

ON THE EXPRESSIVE POWER OF POLYADIC SYNCHRONISATION IN π -CALCULUS

MARCO CARBONE

BRICS, University of Aarhus, Department of Computer Science
Ny Munkegade, building 540, DK-8000 Århus C
Denmark
carbonem@brics.dk*

SERGIO MAFFEIS

*Imperial College London, Department of Computing
Huxley Building, 180 Queen's Gate, London SW7 2BZ
United Kingdom
maffeis@doc.ic.ac.uk*

Abstract. We extend the π -calculus with *polyadic synchronisation*, a generalisation of the communication mechanism which allows channel names to be composite. We show that this operator embeds nicely in the theory of π -calculus, we suggest that it permits divergence-free encodings of distributed calculi, and we show that a limited form of polyadic synchronisation can be encoded weakly in π -calculus. After showing that matching cannot be derived in π -calculus, we compare the expressivity of polyadic synchronisation, mixed choice and matching. In particular we show that the degree of synchronisation of a language increases its expressive power by means of a separation result in the style of Palamidessi's result for mixed choice.

ACM CCS Categories and Subject Descriptors: F.1.2 [Computation by Abstract Devices]: Modes of Computation – *parallelism and concurrency*

Key words: π -calculus, expressivity, matching, polyadic synchronisation, distributed systems

1. Introduction

Process calculi provide a useful framework in which to reason about the theory of concurrent and distributed systems. They are praised both for great simplicity and expressiveness. The π -calculus of Milner *et al.* [1992] is a terse and powerful language which describes the behaviour of concurrent systems, and is endowed with a rich body of theoretical results. However, evidence has been accumulated which suggests that it is inadequate to express certain aspects of *distributed* systems. In fact, the literature is rich with extensions of the π -calculus explicitly designed for modeling such systems.

*BRICS: Basic Research in Computer Science (www.brics.dk), funded by the Danish National Research Foundation.

Following *Occam's razor* principle¹, we study a minimal extension of the π -calculus in which to express concepts relevant to distributed systems such as *localities* and *encryption*. We propose ${}^e\pi$: the π -calculus with *polyadic synchronisation*, a generalisation of the synchronisation mechanism which allows channel names to be composite. The idea is that the subject of an input or output action is no longer restricted to be a single name, but is now a vector of names. For example we allow prefixes such as $a \cdot b(x)$ and $\overline{a \cdot b}\langle v \rangle$, where the channel is identified by vector $a \cdot b$. It turns out that this construct cannot be encoded in π -calculus without introducing divergence.

1.1 Examples of polyadic synchronisation

To justify our proposal, we introduce concrete scenarios and theoretical issues where polyadic synchronisation turns out to be helpful.

1.1.1 Two practical problems

The first example is $ED\pi$ [Carbone *et al.* 2001], where we use a construct which represents atomic transactions as a means to model e-services. Interaction between a client and a server takes place only if both parties agree on a set of service parameters. The second example is the calculus of objects (CO) of Vasconcelos and Tokoro [1993], upon which the TyCO programming language is based [Vasconcelos 1994]. It models distributed object systems where messages are dispatched to a certain object if and only if it provides the method invoked in the request, and it is ready to execute it. Its semantics requires agreement on both object identity and method name, in order for a call to be dispatched.

Both these examples can be generalised as instances of the problem of matching *atomically* vectors of values among different processes (the *matching problem*). The solution consists in bringing the values to be matched directly in the *interface* of each process towards the system, and providing semantic rules that allow interaction if and only if those interfaces are compatible.

1.1.2 Modeling locations

The ability of synchronising on many names at the same time, allows for *localities* to be represented in ${}^e\pi$. Many distributed calculi refer to an explicit notion of location, intended as a unit of distribution where computation takes place. Some of them are presented as extensions of the π -calculus, and are based on the idea that processes running in parallel inside some location can independently migrate

¹ “*Occam's razor* is a logical principle attributed to the medieval philosopher William of Occam (or Ockham). The principle states that one should not make more assumptions than the minimum needed. This principle is often called the principle of parsimony. It underlies all scientific modeling and theory building. It admonishes us to choose from a set of otherwise equivalent models of a given phenomenon the simplest one. In any given model, Occam's razor helps us to “shave off” those concepts, variables or constructs that are not really needed to explain the phenomenon. By doing that, developing the model will become much easier, and there is less chance of introducing inconsistencies, ambiguities and redundancies.” (F. Heylighen).

or communicate with other processes, locally or remotely. Examples of languages of such kind are to be found in [Hennessy and Riely 1998, Amadio *et al.* 1999, Cardelli and Gordon 2000, Sewell *et al.* 1999, and Fournet 1998]. From a pragmatic point of view, it emerged clearly that these models were to some extent more appropriate than the π -calculus to describe physical distribution. From a theoretical point of view, the *necessity* of these variants has not been fully explored. Take the Distributed π -calculus of Hennessy and Riely [1998] (where π -like processes are explicitly enclosed in locations), as a paradigmatic example. Its main reduction rule states that communication among two processes can take place only when they are in the same location:

$$\text{(RCOMM)} \quad l[a\langle v \rangle.P] \mid l[a(x).Q] \mapsto l[P] \mid l[Q\{v/x\}]$$

Our point is that a location can be seen as a name characterising all the interactions in which a process participates: hence it can be modeled as an additional synchronisation parameter in all the communications of a located process. Migration is simply the dynamic (re)binding of the location component of each prefix. For example the result of encoding the $D\pi$ network

$$l[\bar{a}\langle m \rangle.P \mid a(x).\text{go } x.\bar{b}\langle v \rangle.Q] \mid m[b(y).R]$$

is a process in this extension of π -calculus (${}^e\pi$), where the migration construct $\text{go } x$ disappears and the three threads of execution are run in parallel:

$$\bar{l}\cdot\bar{a}\langle m \rangle.P \mid l\cdot a(x).\bar{x}\cdot\bar{b}\langle v \rangle.Q \mid m\cdot b(y).R.$$

Note that rule (RCOMM) reported above, stating that two processes are allowed to react if and only if they share two values (location and channel) at the same time, is another instance of the matching problem. Another example of how localities can be expressed in terms of polyadic synchronisation is in [Carbone and Maffei 2002], where we give a divergence-free encoding of the Local Area π -calculus of Chothia and Stark [2001] in ${}^e\pi$.

1.1.3 Partial restriction and matching

Polyadic synchronisation also enhances the π -calculus in that it allows for *partial restriction*; that is, it gives the ability to restrict only some of the names taking part in a communication. It turns out that thanks to this feature, matching can be expressed as a special form of communication, and therefore is not given as primitive in ${}^e\pi$. Partial restriction allows us also to model cryptographic protocols, as explained below.

1.1.4 Modeling cryptography

We claim that ${}^e\pi$ can express an interesting class of security protocols. In fact, it is possible to use polyadic synchronisation to encode *secure channels*: the sending along public channel a of datum m encrypted under key k is expressed as $\bar{a}\cdot\bar{k}\langle m \rangle.P$,

implying that m can be received only by an agent knowing the secret password k , beside the public name a . With respect to the Spi-calculus of Abadi and Gordon [1998], this first solution lacks for example the power of expressing keys obtained by hashing data. Consider now a different, more expressive way to represent encryption in ${}^e\pi$. We propose constructs for encrypting and decrypting data, such that encrypted messages are represented as *names* (therefore can still be encrypted, sent, or used as keys), and encryption is nondeterministic (encrypting the same message under the same key two times yields different results). These constructs are:

$$\begin{aligned} \llbracket \text{encrypt } m \xrightarrow{k} x \text{ in } P \rrbracket &= (\nu x)(!\overline{x \cdot k} \langle m \rangle \mid P) \\ \llbracket \text{decrypt } x \xrightarrow{k} m \text{ in } P \rrbracket &= x \cdot k(m).P \end{aligned}$$

The first construct encrypts data m under key k and returns the encrypted message as the fresh name x , to be used in all the scope embraced by P . Decryption of message x through key k binds name m in the continuation P to the original message provided that the key is the same as the one used to encrypt it. Note how x , the result of the encryption, is a restricted name whose scope is process P . This is not a limitation because the scope of x can be extended in the standard way using extrusion, allowing modularity in the definition of processes. For example, the process *System* below evolves to a state where R and S share the classified data m , and that A cannot compromise the security of the protocol.

$$\begin{aligned} \text{Receiver} : & \quad (\nu k) \overline{\text{secure}} \langle k \rangle . \text{public}(y) . \text{decrypt } y \xrightarrow{k} w \text{ in } R \\ \text{Sender} : & \quad (\nu m) \text{secure}(z) . \text{encrypt } m \xrightarrow{z} x \text{ in } \overline{\text{public}} \langle x \rangle . S \\ \text{System} : & \quad (\nu \text{secure})(\text{Sender} \mid \text{Receiver}) \mid A \end{aligned}$$

1.2 Overview of the paper

We investigate the expressivity of polyadic synchronisation, using the π -calculus as a general framework. We are interested in finding out whether it must be assumed as primitive or it can be derived, and in understanding how it relates to the other operators of the calculus.

We recall the syntax and the semantics of π -calculus and of some important sub-calculi in Sections 2.1 and 2.2. In Section 2.3 we propose the notion of *sensible encoding* as a subset of the requirements that the encoding of an operator in a language should satisfy in order to be considered meaningful. We actually strengthen the notion of *reasonable encoding* proposed by Palamidessi [2002]: we require a stronger form of uniformity, namely that the encoding respects general substitutions, and we refine the notion of *reasonable* semantics to distinguish deadlocks from livelocks.

We introduce ${}^e\pi$ in Section 3, and in Section 3.2 we show how a restricted form of polyadic synchronisation can be encoded in π -calculus, at the price of introducing divergence.

Section 4 starts with the answer to an open question: we show that matching enhances the expressive power of the π -calculus². In ${}^e\pi$, matching can be encoded up-to strong bisimulation congruence. Also mixed choice is known to increase the expressiveness of π -calculus: we extend the results of Palamidessi [2002] to ${}^e\pi$, showing that polyadic synchronisation does not have the power to encode mixed choice. In Section 4.3 we show that polyadic synchronisation cannot be encoded in the full π -calculus (and therefore it is orthogonal to mixed choice) without introducing divergence, and we generalise the result to higher degrees of synchronisation.

In Section 5 we conclude showing how the π -calculi that we have considered are partially ordered by expressivity, and constitute a complete lattice.

1.3 Previous research related to polyadic synchronisation

Ferrari [1997] extends CCS with composite prefixes in order to model transactions, using a mixed form of polyadic synchronisation and synchronisation between multiple parties. Boudol and Castellani [1988] have studied the impact of considering finite computations as atomic steps on concurrent languages semantics, yet it seems that polyadic synchronisation is not expressible in their framework. Nestmann [1998] has studied the expressive power of the *joint input*, a liberalisation of the *join patterns* of Fournet and Gonthier [1996], in the π -calculus framework: it can be seen as a form of *bi-adic* synchronisation for input processes only. The main reduction rule for joint input is

$$\bar{a}\langle c \rangle \mid \bar{b}\langle d \rangle \mid \{a(x) \mid b(y)\}.P \searrow P\{c/x\}\{d/y\}$$

where $\{a(x) \mid b(y)\}.P$ could be seen as similar to the ${}^e\pi$ process $a \cdot b(x, y).P$. As this example shows, there is a fundamental difference in the way of expressing outputs in the two languages: where the joint input requires multi-way synchronisation, ${}^e\pi$ maintains the interaction confined to two processes. For this reason, joint input does not provide a solution to the matching problem.

Appendix A of Abadi and Gordon [1998] mentions synchronisation on tuples to point out that π -calculus could be made more resistant to security attacks, but the subject is not developed any further. Milner [1991] considered a form of multi-way synchronisation, superseding both polyadic synchronisation and joint input, that raised many interesting but non trivial questions on the theory. Our independent development is strongly biased towards enhancing the expressive power of π -calculus without introducing significant changes in the underlying equational theory.

² Although its usefulness has been challenged in the literature, for example by Merro and Sangiorgi [1998].

2. Preliminaries

2.1 The π -calculus

The π -calculus [Milner *et al.* 1992] is a formalism to describe concurrent execution of communicating processes based on the idea, inherited from CCS, of synchronising over named channels. We introduce briefly the syntax and semantics of the mixed-choice π -calculus with matching (π for short). Compared with the original formulation of the calculus, it drops the full-choice construct in favour of the more well-behaved mixed choice, as found for example in [Sangiorgi and Walker 2001].

Given a countable set of names \mathcal{N} ranged over by a, b, c, x, y, z, w , the syntax of π is defined as follows:

$$\begin{aligned} \text{(PROCESSES)} \quad P &::= 0 \mid P \mid P \mid (\nu x)P \mid !P \mid [x = y]P \mid \Sigma_i \alpha_i.P_i \\ \text{(PREFIXES)} \quad \alpha &::= \tau \mid x(y) \mid \bar{x}\langle y \rangle \end{aligned}$$

The prefix α represents the basic operations of the calculus: $x(y)$ is an input, $\bar{x}\langle y \rangle$ is an output and τ is an internal evolution step. The process $\Sigma_i \alpha_i.P_i$ represents mixed guarded choice. The notation $\Pi_{1..n}P_i$ is a shorthand for polyadic parallel composition $P_1 \mid \dots \mid P_n$. Sums and products are usually finite in π -calculus. The process 0 stands for the inactive process, (νx) and $!$ are respectively restriction and replication. The matching operator $[x = y]P$ behaves like P if x is equal to y , and behaves like 0 otherwise. Where necessary we will write prefixes of *pure* synchronisation in the style of CCS: $\bar{x}.P$ will be a shorthand for $\bar{x}\langle y \rangle.P$ for some y , whereas $x.P$ will stand for $x(y).P$ where y does not appear in P .

The *structural congruence* relation \equiv states when two processes are to be considered syntactically equivalent, and is defined as the least congruence satisfying alpha conversion, the commutative monoidal laws with respect to both $(\mid, 0)$ and $(+, 0)$ and the following axioms:

$$\begin{aligned} (\nu x)P \mid Q &\equiv (\nu x)(P \mid Q) \text{ if } x \notin fn(Q); \\ (\nu x)P &\equiv P \text{ if } x \notin fn(P) \\ [x = x]P &\equiv P. \end{aligned}$$

2.1.1 Semantics

The semantics is given in terms of a labeled transition system (*lts* for short). The actions of the *lts* are the prefixes plus the output of a restricted name $\bar{x}\langle \nu y \rangle$, as reported below.

$$\text{(ACTIONS)} \quad \mu ::= \alpha \mid \bar{x}\langle \nu y \rangle$$

The *subject* of an input or output action is the channel used for communication and the *object* is the parameter. Given an action μ , we recall the notion of its *free names*, *bound names*, and *names* (respectively $fn(\mu)$, $bn(\mu)$, and $n(\mu) = fn(\mu) \cup bn(\mu)$).

μ	$fn(\mu)$	$bn(\mu)$
$x(y), \bar{x}\langle \nu y \rangle$	$\{x\}$	$\{y\}$
$\bar{x}\langle y \rangle$	$\{x, y\}$	\emptyset
τ	\emptyset	\emptyset

Functions fn and bn are extended in the usual way to processes, in particular recalling that $fn((vx)P) \triangleq fn(P) \setminus \{x\}$. A *substitution* is a partial function $\sigma : \mathcal{N} \rightarrow \mathcal{N}$. Given a process P , we will write $P\sigma$ for the capture-avoiding application of σ to the free names of P . Notation $P\{-/-\}$ is equivalent to $P\sigma$ for σ defined only on a single name.

In Table 2.1 we report the *late* lts semantics for π . We omit the symmetric rules for (COMM), (CLOSE) and (PAR). In order to simplify technical passages we adopt in the lts the redundant rule (STRUCT) that accounts for structural congruence.

TABLE 2.1: Labeled transition system for π .

$\frac{}{\Sigma_i \alpha_i. P_i \xrightarrow{\alpha_i} P_i}$	(PREFIX)	$\frac{P \xrightarrow{\bar{x}(y)} P', Q \xrightarrow{x(z)} Q'}{P Q \xrightarrow{\tau} P' Q'\{y/z\}}$	(COMM)		
$\frac{P !P \xrightarrow{\mu} P'}{!P \xrightarrow{\mu} P'}$	(BANG)	$\frac{P \xrightarrow{\bar{x}(vy)} P', Q \xrightarrow{x(y)} Q'}{P Q \xrightarrow{\tau} (vy)(P' Q')}$	(CLOSE)		
$\frac{P \xrightarrow{\mu} P'}{P Q \xrightarrow{\mu} P' Q}$	$bn(\mu) \cap fn(Q) = \emptyset$	(PAR)	$\frac{P \xrightarrow{\mu} P'}{[x = x]P \xrightarrow{\mu} P'}$	(MATCH)	
$\frac{P \equiv Q \quad Q \xrightarrow{\mu} Q' \quad Q' \equiv P'}{P \xrightarrow{\mu} P'} \quad$ (STRUCT)					
$\frac{P \xrightarrow{\mu} P'}{(vy)P \xrightarrow{\mu} (vy)P'}$	$y \notin n(\mu)$	(RES)	$\frac{P \xrightarrow{\bar{x}(y)} P'}{(vy)P \xrightarrow{\bar{x}(vy)} P'}$	$y \neq x$	(OPEN)

2.1.2 Behavioural equivalences

We report the definition of *strong early bisimilarity*, one of the basic behavioural equivalences defined on π processes.

DEFINITION 1. (BISIMILARITY) A binary symmetric relation \mathcal{S} on processes is an early bisimulation if and only if: $P \mathcal{S} Q$ and $P \xrightarrow{\mu} P'$ implies

- (1) if $\mu = x(y)$ then $\forall z. \exists Q' : Q \xrightarrow{x(y)} Q' \wedge P'\{z/y\} \mathcal{S} Q'\{z/y\}$
- (2) if μ is not an input then $\exists Q' : Q \xrightarrow{\mu} Q' \wedge P' \mathcal{S} Q'$.

P is early bisimilar to Q ($P \sim Q$) if $P \mathcal{S} Q$ for some early bisimulation \mathcal{S} .

Full bisimilarity (\sim) is the congruence relation defined as the closure of early bisimilarity under all substitutions σ . The previous relations are called *strong* because they distinguish also processes that differ only by internal actions. It is interesting in some cases to abstract over τ actions and consider bisimulation with respect to visible actions only. Let \Longrightarrow be the reflexive and transitive closure of $\xrightarrow{\tau}$ and $\xRightarrow{\mu}$ be $\Longrightarrow \xrightarrow{\mu} \Longrightarrow$. Moreover let $\xRightarrow{\hat{\mu}}$ be \Longrightarrow if $\mu = \tau$, $\xRightarrow{\mu}$ otherwise. The definition of *Weak early bisimulation* (\approx) is obtained by replacing all of the instances of $Q \xrightarrow{\mu} Q'$ in Definition 1 with $Q \xRightarrow{\hat{\mu}} Q'$.

The semantic rules of π -calculus expressing interaction between processes are (COMM) and (CLOSE), and in both cases the premise of the rule requires that two parallel processes P and Q are able to perform two complementary actions with the same prefix. Therefore, it is natural to define such actions as the observables of a process P : they must in fact be *visible* through the parallel operator by a context Q , in order for interaction to take place. This justifies the following definition.

DEFINITION 2. (BARBS) *The main observability predicates of π -calculus are the barbs defined as:*

$$\begin{array}{ll} P \downarrow_x \triangleq \exists y, Q. P \xrightarrow{x(y)} Q & P \Downarrow_x \triangleq \exists R. (P \Longrightarrow R \wedge R \downarrow_x) \\ P \downarrow_{\bar{x}} \triangleq \exists y, Q. (P \xrightarrow{\bar{x}(y)} Q \vee P \xrightarrow{\bar{x}(y)} Q) & P \Downarrow_{\bar{x}} \triangleq \exists R. (P \Longrightarrow R \wedge R \downarrow_{\bar{x}}) \\ P \Downarrow \triangleq \exists x. (P \downarrow_x \vee P \downarrow_{\bar{x}}) & P \Downarrow \triangleq \exists x. (P \Downarrow_x \vee P \Downarrow_{\bar{x}}) \end{array}$$

A slash on the vertical arrow (e.g. $P \Downarrow_x$) will mean that it is not the case that the property holds. An equivalent characterisation of $P \downarrow_x$ and $P \downarrow_{\bar{x}}$ is syntactical: $P \downarrow_x \triangleq P \equiv (\nu x_1) \dots (\nu x_n) (x(y). Q + M|N)$, $P \downarrow_{\bar{x}} \triangleq P \equiv (\nu x_1) \dots (\nu x_n) (\bar{x}(y). Q + M|N)$ for some name $x \notin x_1, \dots, x_n$, and some processes Q , M , and N .

Rule (MATCH) seems to contradict the intuition that barbs (i.e. observability) should depend on the *structure* of a process, regardless of the actual value of its state (the set of bindings between names and channels). In fact π^m , the sub-calculus without matching, enjoys the following property.

OBSERVATION 2.1. For any process P in π^m , for any name x , and for any substitution σ , $P \downarrow_x$ if and only if $P\sigma \downarrow_{\sigma(x)}$, and $P \downarrow_{\bar{x}}$ if and only if $P\sigma \downarrow_{\overline{\sigma(x)}}$.

PROOF. By cases on the syntax and by definition of barbs. \square

Conversely, this property does not hold in π : given $M \triangleq [x = y]x$, $\sigma_1 = \{z/x\}\{w/y\}$ (where $z \neq w$), and $\sigma_2 = \{z/x\}\{z/y\}$ we have that $M\sigma_1 \Downarrow$ but $M\sigma_2 \downarrow_z$.

REMARK 1. In the light of what we have just said, we advocate a different definition of matching in π :

$$\text{Syntax : } P ::= \dots \mid [x = y]\tau.P \quad \text{Semantics : } \overline{[x = x]\tau.P \xrightarrow{\tau} P} \quad (\text{MATCH})$$

This rule would allow Observation 2.1 to be extended to the whole language, supporting the intuition that the difference introduced in the observability of the process is due to an internal reduction: matching becomes an *operation*, like in most other languages. We will see in Section 4.1 how this definition is supported by a correspondence with the derivation of matching in ${}^e\pi$.

2.2 The π_{Op} family

The asynchronous π -calculus ($a\pi$) proposed independently by Honda and Tokoro [1991], and Boudol [1992], is the sub-calculus without summations and matching, and where output can be prefixed only to the inactive process.

$$P ::= \bar{x}(y).0 \mid M \mid P \mid P \mid (\nu x)P \mid !P$$

$$M ::= 0 \mid x(y).P \mid \tau.P$$

Both the previous references show how to encode the synchronous output $\bar{x}(y).P$ in terms of the simpler asynchronous communication mechanism: the real difference with π consist in the absence of matching and, more remarkably, of choice.

The separate choice π -calculus (π^s) is the sub-calculus of π in which output and input prefixes cannot be present in the same summation. This restriction is captured by modifying the syntax of processes without affecting the semantic rules.

$$P ::= 0 \mid \sum_i \alpha_i^I.P_i \mid \sum_i \alpha_i^O.P_i \mid P \mid P \mid (\nu x)P \mid !P$$

$$\alpha^I ::= \tau \mid x(y)$$

$$\alpha^O ::= \tau \mid \bar{x}(y)$$

Nestmann and Pierce [2000] have shown how to encode input-guarded choice in $a\pi$, and Nestmann [2000] has proposed an encoding of the full π^s .

We denote with π^m the calculus with mixed choice, but without matching. Palamidessi [2002] has proved that π^m is strictly more expressive than π^s .

Another operator sometimes considered in the π -calculus is mismatch : it consists of the production $P ::= [x \neq y].P$, and its semantics can be defined both by an its rule or by a structural congruence rule:

$$\frac{P \xrightarrow{\mu} P'}{[x \neq y]P \xrightarrow{\mu} P'} \quad x \neq y \quad [x \neq x]P \equiv 0$$

All the sub-calculi that we have seen so far can be extended with matching or mismatching. We will denote the extended versions by appending apices to the calculus name. In the sequel, let π_{Op} be defined as $\{a\pi, \pi^s, \pi^m, a\pi^{\bar{\cdot}}, \pi^{s,\bar{\cdot}}, \pi\}$. All the variants of the π -calculus considered in the paper are summarised in Table 3.1.

2.3 Encodings

According to Palamidessi [2002], an encoding is *uniform* if it translates the parallel operator homomorphically and if it respects permutations on free names

$$\llbracket P|Q \rrbracket = \llbracket P \rrbracket \llbracket Q \rrbracket \quad (2.1)$$

$$\forall \sigma \exists \theta \llbracket P\sigma \rrbracket = \llbracket P \rrbracket \theta \quad (2.2)$$

and a *reasonable* semantics is one

“... which distinguishes two processes P and Q whenever there exists a maximal (finite or infinite) computation of Q in which the intended observables (some visible actions) are different from the observables in any (maximal) computation of P .”

Condition (2.1) states that the degree of parallelism in the system must be preserved by the encoding, condition (2.2) states that the structure of the encoding respects permutations of free names. Since we are interested in the problem of encoding specific constructs that may occur in π -calculus terms, it is sensible to strengthen condition (2.2) to account for arbitrary substitutions. In fact, a process can be syntactically placed in the scope of an input, that in π -calculus behaves like a substitution. The following example illustrates our point.

EXAMPLE 1. (ENCODING MISMATCHING) Consider the homomorphic encoding of π^\neq in π with infinite products, where mismatching is translated as follows:

$$\llbracket [x \neq y]P \rrbracket \triangleq \prod_{w \in \mathcal{N} \setminus \{y\}} [x = w]P \quad (\text{if } x \neq y) \quad (2.3)$$

$$\llbracket [x \neq x]P \rrbracket \triangleq 0 \quad (\text{otherwise}) \quad (2.4)$$

This encoding is uniform and is correct in a very strong sense: $\llbracket [x \neq y]P \rrbracket \sim [x \neq y]P$ and $\llbracket [x \neq x]P \rrbracket \equiv [x \neq x]P$. Nonetheless it is not satisfactory because it does not respect arbitrary substitutions:

$$(\nu z)(\bar{z}\langle a, a \rangle | z(x, y). [x \neq y]P) \xrightarrow{\tau} 0 \quad (2.5)$$

($x \neq y$, for the input on z to be defined) and for the encoding to be meaningful, we would expect an equivalent behaviour by its translation, whereas

$$(\nu z)(\bar{z}\langle a, a \rangle | z(x, y).\llbracket [x \neq y]P \rrbracket) \xrightarrow{\tau} P\{a/x\}\{a/y\} \quad (2.6)$$

The problem is that the term $\llbracket [x \neq y]P \rrbracket$ contains syntactically the term $[x = x]P$ that is not affected by the changes made on y after the translation: it does not respect arbitrary substitutions.

Another appealing property that is not needed for the results in [Palamidessi 2002], but that is considered for example in [de Boer and Palamidessi 1994, Nestmann and Pierce 2000, and Nestmann 2000], is termination invariance. We consider a crucial property of a semantics to distinguish inactive processes (deadlocks) from processes involved in infinite internal computations (livelocks). We call *sensible* an encoding $\llbracket - \rrbracket$ which is (strongly) *uniform*, preserves a *reasonable* semantics, and distinguishes deadlocks from livelocks.

3. Polyadic synchronisation in π -calculus

Our proposal is to extend the synchronisation mechanism of π -calculus to the case where channels are denoted by *vectors of names*, allowing interaction to happen only when such vectors match element-wise. Synchronisation remains atomic: we enforce an *all-or-nothing* behaviour. A typical reduction might look like

$$x \cdot y(z).P | \overline{x \cdot y} \langle w \rangle . Q \xrightarrow{\tau} P\{w/z\} | Q$$

3.1 Syntax and semantics of ${}^e\pi$

To define ${}^e\pi$ we need to generalise the syntax for prefixes given in precedence to the one given below, where k and j are any two natural numbers.

$$\text{(PREFIXES)} \alpha ::= \tau \mid x_1 \cdot \dots \cdot x_k(y) \mid \overline{x_1 \cdot \dots \cdot x_j} \langle y \rangle$$

A channel is now a vector of names, the *synchronisation vector*: π -calculus is the instance where only vectors of length one are allowed. Synchronisation vectors will be denoted by letters u and v . The syntax of processes is the same as the one for π^m , and all the definitions given in Section 2.1 are straightforwardly adapted to the case where a vector substitutes a single name in the subjects of actions. As an example, we report two specific reduction rules where vector u has replaced name x .

$$\frac{P \xrightarrow{\overline{u}\langle vy \rangle} P', Q \xrightarrow{u(y)} Q'}{P | Q \xrightarrow{\tau} (vy)(P' | Q')} \quad \text{(CLOSE)} \quad \frac{P \xrightarrow{\overline{u}\langle y \rangle} P'}{(vy)P \xrightarrow{\overline{u}\langle vy \rangle} P'} \quad y \notin u \quad \text{(OPEN)}$$

The only exception regards matching: as we will show in Lemma 4.1, matching can be derived in ${}^e\pi$ and therefore we exclude both the syntactic production and the semantic rule (MATCH) from the definition of ${}^e\pi$. As a consequence, Observation 2.1 holds also for ${}^e\pi$.

It is worth noting that since restriction is defined on names rather than on channels as a whole, process $P \triangleq x \cdot y(z).Q$ is such that $P \downarrow_{x,y}$, but $R \triangleq (vx)x \cdot y(z).Q$ is such that $R \not\downarrow$, even if $y \in fn(R)$.

3.1.1 The $\pi_{\mathbb{N}}$ family

We denote with π_k the sub-language where the length of synchronisation vectors is at most k . A degenerate case is π_0 , where channels are nameless and processes interact through a global ether³: $\langle y \rangle . P \mid (x).Q \longrightarrow P \mid Q\{y/x\}$. The sub-calculi defined in Section 2.2 can be analogously extended to polyadic synchronisation. In particular, π^m is π_1 where prefixes with vectors of length 0 are ruled out, and ${}^e\pi = \bigcup_n \pi_n$. The family of calculi $\{a\pi_0, \pi_0, a\pi_1, \pi_1, a\pi_2, \pi_2, \dots\}$ will be denoted by $\pi_{\mathbb{N}}$ (see also Table 3.1).

³ It is essentially the local communication mechanism of the Ambient Calculus of Cardelli and Gordon [2000].

TABLE 3.1: Some variants of the π -calculus.

	Variants of π -calculus:					The $\pi_{\mathbb{N}}$ family:		
	$\bar{a}.P$	$=$	\neq	$+_s$	$+_m$	$\bar{a}.P$	$+_m$	$a_1 \cdot \dots \cdot a_n$
$a\pi$	-	-	-	-	-	$a\pi_0$	-	0
π^s	✓	-	-	✓	-	$a\pi_n$	-	$\leq n$
π^m	✓	-	-	-	✓	π_0	✓	0
$a\pi^=$	-	✓	-	-	-	π_n	✓	$\leq n$
$a\pi^{=,\neq}$	-	✓	✓	-	-	$^e\pi$	✓	$< \omega$
$\pi^{s,=}$	✓	✓	-	✓	-			
π	✓	✓	-	-	✓			

3.2 Encoding polyadic synchronisation in π -calculus

It is possible to define a strongly uniform encoding of bi-adic synchronisation in polyadic $a\pi^{=,\neq}$, where we use the conditional construct $[a = b]P, Q$ as an abbreviation for $[a = b]P[a \neq b]Q$. By transitivity, the same idea can be used to encode higher degrees of synchronisation. The encoding is a homomorphism, except for the following cases ($w, x, z \notin \text{fn}(P)$):

$$\begin{aligned} \overline{\llbracket b \cdot a \langle c \rangle \rrbracket} &\triangleq (\nu z)(\bar{b} \langle a, c, z \rangle \mid \bar{z}) \\ \llbracket b \cdot a(y).P \rrbracket &\triangleq (\nu w)(\bar{w} \mid !w.b \langle x, y, z \rangle.[x = a](z \mid \llbracket P \rrbracket)), (\bar{b} \langle x, y, z \rangle \mid \bar{w}) \end{aligned}$$

In the encoding of the input we simulate the prefix $b \cdot a(y).P$ in two steps: an input on b and a matching on a , introducing the need to backtrack in case of failure. The parameter z added to the communication distinguishes the behaviour of the two branches of the conditional, and is needed to preserve the soundness of the translation. Note that if matching fails, the original state is restored, introducing the possibility of divergence.

TABLE 3.2: Translation of actions.

$$\begin{aligned} \overline{\llbracket b \cdot a \langle c \rangle \rrbracket} &\triangleq \bar{b} \langle a, c, \nu z \rangle & \overline{\llbracket b \cdot a \langle \nu c \rangle \rrbracket} &\triangleq \bar{b} \langle a, \nu c, \nu z \rangle \\ \llbracket b \cdot a(y) \rrbracket &\triangleq b \langle x, y, z \rangle & \llbracket \tau \rrbracket &\triangleq \tau \end{aligned}$$

In Table 3.2 we define the correspondence among the actions of the source terms and those of the target terms. Note that the correspondence in the observables is not very strong, in particular the translation of an input action replaces a free name with a bound name (in $\llbracket b \cdot a(y) \rrbracket \triangleq b \langle x, y, z \rangle$, x replaces a). We look now at the

properties of the encoding.

PROPOSITION 3.1. *The encoding of $\alpha\pi_2$ in $\alpha\pi^{\neq}$ is strongly uniform.*

PROOF. The encoding is homomorphic with respect to parallel composition. By noting that the encoding preserves free names, and by a straightforward induction, follows that for all substitutions σ , $\llbracket P \rrbracket \sigma = \llbracket P\sigma \rrbracket$. \square

The encoding is sound with respect to \approx , whereas it is not complete, as it can be seen from the following example.

EXAMPLE 2. Let $P \triangleq (\nu a)(\overline{b \cdot a})$ and $Q \triangleq (\nu a)(\overline{c \cdot a})$. We have that $P \approx 0 \approx Q$ whereas $\llbracket P \rrbracket \not\approx_{\pi} \llbracket Q \rrbracket$ since $\llbracket P \rrbracket \downarrow_b$ and $\llbracket P \rrbracket \not\downarrow_c$, but $\llbracket Q \rrbracket \downarrow_c$ and $\llbracket Q \rrbracket \not\downarrow_b$.

The proof of the theorem below is in the Appendix.

THEOREM 3.1. *For all processes P and Q in $\alpha\pi_2$:*

- (1) *If $P \xrightarrow{\tau} Q$ then $\llbracket P \rrbracket \xrightarrow{\tau} \xrightarrow{\tau} \xrightarrow{\tau} \sim \llbracket Q \rrbracket$.*
- (2) *If $\llbracket P \rrbracket \Longrightarrow Q'$ then there exists Q such that $P \Longrightarrow Q$ and $Q' \approx \llbracket Q \rrbracket$.*
- (3) *If $\llbracket P \rrbracket \approx \llbracket Q \rrbracket$ then $P \approx Q$.*

The existence of the encoding shows that *it is possible*, to some extent, to achieve the effects of polyadic synchronisation in π , as long as one is not concerned with termination properties.

4. Expressivity of polyadic synchronisation

4.1 Matching

We show that the matching operator cannot be derived in π -calculus, and therefore needs to be taken as primitive. In ${}^e\pi$ instead it is possible to define matching as a derived operator. This fact constitutes a first separation result between the expressivity of the two languages. For example, in π^m it is not possible to write a tester process able to tell whether or not two arbitrary names denote the same channel, without disturbing the communications on the channel (or channels) denoted by those names. The peculiarity of matching is in fact to allow a process to evolve if and only if two names denote the same observable. Consequently the intended observables to be preserved by a reasonable semantics of matching are all the visible actions performed on the channel names that can be tested for equality, and therefore are all the barbs.

4.1.1 The negative result

We start noting some properties of processes in π and π^m .

OBSERVATION 4.1. For any processes P, Q in π or π^m , if $P \xrightarrow{\tau} Q$ then, for any substitution σ , $P\sigma \xrightarrow{\tau} Q\sigma$.

PROOF. By cases on the syntax the reduction must be defined on two input and output prefixes with the same syntactical subject, and therefore remains executable under any substitution on P . \square

The crucial property that characterises the absence of matching in π^m is that if a substitution does not affect the barbs of a process, then it does not increase its ability to reduce.

PROPOSITION 4.1. *For any process in π^m , for any substitution σ with domain \mathcal{D}_σ , if $(\forall x \in \mathcal{D}_\sigma. P \Downarrow_x \wedge P \Downarrow_{\bar{x}})$ holds and $P\sigma \xrightarrow{\tau} Q\sigma$, then $P \xrightarrow{\tau} Q$.*

PROOF. Similarly to the proof of Observation 4.1, if $P \Downarrow_x$ and $P \Downarrow_{\bar{x}}$ for every $x \in \mathcal{D}_\sigma$, then the reduction $P\sigma \xrightarrow{\tau} Q\sigma$ cannot be obtained by prefixes whose subjects are syntactically different, and therefore the reduction remains executable also without applying the substitution. \square

This law does not hold both in π and in $e\pi$. Consider the substitution $\sigma = \{x/y\}$, the π process $P_\pi \triangleq [x = y]\tau.Q$, and the $e\pi$ process $P_{e\pi} \triangleq (\nu z)(\bar{z} \cdot \bar{x} | z \cdot y.Q)$: both $P_\pi \Downarrow$ and $P_{e\pi} \Downarrow$, but both $P_\pi\sigma$ and $P_{e\pi}\sigma$ can reduce whereas P_π and $P_{e\pi}$ cannot. From Proposition 4.1 and Observation 4.1 follows this useful corollary.

COROLLARY 4.1. *Let σ range over arbitrary substitutions. For any π^m process P , if $P \Downarrow$ then $(\exists \sigma. P\sigma \xrightarrow{\tau} Q\sigma) \Rightarrow (\forall \sigma. P\sigma \xrightarrow{\tau} Q\sigma)$.*

THEOREM 4.1. (MATCHING) *There exists no sensible encoding of $a\pi^=$ in π^m .*

PROOF. Suppose that $\llbracket - \rrbracket$ is such an encoding: we have noted that a reasonable semantics of matching must preserve observations on barbs⁴, and therefore

$$\forall x. P \Downarrow_x \Leftrightarrow \llbracket P \rrbracket \Downarrow_x; \quad \forall x. P \Downarrow_{\bar{x}} \Leftrightarrow \llbracket P \rrbracket \Downarrow_{\bar{x}} \quad (4.1)$$

which implies in particular that $P \Downarrow \Leftrightarrow \llbracket P \rrbracket \Downarrow$. Considering $M \triangleq [x = y]x$ we have that, by definition of matching and by (4.1),

$$\forall \sigma. M\sigma \Downarrow_{\sigma(x)} \Leftrightarrow \llbracket M\sigma \rrbracket \Downarrow_{\sigma(x)} \Leftrightarrow \sigma(x) = \sigma(y) \quad (4.2)$$

where σ is an arbitrary substitution. By strong uniformity the previous condition becomes

$$\forall \sigma \exists \theta. \llbracket M \rrbracket \theta \Downarrow_{\sigma(x)} \Leftrightarrow \sigma(x) = \sigma(y) \quad (4.3)$$

Consider now two substitutions σ_1 and σ_2 such that $\sigma_1(x) = \sigma_1(y)$ and $\sigma_2(x) \neq \sigma_2(y)$. By (4.2) we have that $M\sigma_1 \Downarrow_{\sigma_1(x)}$ and $M\sigma_2 \not\Downarrow$, and by (4.3) follows that

$$\exists \theta_1, \theta_2. \llbracket M \rrbracket \theta_1 \Downarrow_{\sigma_1(x)} \wedge \llbracket M \rrbracket \theta_2 \not\Downarrow \quad (4.4)$$

Expanding the definition of \Downarrow we have that

$$\llbracket M \rrbracket \theta_2 \not\Downarrow \triangleq \forall M'. \llbracket M \rrbracket \theta_2 \Longrightarrow M' \wedge M' \not\Downarrow \quad (4.5)$$

⁴ The result holds also for a weaker notion of observation preservice: replacing (4.1) with $P \Downarrow \Leftrightarrow \llbracket P \rrbracket \Downarrow$.

By reflexivity and transitivity of \Longrightarrow , applying at each step Observation 2.1 and Corollary 4.1, we can conclude that

$$\forall \theta. \llbracket M \rrbracket \theta \Downarrow \quad (4.6)$$

contradicts (4.4), therefore a sensible encoding of matching in π^m cannot exist. \square

REMARK 2. In the proof we have used process $[x = y]x$ as a counterexample for generality: it belongs both to the syntax of Milner *et al.* [1992] and of Sangiorgi and Walker [2001]. This term doesn't enjoy Observation 2.1, but the proof holds also for process $[x = y]\tau.x$, which conforms to our definition of matching (Remark 1), clarifying that the key property characterising the non-encodability of matching is Proposition 4.1.

4.1.2 The positive result

We show that the encoding of matching in ${}^e\pi$ preserves strong full bisimilarity. Our encoding confirms the intuition that matching requires both input and output capabilities on a channel.

LEMMA 4.1. (ENCODING OF MATCHING IN ${}^e\pi$) *Let P be any process in ${}^e\pi^=$ such that $z \notin \text{fn}(P)$, and let $\llbracket [x = y]\tau.P \rrbracket \triangleq (\nu z)(\overline{z \cdot x} \mid z \cdot y.P)$. Then $[x = y]\tau.P$ and $\llbracket [x = y]\tau.P \rrbracket$ are strongly full bisimilar (\sim).*

PROOF. By definition of \sim , given P and Q , $P \sim Q$ if and only if for all σ , $P\sigma \sim Q\sigma$. We split the proof according to the behaviour of σ on x and y :

- $\sigma(x) \neq \sigma(y)$: both processes $[\sigma(x) = \sigma(y)]\tau.(P\sigma)$ and $(\nu z)(\overline{z \cdot \sigma(x)} \mid z \cdot \sigma(y).(P\sigma))$ cannot reduce and cannot perform any barb, therefore they are bisimilar;
- $\sigma(x) = \sigma(y)$: again both processes cannot perform any barb, and both reduce to the same process

$$[\sigma(x) = \sigma(y)]\tau.(P\sigma) \xrightarrow{\tau} P\sigma \quad (4.7)$$

$$(\nu z)(\overline{z \cdot \sigma(x)} \mid z \cdot \sigma(y).(P\sigma)) \xrightarrow{\tau} P\sigma \quad (4.8)$$

By reflexivity of \sim , $P\sigma \sim P\sigma$.

\square

4.2 Mixed choice

We consider now the expressive power of the mixed choice construct. Palamidessi's result, separating π^m from π^s , holds analogously in our setting. The key point is that the impossibility to break the symmetry between identical processes that constitutes the core of the negative result, is not influenced by the ability to synchronise on more than one channel name. In fact, we can show that Lemma 4.1 of Palamidessi [2002] generalises to our setting.

LEMMA 4.2. *Let β be $\bar{u}\langle x \rangle$ or $\bar{u}\langle vx \rangle$, and let P be a process of π_n^s for some n . Assume that P can make two transitions $P \xrightarrow{\beta} Q$ and $P \xrightarrow{v(y)} R$. Then there exists S such that $Q \xrightarrow{v(y)} S$ and $R \xrightarrow{\beta} S$.*

PROOF. Consider the syntactical definition of summations and prefixes in π_n^s , where $k, j \leq n$

$$P ::= \dots \mid \sum_i \alpha_i^I . P_i \mid \sum_i \alpha_i^O . P_i \mid \dots \quad (4.9)$$

$$\alpha^I ::= \tau \mid x_1 \cdot \dots \cdot x_k \langle y \rangle; \quad \alpha^O ::= \tau \mid \overline{x_1 \cdot \dots \cdot x_j} \langle y \rangle \quad (4.10)$$

and recall the syntactical characterisation of barbs given in Definition 2

$$\exists y, Q. P \xrightarrow{x(y)} Q \Leftrightarrow P \equiv (\nu x_1) \dots (\nu x_n) (x(y). Q + M \mid N) \quad (4.11)$$

$$\exists y, Q. (P \xrightarrow{\bar{x}(y)} Q \vee P \xrightarrow{\bar{x}(y)} Q) \Leftrightarrow P \equiv (\nu x_1) \dots (\nu x_n) (\bar{x}(y). Q + M \mid N) \quad (4.12)$$

where x is different from each x_1, \dots, x_n . We have that

$$P \xrightarrow{v(y)} R \Leftrightarrow P \equiv (\nu x_1) \dots (\nu x_n) (v(y). Q_1 + M_1 \mid N_1) \quad (4.13)$$

$$P \xrightarrow{\beta} Q \Leftrightarrow P \equiv (\nu x_1) \dots (\nu x_n) (\bar{u}\langle x \rangle. Q_2 + M_2 \mid N_2) \quad (4.14)$$

where the names in u and v are all different from x_1, \dots, x_n . By (4.9) and (4.10) we have that M_1 cannot contain an output at the top level, and M_2 cannot contain an input. Therefore

$$N_1 \equiv N \mid \bar{u}\langle x \rangle. Q_2 + M_2 \quad (4.15)$$

$$N_2 \equiv N' \mid v(y). Q_1 + M_1 \quad (4.16)$$

that allows us to conclude that $N \equiv N'$ and

$$P \equiv (\nu x_1) \dots (\nu x_n) (\bar{u}\langle x \rangle. Q_1 + M_1 \mid N \mid v(y). Q_1 + M_2) \quad (4.17)$$

By applications of rules (RES), (OPEN), (PAR) and (PREFIX) follows the thesis. \square

REMARK 3. Technically speaking our Theorem 4.1, which shows that π^s is less expressive than $\pi^{s,=}$, restricts the generality of the separation result given by Palamidessi [2002], because Lemma 4.1 shown in that reference is proved only for processes of π^s . Nonetheless it is an easy exercise to adapt the proof of the cited lemma to processes in $\pi^{s,=}$, restoring the full generality of Palamidessi's result.

Given the confluence property shown in Lemma 4.2, we can claim that an analogous of Palamidessi's separation result holds also in our setting.

PROPOSITION 4.2. (MIXED CHOICE) *For any n, m , there exist no uniform encoding of π_n into $a\pi_m$ which preserves a reasonable semantics.*

PROOF. By inspection of the cases in the proof of Theorem 4.2 of Palamidessi [2002], replacing Lemma 4.1 of the reference with Lemma 4.2, we have that a symmetric electoral system cannot be encoded in $a\pi_m$, for any m . On the other hand, Claim 1 of the reference stating the existence of symmetric electoral systems in π^m , holds also for π_n with $n > 0$, noting that π^m is a sub-calculus of π_1 . In the degenerate case of π_0 we have that for example

$$(\langle y \rangle + (x).\bar{o}\langle y \rangle) \mid (\langle z \rangle + (x).\bar{o}\langle z \rangle)$$

is a simple symmetric electoral system with two components. \square

4.3 Polyadic synchronisation

We show our main expressivity result: the expressive power of calculi in $\pi_{\mathbb{N}}$ depends on the degree of synchronisation. In particular, we show that there exists a separation problem: it is not possible to write two non-divergent processes in π_n to detect whether two vectors of $n + 1$ identifiers are equal, whereas it is possible in π_{n+1} .

4.3.1 Matching systems

We define a family of binary relations on processes called *Matching Systems*. In the following, let a *server template* of degree n be a process whose free names (the *process identifiers*) are x_1, \dots, x_n , and let a *client template* be defined as a server template with an additional free name y (the *process index*). In a matching system, copies of a template of each kind (say S and C) are instantiated in parallel as in $C_1\sigma_1 \mid \dots \mid C_h\sigma_h \mid S_1\theta_1 \mid \dots \mid S_k\theta_k$, where a substitution is applied to each process in order to “personalise” its identifiers. If the same substitution is applied to an instance of a client C_i and to one of a server S_j , the two instances are meant to recognise each other and perform some kind of meaningful behaviour. Therefore, a natural requirement is that the recognition process shall be finite. The process index constitutes the unique identity of a client. To represent the end of a (successful) recognition process between C_i and S_j , we require S_j to notify the index i of C_i on an additional global channel o that must be used only for this purpose⁵. We allow special observations on o , of the form $P \downarrow_{\bar{o}\langle i \rangle}$, in order to note the object of the communication as well as the subject. The Matching Problem MP_n consists in finding two processes that constitute a matching system of degree n , according to the following definition.

DEFINITION 3. (MATCHING SYSTEM) *A client template C and a server template S of degree n constitute a Matching System $MS_n(C, S)$ of degree n if and only if*

- *for all finite set of server indexes J ,*
- *for all finite sets of fresh client indexes $I \subset (\mathcal{N} \setminus \{x_1, \dots, x_n, o\})$,*

⁵ Without invalidating the result, we allow this special channel also in π_0 , that normally cannot have channels.

- for any set of substitutions $\{\sigma_i\}_I$ and $\{\theta_j\}_J$ with domain $\{x_1, \dots, x_n\}$ (the process identifiers) and codomain $\mathcal{N} \setminus (I \cup \{o\})$,
- for all the processes P of the form $\prod_{i \in I} C_i\{i/y\}\sigma_i \mid \prod_{j \in J} !S_j\theta_j$

the following properties hold:

- (1) there is no infinite sequence of reductions starting from process P ;
- (2) an output $\bar{o}(i)$ is observable if and only if there are a client i and a server j with the same identifiers: $\forall i \in I. (P \Downarrow_{\bar{o}(i)} \Leftrightarrow \exists j \in J. \sigma_i = \theta_j)$.

Note that the only condition on the server indexes is that J is finite. In fact the server indexes play a role only at the meta-level, and are not needed operationally.

EXAMPLE 3. The π_0 processes $C' = \langle y \rangle$ and $S' = (w).\bar{o}\langle w \rangle$ constitute a matching system $MS_0(C', S')$: C' and S' have no identifiers, and therefore every exchange of indices is legal. For example:

$$\langle i^1 \rangle \langle i^2 \rangle \langle i^3 \rangle ! (w).\bar{o}\langle w \rangle \xrightarrow{\tau} \xrightarrow{\tau} \xrightarrow{\tau} \bar{o}\langle i^1 \rangle \langle \bar{o}\langle i^2 \rangle \rangle \langle \bar{o}\langle i^3 \rangle \rangle ! (w).\bar{o}\langle w \rangle \quad (4.18)$$

and the conditions of Definition 3 are trivially satisfied.

EXAMPLE 4. The $a\pi$ processes $C = \bar{x}\langle y \rangle$ and $S = x(w).\bar{o}\langle w \rangle$ constitute a matching system $MS_1(C, S)$. It is easy to verify that for any possible substitution parallel instances of the processes interact if and only if the channels resulting from a substitution on x are equal, and in that case the identifier of the client is correctly forwarded on channel o by a server.

A crucial property of matching systems is to be *open*, in the sense that a process cannot make assumptions on the parallel context where it is executed. In fact, matching systems are closed under parallel instantiation: given $MS_m(C, S)$ and two instances

$$I, J, \Sigma \triangleq \{\sigma_i\}_I, \Theta \triangleq \{\theta_j\}_J, P \triangleq \prod_{i \in I} C_i\{i/y\}\sigma_i \mid \prod_{j \in J} !S_j\theta_j \quad (4.19)$$

$$I', J', \Gamma \triangleq \{\gamma_i\}_{I'}, \Delta \triangleq \{\delta_j\}_{J'}, P' \triangleq \prod_{i \in I'} C_i\{i/y\}\gamma_i \mid \prod_{j \in J'} !S_j\delta_j \quad (4.20)$$

also the parallel instance

$$I \cup I', J \cup J', \Sigma \cup \Gamma, \Theta \cup \Delta, P'' \triangleq P \mid P' \quad (4.21)$$

is a legal instance of the same matching system, provided that $I \cap I' = \emptyset$.

4.3.2 The $\pi_{\mathbb{N}}$ hierarchy

THEOREM 4.2. (EXPRESSIVITY) *For all non-negative integer numbers n and m , the problem MP_m has a solution in π_n if and only if $n \geq m$.*

PROOF. (\Leftarrow) We give a process in $a\pi_m$ (therefore also in π_n) providing a solution to MP_m :

$$C \triangleq \overline{x_1 \cdot \dots \cdot x_m}(y); \quad S \triangleq x_1 \cdot \dots \cdot x_m(w).\overline{o}(w) \quad (4.22)$$

The degenerate case for $m = 0$ is reported in Example 3. As required by Definition 3, every instance of $MS_m(C, S)$ terminates: there are only a finite number of output prefixes at the top level, and the only outputs in the continuations of the servers are on channel o , but no input on o is allowed in the system. The only eventuality in which an output $\overline{o}(i)$ may be observed is when communication happens between two instances of C and S subject to the same substitution.

(\Rightarrow) Consider the minimal case where $m = n + 1$ and assume that MP_{n+1} has a solution in π_n . We show that this hypothesis leads to a contradiction. Let C, S be those two π_n process templates of degree $n + 1$ such that $MS_{n+1}(C, S)$. They must necessarily satisfy conditions (1) and (2) of Definition 3 for all the well-formed instantiations of their parameters. In particular, we recall that matching systems are closed under parallel instantiations. With a slight abuse of notation, let C_i stand in the sequel for $C_i\{i/y\}$. Consider the instance

$$P \triangleq C_i\sigma_i \mid !S_j\theta_j \quad (4.23)$$

where $\sigma_i = \theta_j$: by condition (2) we have that $P \Downarrow \overline{o}(i)$. It must be the case that neither $C_i\sigma_i \Downarrow \overline{o}(i)$ nor $S_j\theta_j \Downarrow \overline{o}(i)$, otherwise the well-formed instances $P_1 \triangleq C_i\sigma_i$ and $P_2 \triangleq !S_j\theta_j$ would not respect condition (2). Similarly, $\overline{o}(i)$ cannot be made observable by an interaction of $C_i\sigma_i$ with a context not containing $!S_j\theta_j$. We conclude that at least a synchronisation must take place between a client and the corresponding server in order to verify the compatibility of the identifiers.

Since the identifiers to be tested are $n + 1$ and both C and S by hypothesis are π_n processes, a barb presented by a client can contain at most n free names. Without loss of generality, suppose that $C_i\sigma_i \downarrow_u$, and $u = \sigma_i(x_1) \cdot \dots \cdot \sigma_i(x_n)$. For interaction to take place, it must be the case that $S_j\theta_j \downarrow_{\bar{u}}$. Considering θ_k such that $\theta_k = \theta_j$ on the first n identifiers but $\theta_k(x_{n+1}) \neq \theta_j(x_{n+1})$, we have from Observation 2.1 that $S_k\theta_k \downarrow_{\bar{u}}$, and consequently process

$$P_3 \triangleq C_i\sigma_i \mid !S_k\theta_k \quad (4.24)$$

is such that both $P_3 \downarrow_u$ and $P_3 \downarrow_{\bar{u}}$. This shows that in principle C_i can communicate with a process that is not its right partner. Considering now

$$P_4 \triangleq C_i\sigma_i \mid !S_k\theta_k \mid !S_j\theta_j \quad (4.25)$$

it may be the case that $P_4 \xrightarrow{\tau} P'_4 \mid !S_j\theta_j$ and consequently P'_4 must eventually attempt a synchronisation with S_j in order to satisfy condition (2). We have shown that $C_i\sigma_i$ must continue to attempt synchronisation until eventually identifies S_j , since it cannot make assumptions on the parallel context.

Noting that the set $J_\downarrow = \{u \mid S_j\theta_j \downarrow_u\}$ is finite (the free names of $S_j\theta_j$ are its $n + 1$ identifiers, and channel o) we can build a finite instance of $MS_n(C, S)$ that contains an infinite loop, contradicting condition (1)

$$P_\omega \triangleq C_i\sigma_i \mid \prod_{h \in H} !S_h\theta_h \quad (4.26)$$

where $\forall u \in J_{\downarrow}. \exists h \in H. \theta_h(x_1) \cdot \dots \cdot \theta_h(x_n) = u \wedge \theta_h(x_{n+1}) \neq \sigma_i(x_{n+1})$. \square

REMARK 4. Observation 2.1 holds also for π and ${}^e\pi^=$, provided that σ is injective. Therefore direction (\Rightarrow) of the proof holds also if matching is allowed as a primitive operator in C and S , taking care to chose in the counterexample an injective substitution θ_j .

We now look at the separation result from the perspective of encodings.

PROPOSITION 4.3. (POLYADIC SYNCHRONISATION) *There exists no sensible encoding of $a\pi_{n+1}$ in π_n , for any n .*

PROOF. Supposing that $\llbracket - \rrbracket$ is such an encoding, we derive a contradiction. By part (\Leftarrow) of the proof of Theorem 4.2, there are two $a\pi_{n+1}$ processes C and S providing a solution for MP_{n+1} . A sensible encoding preserves the properties of a matching system and therefore we would have that the two π_n processes $\llbracket C \rrbracket$ and $\llbracket S \rrbracket$ provide a solution for MP_{n+1} , contradicting part (\Rightarrow) of Theorem 4.2. \square

5. A hierarchy of expressiveness

To conclude we compare the dialects of π -calculus that we have considered so far by means of a lattice of expressivity induced by the notion of sensible encoding.

DEFINITION 4. *Given two process calculi \mathcal{P} and \mathcal{Q} in a set of calculi \mathcal{S} , we write:*

- $\mathcal{P} \leq \mathcal{Q}$ if there exists a sensible encoding of \mathcal{P} in \mathcal{Q} ;
- $\mathcal{P} \simeq \mathcal{Q}$ if both $\mathcal{P} \leq \mathcal{Q}$ and $\mathcal{Q} \leq \mathcal{P}$;
- $\mathcal{P} \not\leq \mathcal{Q}$ if it is not the case that $\mathcal{P} \leq \mathcal{Q}$;
- $\mathcal{P} < \mathcal{Q}$ if $\mathcal{P} \leq \mathcal{Q}$ and $\mathcal{Q} \not\leq \mathcal{P}$;
- $\mathcal{P} \neq \mathcal{Q}$ if both $\mathcal{P} \not\leq \mathcal{Q}$ and $\mathcal{Q} \not\leq \mathcal{P}$.

LEMMA 5.1. *Let $\mathcal{S} = \pi_{\mathbb{N}} \cup \pi_{Op}$, let $\mathcal{S}_{/\simeq}$ be \mathcal{S} quotiented by \simeq , and for each $[\mathcal{P}]_{/\simeq}, [\mathcal{Q}]_{/\simeq} \in \mathcal{S}_{/\simeq}$, let $[\mathcal{P}]_{/\simeq} \leq_{/\simeq} [\mathcal{Q}]_{/\simeq}$ if and only if $\mathcal{P} \leq \mathcal{Q}$. Then $(\mathcal{S}_{/\simeq}, \leq_{/\simeq})$ is a complete lattice with bottom element $a\pi_0$ and top element ${}^e\pi$, corresponding to the diagram reported in Table 5.1 (where $\mathcal{P} \longrightarrow \mathcal{Q}$ means $\mathcal{P} > \mathcal{Q}$).*

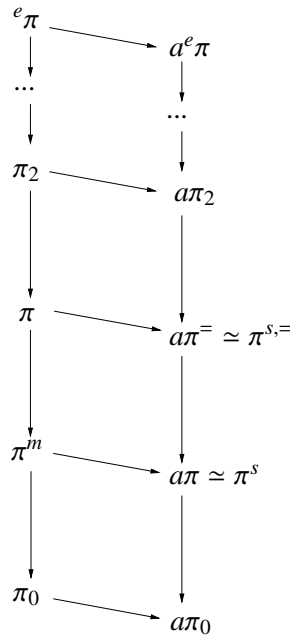
PROOF. Noticing that \leq is a preorder, and \simeq is the equivalence relation induced on \mathcal{S} by \leq , follows immediately that $(\mathcal{S}_{/\simeq}, \leq_{/\simeq})$ is a partial order. We now exhaustively check all the significative relations implied by the diagram.

- $a\pi \simeq \pi^s$, $a\pi^= \simeq \pi^{s=}$: in both cases \leq is an embedding and \geq follows from [Nestmann 2000], noticing in the formulation of Sangiorgi and Walker [2001] that the encoding behaves well with respect to arbitrary substitutions.
- $a\pi < a\pi^=$, $\pi^m < \pi$: in both cases \leq is an embedding and $\not\leq$ follows from Theorem 4.1, in the second case noticing that $a\pi^= \leq \pi$.

- $a\pi < \pi^m$, $a\pi^- < \pi$, $a\pi_n < \pi_n$: \leq is an embedding, $\not\leq$ follows respectively from [Palamidessi 2002], Remark 3, and Proposition 4.2.
- (a) $a\pi_n < a\pi_{n+1}$, $\pi_n < \pi_{n+1}$: in both cases \leq is an embedding and $\not\leq$ follows from Proposition 4.3;
- (b) $a\pi_0 < a\pi$, $\pi_0 < \pi$: in both cases \leq is a simple encoding where anonymous communication is translated by communicating on the same unrestricted channel, and $\not\leq$ follows from Proposition 4.3 noticing that the process given in part (\Leftarrow) of the proof of Theorem 4.2 for $n = 1$ belongs to $a\pi$;
- (c) $a\pi^- < a\pi_2$, $\pi^- < \pi_2$: in both cases \leq is a simple encoding where matching is translated according to Lemma 4.1, and $\not\leq$ follows from Proposition 4.3 and Remark 4.
- $\pi_n \neq a\pi_{n+1}$: $\not\leq$ from Proposition 4.2 and $\not\leq$ from Proposition 4.3.
- $a\pi^- \neq \pi_m$: $\not\leq$ from Proposition 4.3 by noticing that in part (\Leftarrow) of the proof of Theorem 4.1, M belongs to $a\pi$, and $\not\leq$ from Remark 3.

According to the ordering \leq , the results above establish that $a\pi_0$ is the bottom element, ${}^e\pi$ is the top element and each subset X of $\mathcal{S}_{/\simeq}$ has limits in $\mathcal{S}_{/\simeq}$. \square

TABLE 5.1: Expressivity lattice.



From Table 5.1 emerges that the two constructs of polyadic synchronisation and mixed choice can be considered orthogonal. On the other hand, in the light of the results presented in Section 4.1, matching introduces a difference only when binary synchronisation is not available.

REMARK 5. If a comparison operator and a total order on names were provided, the leader election problem could be easily solved in the π -calculus without mixed choice, using for example the *LCR algorithm* of Chang and Roberts [1979]. Analogously, if a composition operator on names was provided as primitive, also the Matching Problem would be solvable in π -calculus: the process identifiers could be composed together to constitute a single channel name.

6. Conclusions and future work

Concluding remarks. We have extended the synchronisation mechanism of π -calculus to allow for polyadic synchronisation, where channels are vectors of names. It is a simple idea that has been adopted implicitly in many other calculi, but a formal treatment of its expressivity has not been given until now.

We have shown that matching cannot be encoded in π -calculus, whereas it is expressed naturally in terms of polyadic synchronisation. We have shown how a restricted form of polyadic synchronisation can be encoded *weakly* in π -calculus and how, in the general case, the higher the degree of synchronisation of a calculus, the greater its expressive power. We have adapted the results on mixed choice to ${}^e\pi$, concluding that polyadic synchronisation and choice are independent from one another.

We have not delved into the question of the expressivity of mismatching, and we conjecture that it is not encodable in ${}^e\pi$ (and consequently in π). Mismatching is seldom considered in the literature, it does not seem to have many applications, and it complicates the equational theory of π -calculus. A remarkable exception is the work of Frontana [2001], who proposes π_B (an extension of π -calculus with a *blocking* operator) to reason about the concept of dynamic binding in process calculi. He shows that π_B and π^{\neq} are mutually encodable. Polyadic synchronisation allows dynamic binding to be expressed in the π -calculus framework in a different way. The $D\pi$ example reported in the introduction, shows how a migrating process can gain access to the local names of a subsystem without requiring any explicit communication: the dynamic binding and re-binding of names is implicit in the semantics.

A possible interpretation of our main expressivity result is that locations are a *fundamental* concept in distributed calculi, since the attempt to encode them in models with simple synchronisation in general introduces divergence. We believe that ${}^e\pi$ has the expressive power to represent nested locations, but only in a static setting. The Ambient Calculus instead, seems to be beyond the reach of the expressive power of the model of synchronisation adopted by π -calculi.

Future work. We identify three major lines of development for ${}^e\pi$:

- (1) identify how the ability to encode cryptography, distribution and concurrent objects in the ${}^e\pi$ setting can improve the understanding of these issues;
- (2) consider how and to what extent the sorting and typing disciplines for π -calculus can be generalised to ${}^e\pi$;
- (3) consider further extensions of the synchronisation mechanism, exploring the ideas of Milner [1991] in the light of the recent work of Nestmann [1998] concerning joint input, and of our work on polyadic synchronisation.

Acknowledgements

This paper is a revised and extended version of a paper appeared in the *Proceedings of the 9th International Workshop on Expressiveness in Concurrency*, volume 68 issue 2, of *Electronic Notes in Theoretical Computer Science*, Elsevier Science, August 2002. The second author is supported by a research grant by Microsoft Research, Cambridge, UK.

We would like to thank Frank Valencia, Mogens Nielsen, Uwe Nestmann, and Antonio Ravara for comments on an earlier version of this work. We are much indebted to the anonymous referees of EXPRESS'02 for the many suggestions that have allowed us to improve a previous version of this paper. We thank Maria Grazia Vigliotti, Andrew Phillips, Nobuko Yoshida and Philippa Gardner for their useful comments and suggestions. We are grateful to Robin Milner for introducing us to his previous experience on the specific subject.

References

- ABADI, M. AND GORDON, A. D. 1998. A Calculus for Cryptographic Protocols: The Spi Calculus. Research Report 149, Digital Equipment Corporation Systems Research Center.
- AMADIO, R., BOUDOL, G., AND LOUSSHAINE, C. 1999. The Receptive Distributed π -Calculus. In *Proceedings of the 5th ECOOP Workshop on Mobile Object Systems (MOS'99)*. Lisbon, Portugal.
- BOUDOL, G. 1992. Asynchrony and the π -Calculus. Rapport de Recherche 1702, INRIA Sofi a-Antipolis.
- BOUDOL, G. AND CASTELLANI, I. 1988. Concurrency and Atomicity. *Theoretical Computer Science* 59, 1-2 (July), 25–84.
- CARBONE, M., COCCIA, M., FERRARI, G., AND MAFFEIS, S. 2001. Process Algebra-Guided Design of Java Mobile Network Applications. Extended Abstract in Inf. Proc. of FMTJP'01.
- CARBONE, M. AND MAFFEIS, S. 2002. On the Expressive Power of Polyadic Synchronisation in π -calculus. In *EXPRESS'02*, Volume 68.2 of *ENTCS*. Elsevier Science Publishers.
- CARDELLI, L. AND GORDON, A. D. 2000. Mobile Ambients. *Theoretical Computer Science* 240, 1, 177–213.
- CHANG, E. J. H. AND ROBERTS, R. 1979. An Improved Algorithm for Decentralized Extrema-Finding in Circular Configurations of Processes. *Communications of the ACM* 22, 5 (May), 281–283.
- CHOTHIA, T. AND STARK, I. 2001. A Distributed π -Calculus with Local Areas of Communication. In *Proceedings of HLCL'00*, Volume 41.2 of *ENTCS*. Elsevier Science Publishers.
- DE BOER, F. S. AND PALAMIDESSI, C. 1994. Embedding as a Tool for Language Comparison. *Information and Computation* 108, 1 (Jan.), 128–157.
- FERRARI, G. 1997. Atomicity and Concurrency Control in Process Calculi. *FUNDINF: Fundamenta Informatica* 29.4, 341–368.
- FOURNET, C. 1998. *The Join-Calculus: a Calculus for Distributed Mobile Programming*. Ph.D. thesis, Ecole Polytechnique.

- FOURNET, C. AND GONTHIER, G. 1996. The Reflexive CHAM and the Join-Calculus. In *Proceedings of the 23rd ACM Symposium on Principles of Programming Languages*, 372–385.
- FRONTANA, J. L. VIVAS. 2001. *Dynamic Binding of Names in Calculi for Mobile Processes*. Ph.D. thesis, Department of Microelectronics and Information Technology.
- HENNESSY, M. AND RIELY, J. 1998. Resource Access Control in Systems of Mobile Agents. In *Proceedings of HLCL '98*, Volume 16.3 of ENTCS. Elsevier Science Publishers, 3–17.
- HONDA, K. AND TOKORO, M. 1991. An Object Calculus for Asynchronous Communication. In *Proceedings of ECOOP*, Volume 512 of LNCS. Springer-Verlag, 133–147.
- MERRO, M. AND SANGIORGI, D. 1998. On Asynchrony in Name-Passing Calculi, Volume 1443 of LNCS. Springer-Verlag, 856–867.
- MILNER, R. 1991. Reduction and Transition Semantics for the π -Calculus with Multi-Names. Unpublished manuscript RM18.
- MILNER, R., PARROW, J., AND WALKER, D. 1992. A Calculus of Mobile Processes, I and II. *Information and Computation* 100, 1 (Sept.), 1–40,41–77.
- NESTMANN, U. 1998. On the Expressive Power of Joint Input. In *EXPRESS'98: Expressiveness in Concurrency (Nice, France, September 7, 1998)*, Volume 16.2 of ENTCS. Elsevier Science Publishers.
- NESTMANN, U. 2000. What Is a Good Encoding of Guarded Choice? *Information and Computation* 156, 287–319.
- NESTMANN, U. AND PIERCE, B. C. 2000. Decoding Choice Encodings. *Journal of Information and Computation* 163, 1–59.
- PALAMIDESSI, C. 1997. Comparing the Expressive Power of the Synchronous and the Asynchronous π -Calculus. In *Conference record of POPL'97*. ACM Press, New York, NY, USA, 256–265.
- PALAMIDESSI, C. 2002. Comparing the Expressive Power of the Synchronous and Asynchronous π -calculi. *Mathematical Structures in Computer Science*. To appear. (Extended version of Palamidessi [1997]), 1–35.
- SANGIORGI, D. AND MILNER, R. 1992. Techniques of Weak Bisimulation up to. Revised version of an article appeared in the Proc. CONCUR'92, Volume 630 of LNCS.
- SANGIORGI, D. AND WALKER, D. 2001. *The π -calculus: a Theory of Mobile Processes*. Cambridge University Press.
- SEWELL, P., WOJCIECHOWSKI, P., AND PIERCE, B. C. 1999. Location Independence for Mobile Agents: A Two-Level Architecture. In *Proceedings of ICCL'98*, Volume 1686 of LNCS. Springer-Verlag, 1–31.
- VASCONCELOS, V. T. 1994. *A Process-Calculus Approach to Typed Concurrent Objects*. PhD thesis, Keio University.
- VASCONCELOS, V. T. AND TOKORO, M. 1993. A Typing System for a Calculus of Objects. In *Object Technologies for Advanced Software*, Volume 742 of LNCS. Springer-Verlag, 460–474.

Appendix A. Proof of Theorem 3.1

In the following we will distinguish between a relation (e.g. structural congruence) in $a\pi^{\bar{\cdot},\bar{\cdot}}$ and the corresponding notion in $a\pi_2$, by labeling the former with the symbol π (e.g. \equiv_π). We report below the lemmata used in the proof of Theorem 3.1.

We start with a simple property based on the definition of barbs.

OBSERVATION 1.1. For any process P in $a\pi_2 \setminus a\pi_1$:

- (1) if $P \xrightarrow{\bar{b}\langle c \rangle} Q$ then $P \equiv \overline{b \cdot a}\langle c \rangle | Q$;
- (2) if $P \xrightarrow{\bar{b}\langle vc \rangle} Q$ then $P \equiv (vc)(\overline{b \cdot a}\langle c \rangle | Q)$;
- (3) if $P \xrightarrow{b\langle a(x) \rangle} Q$ then for some P_1, P_2 , and some \tilde{n} such that $a, b \notin \tilde{n}$, $P \equiv (v\tilde{n})(b \cdot a(x).P_1 | P_2)$ and $Q \equiv (v\tilde{n})(P_1 | P_2)$.

PROOF. Follow directly adapting Definition 2 to $e\pi$. \square

We now establish a strong operational correspondence between the actions of a term and the actions of its encoding.

LEMMA 1.1. *For any process P in $\alpha\pi_2 \setminus \alpha\pi_1$, if $P \xrightarrow{\mu} Q$ then:*

- (1) if $\mu \in \{\overline{b \cdot a}\langle c \rangle, \overline{b \cdot a}\langle \nu c \rangle\}$ then, for any $z \notin \text{fn}(P)$, $\llbracket P \rrbracket \xrightarrow{\llbracket \mu \rrbracket} \bar{z} \mid \llbracket Q \rrbracket$;
- (2) if $\mu = b \cdot a(y)$ then $\exists Q'. \llbracket P \rrbracket \xrightarrow{\tau} \xrightarrow{b(x,y,z)} \pi Q'$ and for any $\sigma : \{x, y, z\} \rightarrow \mathcal{N}$,
 - if $\sigma(x) = a$, $Q'\sigma \sim_{\pi} z\sigma \mid \llbracket Q \rrbracket \sigma$;
 - if $\sigma(x) \neq a$, $Q'\sigma \sim_{\pi} \overline{b}\langle x, y, z \rangle \sigma \mid \llbracket P \rrbracket \sigma$;
- (3) if $\mu = \tau$ then $\llbracket P \rrbracket \xrightarrow{\tau} \xrightarrow{\tau} \xrightarrow{\tau} \pi \sim_{\pi} \llbracket Q \rrbracket$.

PROOF.

- (1) If $\mu = \overline{b \cdot a}\langle c \rangle$, by Observation 1.1 we have that $P \equiv \overline{b \cdot a}\langle c \rangle \mid Q \xrightarrow{\overline{b \cdot a}\langle c \rangle} Q$. By definition of the encoding and lts, follows

$$\llbracket P \rrbracket \equiv_{\pi} (\nu z) \overline{b}\langle a, c, z \rangle \mid \bar{z} \mid \llbracket Q \rrbracket \xrightarrow{\overline{b}\langle a, c, \nu z \rangle} \bar{z} \mid \llbracket Q \rrbracket$$

Note that using alpha conversion on z before applying the (OPEN) rule, it is possible to derive an analogous transition for any $z' \notin \text{fn}(P)$.

The case for $\mu = \overline{b \cdot a}\langle \nu y \rangle$ is analogous to the previous case.

- (2) If $P \xrightarrow{b \cdot a(y)} Q$ then by Observation 1.1 we have that $P \equiv (\nu \tilde{n})(b \cdot a(y).P_1 \mid P_2)$ and $Q \equiv (\nu \tilde{n})(P_1 \mid P_2)$. Consequently, by definition of encoding and lts, $\llbracket P \rrbracket \xrightarrow{\tau} \xrightarrow{b(x,y,z)} \pi Q'$ where

$$Q' \equiv_{\pi} (\nu \tilde{n})(\nu w) \left([x = a](z \mid \llbracket P_1 \rrbracket), (\overline{b}\langle x, y, z \rangle \mid \bar{w}) \mid \right. \\ \left. !w.b(x, y, z).[x = a](z \mid \llbracket P_1 \rrbracket), (\overline{b}\langle x, y, z \rangle \mid \bar{w}) \mid \right. \\ \left. \llbracket P_2 \rrbracket \right)$$

Now for all σ , if $\sigma(x) = a$ then $Q'\sigma \equiv_{\pi}$
 $(\nu \tilde{n})(z\sigma \mid \llbracket P_1 \rrbracket \sigma) \mid (\nu w) (!w.b(x, y, z).[x = a](z \mid \llbracket P_1 \rrbracket), (\overline{b}\langle x, y, z \rangle \mid \bar{w})) \sigma \mid \llbracket P_2 \rrbracket \sigma$
 $\equiv_{\pi} z\sigma \mid (\nu \tilde{n})(\llbracket P_1 \rrbracket \mid \llbracket P_2 \rrbracket) \mid (\nu w) (!w.b(x, y, z).[x = a](z \mid \llbracket P_1 \rrbracket), (\overline{b}\langle x, y, z \rangle \mid \bar{w})) \sigma$
 $\sim_{\pi} z\sigma \mid \llbracket Q \rrbracket \sigma$;

on the other hand, if $\sigma(x) \neq a$ then,

$$Q'\sigma \equiv_{\pi} \\ (\nu \tilde{n})([\sigma(x) = a](z\sigma \mid \llbracket P_1 \rrbracket \sigma) \mid (\nu w)([\sigma(x) \neq a](\overline{b}\langle x, y, z \rangle \sigma \mid \bar{w}) \\ \mid !w.b(x, y, z).[x = a](z \mid \llbracket P_1 \rrbracket), (\overline{b}\langle x, y, z \rangle \mid \bar{w})) \sigma \mid \llbracket P_2 \rrbracket \sigma) \\ \sim_{\pi} \overline{b}\langle x, y, z \rangle \sigma \mid \llbracket P \rrbracket \sigma.$$

Note that in both cases the restrictions on \tilde{n} and w do not interfere with the names substituted by σ , because of the latter being capture-avoiding.

- (3) $\mu = \tau$. A τ action could be achieved by the rules (COM), (PAR), (CLOSE), (BANG) and (RES). We just show the (COM) case as the other ones are similar. Applying rule (COM) we have that $P = P_1 | P_2$ for some P_1, P_2 , and

$$\frac{P_1 \xrightarrow{b \cdot a(y)} Q_1, P_2 \xrightarrow{\overline{b \cdot a(c)}} Q_2}{P_1 | P_2 \xrightarrow{\tau} Q_1\{c/y\} | Q_2}$$

By point (2) there exists Q', Q'' and Q''' such that

- $\llbracket P_1 \rrbracket \xrightarrow{\tau} \pi Q''$,
- $Q' \xrightarrow{b(x,y,z)} \pi Q'$,
- and if $\sigma(x) = a$, $Q' \sigma \sim_{\pi} z \sigma \llbracket Q_1 \rrbracket \sigma$.

By point (1) and rule (CLOSE)

$$\frac{Q' \xrightarrow{b(x,y,z)} \pi Q', \llbracket P_2 \rrbracket \xrightarrow{\overline{b(a,c,yz)}} \pi \bar{z} \llbracket Q_2 \rrbracket, \sigma = \{a, c, z/x, y, z\}}{Q' | \llbracket P_2 \rrbracket \xrightarrow{\tau} \pi(\nu z)(Q' \sigma | \bar{z} \llbracket Q_2 \rrbracket)}$$

where $\sigma(a) = x$ and again by (COM) and (RES)

$$\frac{Q' \sigma \xrightarrow{z} \pi Q''', \bar{z} \llbracket Q_2 \rrbracket \xrightarrow{\bar{z}} \pi \llbracket Q_2 \rrbracket}{(\nu z)(Q' \sigma | (\bar{z} \llbracket P_2 \rrbracket)) \xrightarrow{\tau} \pi(\nu z)(Q''' | \llbracket Q_2 \rrbracket)}$$

Note that z is not free in Q''' or $\llbracket Q_2 \rrbracket$, giving us $(\nu z)(Q''' | \llbracket Q_2 \rrbracket) \equiv_{\pi} Q''' | \llbracket Q_2 \rrbracket$, which allows us to conclude, again by point (2), $Q''' | \llbracket Q_2 \rrbracket \sim_{\pi} \llbracket Q_1 \rrbracket | \llbracket Q_2 \rrbracket$.

□

The following lemma explores the relation between weak actions, from a term to its encoding.

LEMMA 1.2. *For any process P in $a\pi_2 \setminus a\pi_1$*

- (1) *if $P \xRightarrow{\tau} Q$ then $\llbracket P \rrbracket \xRightarrow{\tau} \pi \xrightarrow{\tau} \pi \xrightarrow{\tau} \pi \xrightarrow{\tau} \pi \xRightarrow{\tau} \pi \sim_{\pi} \llbracket Q \rrbracket$;*
- (2) *if $P \xRightarrow{\mu} Q$, where $\mu \in \{\overline{b \cdot a(c)}, \overline{b \cdot a(\nu y)}\}$, then for any $z \notin \text{fn}(P)$, $\llbracket P \rrbracket \xRightarrow{\llbracket \mu \rrbracket} \pi \sim_{\pi} \bar{z} \llbracket Q \rrbracket$;*
- (3) *if $P \xRightarrow{b \cdot a(y)} R \xRightarrow{\tau} Q$ then $\exists R', Q'. \llbracket P \rrbracket \xRightarrow{\tau} \pi R' \xrightarrow{\tau} \pi \xrightarrow{b(x,y,z)} \pi \xRightarrow{\tau} \pi Q'$, and for any $\sigma : \{x, y, z\} \rightarrow \mathcal{N}$,*
 - *if $\sigma(x) = a$, $Q' \sigma \sim_{\pi} z \sigma \llbracket Q \rrbracket \sigma$;*
 - *if $\sigma(x) \neq a$, $Q' \sigma \sim_{\pi} \overline{b(x,y,z)} \sigma | R' \sigma$ and $R' \sim_{\pi} \llbracket R \rrbracket$.*

PROOF.

- (1) By point (3) of Lemma 1.1 and by transitivity of \sim_{π} .
- (2) By points (1) above and (1) of Lemma 1.1.

(3) By points (1) above and (2) of Lemma 1.1.

□

We now establish a limited form of correspondence between strong actions in encoded terms and actions in the original terms. This lemma will be useful to establish the weak correspondence needed to prove soundness.

LEMMA 1.3. *For any process P in $\alpha\pi_2 \setminus \alpha\pi_1$:*

- (1) *If $\llbracket P \rrbracket \xrightarrow{\mu}_{\pi} Q'$, where $\mu \in \{\bar{b}\langle a, c, vz \rangle, \bar{b}\langle a, vc, vz \rangle\}$, then there exists Q such that $P \xrightarrow{\bar{b}\langle c \rangle} Q$ (respectively $\bar{b}\langle vc \rangle$), $Q' \equiv_{\pi} \bar{z} \mid \llbracket Q \rrbracket$ and $z \notin \text{fn}(Q)$.*
- (2) *If $\llbracket P \rrbracket \xrightarrow{\tau}_{\pi} \xrightarrow{b\langle x, y, z \rangle}_{\pi} Q'$ then there exists a, Q such that for any $\sigma : \{x, y, z\} \rightarrow \mathcal{N}$,*
 - *if $\sigma(x) = a$, $Q'\sigma \sim_{\pi} z\sigma \mid \llbracket Q \rrbracket \sigma$;*
 - *if $\sigma(x) \neq a$, $Q'\sigma \sim_{\pi} \bar{b}\langle x, y, z \rangle \sigma \mid \llbracket P \rrbracket \sigma$;**and, if $a \in \text{fn}(P)$ then $P \xrightarrow{b\langle a \rangle} Q$;*

PROOF.

- (1) By Definition 2, if $\llbracket P \rrbracket \xrightarrow{\bar{b}\langle a, c, vz \rangle}_{\pi} Q'$ then by the presence of a bound name as third parameter in the output action and by definition of encoding, $P \equiv \bar{b}\langle a \rangle \langle c \rangle \mid Q$ and $\llbracket P \rrbracket \equiv_{\pi} (vz)(\bar{b}\langle a, c, z \rangle \mid \bar{z}) \mid \llbracket Q \rrbracket$. By rule (OPEN) $\llbracket P \rrbracket \xrightarrow{\bar{b}\langle a, c, vz \rangle} \bar{z} \mid \llbracket Q \rrbracket$ and by construction $z \notin \text{fn}(Q)$.

The case with $\xrightarrow{\bar{b}\langle vc \rangle}$ is analogous.

- (2) Analogous to point (1), reasoning similarly to point (2) of Lemma 1.1.

□

The next two lemmata are technical lemmata needed to prove the weak operational correspondence between tau actions.

LEMMA 1.4. *If $\llbracket P \rrbracket \approx_{\pi} P'$ and $P' \xrightarrow{\tau}_{\pi} Q'$ then $\exists Q. \llbracket P \rrbracket \Longrightarrow_{\pi} \llbracket Q \rrbracket$ and $\llbracket Q \rrbracket \approx_{\pi} Q'$.*

PROOF. If $\llbracket P \rrbracket \approx_{\pi} Q'$ we are done taking $Q \equiv P$. If $\llbracket P \rrbracket \not\approx_{\pi} Q'$, then by the hypothesis $\llbracket P \rrbracket \approx_{\pi} P'$ we have that it must be the case that $\exists R. \llbracket P \rrbracket \Longrightarrow_{\pi} R$, and $R \approx_{\pi} Q'$ but $R \not\approx_{\pi} \llbracket P \rrbracket$. By definition of encoding, the only actions that $\llbracket P \rrbracket$ can perform without preserving weak bisimulation, come from the translation of a communication in ${}^e\pi$. We can chose a trace of actions, originating from $\llbracket P \rrbracket$ such that for each such reduction there is one and only one step of initialisation and termination of the protocol. Since all those additional steps preserve weak bisimilarity, we have that $\llbracket P \rrbracket \Longrightarrow_{\pi} R$ implies $\exists Q. \llbracket P \rrbracket \Longrightarrow_{\pi} \llbracket Q \rrbracket$ where $R \approx_{\pi} \llbracket Q \rrbracket$. By transitivity of \approx_{π} , we conclude. □

LEMMA 1.5. *If $\llbracket P \rrbracket \Longrightarrow_{\pi} \llbracket Q \rrbracket$ then $P \Longrightarrow Q$.*

PROOF. Follows by definition of encoding and by induction on the number of reduction steps. \square

We can now state the weak operational correspondence between actions in encoded terms and actions in source terms.

LEMMA 1.6. *For any process P in $\alpha\pi_2 \setminus \alpha\pi_1$:*

- (1) *If $\llbracket P \rrbracket \xrightarrow{\tau} \pi Q'$ then there exists Q such that $P \Longrightarrow Q$ and $\llbracket Q \rrbracket \approx_\pi Q'$.*
- (2) *If $\llbracket P \rrbracket \xrightarrow{\mu} \pi Q'$ where $\mu \in \{\bar{b}\langle a, c, vz \rangle, \bar{b}\langle a, vc, vz \rangle\}$ then there exists Q such that $P \xrightarrow{\bar{b}\langle c \rangle} Q$ (respectively $\bar{b}\langle a \rangle \langle vc \rangle$) and $Q' \approx_\pi \bar{z} \llbracket Q \rrbracket$ and $z \notin \text{fn}(Q)$.*
- (3) *If $\llbracket P \rrbracket \Longrightarrow \pi R' \xrightarrow{\tau} \pi \xrightarrow{b\langle x, y, z \rangle} \pi Q'$ then there exists a, Q, R such that $P \Longrightarrow R$ and, for any $\sigma : \{x, y, z\} \rightarrow \mathcal{N}$,*
 - \circ *if $\sigma(x) = a$, $Q'\sigma \approx_\pi \bar{z}\sigma \mid \llbracket Q \rrbracket \sigma$;*
 - \circ *if $\sigma(x) \neq a$, $Q'\sigma \approx_\pi \bar{b}\langle x, y, z \rangle \sigma \mid R'\sigma$ and $R' \approx_\pi \llbracket R \rrbracket$;**and if $a \in \text{fn}(P)$ then $R \xrightarrow{b\langle a \rangle} Q$.*

PROOF.

- (1) Follows from the preceding two lemmata.
- (2) Follows by point (1) above and by point (1) in Lemma 1.3.
- (3) Follows by point (1) above and by point (2) in Lemma 1.3.
- (4) Follows by point (1) above and by point (5) in Lemma 1.3.

\square

The following lemma states the soundness of the encoding.

LEMMA 1.7. *If $\llbracket P \rrbracket \approx_\pi \llbracket Q \rrbracket$ then $P \approx Q$.*

PROOF. In the proof we reason up to weak bisimilarity, following a remark of Sangiorgi and Milner [1992]: the technique is sound because we are considering weak actions both for the player and for the adversary in the bisimulation game. We split the proof in four cases.

- \circ $P \xrightarrow{\bar{b}\langle a \rangle} P_1$: by point (2) of Lemma 1.2 $\llbracket P \rrbracket \xrightarrow{\bar{b}\langle a, y, vz \rangle} \pi \sim \bar{z} \mid \llbracket P_1 \rrbracket$, for any $z \notin \text{fn}(P)$; by the hypothesis $\llbracket P \rrbracket \approx_\pi \llbracket Q \rrbracket$, $\llbracket Q \rrbracket \xrightarrow{\bar{b}\langle a, y, vz \rangle} \pi Q' \approx_\pi \bar{z} \mid \llbracket P_1 \rrbracket$; by point (2) of Lemma 1.6 we have that $\exists Q_1. Q \xrightarrow{\bar{b}\langle a \rangle} Q_1$ and $Q' \approx_\pi \bar{z} \mid \llbracket Q_1 \rrbracket$; by transitivity of \approx_π we have that $\bar{z} \mid \llbracket P_1 \rrbracket \approx_\pi \bar{z} \mid \llbracket Q_1 \rrbracket$; by definition of \approx_π , noting that z is fresh, we have $\llbracket P_1 \rrbracket \approx_\pi \llbracket Q_1 \rrbracket$; inductively, $P_1 \approx Q_1$.
- \circ $P \xrightarrow{\bar{b}\langle a \rangle \langle vy \rangle} P_1$: similar to the previous case.

- $P \xRightarrow{b \cdot a(y)} P_1$: by point (3) of Lemma 1.2, there exists P' such that $\llbracket P \rrbracket \xRightarrow{b(x,y,z)}_{\pi} P'$ and for any $\sigma : \{x, y, z\} \rightarrow \mathcal{N}$, if $\sigma(x) = a$, $P' \sigma \sim_{\pi} z \sigma \llbracket P_1 \rrbracket \sigma$, where $z \notin \text{fn}(P)$, by definition of encoding. By the hypothesis $\llbracket P \rrbracket \approx_{\pi} \llbracket Q \rrbracket$ and by definition of \approx_{π} , it must be the case that $\llbracket Q \rrbracket \xRightarrow{b(x,y,z)}_{\pi} Q' \sigma \approx_{\pi} P' \sigma$. By point (4) of Lemma 1.6, we have that there exist a', Q_1 such that, for any $\sigma : \{x, y, z\} \rightarrow \mathcal{N}$, if $\sigma(x) = a'$, then $Q' \sigma \approx_{\pi} z \sigma \llbracket Q_1 \rrbracket \sigma$, and if $a' \in \text{fn}(Q)$ then $Q \xRightarrow{b \cdot a(y)} Q_1$. Considering all the substitutions σ such that $\sigma(x) = a$ and $\sigma(z) = w$ for some fresh name w , from the hypothesis $Q' \sigma \approx_{\pi} P' \sigma$ we have that $a = a'$, obtaining $w \llbracket P_1 \rrbracket \sigma \approx_{\pi} w \llbracket Q_1 \rrbracket \sigma$. From the freshness of w follows that $\llbracket P_1 \rrbracket \sigma \approx_{\pi} \llbracket Q_1 \rrbracket \sigma$, whence by uniformity of the encoding, follows inductively $\forall \sigma. P_1 \sigma \approx_{\pi} Q_1 \sigma$.
- $P \xRightarrow{\tau} P_1$: by point (1) of Lemma 1.2, there exists P' such that $\llbracket P \rrbracket \xRightarrow{\tau} P' \sim_{\pi} \llbracket P_1 \rrbracket$; by the hypothesis $\llbracket P \rrbracket \approx_{\pi} \llbracket Q \rrbracket$ and by definition of \approx_{π} , $\exists Q'. \llbracket Q \rrbracket \xRightarrow{\tau}_{\pi} Q'$, and $P' \approx_{\pi} Q'$; by transitivity of \approx_{π} and by point (1) of Lemma 1.6 we have that $\exists Q_1. Q \xRightarrow{\tau} Q_1$ and $Q' \approx_{\pi} \llbracket Q_1 \rrbracket \approx_{\pi} \llbracket P_1 \rrbracket$; inductively, $P_1 \approx_{\pi} Q_1$.

□

Reassuming, we report the statement of Theorem 3.1 below.

(THEOREM 3.1.) *For all processes P and Q in ap_2 :*

- (1) *If $P \xrightarrow{\tau} Q$ then $\llbracket P \rrbracket \xrightarrow{\tau} \xrightarrow{\tau} \xrightarrow{\tau} \sim \llbracket Q \rrbracket$.*
- (2) *If $\llbracket P \rrbracket \xRightarrow{\tau}_{\pi} Q'$ then there exists Q such that $P \xRightarrow{\tau} Q$ and $Q' \approx \llbracket Q \rrbracket$.*
- (3) *If $\llbracket P \rrbracket \approx \llbracket Q \rrbracket$ then $P \approx Q$.*

PROOF.

- (1) Point (3) of Lemma 1.1.
- (2) Point (1) of Lemma 1.6.
- (3) Lemma 1.7.

□