no arrow leaving its node, and communication only occurs between enabled actions.

Definition 2.10 Let $P = (\mathcal{C}, \mathcal{I}, \mathcal{A}, \vdash)$ be a concurrent net.

- An action α is *enabled* if there is no set X such that $X \vdash \alpha$.
- A *redex* is a pair $\{\alpha, \beta\}$ of enabled actions of opposite polarities with the same subject and arity.
- P reduces along the redex $\{\alpha, \beta\}$ into the net P' obtained by removing α and β from the quotient of P by $\{\tilde{x} = \tilde{y}\}$. If there is an arrow $X \vdash \gamma$ in P with $X \subseteq \{\alpha, \beta\}$, then any arrow $Y \vdash \gamma$ is removed in P'. Otherwise, any arrow $X \vdash \gamma$ in P is replaced by $X \setminus \{\alpha, \beta\} \vdash \gamma$.

If an arrow $\emptyset \vdash \alpha$ appears in the reduction, then α gets enabled and all arrows $Y \vdash \alpha$ are removed, which corresponds to the axiom $1 + x = 1$. This axiom is precisely what is needed in Definition 2.11 to make the graphical formulation equivalent to $\mathcal{C}$-terms. An equivalent approach would be to allow empty sets on the left of $\vdash$ and to define that α is enabled when $\emptyset \vdash \alpha$.

As illustrated by Fig. 1, the reduction of a redex consists in removing the redex and connecting the auxiliary channels of the positive action with those of the negative action. The principal channel of the actions (u in the figure) is still present in the net, minus two actions. Any arrow that points to an action in the redex is removed, which possibly enables other actions.

Definition 2.11 The equivalence $\equiv$ over concurrent nets is the smallest equivalence such that a net with an arrow $X \vdash \alpha$ is equivalent to the same net plus an arrow $Y \vdash \alpha$ for any set of actions Y such that $X \subseteq Y$.

This equivalence is characterised by the operation that removes every arrow $Y \vdash \alpha$ for which there exists $X \vdash \alpha$ with $X \subseteq Y$. Two nets are structurally equivalent if their images by this transformation are isomorphic graphs.

Proposition 2.12 *There is an isomorphism, up to structural equivalence and renaming, between concurrent nets and $\mathcal{C}$-terms with no free location names, with their respective reductions.*

Proof. The canonical form property from Proposition 2.8 provides the translation between both formalisms: for a $\mathcal{C}$-term P, the associated net $[\![P]\!]$ is $(\mathcal{C}, \mathcal{I}, \mathcal{A}, \vdash)$ where $\mathcal{C}$ is the set of channels (bound or free), $\mathcal{I}$ is the set of free channels, $\mathcal{A}$ is the set of locations $\{\ell_i \mid 1 \leqslant i \leqslant n\}$ with ℓ_i labelled by $(\varepsilon_i, u_i, \tilde{x}_i)$. The enabling relation is defined as $\ell_{i,j,1}, \dots, \ell_{i,j,q_{i,j}} \vdash \ell_i$ for each pair (i, j). One easily checks that this translation is a bijection (up to structural equivalence) and that it commutes with reduction, in the sense that there is a translation labelled $P \xrightarrow{\varphi, L} P'$ if and only if $[\![P]\!]$ reduces into $[\![P']\!]$ with equivalence φ. □

As a consequence, in the sequel we refer to processes indifferently using the notations for terms or for nets, whichever is the more suitable.

3 Sub-calculi

Several process calculi can be considered as fragments of concurrent nets.

Definition 3.1 A calculus S is a sub-calculus of $\mathcal{C}$-terms if there is a translation map $t : S \to \mathcal{C}$, modulo the structural equivalences of S and $\mathcal{C}$, that is full and faithful (t is injective on processes and bijective on transitions). A correctness criterion is a characterisation of the image of t.

The solos calculus [8] without replication is a sub-calculus of $\mathcal{C}$-terms by the trivial translation $\alpha \mapsto (\ell)(\ell : \alpha)$. It is the fragment of $\mathcal{C}$-terms with no prefixes. Incidentally, solo diagrams [7] as defined by Laneve, Parrow and Victor are very similar to our graphical syntax, since the diagram for a term in the solos calculus is exactly the dual graph of the concurrent net for its translation, in the sense that the vertices and the edges in the diagram are, respectively, the edges and the nodes in the concurrent net.

The solos calculus is a fragment of the fusion calculus [12]. Fusion terms (without replication or sums) are defined by the grammar

$$P := \mathbf{0} \mid (P \mid P) \mid (x)P \mid \bar{u}(\tilde{x}).P \mid u(\tilde{x}).P$$

Define the translation $[\![\_]\!]_\pi$ by $[\![\alpha.P]\!]_\pi = (\ell)(\langle\pi\rangle\, \ell : \alpha \mid [\![P]\!]_\ell)$ where ℓ is a fresh location, and by homomorphism on all other constructs. The translation $[\![\_]\!]_1$ makes the fusion calculus a sub-calculus of $\mathcal{C}$-terms. The characterisation of the image of this translation requires the formal definition of prefixing:

Definition 3.2 Let P be a concurrent net.

- the *prefix* of an action α is the set $(\_ \vdash \alpha) := \{X \mid X \vdash \alpha\}$,
- P has a *simple prefixing* if for all α, either $(\_ \vdash \alpha) = \emptyset$ or there is a β such that $(\_ \vdash \alpha) = \{\{\beta\}\}$,

- if P has a simple prefixing, the *prefixing relation* of P is the relation $\leftarrow$ over the actions defined by $\beta \leftarrow \alpha$ when $\beta \vdash \alpha$.

Proposition 3.3 *The image of the translation of fusion calculus is the set of processes with simple prefixing in which the prefixing relation is acyclic.*

Proof. In this case the arrows form a directed acyclic graph, with nodes of degree at most 1 by hypothesis, i.e. a forest, which corresponds to the syntactical structure of the prefixes in the fusion term. □

The π-calculus may be seen as the sub-calculus of the fusion calculus where receptions always appear in the form $(\tilde{x})u(\tilde{x}).P$ where the x_i are distinct, i.e. where receptions are binders. Hence our translation of π-calculus is that of fusion calculus except that restrictions are added before every reception. The characterisation of translations of π-terms requires a formal definition of causality which means that one cannot interact on an unknown channel:

Definition 3.4 The causality relation is the binary relation $\Leftarrow$ over actions defined by $\beta \Leftarrow \alpha$ if the action β is negative and the principal channel of α or any of its auxiliary channels is an auxiliary channel of β.

Proposition 3.5 *The image of the translation of π-calculus is the set of processes with a simple prefixing such that the relation $\leftarrow$ is acyclic, the relation $\Leftarrow$ is included in the transitive closure of $\leftarrow$, and the auxiliary channels of negative actions are pairwise distinct.*

Proof. The condition on $\Leftarrow$ imposes that the auxiliary channels of receptions are used only in actions prefixed (possibly indirectly) by this reception, therefore the scope of received channels can always be written $(\tilde{x})u(\tilde{x}).P$. □

4 Encoding Guards

Our notion of guard provides a flexible extension of prefixing that extends fusion calculi with expressive scheduling features. Nevertheless, concurrent nets are still reasonable from an implementation point of view since they can be encoded into its fragment without guards, i.e. the solos calculus. The idea is to use the properties of fusion to translate explicit delaying of actions into delayed unification of names. The same kind of idea was used to encode fusion calculus into solos [8] but the actual encoding depended on the structure of prefixes. The one we provide here takes a more geometric approach.

Our encoding is composed of two steps, as illustrated in Fig. 3. The first step encodes an arbitrary process into a weakly bisimilar one with a prefixing of depth 1, similar to Parrow's *duos* [11]. The second step encodes a process with this simple prefixing into a strongly bisimilar one with no prefix.

The first step consists in associating a forwarder $(u)(u(\tilde{x}) \mid \bar{u}(\tilde{y}))$ to each node $\ell : \alpha$ that appears in another action's prefix. This forwarder is the one

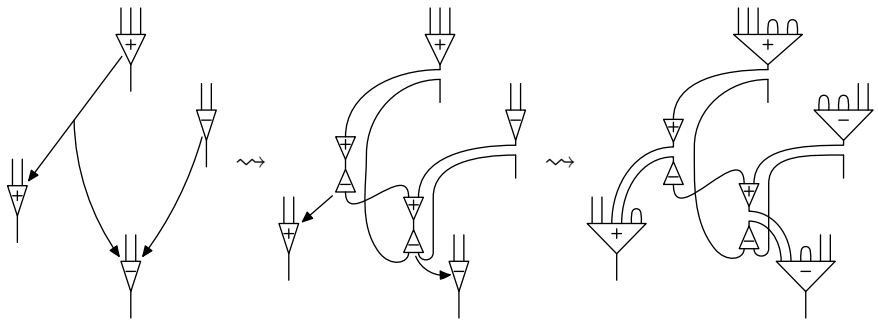


Fig. 3. The two steps of duo translation.

that will delay a fusion, so it is enough to prefix it with $\langle \ell \rangle$ to ensure that the concerned nodes are blocked.

Definition 4.1 Let P be a process. Assume a particular enumeration of P is chosen. For all i, let u'_i be u_i if $\pi_i \equiv 1$ and a fresh name otherwise. Let $(u_{i,j,k})$ be a family of channel names such that $u_{i,j,0} = u_i$ and $u_{i,j,q_{i,j}} = u'_i$ and all other $u_{i,j,k}$ are fresh. The duo translation of P for this enumeration is

$$(\tilde{w}') \prod_{I=1}^{n} (v)\big(\ell_I : {u'}_I^{\varepsilon_I}(\tilde{x}_I) \mid \langle \ell_I \rangle\, v(\tilde{y}_I) \mid \bar{v}(\tilde{z}_I)\big) \quad \text{with } \tilde{w}' = \tilde{w} \cup \{u_{i,j,k} \mid k \neq 0\}$$

where $\tilde{y}_I$ and $\tilde{z}_I$ are families indexed over $\{(i,j,k) \mid \ell_{i,j,k} = \ell_I\}$ defined as

$$y_{I,(i,j,k)} = u_{i,j,k-1} \qquad \text{and} \qquad z_{I,(i,j,k)} = u_{i,j,k}$$

The duo translation of P is the set $[\![P]\!]_d$ of duo translations of P for all possible enumerations, thus duo translation should be understood as a relation between processes rather than a function.

Theorem 4.2 *Every process P is weakly stably bisimilar to each $Q \in [\![P]\!]_d$.*

Sketch of proof. The simulation is achieved by reducing all the forwarders introduced by the translation when necessary. These redexes cannot interfere with any other communication, which ensures bisimilarity. □

Remark that if two processes P and Q have no public locations, then $[\![P \mid Q]\!]_d = [\![P]\!]_d \mid [\![Q]\!]_d$. The advantage of this intermediate form is that it uses a very restricted form of prefixing:

Definition 4.3 A process P has a duo prefixing if there is a partial injection p over $\{1 \dots n\}$ such that P can be written

$$P \equiv (\tilde{w}) \prod_{i=1}^{n} \langle \pi_i \rangle\, \ell_i : u_i^{\varepsilon_i}(\tilde{x}_i) \quad \text{with} \quad \pi_i = \begin{cases} \ell_{p(i)} & \text{if } i \in \mathrm{dom}(p) \\ 1 & \text{otherwise} \end{cases}$$

Lemma 4.4 *For every process P, every $Q \in [\![P]\!]_d$ has a duo prefixing.*

Definition 4.5 Let P be a process with a duo prefixing. Assume an enumeration of P with the notations of definition 4.3. Let (u'_i) be a family of fresh names indexed over the domain of p. The duo encoding of P for this enumeration is $(\tilde{w}\tilde{u}')\prod_{i=1}^{n}(yz)\ell_i : \alpha_i$ where

$$\alpha_i = \begin{cases} u^{\varepsilon_i}(yyzz\tilde{x}_i) & \text{if } i \notin \mathrm{rng}(p) \\ \bar{u}(yyu_{p^{-1}(i)}u'_{p^{-1}(i)}\tilde{x}_i) & \text{if } \varepsilon_i = + \\ u(u_{p^{-1}(i)}u'_{p^{-1}(i)}zz\tilde{x}_i) & \text{if } \varepsilon_i = - \end{cases} \quad \text{with} \quad u = \begin{cases} u'_i & \text{if } i \in \mathrm{dom}(p) \\ u_i & \text{otherwise} \end{cases}$$

The duo encoding $[\![P]\!]_p$ is the set of duo encodings of P for all enumerations.

Theorem 4.6 *Duo encoding is a strong stable bisimulation.*

Sketch of proof. The actions in a process and in any of its encodings are in one-to-one correspondence. The four extra arguments of each action in the translation simulate precisely the unblocking of locations. □

As a corollary of theorems 4.2 and 4.6, any process is weakly stably bisimilar to a process without guards, by composition of the encodings.

5 Conclusions

The calculus of concurrent nets extends the family of π-like calculi by introducing a new and more expressive form of sequentiality constraints. Prefixing is seen as the evolution of a shared state, in a way that recalls synchronisation mechanisms like semaphores. In its graphical form, the calculus makes clear the notions of locality and scoping and expresses prefixing as a form of interaction with a geometric intuition. We show, by means of an encoding, that in a sense the calculus without guards has the same expressive power as the calculus with guards. Hence, while keeping the same theoretical expressiveness, concurrent nets provide new programming features for concurrent calculi.

Interaction nets [6] are a notable example of a concurrent graphical calculus, and several features make them a quite different model, notably the fact that edges always have two vertices and the way replication is performed. Another interesting comparison could be made with proof-nets in linear logic [4], and these objects were a source of inspiration in this work. Using appropriate encodings, proof-nets may actually be considered as a very particular sub-calculus of concurrent nets. This will be detailed in a future paper.

The prefixes in the term calculus are arbitrary monotonic functions, built by conjunction and disjunction on monotonic variables. What makes their encoding possible is the fact that communication by fusion has the same monotonic flavour. The extension to non-monotonic guards, for instance by introducing negation, would strictly extend the expressiveness, actually it would be equivalent to introducing external sums in the calculus. The calculus can also be extended with a replication operator, and the encoding theorems still hold in this case. These extensions will be detailed in a full paper.

References

[1] Cédric Fournet and Georges Gonthier. The reflexive CHAM and the join-calculus. In *Proceedings of POPL'96*, pages 372–385. ACM Press, 1996.

[2] Yuxi Fu. A proof-theoretical approach to communication. In P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, editors, *Proceedings of ICALP'97*, volume 1256 of *Lecture Notes in Computer Science*, pages 325–335. Springer Verlag, 1997.

[3] Philippa Gardner and Lucian Wischik. Explicit fusions. In Mogens Nielsen and Branislav Rovan, editors, *Proceedings of MFCS 2000*, volume 1893 of *Lecture Notes in Computer Science*, pages 373–382. Springer Verlag, 2000.

[4] Jean-Yves Girard. Proof-nets: The parallel syntax for proof-theory. In Paolo Agliano and Aldo Ursini, editors, *Logic and Algebra*. M. Dekker, New York, 1996.

[5] Ole Høgh Jensen and Robin Milner. Bigraphs and transitions. *ACM SIGPLAN Notices*, 38(1):38–49, 2003.

[6] Yves Lafont. Interaction nets. In *Proceedings of POPL'90*, pages 95–108. ACM Press, 1990.

[7] Cosimo Laneve, Joachim Parrow, and Björn Victor. Solo diagrams. In Naoki Kobayashi and Benjamin C. Pierce, editors, *Proceedings of TACS'01*, volume 2215 of *Lecture Notes in Computer Science*, pages 127–144. Springer Verlag, 2001.

[8] Cosimo Laneve and Björn Victor. Solos in concert. In Jiří Wiederman, Peter van Emde Boas, and Mogens Nielsen, editors, *Proceedings of ICALP'99*, volume 1644 of *Lecture Notes in Computer Science*, pages 513–523. Springer Verlag, July 1999.

[9] Robin Milner. Pi-nets: A graphical form of pi-calculus. In Donald Sannella, editor, *Proceedings of ESOP'94*, volume 788 of *Lecture Notes in Computer Science*. Springer Verlag, 1994.

[10] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes (Parts I and II). *Information and Computation*, 100:1–77, 1992.

[11] Joachim Parrow. Trios in concert. In Gordon Plotkin, Colin Stirling, and Mads Tofte, editors, *Proof, Language and Interaction: Essays in Honour of Robin Milner*, pages 621–637. MIT Press, 1998.

[12] Joachim Parrow and Björn Victor. The fusion calculus: Expressiveness and symmetry in mobile processes. In *Proceedings of LICS'98*, pages 176–185, 1998.

A Appendix: Proofs

In this appendix, we detail the technical proofs that we did not include in the main text for lack of space.

A.1 Duo Translation

Note that many enumerations of a given process exist, since these depend on the order of the actions and the formulation of the guards. In particular, a guard $\pi = 1$ may as well be formulated $\pi = 1 + \pi'$ though these are structurally equivalent.

Definition A.1 Let $\tilde{x}$ and $\tilde{y}$ be two sequences of channel names with $|\tilde{x}| = |\tilde{y}|$. The delayed fusion of $\tilde{x}$ and $\tilde{y}$ is $F_{\tilde{x}=\tilde{y}} := (u)(u(\tilde{x}) \mid \bar{u}(\tilde{y}))$. A set of delayed fusions of type φ is a parallel composition of delayed fusions $F_\varphi = F_{\tilde{x}_1=\tilde{y}_1} \mid \cdots \mid F_{\tilde{x}_n=\tilde{y}_n}$ such that φ is the equivalence $\tilde{x}_1 \cdots \tilde{x}_n = \tilde{y}_1 \cdots \tilde{y}_n$.

Lemma A.2 *For any delayed fusion,* $F_{\tilde{x}=\tilde{y}} \xrightarrow{\{\tilde{x}=\tilde{y}\},\emptyset} \mathbf{0}$.

Proof. Obvious from the definition of the transition system. □

Definition A.3 Let $\mathcal{A}$ be a set of processes. The fusion expansion of $\mathcal{A}$ is the set $\mathcal{A}^{\exp} = \big\{(\tilde{x})(F_\varphi \mid P') \;\big|\; \mathrm{dom}(\varphi) \subset \tilde{x}, \forall \sigma : \varphi, \exists P \in \mathcal{A}, (\tilde{x})(P'\sigma) \equiv P\big\}$, where $\forall \sigma : \varphi$ means "for all substitution σ that implements φ."

Lemma A.4 *For any process P, for all $P' \in \{P\}^{\exp}$, $P' \to^* P$.*

Proof. By lemma A.2, the transitions of F_φ produce the effect of φ within the scope of $\tilde{x}$, and they are τ-transitions since all names that are affected by φ are hidden in P' by hypothesis. □

Proof of theorem 4.2. Let $\mathcal{S}$ be the relation over processes that relates each P to all the fusion expansions of its duo translations, i.e. $P \mathcal{S} Q$ for each $Q \in [\![P]\!]_d^{\exp}$. We prove that $\mathcal{S}$ and $\mathcal{S}^{-1}$ are weak simulations.

Let (P, Q) be a pair of processes related by $\mathcal{S}$. By definition, we have $Q = (\tilde{w}')(F_\chi \mid R)$ and for any substitution σ that implements χ, $(\tilde{w}')R\sigma$ is structurally equivalent to some duo translation of P. Pick such a substitution σ. Using the notations of proposition 2.8 and definition 4.1, there is an enumeration of P such that

$$(\tilde{w}')R\sigma \equiv \prod_{i=1}^{n} A_i \quad \text{with} \quad A_i = (v)\big(\ell_i : u_i'^{\varepsilon_i}(\tilde{x}_i) \mid \langle \ell_i \rangle\, v(\tilde{y}_i) \mid \bar{v}(\tilde{z}_i)\big).$$

In the following, for all i we write $\alpha_i = u_i^{\varepsilon_i}(\tilde{x}_i)$ in order to simplify the expression of the enumeration of P.

If a transition $P \xrightarrow{\varphi, L} P'$ exists, then there are two indices a and b such that $\varphi = \{\tilde{x}_a = \tilde{x}_b\} \setminus \tilde{w}$ and $L = \{\ell_a, \ell_b\}$ and the derivation of the transition

can be written

$$\dfrac{\dfrac{\dfrac{}{\ell_a : \alpha_a \xrightarrow{\alpha_a,\{\ell_a\}} \mathbf{0}} \quad \dfrac{}{\ell_b : \alpha_b \xrightarrow{\alpha_b,\{\ell_b\}} \mathbf{0}}}{\ell_a : \alpha_a \mid \ell_b : \alpha_b \xrightarrow{\{\tilde{x}_a=\tilde{x}_b\},L} \mathbf{0}}}{\prod_{i=1}^{n} \langle \pi_i \rangle\, \ell_i : \alpha_i \xrightarrow{\{\tilde{x}_a=\tilde{x}_b\},L} \prod_{i\notin\{a,b\}} \langle \pi_i[1/L] \rangle\, \ell_i : \alpha_i}}{P \xrightarrow{\varphi,L} (\tilde{w}) \prod_{i\notin\{a,b\}} \langle \pi_i[1/L] \rangle\, \ell_i : \alpha_i}$$

and the right-hand side of the last rule is structurally equivalent to P'. After the substitution $1/L$, the family $(\ell_{i,j,k})$ can be reindexed as $(m_{i,j,k})$ with $1 \leqslant i \leqslant n$, $1 \leqslant j \leqslant p_i$ and $1 \leqslant k \leqslant q'_{i,j}$ where $q'_{i,j}$ is $q_{i,j}$ minus the number of occurrences of ℓ_a or ℓ_b in $\ell_{i,j,1} \dots \ell_{i,j,q_{i,j}}$. This way we have for each i:

$$\pi_i[1/L] = \Big(\sum_{j=1}^{p_i} \ell_{i,j,1} \cdots \ell_{i,j,q_{i,j}}\Big)[1/L] = \sum_{j=1}^{p_i} m_{i,j,1} \cdots m_{i,j,q'_{i,j}}$$

which yields an enumeration of P'.

Since the actions at ℓ_a and ℓ_b are enabled, for $i \in \{a, b\}$ we have $u'_i = u_i$ and the following holds:

$$\dfrac{\dfrac{\dfrac{}{\ell_i : \alpha_i.v(\tilde{y}_i) \xrightarrow{\alpha,\{\ell_i\}} v(\tilde{y}_i)}}{\ell_i : \alpha_i \mid \langle \ell_i \rangle\, v(\tilde{y}_i) \mid \bar{v}(\tilde{z}_i) \xrightarrow{\alpha_i,\{\ell_i\}} v(\tilde{y}_i) \mid \bar{v}(\tilde{z}_i)}}{A_i \xrightarrow{\alpha_i,\{\ell_i\}} (v)(v(\tilde{y}_i) \mid \bar{v}(\tilde{z}_i))}$$

and using the notation of definition A.1, with $\psi = \{\tilde{y}_a = \tilde{z}_a, \tilde{y}_b = \tilde{z}_b\}$ and $F_\psi = F_{\tilde{y}_a=\tilde{z}_a} \mid F_{\tilde{y}_b=\tilde{z}_b}$:

$$\dfrac{\dfrac{\dfrac{A_a \xrightarrow{\alpha_a,\{\ell_a\}} F_{\tilde{y}_a=\tilde{z}_a} \qquad A_b \xrightarrow{\alpha_b,\{\ell_b\}} F_{\tilde{y}_b=\tilde{z}_b}}{A_a \mid A_b \xrightarrow{\{\tilde{x}_a=\tilde{x}_b\},L} F_\psi}}{R\sigma \equiv \prod_{j=1}^{n} A_j \xrightarrow{\{\tilde{x}_a=\tilde{x}_b\},L} F_\psi \mid \prod_{i\notin\{a,b\}} A_i}}{(\tilde{w}')R\sigma \xrightarrow{\varphi,L} (\tilde{w}') \Big(F_\psi \mid \prod_{i\notin\{a,b\}} A_i\Big)}$$

Note that the substitution $1/L$ is not necessary in the right-hand sides since the locations ℓ_a and ℓ_b do not occur in F_ψ nor in any A_i with $i \notin \{a, b\}$, by construction. Call $R' = (\tilde{w}')(F_\psi \mid Q')$ the reduct in the last line. By definition, $\tilde{y}_a$ and $\tilde{z}_a$ are families indexed over $\{(i, j, k) \mid \ell_{i,j,k} = \ell_a\}$ defined as

$$y_{a,(i,j,k)} = u_{i,j,k-1} \qquad \text{and} \qquad z_{a,(i,j,k)} = u_{i,j,k}$$

so the fusion $\tilde{y}_a = \tilde{z}_a$ unifies each pair $(u_{i,j,k-1}, u_{i,j,k})$ such that $\ell_{i,j,k} = \ell_a$. The same goes for $\tilde{y}_b$ and $\tilde{z}_b$. By lemma A.4, if τ is a substitution that implements

$\psi = \{\tilde{y}_a = \tilde{z}_a, \tilde{y}_b = \tilde{z}_b\}$, we have

$$R' = (\tilde{w}')\Big(F_\psi \mid \prod_{i\notin\{a,b\}} A_i\Big) \to^* (\tilde{w}') \prod_{i\notin\{a,b\}} A_i\tau \tag{A.1}$$

Besides, $(u_{i,j,k}\tau)$ can be reindexed as $(v_{i,j,k})$ with the same index transformation that transformed $(\ell_{i,j,k})$ into $(m_{i,j,k})$. Define v'_i to be $v_i = u_i$ if $\pi_i[1/L] \equiv 1$ and u'_i otherwise. With these notations we get for each $i \notin \{a, b\}$:

$$A_i\tau = (w)\big(\ell_i : v_i'^{\varepsilon_i}(\tilde{x}_i\tau) \mid \langle\ell_i\rangle\, w(\tilde{y}'_i) \mid \bar{w}(\tilde{z}'_i)\big)$$

Then the process $(\tilde{w}')\prod_{i\notin\{a,b\}} A_i\tau$ is the duo translation of P' for the enumeration we get by reindexing. By equation A.1, we deduce that $R' \in [\![P']\!]_d^{\exp}$, and since $Q \to^* R\sigma \to R'$ we have that $\mathcal{S}$ is a weak simulation.

We now have to prove that $\mathcal{S}^{-1}$ is also a weak simulation. Again, consider a pair (P, Q) in $\mathcal{S}$, with $Q = (\tilde{w}')(F_\chi \mid R)$, and where R is decomposed as

$$R \equiv \prod_{i=1}^{n} A_i \quad \text{with} \quad A_i = (v)\big(\ell_i : u_i'^{\varepsilon_i}(\tilde{x}_i) \mid \langle\ell_i\rangle\, v(\tilde{y}_i) \mid \bar{v}(\tilde{z}_i)\big)$$

If a reduction $Q \xrightarrow{\varphi,L} Q'$ exists, it affects either a redex in F_χ or a redex in R, since the subjects of actions in F_χ are each shared between exactly two opposite actions.

If the reduction happens in F_χ, it concerns a delayed fusion $F_{\tilde{x}=\tilde{y}}$ and we have $Q' = (\tilde{w}')(F_{\chi'} \mid R)\sigma$ where σ is a substitution that implements $\tilde{x} = \tilde{y}$ and χ' is the equivalence generated by what remains of F_χ. Therefore both Q and Q' are in the fusion expansion of $[\![P]\!]_d$, which means that $P\ \mathcal{S}\ Q'$.

If the reduction happens in R, there exist a and b such that u'_a and u'_b are the same channel u, $\varepsilon_a = +$, $\varepsilon_b = -$ and thus $\varphi = \{\tilde{x}_a = \tilde{x}_b\} \setminus \tilde{w}'$ and $L = \{\ell_a, \ell_b\}$. Then if we write $\psi = \{\tilde{y}_a = \tilde{z}_a, \tilde{y}_b = \tilde{z}_b\}$ and $F_\psi = F_{\tilde{y}_a=\tilde{z}_a} \,|\, F_{\tilde{y}_b=\tilde{z}_b}$, the derivation of the reduction can be written

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A_a \xrightarrow{u(\tilde{x}_a),\{\ell_a\}} F_{\tilde{y}_a=\tilde{z}_a}} \quad \dfrac{}{A_b \xrightarrow{\bar{u}(\tilde{x}_b),\{\ell_b\}} F_{\tilde{y}_b=\tilde{z}_b}}}{A_a \mid A_b \xrightarrow{\{\tilde{x}_a=\tilde{x}_b\},L} F_\psi}}{R \xrightarrow{\{\tilde{x}_a=\tilde{x}_b\},L} F_\psi \mid \prod_{i\notin\{a,b\}} A_i}}{Q \xrightarrow{\varphi,L} (\tilde{w}')\big(F_\chi \mid F_\psi \mid \prod_{i\notin\{a,b\}} A_i\big)\tau \equiv Q'}$$

for some substitution τ that implements $\{\tilde{x}_a = \tilde{x}_b\} \setminus \tilde{w}'$. By definition of the translation, the names u'_i in a duo translation are fresh and pairwise distinct except for those that are equal to some u_i, i.e. those for which $\pi_i \equiv 1$. If σ is a substitution that implements χ, $(\tilde{w}')R\sigma$ is a duo translation of P, so $u'_a\sigma = u'_b\sigma$ does imply that $\pi_a \equiv \pi_b \equiv 1$, and the actions at locations ℓ_a and

ℓ_b in P are enabled. As a consequence, the following reduction holds:

$$\dfrac{\dfrac{\dfrac{}{\ell_a : u_a(\tilde{x}_a) \xrightarrow{u(\tilde{x}_a),\{\ell_a\}} \mathbf{0}} \quad \dfrac{}{\ell_b : \bar{u}_b(\tilde{x}_b) \xrightarrow{\bar{u}(\tilde{x}_b),\{\ell_b\}} \mathbf{0}}}{\ell_a : \alpha_a \mid \ell_b : \alpha_b \xrightarrow{\{\tilde{x}_a=\tilde{x}_b\},\{\ell_a,\ell_b\}} \mathbf{0}}}{\prod_{i=1}^{n} \langle \pi_i \rangle\, \ell_i : \alpha_i \xrightarrow{\{\tilde{x}_a=\tilde{x}_b\},\{\ell_a,\ell_b\}} \prod_{i\notin\{a,b\}} \langle \pi_i[1/\ell_a, \ell_b] \rangle\, \ell_i : \alpha_i}}{P \xrightarrow{\varphi,L} \prod_{i\notin\{a,b\}} \langle \pi_i[1/L] \rangle\, \ell_i : \alpha_i\tau \equiv P'}$$

with the same substitution τ as above. It can be verified, with the same arguments as in the first part, that Q' is a fusion expansion of a duo translation of the reduct P', and therefore we have $P \to P'$ and $P' \mathrel{\mathcal{S}} Q'$.

This proves that $\mathcal{S}^{-1}$ is also a weak simulation, therefore the symmetric closure $\mathcal{S} \cup \mathcal{S}^{-1}$ is a weak bisimulation.

The set of barbs (u, L) such that $P \downarrow (u, L)$ is the set of all $(u_i, \{\ell_i\})$ such that $\pi_i \equiv 1$ and $u_i \notin \tilde{w}$. The set of (u, L) such that $(\tilde{w}')R\sigma \downarrow (u, L)$ is the set of $(u'_i, \{\ell_i\})$ where u'_i does not appear in $\tilde{w}'$, that is by definition when u'_i is some u_j that does not appear in $\tilde{w}$. Therefore $P \downarrow (u, L)$ if and only if $(\tilde{w}')R\sigma \downarrow (u, L)$. Moreover, by definition of the fusion expansion, the free channels in Q are those in $(\tilde{w}')R\sigma$, so $(\tilde{w}')R\sigma \downarrow (u, L)$ if and only if $Q \to^* \downarrow (u, L)$. Therefore $\mathcal{S}$ is a weak barbed simulation.

Moreover, it is clear from the definition of the translation that it commutes with arbitrary name substitution since the free names in a process and its translations are the same. Since all locations are supposed private in the definition of the duo translation, the condition on substitution of free locations by 1 in the definition of stable bisimulation is empty, so we have a weak stable bisimulation, which concludes the proof of Theorem 4.2. □

A.2 Duo Encoding

Proof of theorem 4.6. Let $\mathcal{S}$ be the relation over processes that relates each P to every element of $[\![P]\!]_p$. Let (P, Q) be an element of $\mathcal{S}$. We use the notations of definitions 4.3 and 4.5.

If a transition $P \xrightarrow{\varphi,L} P'$ exists, then there are two indices a and b such that u_a and u_b are the same channel u, $\varphi = \{\tilde{x}_a = \tilde{x}_b\} \setminus \tilde{w}$ and $L = \{\ell_a, \ell_b\}$ and the derivation of the transition can be written

$$\dfrac{\dfrac{\dfrac{}{\ell_a : u(\tilde{x}_a) \xrightarrow{u(\tilde{x}_a),\{\ell_a\}} \mathbf{0}} \quad \dfrac{}{\ell_b : \bar{u}(\tilde{x}_b) \xrightarrow{\bar{u}(\tilde{x}_b),\{\ell_b\}} \mathbf{0}}}{\ell_a : u(\tilde{x}_a) \mid \ell_b : \bar{u}(\tilde{x}_b) \xrightarrow{\{\tilde{x}_a=\tilde{x}_b\},L} \mathbf{0}}}{\prod_{i=1}^{n} \langle \pi_i \rangle\, \ell_i : \alpha_i \xrightarrow{\{\tilde{x}_a=\tilde{x}_b\},L} \prod_{i\notin\{a,b\}} \langle \pi_i[1/L] \rangle\, \ell_i : \alpha_i}}{P \xrightarrow{\varphi,L} (\tilde{w}) \prod_{i\notin\{a,b\}} \langle \pi_i[1/L] \rangle\, \ell_i : \alpha_i}$$

and the right-hand side of the last rule is structurally equivalent to P'. Because

of the form of the guards π_i, the substitution $1/L$ may affect at most two guards by replacing them by 1.

The form of the corresponding transition in Q depends on whether a and b are in the range of the partial injection p, which means that four cases have to be considered. The symptomatic case is when both a and b are in the range of p, the other cases are just simpler.

Let a' and b' be the indices such that $a = p(a')$ and $b = p(b')$. Since the actions at ℓ_a and ℓ_b are enabled in P, $\pi_a = \pi_b = 1$ so a' and b' are distinct from a and b. Therefore Q can be written

$$Q \equiv (\tilde{w}\tilde{u}')\Big((z)\ell_a : u(u_{a'}u'_{a'}zz\tilde{x}_a) \;\Big|\; (y)\ell_b : u(yyu_{b'}u'_{b'}\tilde{x}_b) \;\Big|\; \prod_{i\notin\{a,b\}} A_i\Big)$$

Writing A_a and A_b for the first two actions, the following reduction holds:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A_a \xrightarrow{(z)u(u_{a'}u'_{a'}zz\tilde{x}_a),\{\ell_a\}} \mathbf{0}} \quad \dfrac{}{A_b \xrightarrow{(y)u(yyu_{b'}u'_{b'}\tilde{x}_b),\{\ell_b\}} \mathbf{0}}}{A_a \mid A_b \xrightarrow{\{u_{a'}=u'_{a'},u_{b'}=u'_{b'},\tilde{x}_a=\tilde{x}_b\},L} \mathbf{0}}}{\prod_{i=1}^n A_i \xrightarrow{\{u_{a'}=u'_{a'},u_{b'}=u'_{b'},\tilde{x}_a=\tilde{x}_b\},L} \prod_{i\notin\{a,b\}} A_i}}{(\tilde{u}')\prod_{i=1}^n A_i \xrightarrow{\{\tilde{x}_a=\tilde{x}_b\},L} (\tilde{u}')\prod_{i\notin\{a,b\}} A_i\sigma}}{Q \xrightarrow{\varphi,L} (\tilde{w}\tilde{u}')\prod_{i\notin\{a,b\}} A_i\sigma}$$

with $\sigma = [u_{a'}/u'_{a'}, u_{b'}/u'_{b'}]$. Call Q' the reduct in the last rule. The substitution σ affects only the actions at indices a' and b' since the channels $u'_{a'}$ and $u'_{b'}$ appear only in these as subjects, and appear only in A_a and A_b as objects, by injectivity of p. Substituting $u_{a'}$ for $u'_{a'}$ in the encoding of P corresponds to substituting 1 for $\pi_{a'}$ in P, i.e. the effect of σ in Q' is exactly the encoding of the effect of $1/L$ in P', therefore we have $Q \xrightarrow{\varphi,L} Q'$ with $P' \mathrel{\mathcal{S}} Q'$.

In the other cases, where at most one of a and b is in the range of p, the proof of simulation is similar, and as a consequence $\mathcal{S}$ is a simulation.

If a transition $Q \xrightarrow{\varphi,L} Q'$ exists, it affects a pair of actions of indices a and b with the same subject. By construction, this means that this subject is $u_a = u_b$ since all u'_i are fresh and used exactly once as subjects, and thus a and b are not in the domain of p, so the actions at indices a and b in P are enabled. Then it is easy to check that the reduction of this redex in P leads to a transition $P \xrightarrow{\varphi,L} P'$ with the same label, and by the same arguments as above Q' is a duo encoding of P', so $P' \mathrel{\mathcal{S}} Q$. This proves that $\mathcal{S}^{-1}$ is also a strong bisimulation, so the reflexive closure $\mathcal{S} \cup \mathcal{S}^{-1}$ is a strong bisimulation.

Since the public names and enabled actions in P and Q are the same, it is clear that $P \downarrow (u, L)$ if and only if $Q \downarrow (u, L)$ and that $\mathcal{S}$ is closed under arbitrary name substitution, so we have a strong stable bisimulation. □