Aalborg University

Faculty of Engineering and Science

Department of Computer Science

The π-calculus with polyadic synchronization

Joana Martinho

Master Thesis

Software Systems Engineering

Supervisors:

Prof. Luca Aceto
Prof. António Ravara

Delivered: September 24, 2004
Revised: October 27, 2004

Abstract

In this report we study the π-calculus with polyadic synchronization: an extension of the π-calculus that generalizes the synchronization mechanism by allowing channels to be sequences of names. In particular, we extend the π-calculus with polyadic synchronization with cryptographic primitives and then prove the last can be encoded in the regular π-calculus with polyadic synchronization. Further, we prove that the proposed encoding is sound and complete with respect to barbed congruence which we show coincides with early congruence.

Keywords

π-calculus

polyadic synchronization

expressivity

spi-calculus

matching

cryptographic primitives

Contents

List of Tables

Acknowledgements

I would like to thank my supervisors Luca Aceto and António Ravara for their guidance, support and patience during the past year. I would also like to thank Marco Carbone and Michel Boreale for useful comments on the matters discussed here.

This report represents not only part of my knowledge in the particular area of concurrency, but also the culmination of an education that has lasted for the better part of my life. I have been lucky enough to share it with very good people: colleagues, teachers, students, staff. I sincerely hope a general thank you will do to express my gratitude.

My gratitude is extended to all my friends who (like family) put up with my absence for long periods of time, with my forever changing moods, and with my mumbling of theories I hardly ever believe myself. Annabella, Gonçalo and Ana Lara at IST and the 'gang' outside IST shared with me their contagious optimism particularly when there seemed to be no light at the end of the tunnel which made it all the more special. In addition, I thank Thomas for providing me with a shelter whenever I came to Aalborg, with a huge dose of patience to hear me out as I prepared for the presentations, and with his usual tenderness. Last but not least, I thank you, Maria, for believing.

Still, when it comes to actually living with me, no one does it better than my family: parents, big brother, aunt and cousin (aka Puto).

In loving memory of Madrinha who left me – me, a 17 year-old unbeliever – with an undying faith in humanity.

Introduction

The π-calculus with polyadic synchronization proposed in [10] is an extension of the π-calculus in [1] that generalizes the synchronization mechanism by allowing channel names to be composite. The model of interaction among processes is, as in the π-calculus, based on handshaking, i.e., the simultaneous execution of input/output actions. The fact that in the π-calculus with polyadic synchronization communication is only established if the channel vectors match element-wise, enhances its expressive power with respect to the π-calculus. In particular, in [10] it is shown that the matching construct can be encoded in the π-calculus with polyadic synchronization but not in the π-calculus. In addition, it is also proven that the higher the degree of synchronization (i.e. the maximum length of the channel vectors), the higher the expressive power of the calculus.

The cryptographic π-calculus with polyadic synchronization introduced in [10] is an extension of the π-calculus with polyadic synchronization with cryptographic primitives. Much like the spi-calculus introduced in [12] - an extension of the π-calculus with constructs that allow for encryption and decryption of messages - the cryptographic π-calculus with polyadic synchronization can be used to model security protocols.

Goals and Contribution of the Report

The main goal of the present work is to prove that the π-calculus with polyadic synchronization is sufficiently expressive to model security protocols. In order to achieve this goal, we study the π-calculus with polyadic synchronization, extend it with cryptographic primitives, and then prove that we can encode the last calculus in the first. Further, the encoding we propose is sound and complete with respect to barbed congruence. We structure the presentation of our approach to the problem at hand in the following manner:

- In Chapter 1 we introduce the syntax and late labelled transition semantics of the π-calculus with polyadic synchronization as first proposed in [10]. In Section 1.3 we define for the π-calculus with polyadic synchronization four known notions of equivalence in the π-calculus: ground, late, early and open bisimilarity. In addition, we compare these notions and reach an equivalent result to that known in the literature for the π-calculus. Also in Section 1.3 we introduce the notions of barbed bisimilarity, equivalence and congruence, and conclude the last coincides with early congruence.

Although relying on a similar result obtained for the π-calculus in [16], the proof of the coincidence of the notions in the π-calculus with polyadic synchronization requires several adjustments and is, to our knowledge, an original result.

- In Chapter 2 we study the spi-calculus originally introduced in [12], but we rely on the syntax and operational semantics of the spi-calculus proposed in [23]. Our aim is to understand this calculus, used to model and reason about cryptographic protocols, so as to have some leverage to study the cryptographic π-calculus with polyadic synchronization. We synthesize some of the results in [23] and [24], add a few relevant examples, and conclude our analysis with a diagram relating contextual equivalences with bisimilarity notions. In particular, we study the idea behind the proof of the coincidence of barbed equivalence with alley bisimulation.
- In Chapter 3 we introduce the notion of encoding and correctness. In Section 3.2 we study in great detail the encoding of the match construct in the π-calculus with polyadic synchronization in [10]; this constitutes a separation result between the π-calculus and the π-calculus with polyadic synchronization. In Section 3.3 we extend the π-calculus with polyadic synchronization with cryptographic primitives as proposed in [10]. In addition, we give an operational semantics of the new calculus (to our knowledge, this had never been done before), and analyse in detail a cryptographic protocol. We rely on a proposal in [10] for the encoding of the cryptographic constructs in the π-calculus with polyadic synchronization and prove that a similar encoding is both sound and complete with respect to barbed congruence. To our knowledge, this result is original; and it constitutes evidence that the π-calculus with polyadic synchronization can be used to model security protocols.

Outline of the Report

The report is divided in four chapters concerning:

- the study of the π-calculus with polyadic synchronization;
- the study of the spi-calculus;
- the encoding of the matching and cryptographic constructs into the π-calculus with polyadic synchronization;
- the conclusions of the report and the identification of possible avenues for the future development of the work done so far.

Chapter 1

π-calculus with polyadic synchronization

The π-calculus with polyadic synchronization as introduced in [10] is a variant of the π-calculus where the channels can consist of sequences of names and communication is establish if and only if the channel vectors match element-wise.[1]

1.1 Syntax

In this section we introduce the syntax of the calculus in detail and also mention some of the main differences between this and the π-calculus. These differences will be explained in further detail in subsequent sections throughout this chapter.

Definition 1.1.1 *Processes*
Let N be a countable set of names and $x, x_1, ..., x_k, y$ range over N for some $k \in \mathbb{N}$.
The class of processes $\mathcal{P}_S$, *ranged over P, Q is defined by the following grammar:*

$$\begin{array}{lll} P & ::= \mathbf{0} & \textit{inaction} \\ & \mid \pi.P & \textit{prefix} \\ & \mid\, !P & \textit{replication} \\ & \mid (\nu x)P & \textit{restriction} \\ & \mid P|P & \textit{parallel composition} \\ & \mid P + P & \textit{choice} \end{array}$$

where the prefixes π are given by:

$$\begin{array}{lll} \pi & ::= \tau & \textit{silent action} \\ & \mid x_1 \cdot ... \cdot x_k(y) & \textit{input} \\ & \mid \overline{x_1 \cdot ... \cdot x_k}y & \textit{output} \end{array}$$

Decreasing order precedence of operators follows that of the definition, where the prefix operator has the highest precedence.

[1] We call π-calculus with biadic synchronization to the particular case of the π-calculus with polyadic synchronization where the composite channels have at most two names.

Action	Description	$fn(\alpha)$	$bn(\alpha)$	$n(\alpha)$
τ	internal action	$\emptyset$	$\emptyset$	$\emptyset$
$\overline{u}y$	free output	$n(u) \cup \{y\}$	$\emptyset$	$n(u) \cup \{y\}$
$\overline{u}(y)$	bound output	$n(u)$	$\{y\}$	$n(u) \cup \{y\}$
$u(y)$	input	$n(u)$	$\{y\}$	$n(u) \cup \{y\}$

Table 1.1: Actions

Process	Description	$fn(P)$	$bn(P)$	$n(P)$
$\mathbf{0}$	inaction	$\emptyset$	$\emptyset$	$\emptyset$
$\pi.Q$	prefix	$fn(\pi) \cup (fn(Q) \backslash bn(\pi))$	$bn(\pi) \cup bn(Q)$	$n(\pi) \cup n(Q)$
$!Q$	replication	$fn(Q)$	$bn(Q)$	$n(Q)$
$(\nu y)Q$	restriction	$fn(Q) \backslash \{y\}$	$\{y\} \cup bn(Q)$	$\{y\} \cup n(Q)$
$(Q_1 \vert Q_2)$	parallel composition	$fn(Q_1, Q_2)$	$bn(Q_1, Q_2)$	$n(Q_1, Q_2)$
$(Q_1 + Q_2)$	choice	$fn(Q_1, Q_2)$	$bn(Q_1, Q_2)$	$n(Q_1, Q_2)$

Table 1.2: Processes

Note that we use the notation a for $a.\mathbf{0}$, and that we use $(\nu z, w)P$ for $(\nu z)(\nu w)P$.

All operators used here are also present in the π-calculus and their behaviour is as expected. Nonetheless, note that restriction is made on names as in the π-calculus and not on composite channels: this allows for *partial restriction*.

One should also note that in the π-calculus with polyadic synchronization we do not need to include the match operator since it can be encoded in the calculus. This is not possible in a 'sensible' manner using the original π-calculus that, therefore, takes the match operator as a primitive. This separation result between the two calculi was proven in [10]; we refer to it again in detail in Section 3.2.

As in the π-calculus, there are four possible kinds of actions α in the present calculus as seen in Table 1.1. We let $bn(\alpha)$ denote the set of bound names in α, $fn(\alpha)$ the set of free names in α and $n(\alpha)$ the set of all names in α. We let $u = x_1 \cdot ... \cdot x_k$ and $\overline{u} = \overline{x_1 \cdot ... \cdot x_k}$, where $k \in \mathbb{N}$, represent respectively the input and output channel vectors. Then, $fn(u) = fn(\overline{u}) = n(u) = n(\overline{u}) = \{x_1, ..., x_k\}$, and $bn(u) = bn(\overline{u}) = \emptyset$.

The notions of bound and free names in a process P, denoted by $bn(P)$ and $fn(P)$ respectively, follow those of the π-calculus. We now represent that definition in Table 1.2, where $n(P)$ represents the names in the process P. Note that we use the following notation: $fn(P_1, P_2) = fn(P_1) \cup fn(P_2)$, $bn(P_1, P_2) = bn(P_1) \cup bn(P_2)$ and $n(P_1, P_2) = n(P_1) \cup n(P_2)$.

Substitution and α-convertibility are defined as in the π-calculus [3], though we now require that the latter takes into account the possibility of composite channels. Nonetheless, we provide the formal definitions next.

Definition 1.1.2 *Substitution*
Let $w \notin bn(P)$ where $P \in \mathcal{P}_S$. The substitution $\sigma = \{w/z\}$ *applied to process P, written $P\sigma$ or $P\{w/z\}$, is the process obtained by replacing each free occurrence of z in P by w. The* simultaneous substitution $\sigma = \{w_1, ..., w_n/x_1, ..., x_n\}$ *for distinct x_i applied to process P where $w_1, ..., w_n \notin bn(P)$, also written $P\sigma$, is the process obtained by simultaneously replacing each free occurrence of x_i in P by w_i where $1 \leq i \leq n$ and $n \in \mathbb{N}$.*

Note that given a substitution $\sigma = \{w/z\}$ we denote the result of applying σ to z as $\sigma(z)$. In this case, we then have that $\sigma(z) = w$.

Definition 1.1.3 *α-convertibility*
Let $u = x_1 \cdot ... \cdot x_k$ where $k \in \mathbb{N}$. A change of bound names *in a process P is the replacement of a subterm $u(z).Q$ of P by $u(w).Q\{w/z\}$ or the replacement of a subterm $(\nu z)Q$ of P by $(\nu w)Q\{w/z\}$ where in each case w does not occur in Q.*
*Two processes P and Q are α-*convertible*, written $P =_\alpha Q$, if Q can be obtained from P by a finite number of changes of bound names.*

Note that substitution may imply the renaming via α-conversion of bound actions to avoid unwanted captures of free names. We now present an example that shows the danger of performing the substitution $\sigma = \{w/z\}$ in a process P such that $w \in bn(P)$.

Example 1.1.4 *Let $P = w(w).\overline{w}z$. By definition of substitution $P\{w/z\} =_\alpha (w(v).\overline{v}z)\{w/z\} = w(v).\overline{v}w$ where we have performed the renaming of bound names so as to respect the conditions of substitution. We easily notice that the occurrences of w are still free, so the former z was not captured. However, if we do not perform the renaming operation, we have that $P\{w/z\} = w(w).\overline{w}w$ and the former z is now bound.*

The following examples show how α-convertibility can be established.

Example 1.1.5 *Let $P = (\nu y)(y \cdot z(a)|z(y))|\overline{z}a$. Then we can obtain through α-convertibility the process $Q = (\nu w)(w \cdot z(a)|z(y))|\overline{z}a$ where we replaced the subterm $(\nu y)P'$ of P with $(\nu w)P'\{w/y\}$, where $P' = y \cdot z(a)|z(y)$ and w is new.*

Example 1.1.6 *Let $P = x \cdot y(a).a(b)|\overline{x \cdot y}z$. Then we can obtain through α-convertibility the process $Q = x \cdot y(c).c(b)|\overline{x \cdot y}z$ where we replaced the subterm $x \cdot y(a).P'$ of P with $x \cdot y(c).P'\{c/a\}$ where $P' = a(b)$ and c is new.*

The importance of α-convertibility is illustrated in the Example 1.2.3 provided after the introduction of the labelled transition relation of the π-calculus with polyadic synchronization because only then can we fully understand the relevance of α-convertibility in the evolution of a process.

1.2 Late Labelled Transition Semantics

In this section we introduce the late labelled transition semantics of the π-calculus with polyadic synchronization. In addition, we provide some examples that reflect the differences between this and the π-calculus.

Definition 1.2.1 *Late labelled transition relation*
Let $u = x_1 \cdot ... \cdot x_k$, where $k \in \mathbb{N}$. The late labelled transition relation $\xrightarrow{\alpha} \subseteq \mathcal{P}_S \times \mathcal{P}_S$*, where α is a possible action, is the smallest relation generated by the set of rules in Table 1.3*[2]*.*

The rules follow in a straightforward manner those of the π-calculus where we now consider vectors of names as channels. Note once again that in the restriction rule, RES, we consider singular and not composite names. This characteristic of the calculus and the fact that we enforce an all-or-nothing behaviour, that is, we require the match of all the names in the vector channel to allow synchronization, we can define the notion of *partial restriction*. The following example reflects the consequences of this type of restriction.

Example 1.2.2 *Let $P = (\nu x_1)x_1 \cdot x_2(y)$ and $Q = x_1 \cdot x_2(y)$. Then, P cannot perform the input action because of the restriction in one of its channel names (although the other is free), while Q can. Also note that this example can be generalized to channel vectors composed of n names, i.e., where $u = x_1 \cdot ... \cdot x_n$, $P = (\nu x_i)u(y)$ and $Q = u(y)$. Because of the restriction on the channel name x_i where $i \in \{1, ..., n\}$, P is unable to perform the input action.*

Now that we have introduced the labelled transition relation for processes in the π-calculus with polyadic synchronization, we can once again reflect on the importance of α-convertibility. The following example accounts for the relevance of this operation.

Example 1.2.3 *Let $P = (S|Q)|R$ where $S = x \cdot z(y).y(b)$, $Q = z \cdot y(b)$ and $R = \overline{x \cdot z}c$. We expect S to synchronize with R in such a way that $P \xrightarrow{\tau} (c(b)|z \cdot w(b))|\mathbf{0}$, but in order for this to be achievable we need to perform an α-conversion, else the side condition of the PAR1 rule is not satisfied.*

$$\dfrac{\dfrac{\dfrac{\dfrac{--}{x \cdot z(a).a(b) \xrightarrow{x \cdot z(a)} a(b)}\ \text{PREFIX}}{x \cdot z(y).y(b) \xrightarrow{x \cdot z(a)} a(b)}\ \text{CONV}}{x \cdot z(y).y(b)|z \cdot y(b) \xrightarrow{x \cdot z(a)} a(b)|z \cdot y(b)}\ \text{PAR1} \qquad \dfrac{--}{\overline{x \cdot z}c \xrightarrow{\overline{x \cdot z}c} \mathbf{0}}\ \text{PREFIX}}{P \xrightarrow{\tau} (c(b)|z \cdot y(b))|\mathbf{0}}\ \text{COMM}$$

[2] Note that not included in the table are four rules: the symmetric form CH2 of CH1 which has $Q + P$ instead of $P + Q$, and the symmetric forms PAR2, COMM2 and CLOSE2 of PAR1, COMM1, CLOSE1 in which the roles of the left and right components are swapped.

(PREFIX)	$\dfrac{-}{\alpha.P \xrightarrow{\alpha} P}$	(CH1)	$\dfrac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'}$
(PAR1)	$\dfrac{P \xrightarrow{\alpha} P'}{P\|Q \xrightarrow{\alpha} P'\|Q}$	where $bn(\alpha) \cap fn(Q) = \emptyset$	
(RES)	$\dfrac{P \xrightarrow{\alpha} P'}{(\nu y)P \xrightarrow{\alpha} (\nu y)P'}$	where $y \notin n(\alpha)$	
(REP-ACT)	$\dfrac{P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P'\|!P}$		
(REP-COMM)	$\dfrac{P \xrightarrow{\overline{u}y} P' \quad P \xrightarrow{u(z)} P''}{!P \xrightarrow{\tau} (P'\|P''\{y/z\})\|!P}$		
(REP-CLOSE)	$\dfrac{P \xrightarrow{\overline{u}(y)} P' \quad P \xrightarrow{u(y)} P''}{!P \xrightarrow{\tau} (\nu y)(P'\|P'')\|!P}$	where $y \notin fn(P)$	
(OPEN)	$\dfrac{P \xrightarrow{\overline{u}y} P'}{(\nu y)P \xrightarrow{\overline{u}(y)} P'}$	where $y \notin n(\overline{u})$	
(CLOSE1)	$\dfrac{P \xrightarrow{\overline{u}(y)} P' \quad Q \xrightarrow{u(y)} Q'}{P\|Q \xrightarrow{\tau} (\nu y)(P'\|Q')}$		
(COMM1)	$\dfrac{P \xrightarrow{\overline{u}y} P' \quad Q \xrightarrow{u(z)} Q'}{P\|Q \xrightarrow{\tau} P'\|Q'\{y/z\}}$		
(CONV)	$\dfrac{P \xrightarrow{\alpha} P'}{Q \xrightarrow{\alpha} P'}$	if $Q =_\alpha P$	

Table 1.3: Late transition rules

Note that if we had not performed the α-conversion and had disrespected the side condition of the PAR1 rule then P would have evolved through a τ action into $c(b)|z \cdot c(b)$.

1.3 Observational Semantics

In this section we seek to introduce different notions of bisimulation: a technique that allows us to equate processes in the π-calculus with polyadic synchronization. We define these notions of bisimulation following those known in literature [1, 4] for the π-calculus with the necessary adjustments.

The first notion we will consider is that of ground bisimulation where there is no name instantiation.

Definition 1.3.1 *Ground bisimilarity*
A binary symmetric relation $\mathcal{S}$ is a ground bisimulation *if $P\mathcal{S}Q$ implies:*

- *if $P \xrightarrow{\alpha} P'$ where $bn(\alpha) \cap fn(P, Q) = \emptyset$ then there is a Q' such that $Q \xrightarrow{\alpha} Q'$ and $P'\mathcal{S}Q'$.*

Two processes P and Q are ground bisimilar *if $P\mathcal{S}Q$ for some ground bisimulation $\mathcal{S}$.* Ground bisimilarity*, written $\sim_g$, is the largest ground bisimulation*[3]*.*

The notion of ground bisimilarity is very simple since a process merely has to imitate the other in its possible transitions and vice versa without considering name instantiation. Unfortunately, as in the π-calculus, a consequence of this is that ground bisimilarity is not preserved by the parallel composition operator as seen in the following example. We generalize this example to channel vectors of n names, where $n \in \mathbb{N}$ in Example 1.3.3.

Example 1.3.2 *Let $P = (\nu a)(z(w).\overline{a \cdot w}c|a \cdot y(b))$ and $Q = z(w)$. Then both P and Q are ground bisimilar since after performing the input action they both become inactive. Let $S = \overline{z}y$ and consider $P' = P|S$ and $Q' = Q|S$. Then P' can perform an internal action through synchronization between P and S and evolve into $(\nu a)(\overline{a \cdot y}c|a \cdot y(b))$ which can also perform an internal action. Thus P' can perform two consecutive internal actions while Q' can only perform one internal action and then becomes inactive. We can then conclude that although P and Q are ground bisimilar, P' and Q' are not ground bisimilar.*

Example 1.3.3 *Let $u = x_1 \cdot ... \cdot x_n$, $P = (\nu x_i)(z(w).\overline{u \cdot w}c|u \cdot y(b))$ where $i \in \{1, ..., n\}$, and $Q = z(w)$. Then both P and Q become inactive after performing the input action, and so $P \sim_g Q$. Let $S = \overline{z}y$ and consider $P' = P|S$ and $Q' = Q|S$. Then P' can evolve into $(\nu x_i)(\overline{u \cdot y}c|u \cdot y(b))$ which can perform another internal action, while Q' can only perform one τ action. Thus, although $P \sim_g Q$, we have that $P \not\sim_g Q$.*

[3] The existence and uniqueness of a largest bisimulation is a direct result of the Knoster-Tarski's Fixed Point Theorem.

In the following example we show that ground bisimilarity is not preserved by replication. Note that in the definition of P we use polyadic CCS-like prefixes $\overline{a \cdot w}$ and $a \cdot y$ where no item is being sent or expected to be received. We do this to highlight the fact that what could be transmitted is irrelevant, the problem lies on the synchronization of the composite channels. Thus, in general, $\overline{u}.P$ will be used as shorthand for $\overline{u}y.P$ for some y, and $u.P$ will be used as shorthand for $u(y).P$ where $y \notin fn(P)$.

Example 1.3.4 *Let $P = (\nu a)(z(w).\overline{a \cdot w}|a \cdot y).\overline{z}x + \overline{z}y$ and $Q = z(w) + \overline{z}y$, where w and y are distinct. Then $P \sim_g Q$, but $!P \not\sim_g !Q$ since two copies of P and two copies of Q can synchronize and the resulting processes are not bisimilar. In detail, $!P \xrightarrow{\tau}\xrightarrow{\tau}\xrightarrow{\overline{z}x} P'$, but no descendant of $!Q$ can ever perform an output action $\overline{z}x$.*

Nonetheless, ground bisimilarity is preserved, just like in the π-calculus, by some operators as stated and proved in Proposition 1.3.6.

Lemma 1.3.5 *$\sim_g$ is preserved by the restriction operator*

PROOF: Let P, $Q \in \mathcal{P}_S$ such that $P \sim_g Q$. We establish the proof by performing a case analysis on the rule used to infer an action for $(\nu x)P$, where we assume that $bn(\alpha) \cap fn(P, Q) = \emptyset$.

- Application of rule RES where $x \notin n(\alpha)$:

$$\frac{P \xrightarrow{\alpha} P'}{(\nu x)P \xrightarrow{\alpha} (\nu x)P'}$$

By definition of $\sim_g$, since $P \xrightarrow{\alpha} P'$ we have that $Q \xrightarrow{\alpha} Q'$ and $P' \sim_g Q'$. Therefore, by application of rule RES, $(\nu x)Q \xrightarrow{\alpha} (\nu x)Q'$ where $x \notin n(\alpha)$.

- Application of rule OPEN where $\overline{u} = \overline{x_1 \cdot ... \cdot x_k}$ for some $k \in \mathbb{N}$ and $x \notin n(\overline{u})$:

$$\frac{P \xrightarrow{\overline{u}x} P'}{(\nu x)P \xrightarrow{\overline{u}(x)} P'}$$

By definition of $\sim_g$, since $P \xrightarrow{\overline{u}x} P'$ we have that $Q \xrightarrow{\overline{u}x} Q'$ and $P' \sim_g Q'$. Therefore, by application of rule OPEN, $(\nu x)Q \xrightarrow{\overline{u}(x)} Q'$ where $x \notin n(\overline{u})$.

□

Proposition 1.3.6 *$\sim_g$ is preserved by all operators except parallel composition and replication.*

PROOF: We split the proof into three parts, reflecting the preservation of ground bisimilarity by each of the three operators.
If P, P', $Q \in \mathcal{P}_S$ such that $P \sim_g Q$ then:

- $\pi.P \sim_g \pi.Q$ where π is a prefix: similar to proof in [1]
- $P + P' \sim_g Q + P'$: similar to proof in [1]
- $(\nu x)P \sim_g (\nu x)Q$: proven in Lemma 1.3.5

Examples 1.3.3, 1.3.4 prove that ground bisimilarity is not preserved by parallel composition nor by replication.

□

Thus the congruence properties appear to steam directly from those of the π-calculus. However, it was also proven in [9, 14] that ground bisimilarity was a full congruence in the asynchronous π-calculus without match and this result does not hold if we consider the asynchronous π-calculus with polyadic synchronization as seen in Example 1.3.3. The reason why this result does not hold is related to the fact that match does not need to be considered as a primitive in the π-calculus with polyadic synchronization (synchronous or asynchronous) since it can be derived.

Before we introduce other notions of bisimilarity, we state the following result which is used in the proofs ahead together with the notion of 'bisimulation up to' which we introduce next. Note that the first lemma concerns the commonly denoted *structural properties* which are preserved by the notion of bisimilarity.

Definition 1.3.7 *Structural congruence*
Structural congruence*, written $\equiv$, is the smallest congruence on the processes that satisfies the following axioms where P, Q, $R \in \mathcal{P}_S$ and z, $w \in N$. Any two processes related by these axioms are called* structurally congruent*.*

- $P + (Q + R) \equiv (P + Q) + R$
- $P + Q \equiv Q + P$
- $P + \mathbf{0} \equiv P$
- $P|(Q|R) \equiv (P|Q)|R$
- $P|Q \equiv Q|P$
- $P|\mathbf{0} \equiv P$
- $(\nu z)(\nu w)P \equiv (\nu w)(\nu z)P = (\nu w, z)P$
- $(\nu z)\mathbf{0} \equiv \mathbf{0}$
- $(\nu z)(P|Q) \equiv P|(\nu z)Q$ *if* $z \notin fn(P)$
- $!P \equiv P|!P$

Lemma 1.3.8 [4] *Let P, $Q \in \mathcal{P}_S$. If $P \equiv Q$ then $P \sim_g Q$.*

The notion of *bisimulation up to* follows that introduced for CCS in [7] and so does the result presented as 1.3.10.

Definition 1.3.9 [5]*Ground bisimulation up to $\sim_g$*
A binary symmetric relation $\mathcal{S}$ is a ground bisimulation up to $\sim_g$ *if $P\mathcal{S}Q$ implies:*

- *if $P \xrightarrow{\alpha} P'$ where $bn(\alpha) \cap fn(P, Q) = \emptyset$ then there is a Q' such that $Q \xrightarrow{\alpha} Q'$ and $P' \sim_g \mathcal{S} \sim_g Q'$.*

Proposition 1.3.10 [6] *If $P\mathcal{S}Q$ where $\mathcal{S}$ is a ground bisimulation up to $\sim_g$ then $P\mathcal{S}'Q$ where $\mathcal{S}'$ is a ground bisimulation.*

Proof: The proof follows that in [7] with the necessary adjustments since we are considering the π-calculus with polyadic synchronization. The proof can be split into proving firstly that $\sim_g \mathcal{S} \sim_g$ is a ground bisimulation and secondly that $\mathcal{S}$ is included in $\sim_g \mathcal{S} \sim_g$.

□

Note that we take for granted the reader remembers these notions since we use the Lemma 1.3.8 combined with Proposition 1.3.10 so as to simplify many of the proofs throughout the report.

We now introduce the notions of late and early bisimilarity which differ in their treatment of name instantiation for input actions.

In late bisimilarity we require that the derivative of a process simulates the derivative of the other process (and vice versa) for all possible instantiations of the bound parameter. It is called late because the choice of the name instantiation is made after the choice of the derivative.

Definition 1.3.11 *Late bisimilarity*
Let $u = x_1 \cdot ... \cdot x_k$ where $k \in \mathbb{N}$.
A binary symmetric relation $\mathcal{S}$ is a late bisimulation *if $P\mathcal{S}Q$ implies:*

- *if $P \xrightarrow{\alpha} P'$ where $\alpha = \overline{u}y, \overline{u}(y)$ or τ and $bn(\alpha) \cap fn(P, Q) = \emptyset$ then there is a Q' such that $Q \xrightarrow{\alpha} Q'$ and $P'\mathcal{S}Q'$.*

- *if $P \xrightarrow{u(y)} P'$ where $y \notin fn(P, Q)$ then there is a Q' such that $Q \xrightarrow{u(y)} Q'$ and for each w, $P'\{w/y\}\mathcal{S}Q'\{w/y\}$.*

[4] The same result holds for all the notions of bisimilarity presented in this section.

[5] An analogous definition can be presented for all notions of bisimilarity introduced in this section.

[6] The same result holds for any of the notions of bisimilarity presented in this section.

Two processes P and Q are late bisimilar *if $P\mathcal{S}Q$ for some late bisimulation $\mathcal{S}$.*
Late bisimilarity*, written $\sim_l$, is the largest late bisimulation.*

In early bisimilarity we require that under the same possible name instantiation there is a derivative of each of the processes that simulates the other and vice versa. It is named early because the choice of the name instantiation is made before the choice of the derivative.

Definition 1.3.12 *Early bisimilarity*
Let $u = x_1 \cdot ... \cdot x_k$ where $k \in \mathbb{N}$.
A binary symmetric relation $\mathcal{S}$ is an early bisimulation *if $P\mathcal{S}Q$ implies:*

- *if $P \xrightarrow{\alpha} P'$ where $\alpha = \overline{u}y, \overline{u}(y)$ or τ and $bn(\alpha) \cap fn(P,Q) = \emptyset$ then there is a Q' such that $Q \xrightarrow{\alpha} Q'$ and $P'\mathcal{S}Q'$.*

- *if $P \xrightarrow{u(y)} P'$ where $y \notin fn(P,Q)$ then for each w there is a Q' such that $Q \xrightarrow{u(y)} Q'$ and $P'\{w/y\}\mathcal{S}Q'\{w/y\}$.*

Two processes P and Q are early bisimilar *if $P\mathcal{S}Q$ for some early bisimulation $\mathcal{S}$.*
Early bisimilarity*, written $\sim_e$, is the largest early bisimulation.*

Similarly to what happens in the π-calculus, in the π-calculus with polyadic synchronization both late and early bisimilarity are not preserved by input prefixing. This is evidenced by the following example where we consider processes in the π-calculus with biadic synchronization.

Example 1.3.13 *Let $P = (\nu a)(\overline{a \cdot z}c | a \cdot y(b))$ and $Q = \mathbf{0}$. Since both P and Q are unable to perform any action, we have that P and Q are late and early bisimilar. Now consider the processes $P' = z(y).P$ and $Q' = z(y).\mathbf{0}$. Then $P' \xrightarrow{z(y)} P$ and $Q' \xrightarrow{z(y)} \mathbf{0}$, but for $w = z$ we have that $P\{w/y\} \xrightarrow{\tau}$ while $\mathbf{0} \not\xrightarrow{\tau}$. Thus, P' and Q' are neither early nor late bisimilar.*

We can now generalize the previous example for the π-calculus with polyadic synchronization, by considering channel vectors of n names, where $n \in \mathbb{N}$.

Example 1.3.14 *Let $u = x_1 \cdot ... \cdot x_n$, $P = (\nu x_i)(\overline{u \cdot z}c | u \cdot y(b))$ where $i \in \{1, ..., n\}$ and $Q = \mathbf{0}$. Since both P and Q are inactive, we have that $P \sim_e Q$ and $P \sim_l Q$. Now consider the processes $P' = z(y).P$ and $Q' = z(y).\mathbf{0}$. Then $P' \xrightarrow{z(y)} P$ and $Q' \xrightarrow{z(y)} \mathbf{0}$, and for $w = z$ we have that $P\{w/y\} \xrightarrow{\tau}$ while $\mathbf{0} \not\xrightarrow{\tau}$. Thus, $P' \not\sim_l Q'$ and $P' \not\sim_e Q'$.*

As in the π-calculus, late and early bisimilarity are preserved by all other operators as shown in the following propositions.

Proposition 1.3.15 *$\sim_l$ is preserved by all operators except input prefixing.*

PROOF: The proof follows that in [1] for the output prefixing, choice and parallel composition operators and [15] for the replication operator. Example 1.3.14 proves that $\sim_l$ is not preserved by input prefixing.

□

Proposition 1.3.16 $\sim_e$ *is preserved by all operators except input prefixing.*

PROOF: The proof is similar to that of Proposition 1.3.15 and Example 1.3.14 also shows that $\sim_e$ is not preserved by input prefixing.

□

In the original π-calculus congruences for late and early bisimilarity were achieved by closing the equivalences over all name substitutions [1]. We now introduce similar notions for the π-calculus with polyadic synchronization and get similar results.

Definition 1.3.17 *Late congruence*
Let P, $Q \in \mathcal{P}_S$. The two processes are late congruent*, written $P \simeq_l Q$, if for all substitutions σ we have that $P\sigma \sim_l Q\sigma$.*

Definition 1.3.18 *Early congruence*
Let P, $Q \in \mathcal{P}_S$. The two processes are early congruent*, written $P \simeq_e Q$, if for all substitutions σ we have that $P\sigma \sim_e Q\sigma$.*

The relation between the notions of late bisimilarity and late congruence, and of early bisimilarity and early congruence, are shown in the following propositions.

Proposition 1.3.19 $\simeq_l \subset \sim_l$

PROOF: The inclusion follows directly from the definitions of late bisimilarity and late congruence because for P, $Q \in \mathcal{P}_S$, if $P \simeq_l Q$ then for all substitutions σ we have that $P\sigma \sim_l Q\sigma$. In particular, this is true for the identity substitution, that is, $P \sim_l Q$.
The following example is evidence of the strictness of the inclusion. Let P, $Q \in \mathcal{P}_S$ and distinct x, y, z, $w \in N$. If $P = (\nu w)(\overline{w \cdot x}a|w \cdot y(b))$ and $Q = (\nu w)(\overline{w \cdot x}a|w \cdot z(b))$, then $P \sim_l Q$ since both processes are inactive. Nonetheless, for $\sigma = \{y/x\}$, $P\sigma$ can perform a τ action, but $Q\sigma$ remains inactive. Thus, $P\sigma \not\sim_l Q\sigma$, and so $P \not\simeq_l Q$.

□

Proposition 1.3.20 $\simeq_e \subset \sim_e$

PROOF: The inclusion follows directly from the definitions of early bisimilarity and early congruence and is similar to that of Proposition 1.3.19. The same example given in Proposition 1.3.19 can be used to prove the strictness of the inclusion, since $P \sim_e Q$ but $P \not\simeq_e Q$.

□

The notion of open bisimilarity for the π-calculus was introduced in [4] and proven to be a congruence relation in that calculus. We now present the same notion for the π-calculus with polyadic synchronization.

Definition 1.3.21 *Open bisimilarity*
A binary symmetric relation $\mathcal{S}$ is an open bisimulation *if $P\mathcal{S}Q$ implies for every substitution σ:*

- *If $P\sigma \xrightarrow{\alpha} P'$ with $bn(\alpha) \cap fn(P\sigma, Q\sigma) = \emptyset$ then there is a Q' such that $Q\sigma \xrightarrow{\alpha} Q'$ and $P'\mathcal{S}Q'$.*

Two processes P and Q are open bisimilar *if $P\mathcal{S}Q$ for some open bisimulation $\mathcal{S}$.*
Open bisimilarity*, written $\sim_o$, is the largest open bisimulation.*

Proposition 1.3.22 *$\sim_o$ is preserved by all operators*

Proof: The proof follows that in [4].

□

We now analyse the relationships between the bisimulation relations previously defined and present a general diagram that summarizes these results in Corollary 1.3.29. The results and proofs are similar to those presented for the π-calculus [1, 5].

In order to prove that the largest open bisimulation is itself a late bisimulation, which is enough to prove that $\sim_o \subset \sim_l$, we first introduce the following lemma.

Lemma 1.3.23 *The relation $\{(P\sigma, Q\sigma) : P \sim_o Q\} \cup \sim_o$, where σ is a substitution, is an open bisimulation.*

Proof: Let $P_1 = P\sigma_1$ and $Q_1 = Q\sigma_1$ where σ_1 is a substitution.
By hypothesis $P \sim_o Q$ and for some substitution σ_2 we have that $P\sigma_2 \xrightarrow{\alpha} P'$. Since $P \sim_o Q$, by definition of open bisimilarity, we know that there is a Q' such that $Q\sigma_2 \xrightarrow{\alpha} Q'$ and $P' \sim_o Q'$. Also, note that the composition of two substitutions is itself a substitution, so by definition of open bisimilarity, if $P \sim_o Q$ and $P\sigma_1\sigma_2 \xrightarrow{\alpha} P'$ then there is a Q' such that $Q\sigma_1\sigma_2 \xrightarrow{\alpha} Q'$ and $P' \sim_o Q'$.

□

Proposition 1.3.24 $\sim_o \subset \sim_l$

Proof: We split the proof in two parts, first proving that $\sim_o$ is itself a late bisimulation and then the strictness of the inclusion.

- Let $u = x_1 \cdot \ldots \cdot x_k$ for some $k \in \mathbb{N}$, and let P, $Q \in \mathcal{P}_S$ such that $P \sim_o Q$.
  By definition of open bisimilarity we know that for every substitution σ if $P\sigma \xrightarrow{\alpha} P'$ then there is a Q' such that $Q\sigma \xrightarrow{\alpha} Q'$ and $P' \sim_o Q'$. Considering the substitution σ as the identity, $\sim_o$ immediately satisfies the first condition to be a late bisimulation. Similarly, if $P \xrightarrow{u(y)} P'$ where $y \notin fn(P, Q)$ then there is a Q' such that $Q \xrightarrow{u(y)} Q'$ and $P' \sim_o Q'$ (using the identity substitution). By Lemma 1.3.23 we have that $P'\sigma \sim_o Q'\sigma$ for every substitution σ. In particular, $P'\{w/y\} \sim_o Q'\{w/y\}$ for all w.

- In order to prove the strictness of the inclusion, consider the following processes $P = (\nu x)(x \cdot y(a) | \overline{x \cdot z}b)$ and $Q = (\nu y)(x \cdot y(a) | \overline{z \cdot y}b)$. Then P and Q are late bisimilar because both processes are inactive. However, P and Q are not open bisimilar since for $\sigma = \{z/y\}$ we have that $P\sigma \xrightarrow{\tau}$ but $Q =_\alpha Q' = (\nu k)(x \cdot k(a) | \overline{z \cdot k}b)$ and $Q'\sigma \overset{\tau}{\not\longrightarrow}$ (note that we could not apply σ directly to Q because y occurs bound in Q).

□

We shall now prove that open bisimilarity is not only a late bisimulation but also a late congruence.

Proposition 1.3.25 $\sim_o \subset \simeq_l$

Proof: We split the proof in two parts: first proving that $\sim_o$ is itself a late congruence, and then proving the strictness of the inclusion.

- Let P, $Q \in \mathcal{P}_S$ and $P \sim_o Q$.
  By definition of open bisimilarity we know that for every substitution σ if $P\sigma \xrightarrow{\alpha} P'$ then there is a Q' such that $Q\sigma \xrightarrow{\alpha} Q'$ and $P' \sim_o Q'$. By Lemma 1.3.23 we know that $P'\sigma \sim_o Q'\sigma$, and by Proposition 1.3.24 we have that $P'\sigma \sim_l Q'\sigma$. Thus, for every substitution σ we have that $P\sigma \sim_l Q\sigma$, that is, $P \simeq_l Q$.

- In order to prove the strictness of the inclusion, consider the following processes $P = c(a).(\tau.\tau + \tau)$ and $Q = (\nu z)c(a).(\tau.\tau + \tau + \tau.(\overline{z \cdot a}|z \cdot b).\tau)$. Then P and Q are late bisimilar since if $Q \xrightarrow{c(a)} (\nu z)(\tau.\tau + \tau + \tau.(\overline{z \cdot a}|z \cdot b).\tau)$ then $P = \tau.\tau + \tau$ and for $w \neq b$ $Q\{w/a\} \sim_l \tau$ or $Q\{b/a\} \sim_l \tau.\tau$. In addition, P and Q are late congruent since the only relevant substitutions would be $\sigma = \{b/a\}$ or $\sigma = \{a/b\}$ but these are impossible because a is bound in Q. Nonetheless, P and Q are not open bisimilar. If $Q \xrightarrow{c(a)}\xrightarrow{\tau} (\nu z)((\overline{z \cdot a}|z \cdot b).\tau)$ then $P \xrightarrow{c(a)}\xrightarrow{\tau} P'_1 = \tau$ or $P \xrightarrow{c(a)}\xrightarrow{\mathbf{0}} P'_2 = \tau$; but for $w \neq b$ then $Q\{w/a\} \overset{\tau}{\not\longrightarrow}$ but $P'_1\{w/a\} \xrightarrow{\tau}$, and $Q\{b/a\} \xrightarrow{\tau}$ but $P'_2\{b/a\} \overset{\tau}{\not\longrightarrow}$. Thus it does not hold that for all substitutions σ if $Q\sigma \xrightarrow{\alpha} Q'$ then $P\sigma \xrightarrow{\alpha} P'$ and $P' \sim_o Q'$.

□

We now prove that late bisimilarity is itself an early bisimulation and present an example that proves the reverse does not hold.

Proposition 1.3.26 $\sim_l \subset \sim_e$

PROOF: We split the proof in two parts, first proving that $\sim_l$ is itself an early bisimulation and then the strictness of the inclusion.

- Let $u = x_1 \cdot ... \cdot x_k$ for some $k \in \mathbb{N}$, and let P, $Q \in \mathcal{P}_S$ such that $P \sim_l Q$.
  The first condition of early bisimilarity is identical to that of late, so we need only concern ourselves with the input clause.
  By definition of late bisimilarity, if $P \xrightarrow{u(y)} P'$ and $y \notin fn(P, Q)$ then there is a Q' such that $Q \xrightarrow{u(y)} Q'$ and for all w we have that $P'\{w/y\} \sim_l Q'\{w/y\}$. Then, if $P \xrightarrow{u(y)} P'$ with $y \notin fn(P, Q)$ then for each w there is a Q' such that $Q \xrightarrow{u(y)} Q'$ and $P'\{w/y\} \sim_l Q'\{w/y\}$ which corresponds to the second condition of the definition of early bisimulation.

- In order to prove the strictness of the inclusion, consider the following processes $P = x(y).\tau + x(y)$ and $Q = P + x(y).(y \cdot u(a)|\overline{z \cdot u}b)$. Then, $Q \xrightarrow{x(y)} Q' = y \cdot u(a)|\overline{z \cdot u}b$ and $P \xrightarrow{x(y)} P_1' = \tau$ or $P \xrightarrow{x(y)} P_2' = \mathbf{0}$. So, on the one hand, we have that P and Q are not late bisimilar because for $w \neq z$ we have that $P_1'\{w/y\} \xrightarrow{\tau}$ but $Q'\{w/y\} \overset{\tau}{\not\longrightarrow}$, and for $w = z$ we have that $Q'\{w/y\} \xrightarrow{\tau}$ but $P_2'\{w/y\} \overset{\tau}{\not\longrightarrow}$. On the other hand, we have that P and Q are early bisimilar since for $w = z$ we have that $P_1'\{w/y\}$ can match $Q'\{w/y\}$, and for $w \neq z$ then both $Q'\{w/y\}$ and $P_2'\{w/y\}$ are inactive.

□

The same result holds if we consider the notions of late and early congruences instead of late and early bisimilarity, as shown in the following proposition.

Proposition 1.3.27 $\simeq_l \subset \simeq_e$

PROOF: The inclusion follows directly from the definitions of late and early congruences as well as from Proposition 1.3.26. The strictness of the inclusion is shown by the example provided in the proof of Proposition 1.3.26. The processes $P = x(y).\tau + x(y)$ and $Q = P + x(y).(y \cdot u(a)|\overline{z \cdot u}b)$ are not only early bisimilar but also early congruent. Since the processes P and Q are not late bisimilar, by Proposition 1.3.19 $P \not\simeq_l Q$.

□

Our last result shows that if two processes are early bisimilar then they are also ground bisimilar. In addition, we provide an example that proves the reverse does not hold.

Proposition 1.3.28 $\sim_e \subset \sim_g$

PROOF: We split the proof in two parts, first proving that $\sim_e$ is itself a ground bisimulation and then the strictness of the inclusion.

- Let $u = x_1 \cdot ... \cdot x_k$ for some $k \in \mathbb{N}$, and let P, $Q \in \mathcal{P}_S$.
  The first clause of early bisimulation is identical to that of ground so we need only concern ourselves with the input clause.
  By definition of early bisimilarity we have that if $P \xrightarrow{u(y)} P'$ where $y \notin fn(P, Q)$ then for each w there is a Q' such that $Q \xrightarrow{u(y)} Q'$ and $P'\{w/y\} \sim_e Q'\{w/y\}$. In particular, if we let $w = y$ the second clause is identical to the one for ground bisimilarity in the case of input.

- In order to prove that the inclusion is strict, consider the following processes $P = (\nu a)(z(w).\overline{a \cdot w}c|a \cdot y(b))$ and $Q = z(w)$. Then, P and Q are ground bisimilar because both become inactive after performing the input action. Nonetheless, P and Q are not early bisimilar since $P \xrightarrow{z(w)} P'$ and $Q \xrightarrow{z(w)} Q'$ but $P'\{y/w\}$ can perform an internal action while $Q'\{y/w\}$ can not.

□

In this section we have introduced the observational semantics of the π-calculus with polyadic synchronization, presented several definitions of bisimulation and established the correspondence between these relations. We now summarize these results in the following diagram where → stands for strict inclusion $\subset$.

Corollary 1.3.29

$$\begin{tikzcd}
\sim_o \arrow[r] \arrow[dr] & \sim_l \arrow[r] & \sim_e \arrow[r] & \sim_g \\
 & \simeq_l \arrow[u] \arrow[r] & \simeq_e \arrow[u] \arrow[ur] &
\end{tikzcd}$$

We now prove that the contextual equivalence coincides with early bisimilarity; we rely on a similar result obtained for the π-calculus in [16].

Definition 1.3.30 *Barbs*
A barb *is an input or output channel. The predicate* P exhibits barb β*, written* $P \downarrow_\beta$*, is defined by:*

- $P \downarrow_u$ *if* P *can perform an input action on channel* u

- $P \downarrow_{\overline{u}}$ *if* P *can perform an output action on channel* u

Note that the predicate just defined concerns only visible and immediate actions, as seen in the following example.

Example 1.3.31 *Let all considered names be distinct,* $P = (x \cdot y(a)|\overline{x \cdot y}b).\overline{z}c + \overline{w \cdot x}c$ *and* $Q = (\nu x)(x \cdot y(a)|\overline{x \cdot y}b).\overline{z}c$*. Then,* $P \downarrow_{x \cdot y}$*,* $P \downarrow_{\overline{x \cdot y}}$*,* $P \downarrow_{\overline{w \cdot x}}$ *but* $P \not\downarrow_{\overline{z}}$*. Further,* Q *exhibits no barbs.*

We now introduce the notion of barbed bisimilarity as proposed in [21].

Definition 1.3.32 *Barbed bisimilarity*
A binary symmetric relation $\mathcal{S}$ is a barbed bisimulation *if $P\mathcal{S}Q$ implies:*

- *if $P \downarrow_\beta$ then $Q \downarrow_\beta$ for each barb β*

- *if $P \xrightarrow{\tau} P'$ then there exists a Q' such that $Q \xrightarrow{\tau} Q'$ and $P'\mathcal{S}Q'$*

Two processes P and Q are barbed bisimilar *if $P\mathcal{S}Q$ for some barbed bisimulation $\mathcal{S}$.* Barbed bisimilarity*, written $\sim_b$, is the greatest barbed bisimulation.*

The following example shows the difference between barbed bisimilarity and the notions of bisimilarity introduced.

Example 1.3.33 *Let $P = \overline{m}n.\overline{m}n$ and $Q = \overline{m}n$. Then, P and Q are barbed bisimilar since their only barb is $\overline{m}$. However, P and Q are not ground, nor late, nor early nor open bisimilar since $P \xrightarrow{\overline{m}n} P'$ and $Q \xrightarrow{\overline{m}n} Q'$ but P' and $Q' = \mathbf{0}$ are not bisimilar since P' can still perform an output action while Q' is inactive.*

Note also that barbed bisimilarity is not a congruence since it is not preserved by parallel composition, nor by replication, nor by substitution as seen in the following examples. Example 1.3.36 is a proposed exercise in [3].

Example 1.3.34 *Let $P = \overline{m}n.\overline{m}n$, $Q = \overline{m}n$ and $R = m(x)$. As seen in the previous example $P \sim_b Q$, and trivially $R \sim_b R$. Nonetheless, $P|R \not\sim_b Q|R$ since $P|R \xrightarrow{\tau} P' = \overline{m}n$ and $P' \downarrow_{\overline{m}}$ but $Q|R \xrightarrow{\tau} \mathbf{0}$.*

Example 1.3.35 *Let $P = \overline{m}n.\overline{a}b + m(x)$, $Q = \overline{m}n.\overline{b}a + m(x)$ and a, b be distinct names. Then, P and Q are barbed bisimilar since they exhibit exactly the same barbs: $\overline{m}$ and m. Nonetheless, $!P$ and $!Q$ are not barbed bisimilar since two copies of P and two copies of Q can synchronize but the resulting processes do not exhibit the same barb, i.e., $!P \xrightarrow{\tau} P'$ and $P' \downarrow_{\overline{a}}$ but $!Q \xrightarrow{\tau} Q'$ and $Q' \downarrow_{\overline{b}}$.*

Example 1.3.36 *Let $P = \overline{m}|n$ and $Q = \overline{m}.n + n.\overline{m}$. Then, P and Q are barbed bisimilar since they have the same barbs. We only analyse the case when P starts: if $P \downarrow_{\overline{m}}$ then $Q \downarrow_{\overline{m}}$ and if $P \downarrow_n$ then $Q \downarrow_n$. However, if we consider the substitution $\sigma = \{n/m\}$, we have that $P\sigma$ and $Q\sigma$ are not barbed bisimilar, since $P\sigma \xrightarrow{\tau}$ but $Q\sigma \not\xrightarrow{\tau}$.*

Nonetheless, barbed bisimilarity is preserved by some operators, as shown next.

Proposition 1.3.37 *$\sim_b$ is preserved by prefixing, restriction and choice operators.*

Proof: Let P, Q, $R \in \mathcal{P}_S$ be such that $P \sim_b Q$, and let $u = x_1 \cdot ... \cdot x_n$ where $n \in \mathbb{N}$. Then:

- $\alpha.P \sim_b \alpha.Q$ since by applying the rule PREFIX we can conclude that $i)$ if $\alpha = u(y)$ or $\overline{u}y$ we have that both $P \downarrow_u$ and $Q \downarrow_u$ or both $P \downarrow_{\overline{u}}$ and $Q \downarrow_{\overline{u}}$ respectively; $ii)$ if $\alpha = \tau$ we have that $\tau.P \xrightarrow{\tau} P$ and $\tau.Q \xrightarrow{\tau} Q$ and by hypothesis $P \sim_b Q$.
- $(\nu x)P \sim_b (\nu x)Q$ since by applying rules RES or OPEN we have that $(\nu x)P \downarrow_\beta$ if and only if $(\nu x)Q \downarrow_\beta$. In addition, if by applying RES $(\nu x)P \xrightarrow{\tau} (\nu x)P'$ then we had that $P \xrightarrow{\tau} P'$ and since $P \sim_b Q$ we would also have that $Q \xrightarrow{\tau} Q'$ and hence $(\nu x)Q \xrightarrow{\tau} (\nu x)Q'$ where $P' \sim_b Q'$ and so $(\nu x)P' \sim_b (\nu x)Q'$ as expected.
- $P + R \sim_b Q + R$ since $P + R$ exhibits a barb β if and only if so does P or R. Analogously, $Q + R$ exhibits a barb β if and only if so does Q or R. Since by hypothesis we have that $P \sim_b Q$, we can conclude that $P + R$ and $Q + R$ exhibit the same barbs.

□

Therefore, the notion of barbed congruence was put forth in [21] while the less demanding notion of barbed equivalence was presented in [16]. Note that barbed equivalence and barbed congruence do not coincide, as shown in Example 1.3.36 since the referred processes are barbed equivalent but not barbed congruent.

Definition 1.3.38 *Barbed equivalence*
Two processes P and Q are barbed equivalent*, written $\sim_{beq}$, if for every process R we have that $P|R \sim_b Q|R$.*

In order to define barbed congruence we must first introduce the notion of context.

Definition 1.3.39 *Context*
A context *is obtained when the hole $[\cdot]$ replaces a process in $P \in \mathcal{P}_S$. The process obtained by replacing the $[\cdot]$ in C by P, where C is a context and P a process, is denoted by $C[P]$.*

Definition 1.3.40 *Barbed congruence*
Two processes P and Q are barbed congruent*, written $\simeq_b$, if for each context $C[\cdot]$ it holds that $C[P] \sim_b C[Q]$.*

We now extend the result that establishes an alternative definition of barbed congruence in the π-calculus as done in [3] to the π-calculus with polyadic synchronization. The proof of Lemma 1.3.41 relies heavily on the one presented in [3].

Lemma 1.3.41 *$P \simeq_b Q$ if and only if $P\sigma \sim_{beq} Q\sigma$ for any substitution σ*

Proof:

($\Rightarrow$) Let P, $Q \in \mathcal{P}_S$ be such that $P \simeq_b Q$. Also, let $u = x_1 \cdot ... \cdot x_k$ where $k \in \mathbb{N}$ and $x_1, ..., x_k$ are fresh, and $\sigma = \{\tilde{y}/\tilde{z}\}$ be a substitution where $\tilde{y} = y_1, ..., y_n$ and $\tilde{z} = z_1, ...z_n$. Given $C = \overline{u}y_1...\overline{u}y_n \,|\, u(z_1)...u(z_n).[\cdot] \,|\, R$ we know that $C[P] \sim_b C[Q]$ since both processes exhibit the same barbs. In addition, by performing n internal actions $C[P] \xrightarrow{\tau} ... \xrightarrow{\tau} P\sigma|R$, and the only process that does not exhibit barb $\overline{u}$ to which $C[Q]$ reduces in n steps is $Q\sigma|R$. Thus $P\sigma|R \sim_b Q\sigma|R$, that is, $P\sigma \sim_{beq} Q\sigma$.

($\Leftarrow$) Let P, $Q \in \mathcal{P}_S$ and σ be a substitution such that $P\sigma \sim_{beq} Q\sigma$, that is, $P\sigma|R \sim_b Q\sigma|R$ for any R. Since $\simeq_b$ is the largest congruence in $\sim_b$ it suffices to show that for any context C we have that $C[P]\sigma|R \sim_b C[Q]\sigma|R$. The proof is done by induction on C and we consider only the relevant transitions.

- $C = u(y).C'$ where $u = x_1 \cdot ... \cdot x_k$ and $k \in \mathbb{N}$. If by application of rules PREFIX and PAR1 $C[P]\sigma|R \xrightarrow{\sigma(u(y))} C'[P]\sigma|R$, then $C[Q]\sigma|R \xrightarrow{\sigma(u(y))} C'[Q]\sigma|R$ and by induction hypothesis $C'[P]\sigma|R \sim_b C'[Q]\sigma|R$. If by application of rule COMM $C[P]\sigma|R \xrightarrow{\tau} C'[P]\sigma\{z/y\}|R'$ (where we assume $R \xrightarrow{\overline{u}z} R'$), then $C[Q]\sigma|R \xrightarrow{\tau} C'[Q]\sigma\{z/y\}|R'$ and by induction hypothesis $C'[P]\sigma\{z/y\}|R' \sim_b C'[Q]\sigma\{z/y\}|R'$. If by application of rule CLOSE $C[P]\sigma|R \xrightarrow{\tau} (\nu y)C'[P]\sigma|R'$ (where we assume $R \xrightarrow{\overline{u}(y)} R'$), then $C[Q]\sigma|R \xrightarrow{\tau} (\nu y)C'[Q]\sigma|R'$; by induction hypothesis $C'[P]\sigma|R' \sim_b C'[Q]\sigma|R'$ and by Proposition 1.3.37 $\sim_b$ is closed under restriction, so $(\nu y)C'[P]\sigma|R' \sim_b (\nu y)C'[Q]\sigma|R'$.

The other cases can be handled in a similar way (for $C =!C'$ check [3]).

□

In [16] an alternative characterization of barbed equivalence was obtained by proving it coincided with the notion of early bisimulation. We now extend that proof in order to establish a similar result for the π-calculus with polyadic synchronization. Also, note that we only establish the strong result; we believe the weak can be established also following the proof in [16] and the one for the strong case we will present.

Before we establish the result, we will first introduce some auxiliary lemmas about countable sets which we will rely upon but not prove. Note that we consider a set to be *countable* if it is either finite or has the same cardinality as $\mathbb{N}$.

Lemma 1.3.42 *Countable Sets*

- *Every subset of a countable set is again countable.*
- *The countable union of countable sets is countable*

In order to begin the proof, we must define the following three sets which we will use. Firstly, let F be a finite set of pairs of names, $F_1 = \pi_1(F)$ and $F_2 = \pi_2(F)$ where π_1, π_2 represent respectively the projection of F on the first and second component of the pairs. Also, let F_1 include the free names of two processes P and Q and a finite set of names represented by $\{\tilde{x}\}$, that is, $fn(P,Q) \cup \{\tilde{x}\} \subseteq F_1$. Also note that $(a, a') \in F$ if and only if $a \in F_1$ and $a' \in F_2$.
Secondly, let Y a countable infinite set of pairs of names, $Y_1 = \pi_1(Y)$ and $Y_2 = \pi_2(Y)$ be such that $(y, y') \in Y$ if and only if $y \in Y_1$ and $y' \in Y_2$. Also, Y_1 must be composed of names not present in F, that is, if we let $n(F)$ and $n(Y)$ denote the set of names which occur in F and Y respectively, then $n(Y) \cap n(F) = \emptyset$.

Thirdly, let $H\langle F\rangle$ be a set of pairs of channels built from the names in F_1; we will prove H is countable. In addition, let $H_1 = \pi_1(H)$ and $H_2 = \pi_2(H)$ be such that $(c, c'') \in H$ if and only if $c \in H_1$ and $c'' \in H_2$. The set H_1 is the union of the composite channels built from all possible permutations of subsets of F_1 which are denoted by $!\mathcal{P}(F_1)$, i.e., $H_1 = \cup_{A\in !\mathcal{P}(F_1)} make{-}channel(A)$ where $make{-}channel(a_1, ..., a_n) = a_1 \cdot ... \cdot a_n$. By Lemma 1.3.42 every subset of F_1 is countable (because F_1 is countable) and $\mathcal{P}(F_1) = \cup_{A\subseteq F_1} A$ is also countable since it is a countable union of countable sets. Thus, H_1 is a countable set of composite channels.

Theorem 1.3.43 *$\sim_e$ coincides with $\sim_{beq}$*

PROOF:

($\Rightarrow$) We prove that $\sim_e$ is a barbed bisimulation. Let $u = x_1 \cdot ... \cdot x_n$ where $n \in \mathbb{N}$ and let P, $Q \in \mathcal{P}_S$ be such that $P\mathcal{S}Q$ where $\mathcal{S}$ is an early bisimulation. Then, if:

1. $P \xrightarrow{u(y)} P'$ then $P \downarrow_u$ and since P and Q are early bisimilar for each w there is a Q' such that $Q \xrightarrow{u(y)} Q'$ and $P'\{w/y\}\mathcal{S}Q'\{w/y\}$. Thus, since $Q \xrightarrow{u(y)} Q'$, we have that $Q \downarrow_u$.
2. $P \xrightarrow{\alpha} P'$ where $\alpha = \overline{u}y$ or $\overline{u}(y)$ then $P \downarrow_{\overline{u}}$ and since P and Q are early bisimilar there is a Q' such that $Q \xrightarrow{\alpha} Q'$ which implies that $Q \downarrow_{\overline{u}}$.
3. $P \xrightarrow{\tau} P'$ since P and Q are early bisimilar there is a Q' such that $Q \xrightarrow{\tau} Q'$ (and $P'\mathcal{S}Q'$). Note that to P' and Q' we can apply the same reasoning as to the P and Q we started from till a visible action is performed by both processes (or their descendants) like in case 1 or 2, or till both processes are inactive.

Therefore, we can conclude that P and Q are barbed bisimilar. Since early bisimulation is preserved by all operators except input prefixing (Lemma 1.3.16), we have that for any R, $P|R \sim_e Q|R$, and thus $P|R \sim_b Q|R$. We then comply with the necessary requirements of barbed equivalence and establish that $P \sim_{beq} Q$.

($\Leftarrow$) Given $\mathcal{S} = \{(P, Q) : F, Y, \tilde{x}$ *exist such that* $(\nu\tilde{x})C^{F,Y}[P] \sim_b (\nu\tilde{x})C^{F,Y}[Q]\}$ where $\tilde{x}$, F, $H\langle F\rangle$, Y are related as explained before, we prove that $\mathcal{S}$ is an early bisimulation. The context $C[\cdot]$ is defined as $C[\cdot] = [\cdot] \,|\, V\langle F, Y\rangle$ where:

$V\langle F, Y\rangle =$

$$\sum_{(c,c'')\in H\langle F\rangle} \quad \sum_{(b,b')\in F\cup(y,y')} \overline{c}b.(c'' + b' + in + V\langle F \cup (y, y'), Y\backslash(y, y')\rangle) \tag{1.1}$$

$$+ \sum_{(c,c'')\in H\langle F\rangle} c(y).(c''+y'+out+V\langle F\cup(y, y'), Y\backslash(y, y')\rangle+(\nu t) \sum_{(b,b')\in F} (\overline{t \cdot b}|t{\cdot}y).b' \tag{1.2}$$

The first 1.1 and second 1.2 summands are used to test respectively the input and output actions of P or Q[7]. Note in 1.1 that all possible inputs are considered in the inner summation just like early bisimulation requires. In 1.2 the last term in the summation concerns the case of bound output in which the outputted name will not be found in H, as opposed to the case of free output. Further, notice that the names in, out are not in $n(P, Q)$, these names are used to show which type of action (input or output, respectively) was performed.
The relation between F and Y can now be further analysed, since the names taken from Y are used to augment F (and hence H) via name-communication. Note that on the definition of $V\langle F, Y\rangle$ the pair (y, y') is drawn from Y.
We now prove the core of the theorem by splitting the proof into the four possible actions of P.

1. If $P \xrightarrow{\tau} P'$, then we can infer that $(\nu\tilde{x})C^{F,Y}[P] = (\nu\tilde{x})(P|V\langle F, Y\rangle) \xrightarrow{\tau} (\nu\tilde{x})(P'|V\langle F, Y\rangle) = R$. Since by hypothesis $(\nu\tilde{x})C^{F,Y}[P] \sim_b (\nu\tilde{x})C^{F,Y}[Q]\}$, then $(\nu\tilde{x})C^{F,Y}[Q] \xrightarrow{\tau} T$, and R may have as barbs only c such that $c \in H_1$, and so should T. Note that $(\nu\tilde{x})C^{F,Y}[Q]$ could have performed a τ action by *i*) interaction between $V\langle F, Y\rangle$ and process Q where Q performed an input action; *ii*) interaction between $V\langle F, Y\rangle$ and process Q where Q performed an free or bound output action; *iii*) there is no interaction with $V\langle F, Y\rangle$ and Q performs a τ action by itself. Both *i*) and *ii*) are impossible since at least $T \downarrow_{in}$ or $T \downarrow_{out}$ and $\{in, out\} \cap H_1 = \emptyset$. Thus, $(\nu\tilde{x})C^{F,Y}[Q] \xrightarrow{\tau} T = (\nu\tilde{x})(Q'|V\langle F, Y\rangle)$, that is $Q \xrightarrow{\tau} Q'$ and $(P', Q') \in \mathcal{S}$.

2. If $P \xrightarrow{c(y)} P'$, then we can infer that $(\nu\tilde{x})C^{F,Y}[P] = (\nu\tilde{x})(P|V\langle F, Y\rangle) \xrightarrow{\tau} (\nu\tilde{x})(P'\{b/y\}|V_1) = R$ where $V_1 = c''+b'+in+V\langle F\cup(y, y'), Y\backslash(y, y')\rangle$. Then, R has as barbs at least c'', b', and in. Note that $(\nu\tilde{x})C^{F,Y}[Q]$ could have performed a τ action by the *i*), *ii*) and *iii*) reasons mentioned in case 1. The situation *ii*) where Q performed a free or bound output is impossible since in that case $T \downarrow_{out}$ but $R \not\downarrow_{out}$. The situation *iii*) is impossible since in that case, e.g., $T \not\downarrow_{b'}$. The only possible situation is then if Q performed an input action of the type $\alpha = c_1(b_1)$. Note that the following equalities have to hold $c_1 = c$ and $b_1 = b$ so that $T \downarrow_{c''}$ and $T \downarrow_{b'}$ as does R. Thus, we have that if $P \xrightarrow{c(y)} P'$ then for every possible b we have that $Q \xrightarrow{c(y)} Q'$ and $R \sim_b T$, that is, $(P'\{b/y\}, Q'\{b/y\}) \in \mathcal{S}$.

3. If $P \xrightarrow{\bar{c}z} P'$, then we can infer that $(\nu\tilde{x})C^{F,Y}[P] = (\nu\tilde{x})(P|V\langle F, Y\rangle) \xrightarrow{\tau} (\nu\tilde{x})(P'|V_2) = R$ where $V_2 = c''+z'+out+V\langle F, Y\rangle+(\nu t)\sum_{(b,b')\in F}(\overline{t\cdot b}|t\cdot y).b'$ and t is fresh, while $z \in fn(P) \subseteq F_1$. Then, R has as barbs at least c'', z' and out. Since by hypothesis $(\nu\tilde{x})C^{F,Y}[P] \sim_b (\nu\tilde{x})C^{F,Y}[Q]\}$, then $(\nu\tilde{x})C^{F,Y}[Q] \xrightarrow{\tau} T$ and T should have the same barbs as R. Note that $(\nu\tilde{x})C^{F,Y}[Q]$ could have performed a τ action by the *i*), *ii*) and *iii*) reasons mentioned in case 1. The

[7] Note that the summations in 1.1 and 1.2 are finite.

situation $i)$ is impossible since $T \downarrow_{in}$ but $R \not\downarrow_{in}$, and the situation $iii)$ is impossible since, e.g., $T \not\downarrow_{c''}$. Then we are in a situation where Q has to do an output on the same channel as P (so it has c'' as a barb too), but we must still prove the output has to be free. This happens because since P performed a free output, R can do a τ action from the last summation in V_2 and the resulting process has as an unique barb z'. However, if Q performs a bound output, the summation in V_2 is an inactive process, and even if $(\nu\tilde{x})C^{F,Y}[Q]$ performs a τ action by the cases $i)$, $ii)$ and $iii)$, in both $ii)$ and $iii)$ the resulting process would have either *in* or *out* as a barb. In the case of $i)$ then there would be no interaction with $V\langle F, Y\rangle$, and so the resulting process would not have z' as a barb. Thus, Q has to perform $\alpha = \overline{c}z$ as did P.

4. If $P \xrightarrow{\overline{c}(y)} P'$, then we can infer that $(\nu\tilde{x})C^{F,Y}[P] = (\nu\tilde{x})(P|V\langle F, Y\rangle) \xrightarrow{\tau} (\nu\tilde{x})(\nu y)(P'|V_3) = R$ where $V_3 = c'' + y' + out + V\langle F \cup (y, y'), Y\backslash(y, y')\rangle + (\nu t)\sum_{(b,b')\in F}(\overline{t \cdot b}|t \cdot y).b'$. The rest of the proof is very similar to case 3 where the difference between bound and free output is analysed.

□

Corollary 1.3.44 $\simeq_e$ *coincides with* $\simeq_b$

PROOF: Let P, $Q \in \mathcal{P}_S$.

($\Rightarrow$) If $P \simeq_e Q$ then by Definition 1.3.18 $P\sigma \sim_e Q\sigma$ for any substitution σ. By Theorem 1.3.43 we know that $P\sigma \sim_{beq} Q\sigma$, and by Lemma 1.3.41 $P \simeq_b Q$.

($\Leftarrow$) If $P \simeq_b Q$ then by Lemma 1.3.41 we know that $P\sigma \sim_{beq} Q\sigma$ for any substitution σ. By Theorem 1.3.43 we have that $P\sigma \sim_e Q\sigma$, and therefore $P \simeq_e Q$.

□

Chapter 2

Spi-Calculus

The spi-calculus was introduced in [12] as an extension of the π-calculus with cryptographic primitives, and it is designed to model and analyse security protocols.

2.1 Syntax

In this section we introduce the syntax of a variant of the spi-calculus that was originally presented in [12]. The spi-calculus used is the one in [23] following that presented in [24]. We start by defining the set of messages, expressions, and decryption-free expressions, and by explaining their role in the spi-calculus syntax.

Definition 2.1.1 *Messages*
Let $\mathcal{N}$ be a countable set of names ranged over a, b, c,..., k, l,..., x, y, z. The set of messages *$\mathcal{M}$ ranged over M, N is defined by the following grammar:*

M ::= a *name*
      | $\{M\}_k$ *shared-key encryption*

Note that the names are untyped and so can be used as channels, keys, variables or cipher texts. In order to simplify our examples, we often let a, b, c represent channels or cipher texts, k, l represent keys, and x, y, z represent variables. Also note that although we allow nested encryption, we require that on the encryption of the cipher text M under the key n, n is a name. As an example, we do not allow messages of the type $\{a\}_{\{b\}_k}$.

Definition 2.1.2 *Expressions*
Let $\mathcal{N}$ be a countable set of names. The set of expressions *$\mathcal{E}$ ranged over η, ζ is defined by the following grammar:*

η ::= a *name*
      | $\{\eta\}_\eta$ *encryption*
      | $dec_\eta(\eta)$ *decryption*

The set of decryption-free expressions *$\mathcal{D}$ ranged over δ, ϵ is defined by the following grammar:*

δ ::= a *name*
      | $\{\delta\}_\delta$ *encryption*

Note that when considering expressions these may be the result of arbitrary encryptions and decryptions, while messages are decryption-free and the encryption is done under a key (name). The introduction of decryption-free expressions allows us to ensure that decryption constructs occur only in the 'let' operator of processes (See Definition 2.1.4).

We now introduce the set of logical formulae which are used as Boolean guards for spi-calculus processes - a generalization of the matching construct present in the π-calculus.

Definition 2.1.3 *Formulae*
Let $z \in \mathcal{N}$ and η, $\zeta \in \mathcal{E}$. The set of logical formulae Φ *ranged over ϕ, ψ is defined by the following grammar:*

$$
\begin{array}{lll}
\phi ::= & tt & \textit{true} \\
 & \mid name(\eta) & \textit{name predicate} \\
 & \mid [\eta = \zeta] & \textit{equality} \\
 & \mid let\, z = \eta\, in\, \phi & \textit{decryption} \\
 & \mid \phi \wedge \phi & \textit{conjunction} \\
 & \mid \neg\phi & \textit{negation}
\end{array}
$$

The name predicate $name(\eta)$ checks whether the argument η is or not a plain name. The decryption operator $let\, z = \eta\, in\, \phi$, binds the value of η (calculated through the rules in Table 2.1) to z within ϕ.

Definition 2.1.4 *Processes*
Let n, $x \in \mathcal{N}$, $\delta \in \mathcal{D}$, $\zeta \in \mathcal{E}$ and $\phi \in \Phi$. The class of processes $\mathcal{P}_{spi}$ ranged over P, Q is defined by the following grammar:

$$
\begin{array}{lll}
P ::= & \mathbf{0} & \textit{inaction} \\
 & \mid \pi.P & \textit{prefix} \\
 & \mid !P & \textit{replication} \\
 & \mid (\nu n)P & \textit{restriction} \\
 & \mid P|P & \textit{parallel composition} \\
 & \mid P + P & \textit{choice} \\
 & \mid \phi P & \textit{Boolean guard} \\
 & \mid let\, x = \zeta\, in\, P & \textit{decryption}
\end{array}
$$

where the prefixes π are given by:

$$
\begin{array}{lll}
\pi ::= & \delta(x) & \textit{input} \\
 & \mid \overline{\delta}\delta & \textit{output}
\end{array}
$$

As a brief explanation of some of the constructs not present in the π-calculus, note that, as would be expected, the process ϕP behaves as P if and only if ϕ evaluates to true. In addition, note that if the evaluation of ζ fails in a process like $let\, x = \zeta\, in\, P$ [1] the whole process is stuck. Also note that x is bound in $let\, x = \zeta\, in\, P$; the other constructs behave

[1] The reader already familiar with the spi-calculus should note that the 'let' construct is equivalent to the originally proposed 'case of'. Thus, processes like $let\, x = dec_k(M)\, in\, P$ and $case\, M\, of\, \{x\}_k\, in\, P$ are the same.

exactly as in the π-calculus.

In the spi-calculus we consider three possible actions: the internal action τ, the free input action aM and the output action $(\nu\tilde{b})\overline{a}\langle M\rangle$ where $\{a, \tilde{b}\} \subseteq \mathcal{N}$ and $M \in \mathcal{M}$. Whenever $\{\tilde{b}\} = \emptyset$ we write $\overline{a}\langle M\rangle$ instead of $(\nu\tilde{b})\overline{a}\langle M\rangle$. We will also note that only names are allowed as channels, otherwise the process performing the action is stuck.

The notion of substitution follows that of the π-calculus with the sole difference that instead of considering substitutions as mappings of $\mathcal{N}$ to $\mathcal{N}$, we consider mappings of $\mathcal{N}$ to $\mathcal{M}$, where $\mathcal{N}$ is the set of names and $\mathcal{M}$ the set of messages. The formal definition of substitution is presented in Section 2.3. The notion of α-conversion follows that presented in Chapter 1 for the π-calculus with polyadic synchronization.

2.2 Late Labelled Transition Semantics

In this section we introduce the early labelled transition semantics following [23] instead of [24]. This choice is based on the fact that while in [24] two sets of operational semantics (one for the processes and one for the environment) are developed, in [23] the environment behaviour is built into the definition of bisimulation, following the work in [22].

In order to establish a transition relation based on a set of rules, we must first be able to evaluate expressions and Boolean guards. Following [23], the evaluation function for expressions $e(\cdot) : \mathcal{E} \to \mathcal{M} \cup \{\bot\}$ and formulae $e(\cdot) : \Phi \to \{tt, ff\}$ is defined recursively according to Table 2.1[2].

We now introduce the early labelled transition relation for the spi-calculus.

Definition 2.2.1 *Early labelled transition relation*
The early labelled transition relation $\xrightarrow{\alpha} \subseteq \mathcal{P}_S \times \mathcal{P}_S$, *where α is a possible action, is the smallest relation generated by the set of rules in Table 2.2.*

Most of the rules come straightforwardly from the π-calculus. The rules S-GUARD and S-LET establishing the behaviour of process of the type ϕP and $let\, z = \zeta\, in\, P$ have been described in Section 2.1 when we introduced these constructs.

2.3 Observational Semantics

In this section we present a notion of bisimulation that takes into account the behaviour of the environment. Although several notions of bisimulation have been put forth, we rely on that of *alley bisimulation* since it was proven it coincides with barbed equivalence [See

[2] It should be clear that the first three rules correspond to the evaluation of expressions and the rest to the evaluation of formulae.

$$
\begin{array}{l}
e(a) = a \\
e(\{\eta\}_\zeta) = \begin{cases} \{M\}_k & \text{if } e(\eta) = M \in \mathcal{M} \text{ and } e(\zeta) = k \in \mathcal{N} \\ \perp & \text{otherwise} \end{cases} \\
e(dec_\zeta(\eta)) = \begin{cases} M & \text{if } e(\eta) = \{M\}_k \in \mathcal{M} \text{ and } e(\zeta) = k \in \mathcal{N} \\ \perp & \text{otherwise} \end{cases} \\
\\
e(tt) = tt \\
e(\phi \wedge \psi) = e(\phi) \wedge e(\psi) \\
e(\neg\phi) = \neg e(\phi) \\
e(let\, z = \zeta\, in\, \phi) = \begin{cases} e(\phi\{M/z\}) & \text{if } e(\zeta) = M \in \mathcal{M} \\ ff & \text{otherwise} \end{cases} \\
e(name(\zeta)) = \begin{cases} tt & \text{if } \zeta \in \mathcal{N} \\ ff & \text{otherwise} \end{cases} \\
e([\zeta = \eta]) = \begin{cases} tt & \text{if } \zeta = \eta \in \mathcal{M} \\ ff & \text{otherwise} \end{cases}
\end{array}
$$

Table 2.1: Evaluation in the spi-calculus

Corollary 2.3.27]. In order to introduce this notion, we must first describe in detail the environment, so as to be able to compare the processes.

We rely on a set of assumptions that are generally accepted and referred to in the literature (e.g. [24, 23]) and which we now enumerate.

1. A message $\{M\}_k$ can only be decrypted using k, and can only be produced by encrypting M under k. If k is kept secret, then no attacker can guess or forge k.

2. There is enough redundancy in the structure of messages to tell whether the decryption of a message with a given key has actually succeeded or not.

3. There is enough redundancy in the structure of messages to tell their role (name or compound cipher text).

4. The only way to form a new key is to get a fresh name from a primitive set of names.

Note that Assumption 3 is necessary since, as mentioned earlier, we only allow channels to be names.

We now present concepts that are important to understand the role of the environment. We start by formally introducing the definition of substitution and other useful notation.

Definition 2.3.1 *Substitution*
A substitution σ *is a finite partial map from the set of names* $\mathcal{N}$ *to the set of messages* $\mathcal{M}$*. The* domain *and* codomain *of substitution* σ *are written* $dom(\sigma)$ *and* $range(\sigma)$ *respectively. We write* $\sigma\{M/x\}$ *for* $\sigma \cup \{(x, M)\}$ *where* $x \notin dom(\sigma)$*.*

$$\text{(S-OUT)} \quad \frac{-}{\overline{a}M.P \xrightarrow{\overline{a}\langle M\rangle} P} \qquad \text{(S-INP)} \quad \frac{-}{a(x).P \xrightarrow{aM} P\{M/x\}}$$

$$\text{(S-CH1)} \quad \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'} \qquad \text{(S-CH2)} \quad \frac{P \xrightarrow{\alpha} P'}{Q + P \xrightarrow{\alpha} P'}$$

$$\text{(S-PAR1)} \quad \frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q} \qquad \text{(S-PAR2)} \quad \frac{P \xrightarrow{\alpha} P'}{Q|P \xrightarrow{\alpha} Q|P'} \qquad \text{where } bn(\alpha) \cap fn(Q) \; = \; \emptyset$$

$$\text{(S-RES)} \quad \frac{P \xrightarrow{\alpha} P'}{(\nu m)P \xrightarrow{\alpha} (\nu m)P'} \quad \text{where } m \notin n(\alpha)$$

$$\text{(S-REP-ACT)} \quad \frac{P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P'|!P} \qquad \text{where } bn(\alpha) \cap fn(P) \; = \; \emptyset$$

$$\text{(S-REP-COMM)} \quad \frac{P \xrightarrow{(\nu\tilde{n})\overline{m}\langle M\rangle} P' \quad P \xrightarrow{m(x)} P''}{!P \xrightarrow{\tau} ((\nu\tilde{n})(P'|P''))|!P} \qquad \text{where } \tilde{n} \notin fn(P)$$

$$\text{(S-OPEN)} \quad \frac{P \xrightarrow{(\nu\tilde{n})\overline{m}\langle M\rangle} P'}{(\nu k)P \xrightarrow{(\nu\tilde{n},k)\overline{m}\langle M\rangle} P'} \quad \text{where } k \notin \{m, \tilde{n}\},\ k \in n(M)$$

$$\text{(S-COMM1)} \quad \frac{P \xrightarrow{(\nu\tilde{n})\overline{m}\langle M\rangle} P' \quad Q \xrightarrow{m(M)} Q'}{P|Q \xrightarrow{\tau} (\nu\tilde{n})(P'|Q')} \qquad \text{(S-COMM2)} \quad \frac{P \xrightarrow{(\nu\tilde{n})\overline{m}\langle M\rangle} P' \quad Q \xrightarrow{m(M)} Q'}{Q|P \xrightarrow{\tau} (\nu\tilde{n})(Q'|P')} \quad \text{where } \tilde{n} \notin fn(Q)$$

$$\text{(S-GUARD)} \quad \frac{P \xrightarrow{\alpha} P'}{\phi P \xrightarrow{\alpha} P'} \qquad \text{if } e(\phi) = tt$$

$$\text{(S-LET)} \quad \frac{P\{e(\zeta)/z\} \xrightarrow{\alpha} P'}{let\, z = \zeta\, in\, P \xrightarrow{\alpha} P'} \qquad \text{if } e(\zeta) \neq \bot$$

$$\text{(S-CONV)} \quad \frac{P \xrightarrow{\alpha} P'}{Q \xrightarrow{\alpha} P'} \qquad \text{if } Q =_\alpha P$$

Table 2.2: Early transition rules

Following the idea in [24] that it is possible to obtain an irreducible message after decrypting it using known keys, we present the notion of analysis, irreducibility and of core as introduced in [23]. Any set of messages, in particular the messages in the codomain of a substitution, may be reduced via decryption using the notion of analysis. The set of irreducibles contains only the messages that cannot be further reduced.

Definition 2.3.2 *Analysis / Irreducibility*
Let $S \subseteq \mathcal{M}$, where $\mathcal{M}$ is the set of messages. The analysis *$\mathcal{A}(S)$ is the smallest subset of $\mathcal{M}$ containing S and satisfying*

$$\text{(SET-DEC)} \quad \frac{\{M\}_k \in \mathcal{A}(S) \quad k \in \mathcal{A}(S)}{M \in \mathcal{A}(S)}$$

The irreducibles *$\mathcal{I}(S)$ are defined as $\mathcal{I}(S) = \mathcal{A}(S) \backslash \{\{M\}_k : k \in \mathcal{A}(S)\}$.*

Definition 2.3.3 *Core*
Let σ be a substitution, and $\mathcal{I}(\sigma)$, $\mathcal{A}(\sigma)$ be shorthand for $\mathcal{I}(range(\sigma))$ and $\mathcal{A}(range(\sigma))$. Then, for each message M,

$$core_\sigma(M) = \begin{cases} core_\sigma(M') & \text{if } M = \{M'\}_a \text{ and } a \in \mathcal{I}(\sigma) \\ M & \text{otherwise} \end{cases}$$

Also, we introduce the following notation by letting $C(\sigma, x) = core_\sigma(\sigma(x))$.

Note that with the auxiliary notion of $C(\sigma, x)$, we can define the irreducibles in an alternative way: $\mathcal{I}(\sigma) = \{C(\sigma, x) : x \in dom(\sigma)\}$.

In the following examples we show how the concepts just introduced work in practice.

Example 2.3.4 *Let $S = \{k, \{\{a\}_k\}_k, \{\{b\}_h\}_k, \{\{c\}_k\}_h, \{k\}_k\}$, where a, b, c, h, $k \in \mathcal{N}$. Then, $\mathcal{A}(S) = S \cup \{\{a\}_k, a, \{b\}_h, k\} = \{k, \{\{a\}_k\}_k, \{\{b\}_h\}_k, \{\{c\}_k\}_h, \{k\}_k, \{a\}_k, a, \{b\}_h\}$ and $\mathcal{I}(S) = \mathcal{A}(S) \backslash \{ \{\{a\}_k\}_k, \{\{b\}_h\}_k, \{k\}_k, \{a\}_k \} = \{k, \{\{c\}_k\}_h, a, \{b\}_h\}$.*

Example 2.3.5 *Let $\sigma = \{k/x_1, \{\{a\}_k\}_k/x_2, \{\{b\}_h\}_k/x_3, \{\{c\}_k\}_h/x_4, \{k\}_k/x_5\}$. Then, $core_\sigma(k) = k$, $core_\sigma(\{\{a\}_k\}_k) = core_\sigma(\{a\}_k) = core_\sigma(a) = a$, $core_\sigma(\{\{b\}_h\}_k) = core_\sigma(\{b\}_h) = \{b\}_h$, $core_\sigma(\{\{c\}_k\}_h) = \{\{c\}_k\}_h$ and $core_\sigma(\{k\}_k) = core_\sigma(k) = k$. Therefore, $\mathcal{I}(\sigma) = \mathcal{I}(range(\sigma)) = \{C(\sigma, x_i) : x_i \in dom(\sigma)\} = \{k, a, \{b\}_h, \{\{c\}_k\}_h\}$.*

We now introduce the definition of consistency based on the notion that two substitutions are consistent if they decrypt messages in precisely corresponding ways. In addition, we consider a stronger notion of consistency also proposed in [24]. The notions of *consistent* and *strongly consistent* environments lead to the notions of *alley* and *trellis bisimulation* respectively, which we will introduce afterwards.

Definition 2.3.6 *Consistent environments*
A pair of substitutions (σ, ρ) *is* consistent*, written* $\sigma \sim \rho$ *if and only if*

1. $dom(\sigma) = dom(\rho) = \{x_1, ..., x_n\}$ *where* $n \in \mathbb{N}$
2. $C(\sigma, x_i) \in \mathcal{N}$ *if and only if* $C(\rho, x_i) \in \mathcal{N}$
3. $C(\sigma, x_i) = C(\sigma, x_j)$ *if and only if* $C(\rho, x_i) = C(\rho, x_j)$
4. *for each* $i = 1, 2, ..., n$ *there is a tuple* $\tilde{\imath} = \imath_1, ..., \imath_n$ *such that*
   $\sigma(x_i) = \{...\{C(\sigma, x_i)\}_{C(\sigma, x_{\imath_1})} ...\}_{C(\sigma, x_{\imath_m})}$
   $\rho(x_i) = \{...\{C(\rho, x_i)\}_{C(\rho, x_{\imath_1})} ...\}_{C(\rho, x_{\imath_m})}$

Definition 2.3.7 *Strongly consistent environments*
A pair of substitutions (σ, ρ) *is* strongly consistent*, written* $\sigma \sim_s \rho$ *if and only if* $\sigma \sim \rho$ *and whenever* $C(\sigma, x_i) \in \mathcal{N}$ *or* $C(\rho, x_i) \in \mathcal{N}$ *then* $C(\sigma, x_i) = C(\rho, x_i)$.

We now compare with respect to consistency several environment, the first three are cited from [23].

Example 2.3.8 *Consider the following environments:*

$$\sigma_1 = \{a/x_1, \{b\}_a/x_2, \{\{c\}_a\}_k/x_3\}$$

$$\sigma_2 = \{a/x_1, \{b\}_a/x_2, \{\{d\}_k\}_k/x_3\}$$

$$\sigma_3 = \{a/x_1, \{c\}_a/x_2, \{\{c\}_k\}_k/x_3\}$$

$$\sigma_4 = \{d/x_1, \{b\}_k/x_2\}$$

$$\sigma_5 = \{a/x_1, \{\{b\}_b\}_k/x_2\}$$

- $\sigma_1 \sim \sigma_2$ *since 1.* $dom(\sigma_1) = dom(\sigma_2) = \{x_1, x_2, x_3\}$*; 2.* $C(\sigma_1, x_1) = a \in \mathcal{N} \Leftrightarrow C(\sigma_2, x_1) = a \in \mathcal{N}$ *and* $C(\sigma_1, x_2) = b \in \mathcal{N} \Leftrightarrow C(\sigma_2, x_2) = b \in \mathcal{N}$*; 3. not applicable; 4.* $\sigma_1(x_1) = a = \sigma_2(x_1)$, $\sigma_1(x_2) = \{b\}_a = \{C(\sigma_1, x_2)\}_{C(\sigma_1, x_1)}$ *and* $\sigma_2(x_2) = \{b\}_a = \{C(\sigma_2, x_2)\}_{C(\sigma_2, x_1)}$, $\sigma_1(x_3) = \{\{c\}_a\}_k = C(\sigma_1, x_3)$ *and* $\sigma_2(x_3) = \{\{d\}_k\}_k = C(\sigma_2, x_3)$. *Note that if both* σ_1 *and* σ_2 *acquire knowledge of* k*, i.e.,* $\sigma_1' = \sigma_1\{k/x_4\}$ *and* $\sigma_2' = \sigma_2\{k/x_4\}$*, then* $\sigma_1' \not\sim \sigma_2'$*. The problem resides on the fact that* $\sigma_1'(x_3) = \{\{c\}_a\}_k = \{\{C(\sigma_1', x_3)\}_{C(\sigma_1', x_1)}\}_{C(\sigma_1', x_4)}$ *but* $\sigma_2'(x_3) = \{\{d\}_k\}_k = \{\{C(\sigma_2', x_3)\}_{C(\sigma_2', x_4)}\}_{C(\sigma_2'(x_4))}$*, that is the decryptions are not made in the same way, thus violating condition 4.*

- $\sigma_1 \sim \sigma_3$ *since 1.* $dom(\sigma_1) = dom(\sigma_3)$*; 2.* $C(\sigma_1, x_1) = a \in \mathcal{N} \Leftrightarrow C(\sigma_3, x_1) = a \in \mathcal{N}$ *and* $C(\sigma_1, x_2) = b \in \mathcal{N} \Leftrightarrow C(\sigma_3, x_2) = c \in \mathcal{N}$*; 3. not applicable; 4.* $\sigma_1(x_1) = a = \sigma_3(x_1)$, $\sigma_1(x_2) = \{b\}_a = \{C(\sigma_1, x_2)\}_{C(\sigma_1, x_1)}$ *and* $\sigma_3(x_2) = \{c\}_a = \{C(\sigma_3, x_2)\}_{C(\sigma_3, x_1)}$, $\sigma_1(x_3) = \{\{c\}_a\}_k = C(\sigma_1, x_3)$ *and* $\sigma_3(x_3) = \{\{c\}_k\}_k = C(\sigma_3, x_3)$. *Note that if both* σ_1 *and* σ_3 *acquire knowledge of* k*, i.e.,* $\sigma_1' = \sigma_1\{k/x_4\}$ *and* $\sigma_3' = \sigma_3\{k/x_4\}$*, then* $\sigma_1' \not\sim \sigma_3'$*. The problem resides on the fact that* $C(\sigma_3', x_2) = C(\sigma_3', x_3) = c$ *but* $C(\sigma_1', x_2) = b \neq c = C(\sigma_1', x_3)$*, thus violating condition 3.*

- $\sigma_2 \sim \sigma_3$ *since 1.* $dom(\sigma_2) = dom(\sigma_3)$*; 2.* $C(\sigma_2, x_1) = a \in \mathcal{N} \Leftrightarrow C(\sigma_3, x_1) = a \in \mathcal{N}$ *and* $C(\sigma_2, x_2) = b \in \mathcal{N} \Leftrightarrow C(\sigma_3, x_2) = c \in \mathcal{N}$*; 3. not applicable; 4.* $\sigma_2(x_1) = a = \sigma_3(x_1)$, $\sigma_2(x_2) = \{b\}_a = \{C(\sigma_2, x_2)\}_{C(\sigma_2,x_1)}$ *and* $\sigma_3(x_2) = \{c\}_a = \{C(\sigma_3, x_2)\}_{C(\sigma_3,x_1)}$, $\sigma_2(x_3) = \{\{d\}_k\}_k = C(\sigma_2, x_3)$ *and* $\sigma_3(x_3) = \{\{c\}_k\}_k = C(\sigma_3, x_3)$*. Note that if both* σ_2 *and* σ_3 *acquire knowledge of* k*, i.e.,* $\sigma'_2 = \sigma_2\{k/x_4\}$ *and* $\sigma'_3 = \sigma_3\{k/x_4\}$*, then* $\sigma'_2 \not\sim \sigma'_3$*. The problem resides on the fact that* $C(\sigma'_3, x_2) = C(\sigma'_3, x_3) = c$ *but* $C(\sigma'_2, x_2) = b \neq d = C(\sigma'_2, x_3)$*, thus violating condition 3.*

- $\sigma_4 \sim \sigma_5$ *since 1.* $dom(\sigma_4) = dom(\sigma_5)$*; 2.* $C(\sigma_4, x_1) = a \in \mathcal{N} \Leftrightarrow C(\sigma_5, x_1) = d \in \mathcal{N}$*; 3. not applicable; 4.* $\sigma_4(x_1) = a$ *and* $\sigma_5(x_1) = d$, $\sigma_4(x_2) = \{b\}_k$ *and* $\sigma_5(x_2) = \{\{b\}_b\}_k$*. Note that if both* σ_4 *and* σ_5 *acquire knowledge of* k*, i.e.,* $\sigma'_4 = \sigma_4\{k/x_3\}$ *and* $\sigma'_5 = \sigma_5\{k/x_3\}$*, then* $\sigma'_4 \not\sim \sigma'_5$*. The problem resides on the fact that* $C(\sigma'_4, x_2) = b \in \mathcal{N}$ *but* $C(\sigma'_5, x_2) = \{b\}_b \notin \mathcal{N}$*, thus violating condition 2.*

- $\sigma_4 \not\sim \sigma_1$, $\sigma_4 \not\sim \sigma_2$, $\sigma_4 \not\sim \sigma_3$, $\sigma_5 \not\sim \sigma_1$, $\sigma_5 \not\sim \sigma_2$, $\sigma_5 \not\sim \sigma_3$ *since* $dom(\sigma_4) = dom(\sigma_5) \neq dom(\sigma_1) = dom(\sigma_2) = dom(\sigma_3)$*, thus violating condition 1.*

Note that for two environments to be consistent it is not necessary that their knowledge of names is identical as seen in Example 2.3.8 in the case where $\sigma_4 \sim \sigma_5$, whereas for two environments to be strongly consistent it is so. Therefore, of the environments considered in Example 2.3.8, $\sigma_1 \sim_s \sigma_2$ but $\sigma_1 \not\sim_s \sigma_3$ and $\sigma_2 \not\sim_s \sigma_3$ since $C(\sigma_1, x_2) = C(\sigma_2, x_2) = b \neq c = C(\sigma_3, x_2)$. It is immediate that $\sigma_4 \not\sim_s \sigma_5$. In addition, note that if $\sigma \not\sim \rho$ then $\sigma \not\sim_s \rho$ by definition of consistent and strongly consistent environments.

We now introduce the notion of synthesis of consistent environments as in [23] which will be used in the definition of alley bisimulation.

Definition 2.3.9 *Synthesis*
If $\sigma \sim \rho$ *we write* $(\sigma, \rho) \vdash M \leftrightarrow N$ *if and only if there is a* ζ *such that* $n(\zeta) \subseteq dom(\sigma)$, $e(\sigma(\zeta)) = M$ *and* $e(\rho(\zeta)) = N$*. The* synthesis *of a consistent pair of substitutions* (σ, ρ) *is defined as* $\mathcal{S}(\sigma, \rho) = \{(M, N) : (\sigma, \rho) \vdash M \leftrightarrow N\}$*.*

We now present the notion of consistent and strongly consistent alley relation, which will allow us to define the notions of alley bisimulation and trellis bisimulation used to compare processes.

Definition 2.3.10 *Consistent alley relation*
Let (σ, ρ) *be a substitution pair, and* P, $Q \in \mathcal{P}_{spi}$*. An* alley process pair *is a triple* $((\sigma, \rho), P, Q)$ *with* $dom(\sigma) = dom(\rho)$*. An* alley relation $\mathcal{R}$ *is a set of alley process pairs. We write* $(\sigma, \rho) \vdash P\mathcal{R}Q$ *if* $((\sigma, \rho), P, Q) \in \mathcal{R}$*. An alley relation* $\mathcal{R}$ *is* consistent *if* $(\sigma, \rho) \vdash P\mathcal{R}Q$ *implies that* $\sigma \sim \rho$*, and* strongly consistent *if* $(\sigma, \rho) \vdash P\mathcal{R}Q$ *implies that* $\sigma \sim_s \rho$*.*

Unlike what was done in Chapter 1, here we introduce the weak versions of the equivalence relations. Therefore, we must abstract from the internal actions of processes by considering transitions of the type $\Longrightarrow$ which represent an unbounded number of successive τ actions (possibly zero), and $\overset{\alpha}{\Longrightarrow}$ which represent the transition(s) $\Longrightarrow\xrightarrow{\alpha}\Longrightarrow$.

Definition 2.3.11 *Weak alley bisimilarity*
A consistent symmetric alley relation $\mathcal{R}$ is an alley bisimulation *if $(\sigma, \rho) \vdash P\mathcal{R}Q$ implies:*

- *if $P \xrightarrow{\tau} P'$ then there is a Q' such that $Q \Longrightarrow Q'$ and $(\sigma, \rho) \vdash P'\mathcal{R}Q'$*
- *if $P \xrightarrow{a\,M} P'$ and there are ζ, $\tilde{b}$, b such that $e(\sigma(\zeta)) = M$ with $(\sigma, \rho) \vdash a \leftrightarrow b$, $\{\tilde{b}\} = n(\zeta)\backslash dom(\sigma)$ and $\{\tilde{b}\} \cap fn(P, Q, \sigma, \rho) = \emptyset$, then there exist $\tilde{c}$, Q' with $\{\tilde{c}\} \subset \mathcal{N}$, $|\{\tilde{c}\}| = |\{\tilde{b}\}|$, $\{\tilde{c}\} \cap dom(\sigma) = \emptyset$ such that $Q \overset{b\,e(\rho(\zeta))}{\Longrightarrow} Q'$ and $(\sigma\{\tilde{b}/\tilde{c}\}, \rho\{\tilde{b}/\tilde{c}\}) \vdash P'\mathcal{R}Q'$*
- *if $P \xrightarrow{(\nu\tilde{c})\overline{a}\langle M\rangle} P'$ with $fn(P, \sigma) \cap \{\tilde{c}\} = \emptyset$ and $(\sigma, \rho) \vdash a \leftrightarrow b$ then there are Q', N, $\tilde{d}$, x with $fn(Q, \rho) \cap \{\tilde{d}\} = \emptyset$ such that $Q \overset{(\nu\tilde{d})\overline{b}\langle N\rangle}{\Longrightarrow} Q'$ and $(\sigma\{M/x\}, \rho\{N/x\}) \vdash P'\mathcal{R}Q'$*

Given a pair of substitutions (σ, ρ), two processes P and Q are weakly alley bisimilar *if $P\mathcal{R}Q$ for some alley bisimulation $\mathcal{R}$.* Alley bisimilarity*, written $\approx_a$, is the largest alley bisimulation.*

With respect to the definition of alley bisimulation just presented, note that when an internal action is performed nothing is revealed so the environments remain unchanged, when an input action is performed the environments are extended with the new names created, and when an output action is performed the environments are extended with the new messages.

Definition 2.3.12 *Weak trellis bisimilarity*
A weak trellis bisimulation *is a strongly consistent alley bisimulation. Given a pair of substitutions (σ, ρ), two processes P and Q are* weakly trellis bisimilar *if $P\mathcal{R}Q$ for some trellis bisimulation $\mathcal{R}$.* Trellis bisimilarity*, written $\approx_t$, is the largest trellis bisimulation.*

Note that since strongly consistent environments are also consistent, then whenever two processes are trellis bisimilar they are also alley bisimilar. The opposite is not true, as seen in the Example 2.3.14.

We now examine some of the examples mentioned in [23][3] and in [24].

Example 2.3.13 *Let $\sigma = \rho = \{a/x_1\}$, $P = (\nu n, k, l)\overline{a}\{\{n\}_k\}_l.P'$ and $Q = (\nu n, k)\overline{a}\{n\}_k.Q'$ where $P' = Q' = (\nu m)\overline{a}m$. An alley bisimulation relating P and Q is given by*
$R = \{((\sigma, \rho), P, Q),$
$((\sigma\{\{\{n\}_k\}_l/x_2\}, \rho\{\{n\}_k/x_2\}), P', Q'),$
$((\sigma\{\{\{n\}_k\}_l/x_2, m/x_3\}, \rho\{\{n\}_k/x_2, m/x_3\}), \mathbf{0}, \mathbf{0})\}.$
In other words, if $P \xrightarrow{(\nu n,k,l)\overline{a}\langle\{\{n\}_k\}_l\rangle} P'$ and $(\sigma, \rho) \vdash a \leftrightarrow a$ then $Q \xrightarrow{(\nu n,k)\overline{a}\langle\{n\}_k\rangle} Q'$ and the resulting environments are consistent since neither k nor l are known. In addition, $P' \xrightarrow{(\nu m)\overline{a}\langle m\rangle} \mathbf{0}$ and $Q' \xrightarrow{(\nu m)\overline{a}\langle m\rangle} \mathbf{0}$ and the resulting environments are once again consistent.

[3] The examples proposed are used to differentiate the notion of hedged bisimulation therein introduced (and proven to coincide with alley bisimulation) from other notions of bisimulation introduced for the spi-calculus.

Further, note that the pairs of environments considered are actually strongly consistent, whereby the relation presented is not only an alley bisimulation but also a trellis bisimulation.

Example 2.3.14 *Let* $\sigma = \rho = \{a/x_1\}$, $P = (\nu m, k, l)\overline{a}\{\{m\}_l\}_k.(\overline{a}m + \overline{a}l)$ *and* $Q = (\nu m, k)\overline{a}\{m\}_k.\overline{a}m$. *An alley bisimulation relating* P *and* Q *is given by*
$\mathcal{R} = \{((\sigma, \rho), P, Q),$
$((\sigma\{\{\{m\}_l\}_k/x_2\}, \rho\{\{m\}_k/x_2\}), \overline{a}m + \overline{a}l, \overline{a}m),$
$((\sigma\{\{\{m\}_l\}_k/x_2, m/x_3\}, \rho\{\{m\}_k/x_2, m/x_3\}), \mathbf{0}, \mathbf{0}),$
$((\sigma\{\{\{m\}_l\}_k/x_2, l/x_3\}, \rho\{\{m\}_k/x_2, m/x_3\}), \mathbf{0}, \mathbf{0})\}$.
In particular note that, given $P' = \overline{a}m + \overline{a}l$ *and* $Q' = \overline{a}m$, *whether* $P' \xrightarrow{\overline{a}\langle m\rangle} \mathbf{0}$ *or* $P' \xrightarrow{\overline{a}\langle l\rangle} \mathbf{0}$ *it can be matched by* $Q' \xrightarrow{\overline{a}\langle m\rangle} \mathbf{0}$ *since* l *and* m *can not be distinguished by the environment (this because* k *is kept secret). Nonetheless,* $P \not\approx_t Q$ *since the environments resulting from the transitions* $P' \xrightarrow{\overline{a}\langle l\rangle} \mathbf{0}$ *and* $Q' \xrightarrow{\overline{a}\langle m\rangle} \mathbf{0}$ *are not strongly consistent because* $m \neq l$.

Example 2.3.15 *Let* $\sigma = \rho = \{a/x_1\}$, $P = (\nu k, n)\overline{a}\{n\}_k.P'$ *where* $P' = (\nu m)\overline{a}m$, *and* $Q = (\nu k, n)\overline{a}\{n\}_k.Q'$ *where* $Q' = \overline{a}n$ *and* $n \neq a$. *An alley bisimulation relating* P *and* Q *is given by*
$\mathcal{R} = \{((\sigma, \rho), P, Q),$
$((\sigma\{\{n\}_k/x_2\}, \rho\{\{n\}_k/x_2\}), P', Q'),$
$((\sigma\{\{n\}_k/x_2, m/x_3\}, \rho\{\{n\}_k/x_2, n/x_3\}), \mathbf{0}, \mathbf{0})\}$.
Note that the two processes are alley bisimilar since the environments do not distinguish between the cipher text n *(since* k *is not revealed) and name* m. *Nonetheless, the two processes are not trellis bisimilar since the environments are not strongly consistent because* $m \neq n$.

The next three examples were proposed in [24] in order to highlight certain particularities of the introduced notion of alley bisimulation.

Example 2.3.16 *Let* $\sigma = \rho = \{a/x_1, b/x_2, c/x_3\}$ *where* a, b, c *are distinct,* $P = (\nu k)\overline{a}\{b\}_k.\overline{a}k$ *and* $Q = (\nu k)\overline{a}\{c\}_k.\overline{a}k$. *Then* P *and* Q *are not alley bisimilar since* $P \xrightarrow{(\nu k)\overline{a}\langle\{b\}_k\rangle} P' = \overline{a}k$ *and* $Q \xrightarrow{(\nu k)\overline{a}\langle\{c\}_k\rangle} Q' = \overline{a}k$ *where* (σ, ρ) *is extended to* $(\sigma', \rho') = (\sigma\{\{b\}_k/x_4\}, \rho\{\{c\}_k/x_4\})$. *Note that so far there is no inconsistency in the environments. However, when* k *is revealed, i.e.,* $P' \xrightarrow{\overline{a}\langle k\rangle} \mathbf{0}$ *and* $Q' \xrightarrow{\overline{a}\langle k\rangle} \mathbf{0}$ *then* (σ', ρ') *is extended to* $(\sigma'', \rho'') = (\sigma'\{k/x_5\}, \rho'\{k/x_5\})$ *which is no longer consistent. Although* $dom(\sigma'') = dom(\rho'')$ *and for* $i \in \{1, ..., 5\}$ $C(\sigma'', x_i) \in \mathcal{N}$ *if and only if* $C(\rho'', x_i) \in \mathcal{N}$, *it is not true for* i, $j \in \{1, ..., 5\}$ *that* $C(\sigma'', x_i) = C(\sigma'', x_j)$ *if and only if* $C(\rho'', x_i) = C(\rho'', x_j)$. *In particular,* $C(\sigma'', x_2) = C(\sigma'', x_4) = b$ *but* $C(\rho'', x_2) = b \neq c = C(\rho'', x_4)$; *this is caused by the revelation of the cipher texts once the key* k *is known.*

Example 2.3.17 *Let* $\sigma = \rho = \{c/x_1\}$, $P = (\nu a, k)\overline{c}\{k\}_k.\overline{c}a$ *and* $Q = (\nu a, k)\overline{c}\{\{k\}_a\}_k.\overline{c}a$. *A consistent alley bisimulation relating* P *and* Q *is given by*

$\mathcal{R} = \{((\sigma, \rho), P, Q),$
$((\sigma\{\{k\}_k/x_2\}, \rho\{\{\{k\}_a\}_k/x_2\}), (\nu a)\overline{c}a =_\alpha (\nu b)\overline{c}b, \overline{c}a),$
$((\sigma\{\{k\}_k/x_2, b/x_3\}, \rho\{\{\{k\}_a\}_k/x_2, a/x_3\}), \mathbf{0}\,, \mathbf{0}\,)\}.$
Since the 'external' encryption key k is not revealed, the consistency of the environments is preserved. Also, note that we did an α-conversion on the process resulting from the output transition of P: $P' = (\nu a)\overline{c}a =_\alpha (\nu b)\overline{c}b$ so as to distinguish the name a (renamed to b) which occurs restricted in P' from that which appears free in $Q' = \overline{c}a$.

Example 2.3.18 *Let $\sigma = \rho = \{c/x_1\}$, $P = (\nu a, k)\overline{c}\{k\}_k.\overline{c}a.\overline{a}c$ and $Q = (\nu a, k)\overline{c}\{\{k\}_a\}_k.\overline{c}a.\overline{a}c$. A consistent alley bisimulation relating P and Q is given by*
$\mathcal{R} = \{((\sigma, \rho), P, Q),$
$((\sigma\{\{k\}_k/x_2\}, \rho\{\{\{k\}_a\}_k/x_2\}), (\nu a)\overline{c}a.\overline{a}c =_\alpha (\nu b)\overline{c}b.\overline{b}c, \overline{c}a.\overline{a}c),$
$((\sigma\{\{k\}_k/x_2, b/x_3\}, \rho\{\{\{k\}_a\}_k/x_2, a/x_3\}, \overline{b}c, \overline{a}c),$
$((\sigma\{\{k\}_k/x_2, b/x_3, c/x_4\}, \rho\{\{\{k\}_a\}_k/x_2, a/x_3, c/x_4\}, \mathbf{0}\,, \mathbf{0}\,)\}.$
Once again, note that we did an α-conversion on the process resulting from the output transition of P: $P' = (\nu a)\overline{c}a.\overline{a}c =_\alpha (\nu b)\overline{c}b.\overline{b}c$ so as to distinguish the name a that is restricted in P' from that which appears free in $Q' = \overline{c}a.\overline{a}c$.

We now introduce the notions of weak barbed bisimilarity and weak barbed equivalence in a similar way as these were introduced in Chapter 1. A generalized version of the latter was shown to coincide with weak early bisimulation for the spi-calculus in [24]. Note that $P \Downarrow_u$ or $P \Downarrow_{\overline{u}}$ if $P \overset{\alpha}{\Longrightarrow}$ where $\alpha = u(y)$ or $\alpha \in \{\overline{u}y, \overline{u}(y)\}$ respectively.

Definition 2.3.19 *Weak barbed bisimilarity*
A binary symmetric relation $\mathcal{S}$ is a weak barbed bisimulation *if $P\mathcal{S}Q$ implies:*

- *if $P{\downarrow_\beta}$ then $Q{\Downarrow_\beta}$ for each barb β*
- *if $P \xrightarrow{\tau} P'$ then there is a Q' such that $Q \Longrightarrow Q'$ and $P'\mathcal{S}Q'$*

Two processes P and Q are weakly barbed bisimilar *if $P\mathcal{S}Q$ for some weak barbed bisimulation $\mathcal{S}$.* Weak barbed bisimilarity*, written $\approx_b$, is the largest barbed bisimulation.*

Definition 2.3.20 *Weak barbed equivalence*
Two processes P and Q are weakly barbed equivalent*, written $P \approx_{beq} Q$, if for every process R we have that $P|R \approx_b Q|R$.*

Note that, as with the strong versions, $\approx_{beq} \subset \approx_b$ but the opposite inclusion does not hold. In addition, note that $\sim_{beq} \subset \approx_{beq}$ and $\sim_b \subset \approx_b$, that is, two processes (strongly) barbed equivalent [resp. bisimilar] are also weakly barbed equivalent [resp. bisimilar].

We now introduce the notion of generalized barbed equivalence and prove that it coincides with the notion of barbed equivalence under certain conditions, following the result mentioned in [24].

Definition 2.3.21 *Generalized barbed bisimilarity*
*Let P, $Q \in \mathcal{P}_{spi}$ and $(\sigma, \rho) = ([M_i/x_i]_{i\in I}, [M'_i/x_i]_{i\in I})$ be a consistent pair of substitutions. Let $N_i = C(\sigma, x_i)$ and $N'_i = C(\rho, x_i)$ for each $i \in I$. A binary symmetric relation $\mathcal{S}$ of processes is a (σ, ρ)-*barbed bisimulation *if $P\mathcal{S}Q$ implies:*

- *if $P \downarrow_{N_i}$ then $Q \Downarrow_{N'_i}$ for each $i \in I$*

- *if $P \xrightarrow{\tau} P'$ then there is a Q' such that $Q \Longrightarrow Q'$ and $P'\mathcal{S}Q'$*

*Two processes P and Q are (σ, ρ)-*barbed bisimilar*, written $(\sigma, \rho) \vdash P \approx_b Q$, if $P\mathcal{S}Q$ for some (σ, ρ)-barbed bisimulation $\mathcal{S}$. (σ, ρ)-*barbed bisimilarity *is the largest (σ, ρ)-barbed bisimulation.*

Note that there are significant differences between the notions of barbed bisimulation and generalized barbed bisimulation. Namely, only the names known to the environments, N_i's and N''_i's, are checked. Second, the names N_i and N'_i are not required to be equal. Also note that $\approx_b$ is only closed under the contexts that can be obtained via instantiation with σ and ρ.

Definition 2.3.22 *Generalized barbed equivalence*
*Two processes P and Q are (σ, ρ)-*barbed equivalent*, written $(\sigma, \rho) \vdash P \approx_{beq} Q$, if for every process R with $fn(R) \subseteq dom(\sigma)$we have that $(\sigma, \rho) \vdash P|R\sigma \approx_b Q|R\rho$.*

The following proposition relates the notions of barbed equivalence and generalized barbed equivalence and was proposed in [24]. Note that $\sigma_V = \{x'_1/x_1, ...x'_n/x_n\}$ if $V = \{x'_1, ..., x'_n\}$ where $n \in \mathbb{N}$.

Proposition 2.3.23 *$P \approx_{beq} Q$ if and only if $(\sigma_V, \sigma_V) \vdash P \approx_{beq} Q$ for some V containing $fn(P, Q)$.*

We only provide the idea behind the proof of this result, which relies on the following considerations: *i*) the processes P and Q being compared perform every visible action compared on the same channel, else condition 3. for a consistent environment [See Definition 2.3.6] is violated. *ii*) the environments can not be extended by output actions from P and Q with a and b where $a \neq b$ and $a, b \in fn(P, Q)$ since a process put in parallel with both P and Q distinguishes them. As an example, let $P = \overline{c}a$ and $Q = \overline{c}b$; if $R = c(x).\overline{x}$ then $P|R \xrightarrow{\tau}\downarrow_{\overline{a}}$ and $Q|R \xrightarrow{\tau}\downarrow_{\overline{b}}$ but $a \neq b$. *iii*) the environments can not be extended by output actions from P and Q with a and b where $a \neq b$ and $a \in fn(P)$ or $b \in fn(Q)$ exclusively, since a process put in parallel with both P and Q distinguishes them. As an example, let $P = \overline{c}a$ and $Q = (\nu b)\overline{c}b$; if $R = c(x).\overline{x}$ then $P|R \xrightarrow{\tau}\downarrow_{\overline{a}}$ but $Q|R \xrightarrow{\tau}\not\downarrow_{\overline{a}}$.

We now introduce the notion of structurally image-finite processes so we can present the main result of [24]: the coincidence between alley bisimulation and barbed equivalence.

Definition 2.3.24 *Structurally image-finite process*
A process $P \in \mathcal{P}_{spi}$ is structurally image-finite *if for each visible trace s*[4] *the set of equivalence classes $\{P' : P \overset{s}{\Longrightarrow} P'/_{\equiv}\}$ is finite, where the structural equivalence $\equiv$ was defined in Definition 1.3.7.*

Proposition 2.3.25 *Let P, $Q \in \mathcal{P}_{spi}$ and σ, ρ be consistent environments. If $(\sigma, \rho) \vdash P \approx_a Q$ then $(\sigma, \rho) \vdash P \approx_{beq} Q$ (soundness). If P, Q are structurally image-finite processes, whenever $(\sigma, \rho) \vdash P \approx_{beq} Q$ then $(\sigma, \rho) \vdash P \approx_a Q$ (completeness).*

Note that, in particular, the following result holds since if $\sigma = \rho$, (σ, ρ) is a consistent environment-pair:

Corollary 2.3.26 *Let P, $Q \in \mathcal{P}_{spi}$ be structurally image-finite processes such that $fn(P, Q) \subseteq V$. Then, $(\sigma_V, \sigma_V) \vdash P \approx_a Q$ if and only if $(\sigma_V, \sigma_V) \vdash P \approx_{beq} Q$.*

The following result is a combination of Corollary 2.3.26 with Proposition 2.3.23.

Corollary 2.3.27 *Let P, $Q \in \mathcal{P}_{spi}$ be structurally image-finite processes such that $fn(P, Q) \subseteq V$. Then, $(\sigma_V, \sigma_V) \vdash P \approx_a Q$ if and only if $P \approx_{beq} Q$.*

We now recall Example 2.3.16 and analyse it with respect to barbed equivalence. Note that we consider this particular example because $P \not\approx_{beq} Q$ which is easy to prove when recurring to the definition of barbed equivalence. An arbitrary proof that $P \approx_{beq} Q$ requires quantification over all processes $R \in \mathcal{P}_{spi}$ which is difficult to accomplish. Thus, the more important is the result in Corollary 2.3.27 which establishes the coincidence of barbed equivalence with the more tractable notion of alley bisimulation.

Example 2.3.28 *Following Example 2.3.16, let $\sigma_V = \{a/x_1, b/x_2, c/x_3\}$, $P = (\nu k)\overline{a}\{b\}_k.\overline{a}k$ and $Q = (\nu k)\overline{a}\{c\}_k.\overline{a}k$. Then $P \not\approx_{beq} Q$ since if $R = a(x).a(y).let\, z = dec_y(x).\overline{z}$ then $P|R \xrightarrow{\tau}\xrightarrow{\tau}\downarrow_{\overline{b}}$ but $Q|R \xrightarrow{\tau}\xrightarrow{\tau}\downarrow_{\overline{c}}$ and $b \neq c$.*

The following diagram exhibits the relations between the notions presented for the spi-calculus, where $\rightarrow$ stands for strict inclusion $\subset$, and $\leftrightarrow$ for coincidence.

$$\begin{tikzcd}
\approx_b & \approx_{beq} \arrow[l] \arrow[d, leftrightarrow, "{(\sigma_V,\sigma_V)}"] & \\
& \approx_a & \approx_t \arrow[l]
\end{tikzcd}$$

[4] A *trace* is a sequence of visible actions $\alpha_1, ..., \alpha_n, ...$ such that for all $i \neq j$, $bn(\alpha_i) \cap bn(\alpha_j) = \emptyset$ and for all $j < i$, $bn(\alpha_i) \cap fn(\alpha_j) = \emptyset$. (These conditions are to ensure that the bound names of the trace are fresh).

Chapter 3

Encodings

3.1 Correctness of Encodings

In this section we introduce the notion of encoding and study some measures of correctness proposed in the literature. We start by formally presenting the definition of encoding.

Definition 3.1.1 *Encoding*
An encoding *is a function* $[|.|] : S \rightarrow T$ *where* S *represents the source calculus which is being encoded in the target calculus* T*.*

Encodings are extremely important to determine the expressiveness of a calculus in relation to another; if a calculus represents the target language of an encoding of the source calculus then it is considered at least as expressive as the source language.

In this section, following the work in [17, 11, 18], we attempt to provide a definition of 'good' encoding. We rely on some notions already known but also on a certain amount of intuition.

Definition 3.1.2 *Semantic equivalence*
A source term S *and its translation* $[|S|]$ *are* semantically equivalent *if* $S \asymp [|S|]$ *where* $\asymp$ *denotes a notion of equivalence.*

The former definition has two requirements that are not easily (if at all) satisfied: that the direct comparison between term and translation is possible, and the existence of an applicable 'good' equivalence relation. Intuitively, the stronger the employed equivalence, the stronger our belief that the encoding is correct. Several possible equivalence relations were introduced and studied in the previous sections [See Section 1.3 and Section 2.3]; nonetheless many more equivalence relations have been presented (See e.g. [16]), some of which can be more adequate as a choice of correctness requirement for an encoding for a certain problem.

If a direct comparison between term and translation is not possible, it is useful to consider the notion of *full abstraction* following [16]; the full abstraction problem was first studied in [19, 20].

Definition 3.1.3 *Full abstraction*
Let S_1, S_2 be two elements of the source language and $[|S_1|]$, $[|S_2|]$ be their respective encodings into the target language. Then full abstraction *requires that $S_1 \asymp_s S_2$ if and only if $[|S_1|] \asymp_t [|S_2|]$ where $\asymp_s$ and $\asymp_t$ are equivalence relations in the source and target languages respectively.*

The reflection of equivalence - *adequacy* - provides *behavioural soundness*, while the preservation of equivalence provides *behavioural completeness.*

In the case when full abstraction cannot be achieved for any known equivalence relation, certain requirements like compositionality and the preservation of some intended semantics may be enough to ensure the 'quality' of the encoding. The notions of uniform and reasonable encoding were presented in [11], and those of strongly uniform and sensible encoding in [10].

Definition 3.1.4 *Uniform encoding*
Let σ be a substitution, S_1, S_2 be two elements of the source language and $[|S_1|]$, $[|S_2|]$ be their respective encodings into the language target. The encoding is uniform *if the following properties hold:*

- $[|S_1 \mid S_2|] = [|S_1|] \mid [|S_2|]$

- $[|S_1\sigma|] = [|S_1|]\sigma$

The encoding is strongly uniform *if the second condition of the uniform encoding accounts for arbitrary substitutions, i.e., the second condition reads $\forall\sigma\,\exists\theta\,[|S_1\sigma|] = [|S_1|]\theta$.*

Definition 3.1.5 *Reasonable encoding*
An encoding is reasonable *if it preserves a reasonable semantics: a semantics which distinguishes two processes P and Q whenever in some computation of P the actions on a certain intended channel are different from those in any computation of Q.*

Definition 3.1.6 *Sensible encoding*
An encoding is sensible *if it is (strongly) uniform, reasonable and distinguishes deadlocks from livelocks.*

3.2 Encoding of Match

Encodings are not only meaningful as a way of determining the expressivity of a calculus, but also in establishing which operators should be considered primitives of a certain calculus and which can be derived.

In this section, we prove that the match operator can be encoded in the π-calculus with polyadic synchronization; thus, unlike in the π-calculus, it does not need to be taken as a primitive. This fact results directly from the possibility of partial restriction in the π-calculus with polyadic synchronization.

Note that, following [10], we do not consider the usual match operator $[x = y]$ but $[x = y]\tau$ instead. In this sense, a process $P = [x = y]\tau.P'$ can evolve, if indeed x and y are equal, by means of an internal action into P'; the rule MATCH is then replaced by $\tau-$MATCH.

$$\frac{P \xrightarrow{\alpha} P'}{[x = x]P \xrightarrow{\alpha} P'}\ \text{MATCH} \qquad\qquad \frac{}{[x = x]\tau.P \xrightarrow{\tau} P}\ \tau - MATCH$$

This definition of match has the advantage of not only making the encoding simpler, but also that the observability of actions depends of the structure of the process and not on the set of bindings between names and channels as seen in [10]. Thus, in the π-calculus with the match operator $[x = y]\tau$, any process P for any name x and any substitution σ, can perform the action $x(y)$, $\overline{x}y$ or $\overline{x}(y)$, if and only if $P\sigma$ can perform the action $\sigma(x)(y)$, $\overline{\sigma(x)}y$ or $\overline{\sigma(x)}(y)$ respectively. This property also holds in the π-calculus with polyadic synchronization.

Proposition 3.2.1 *Encoding of match with respect to late congruence*[1]
Let $P \in \mathcal{P}_S$ such that $z \notin fn(P)$. If $[\![\, [x = y]\tau.P \,]\!] = (\nu z)(\overline{z \cdot x}|z \cdot y.P)$, then $[x = y]\tau.P$ and $[\![\, [x = y]\tau.P \,]\!]$ are late congruent.

Proof: Let $Q = [x = y]\tau.P$ and $R = (\nu z)(\overline{z \cdot x}|z \cdot y.P)$, where $P \in \mathcal{P}_S$ such that $z \notin fn(P)$, and σ be an arbitrary substitution. We show that $P\sigma \sim_l Q\sigma$.
We split the proof according to the behaviour of σ on x and on y.

- $\sigma(x) \neq \sigma(y)$: both processes $Q\sigma = [\sigma(x) = \sigma(y)]\tau.(P\sigma)$ and $R\sigma = (\nu z)(\overline{z \cdot \sigma(x)}|z \cdot \sigma(y).(P\sigma))$ are inactive; in the first the match is unsuccessful, and in the latter the synchronization is not possible because the channel vectors do not match element wise.

- $\sigma(x) = \sigma(y)$: both processes $Q\sigma = [\sigma(x) = \sigma(y)]\tau.(P\sigma)$ and $R\sigma = (\nu z)(\overline{z \cdot \sigma(x)}|z \cdot \sigma(y).(P\sigma))$ may perform a deterministic τ action and evolve into $P\sigma$; in the first the match is successful, and in the latter the τ action results from the synchronization

[1] We could prove the same result resorting to an even stronger notion of equivalence: open bisimilarity.

of the vectors. Actually, $R\sigma \xrightarrow{\tau} (\nu z)(\mathbf{0}\,|P\sigma) \sim_l P\sigma$ since we may assume that $z \notin fn(P\sigma)$.

Therefore, by definition of late congruence, we have that $Q \simeq_l R$, that is we have that $[x = y]\tau \simeq_l [|\,[x = y]\tau.P\,|]$.

□

Note that the restriction on z in Proposition 3.2.1 is crucial since it prevents the input and output transitions through $z \cdot y$ and $\overline{z \cdot x}$, respectively, of $R = (\nu z)(\overline{z \cdot x}|z \cdot y.P)$. Thus, the process R can only perform the internal communication referred to in the proof of the proposition.

We could also have opted to consider the following encoding of the match operator: $[|\,[x = y]P\,|] = (\nu z)(\overline{z \cdot x}|z \cdot y.P)$ and prove that $[x = y]P$ and its respective encoding were weakly late congruent. However, it was proven in [4] for the π-calculus that the intuitive weak version of late bisimilarity was no longer an equivalence relation. The same result also holds for the π-calculus with polyadic synchronization since it corresponds to the particular case of the calculus when the channel vectors are unitary. The fact that the intuitive notion of weak late bisimilarity is not a transitive relation is also true for the intuitive notion of weak late congruence. The example provided in [4] presents not only weak late bisimilar processes but also weak late congruent processes such that although $P_1 \simeq_l P_2$ and $P_2 \simeq_l P_3$, it does not hold that $P_1 \simeq_l P_3$ [See Appendix].

The definition of an alternative notion of weak late bisimilarity and subsequently of weak late congruence only add to the complexity of the proof of Proposition 3.2.1 and this is the reason why we prefer to follow [4] and present instead the alternative notion of match operator.

In [10] it was also proven that the match operator cannot be derived in the π-calculus since there was no sensible encoding of that operator. In addition, in [10] it was shown that there is no sensible encoding of the π-calculus with polyadic synchronization into the π-calculus[2] which yields the following result.

Corollary 3.2.2 *The π-calculus with polyadic synchronization is more expressive than the π-calculus.*

Proof: The π-calculus is embedded in the π-calculus with polyadic synchronization with the match operator. Following Proposition 3.2.1, we know that the match operator can be derived in the π-calculus with polyadic synchronization, and thus needs not be considered as a primitive. The encoding described in that proposition is not only sensible with respect to late congruence, which would be enough to establish the expressiveness relation between the calculi, but the source term in the π-calculus and its respective encoding in

[2] Actually, the result is even stronger stating that the higher the degree of synchronization of a language - the maximum length of the composite channels - the more expressive the calculus.

the π-calculus with polyadic synchronization are semantically equivalent with respect to late congruence.
The lack of a sensible encoding of the π-calculus into the π-calculus with polyadic synchronization serves to prove that the latter is (strictly) more expressive than the first, that is, that the expressivity of the calculi cannot be the same.

□

3.3 Encoding of Cryptographic Primitives

In this section we prove that cryptographic primitives can be encoded in the π-calculus with polyadic synchronization; thus proving that this calculus does not need to be extended with those primitives. In Chapter 2, we analysed the spi-calculus: an extension of the π-calculus with cryptographic primitives in order to understand the importance and particularities of encryption and decryption of messages. The relation between the π-calculus and spi-calculus was studied in [25] from an innovative perspective[3], which resulted in the establishment of an encoding of spi-calculus into π-calculus with respect to may-testing equivalence[4].

To our knowledge, the first mention of a possible encoding of a calculus with cryptographic primitives into a calculus with polyadic synchronization was put forth in [12]. The idea can be summarized in the following way: the sending of a message m encrypted under a key k over a channel a can be seen as $\overline{a \cdot k}m.P$. In order to receive this message, the other party needs to know the channel where the message is being transmitted and the key, which could be represented as $a \cdot k(m).P$. There are several problems with this approach [12, 10]; an example is that one cannot represent nested encryptions.

A solution for a different encoding of the cryptographic π-calculus with polyadic synchronization into the π-calculus with polyadic synchronization was proposed in [10]. The proposed encoding is a homomorphism, except for the following new constructs:

$$\begin{aligned} [|\, \mathit{encrypt}\, m \hookrightarrow^{k} x\, \mathit{in}\, P \,|] &= (\nu x)(!\overline{x \cdot k}m \mid [|\, P \,|]) \\ [|\, \mathit{decrypt}\, x \hookrightarrow^{k} m\, \mathit{in}\, P \,|] &= x \cdot k(m).[|\, P \,|] \end{aligned}$$

Note that the constructs just presented are more expressive than those proposed in [12] since the encrypted messages are represented as names which can still be encrypted, sent or used as keys[5], and that the encryption is nondeterministic (encrypting the same message

[3] In particular, the spi-calculus processes are viewed as objects with a number of predefined methods, and where encryption corresponds to creating an object with a special method for decryption.

[4] In may-testing equivalence, the basis for comparing processes is the result of experiments in which the processes are tested by composing them with special processes. As mentioned in [25] the notion of equivalence employed is rather weak, so we will not concern ourselves with its detailed analysis.

[5] Note that in the spi-calculus studied in Chapter 2 we did not consider messages of the type $\{a\}_{\{b\}_k}$. We follow the same restriction here.

$$\text{(ENC)}\qquad \frac{P \xrightarrow{\alpha} P'}{\mathit{encrypt}\, m \looparrowright^{k} x\, \mathit{in}\, P \xrightarrow{\alpha} \mathit{encrypt}\, m \looparrowright^{k} x\, \mathit{in}\, P'}$$

where if $\alpha \in \{\overline{u}y, \overline{u}(y), u(y)\}$ then $x \notin n(u)$ and $\alpha \neq \overline{u}x$

$$\text{(ENC-OPEN)}\qquad \frac{P \xrightarrow{\overline{u}x} P'}{\mathit{encrypt}\, m \looparrowright^{k} x\, \mathit{in}\, P \xrightarrow{\overline{u}(x)} \overline{x \cdot k}m|P'}$$

where $x \notin n(u)$

$$\text{(DEC)}\qquad \frac{--}{\mathit{decrypt}\, x \looparrowright^{k} y\, \mathit{in}\, P \xrightarrow{x \cdot k(y)} P}$$

Table 3.1: Late transition rules for the cryptographic constructs

under the same key yields different results). The first construct, encrypts the cipher text m under k and returns the encrypted message as the fresh name x to be used in all the scope embraced by P. The decryption of message x through the key k (used to encrypt the message) binds the name m in the continuation of P to the original message. Nonetheless we rely on a slightly different and simpler encryption construct:

$$[\![\, \mathit{encrypt}\, m \looparrowright^{k} x\, \mathit{in}\, P \,]\!] \;=\; (\nu x)(\overline{x \cdot k}m \mid [\![\, P \,]\!])$$

Consequently we require that encrypted messages stop being 'available' as soon as decryption is done. Thus, whenever we want to encrypt a certain message m under a key k twice we have a process like $P = \mathit{encrypt}\, m \looparrowright^{k} x_1\, \mathit{in}\, \mathit{encrypt}\, m \looparrowright^{k} x_2\, \mathit{in}\, P'$.

The transition semantics of the cryptographic π-calculus with polyadic synchronization includes, besides the rules of the π-calculus with polyadic synchronization in Table 1.3, the rules in Table 3.1. We now explain the reasoning behind the inclusion of each of the rules in Table 3.1, and reiterate the side conditions of these rules.

- The rule ENC determines that an action of P is also an action of $\mathit{encrypt}\, m \looparrowright^{k} x\, \mathit{in}\, P$, thus implying that messages can be encrypted at any time. This rule has as side conditions that if $\alpha \in \{\overline{u}y, \overline{u}(y), u(y)\}$ then $x \notin n(u)$, and $\alpha \neq \overline{u}x$. The last case is handled by the rule ENC-OPEN.
- The rule ENC-OPEN determines that when an encrypted message is sent over a channel its scope is opened to include the process that receives it (via application of rule CLOSE). Also note that once the encrypted message x is sent, the possibility

of decryption is assured since any process with knowledge of x and the key used to encrypt it k can synchronize with it and get hold of the original message m. This rule has as side condition that $x \notin n(u)$.

- The rule DEC shows that a process with knowledge of the encrypted message x and encryption key k awaits for the sending of the original cipher text on the channel $x \cdot k$ to evolve.

The notion of structural congruence presented in Definition 1.3.7 can be extended to make use of the new constructs with the following rules:

$$\mathit{encrypt}\, m \looparrowright^k x\, \mathit{in}\, P \equiv \mathit{encrypt}\, m \looparrowright^k x\, \mathit{in}\, Q \text{ if } P \equiv Q$$

$$\mathit{decrypt}\, x \looparrowright^k m\, \mathit{in}\, P \equiv \mathit{decrypt}\, x \looparrowright^k m\, \mathit{in}\, Q \text{ if } P \equiv Q$$

The notions of bisimilarity introduced in Section 1.3 can be adjusted to consider processes in the cryptographic π-calculus with polyadic synchronization. We assume that the notion of early congruence in the cryptographic π-calculus with polyadic synchronization can be obtained in the same manner as in Definition 1.3.18 and that the result in Corollary 1.3.44 can be extended to the cryptographic π-calculus with polyadic synchronization.

Before we attempt to prove the soundness and completeness of the encoding, we first analyse a cryptographic protocol proposed in [10] from the source and target language view points.

Example 3.3.1 *Consider the processes:*
$P = (\nu k)\overline{secure}k.public(y).decrypt\, y \looparrowright^k w\, in\, R$
$Q = (\nu m)secure(z).encrypt\, m \looparrowright^z x\, in\, \overline{public}x.S$
and the cryptographic protocol defined as $(\nu secure)(P|Q)|A$ *where* A *is a possible attacker. Intuitively the attacker can never have access to the key* k *since the transfer of the knowledge of the key is done on a secure channel, thus we will not concern ourselves with the process* A*. Obviously, we consider in the target language the following processes:*
$[\!|\, P \,|\!] = (\nu k)\overline{secure}k.public(y).y \cdot k(w).[\!|\, R \,|\!]$
$[\!|\, Q \,|\!] = (\nu m)secure(z).(\nu x)(\overline{x \cdot z}(m)|\overline{public}x.[\!|\, S \,|\!])$

We split the analysis in two, first considering the transitions in the source language and then in the target language.[6]

- *Source Language: Cryptographic π-calculus with polyadic synchronization*

  ◆ *The key is transmitted:* $(\nu secure)(P|Q) \xrightarrow{\tau} (\nu secure, k)(P'|Q')$*, where*
  $P' = public(y).decrypt\, y \looparrowright^k w\, in\, R$
  $Q' = (\nu m)encrypt\, m \looparrowright^k x\, in\, \overline{public}x.S$

[6] We do not include the α-conversions on the derivation trees so as to make them more legible. These are implicit.

$Q_1 = \mathit{secure}(z).\mathit{encrypt}\,m \looparrowright^z x\,\mathit{in}\,\overline{\mathit{public}}x.S$
$Q_1' = \mathit{encrypt}\,m \looparrowright^z x\,\mathit{in}\,\overline{\mathit{public}}x.S$
This transition corresponds to the following derivation tree:

$$\dfrac{\dfrac{\dfrac{--}{\overline{\mathit{secure}}k.\mathit{public}(y).\mathit{decrypt}\,y \looparrowright^k w\,\mathit{in}\,R \xrightarrow{\overline{\mathit{secure}}k} P'}\,\text{PREFIX}}{P = (\nu k)\overline{\mathit{secure}}k.\mathit{public}(y).\mathit{decrypt}\,y \looparrowright^k w\,\mathit{in}\,R \xrightarrow{\overline{\mathit{secure}}(k)} P'}\,\text{OPEN} \qquad \dfrac{\dfrac{--}{Q_1 \xrightarrow{\mathit{secure}(z)} Q_1'}\,\text{PREFIX}}{Q \xrightarrow{\mathit{secure}(z)} Q'}\,\text{RES}}{\dfrac{P|Q \xrightarrow{\tau} (\nu k)(P'|Q')}{(\nu \mathit{secure})(P|Q) \xrightarrow{\tau} (\nu \mathit{secure}, k)(P'|Q')}\,\text{RES}}\,\text{CLOSE1}$$

- *The encrypted message is transmitted:*
$(\nu \mathit{secure}, k)(P'|Q') \xrightarrow{\tau} (\nu \mathit{secure}, k, m)(\mathit{encrypt}\,m \looparrowright^k x\,\mathit{in}\,(P''|S))$, *where*
$P'' = \mathit{decrypt}\,x \looparrowright^k w\,\mathit{in}\,R$
$Q'' = (\nu m)(\overline{x \cdot k}m|S)$
This transition corresponds to the following derivation tree:

$$\dfrac{\dfrac{\dfrac{--}{P' \xrightarrow{\mathit{public}(x)} P''}\,\text{PREFIX} \qquad \dfrac{\dfrac{\dfrac{--}{\overline{\mathit{public}}x.S \xrightarrow{\overline{\mathit{public}}x} S}\,\text{PREFIX}}{\mathit{encrypt}\,m \looparrowright^k x\,\mathit{in}\,\overline{\mathit{public}}x.S \xrightarrow{\overline{\mathit{public}}(x)} \overline{x \cdot k}m|S}\,\text{ENC-OPEN}}{Q' \xrightarrow{\overline{\mathit{public}}(x)} Q''}\,\text{RES}}{P'|Q' \xrightarrow{\tau} (\nu x)(P''|Q'')}\,\text{CLOSE2}}{(\nu \mathit{secure}, k)(P'|Q') \xrightarrow{\tau} (\nu \mathit{secure}, k, x)(P''|Q'')}\,\text{RES}$$

- *The encrypted message is decrypted:*
$(\nu \mathit{secure}, k, x)(P''|Q'') \xrightarrow{\tau} (\nu \mathit{secure}, k, x, m)(R|(\mathbf{0}\,|S)) \equiv (\nu \mathit{secure}, k, x, m)(R|S)$.
This transition corresponds to the following derivation tree:

$$\dfrac{\dfrac{\dfrac{--}{P'' \xrightarrow{x \cdot k(m)} R}\,\text{DEC} \qquad \dfrac{\dfrac{\dfrac{--}{\overline{x \cdot k}m \xrightarrow{\overline{x \cdot k}m} \mathbf{0}}\,\text{PREFIX}}{\overline{x \cdot k}m|S \xrightarrow{\overline{x \cdot k}m} \mathbf{0}\,|S}\,\text{PAR1}}{Q'' \xrightarrow{\overline{x \cdot k}(m)} \mathbf{0}\,|S}\,\text{OPEN}}{P''|Q'' \xrightarrow{\tau} (\nu m)(R|(\mathbf{0}\,|S))}\,\text{CLOSE2}}{(\nu \mathit{secure}, k, x)(P''|Q'') \xrightarrow{\tau} (\nu \mathit{secure}, k, x, m)(R|S)}\,\text{RES}$$

- *Target Language: π-calculus with polyadic synchronization*

  - ♦ *The key is transmitted:* $(\nu secure)([\!|\, P \,|\!] \mid [\!|\, Q \,|\!]) \xrightarrow{\tau} (\nu secure, k)(P'|Q')$, *where*
    $P' = public(y).y \cdot k(w).[\!|\, R \,|\!]$
    $Q' = (\nu m, x)(\overline{x \cdot k}m|\overline{public}x.[\!|\, S \,|\!])$
    $Q_1 = secure(z).(\nu x)(\overline{x \cdot z}m|\overline{public}x.[\!|\, S \,|\!])$
    $Q_1' = (\nu x)(\overline{x \cdot z}m|\overline{public}x.[\!|\, S \,|\!])$
    This transition corresponds to the following derivation tree:

$$\dfrac{\dfrac{\dfrac{\dfrac{--}{\overline{secure}k.public(y).y \cdot k(w).R \xrightarrow{\overline{securek}} P'}\ \text{PREFIX}}{[\!|\, P \,|\!] \xrightarrow{\overline{secure}(k)} P'}\ \text{OPEN} \qquad \dfrac{\dfrac{--}{Q_1 \xrightarrow{secure(z)} Q_1'}\ \text{PREFIX}}{Q \xrightarrow{secure(z)} Q'}\ \text{RES}}{[\!|\, P \,|\!] \mid [\!|\, Q \,|\!] \xrightarrow{\tau} (\nu k)(P'|Q')}\ \text{CLOSE1}}{(\nu secure)([\!|\, P \,|\!] \mid [\!|\, Q \,|\!]) \xrightarrow{\tau} (\nu secure, k)(P'|Q')}\ \text{RES}$$

  - ♦ *The encrypted message is transmitted:*
    $(\nu secure, k)(P'|Q') \xrightarrow{\tau} (\nu secure, k, x)(x \cdot k(w).[\!|\, R \,|\!] \mid (\nu m)(\overline{x \cdot k}m \mid [\!|\, S \,|\!]))$
    This transition corresponds to the following derivation tree:

$$\dfrac{\dfrac{\dfrac{--}{P' \xrightarrow{public(x)} x \cdot k(w).[|R|]}\ \text{PREFIX} \qquad \dfrac{\dfrac{\dfrac{\dfrac{--}{\overline{public}x.[|S|] \xrightarrow{\overline{public}x} [|S|]}\ \text{PREFIX}}{\overline{x \cdot k}m|\overline{public}x.[|S|] \xrightarrow{\overline{public}x} \overline{x \cdot k}m|[|S|]}\ \text{PAR2}}{(\nu m)(\overline{x \cdot k}m|\overline{public}x.[|S|]) \xrightarrow{\overline{public}x} (\nu m)(\overline{x \cdot k}m|[|S|])}\ \text{RES}}{Q' \xrightarrow{\overline{public}(x)} (\nu m)(\overline{x \cdot k}m|[|S|])}\ \text{OPEN}}{P'|Q' \xrightarrow{\tau} (\nu x)(x \cdot k(w).[|R|]|(\nu m)(\overline{x \cdot k}m|[|S|]))}\ \text{CLOSE2}}{(\nu secure, k)(P'|Q') \xrightarrow{\tau} (\nu secure, k, x)(x \cdot k(w).[|R|]|(\nu m)(\overline{x \cdot k}m|[|S|]))}\ \text{RES}$$

  - ♦ *The encrypted message is decrypted:*
    $(\nu secure, k, x)(x{\cdot}k(w).[\!|\, R \,|\!] \mid (\nu m)(\overline{x \cdot k}m \mid [\!|\, S \,|\!])) \xrightarrow{\tau} (\nu secure, k, x, m)([\!|\, R \,|\!] \mid [\!|\, S \,|\!])$
    This transition corresponds to the following derivation tree:

$$\cfrac{\cfrac{\cfrac{--}{x\cdot k(w).[|R|] \xrightarrow{x\cdot k(m)} [|R|]}\ \text{PREFIX} \qquad \cfrac{\cfrac{\cfrac{--}{\overline{x\cdot k}m \xrightarrow{\overline{x\cdot km}} \mathbf{0}}\ \text{PREFIX}}{\overline{x\cdot k}m|[|S|] \xrightarrow{\overline{x\cdot km}} \mathbf{0}\,|[|S|]}\ \text{PAR2}}{(\nu m)(\overline{x\cdot k}m|[|S|]) \xrightarrow{\overline{x\cdot k}(m)} [|S|]}\ \text{OPEN}}{x\cdot k(w).[|R|]|(\nu m)(\overline{x\cdot k}m|[|S|]) \xrightarrow{\tau} (\nu m)([|R|]|[|S|])}\ \text{CLOSE2}}{(\nu secure, k, x)(x\cdot k(w).[|R|]|(\nu m)(\overline{x\cdot k}m|[|S|])) \xrightarrow{\tau} (\nu secure, k, x, m)([|R|]|[|S|])}\ \text{RES}$$

Before we prove the soundness and completeness of the proposed encoding, we first introduce the following lemma.

Lemma 3.3.2 $[|\,P\sigma\,|] = [|\,P\,|]\sigma$, *for any substitution* σ

PROOF: We split the proof according to the constructs in the cryptographic π-calculus with polyadic synchronization. The only relevant cases are those of the new constructs.

- If $P = \mathit{encrypt}\, m \looparrowright^k x\, \mathit{in}\, P'$ then $[|\,P\,|] = (\nu x)(\overline{x\cdot k}m|\,[|\,P'\,|])$. Let $\sigma = \{m'/m\}$[7]. Then, $P\sigma = \mathit{encrypt}\, m' \looparrowright^k x\, \mathit{in}\, P'\{m'/m\}$ and $[|\,P\sigma\,|] = (\nu x)(\overline{x\cdot k}m'|\,[|\,P'\{m'/m\}\,|])$ and we have that $[|\,P\sigma\,|] = (\nu x)(\overline{x\cdot k}m'|\,[|\,P'\,|]\{m'/m\})$.
  In addition, $[|\,P\,|]\sigma = ((\nu x)(\overline{x\cdot k}m|\,[|\,P'\,|]))\sigma = (\nu x)(\overline{x\cdot k}m'|\,[|\,P'\,|]\{m'/m\})$.

- If $P = \mathit{decrypt}\, x \looparrowright^k y\, \mathit{in}\, P'$ then $[|\,P\,|] = x\cdot k(y).[|\,P'\,|]$. Let $\sigma = \{x'/x\}$[8]. Then, $P\sigma = \mathit{decrypt}\, x' \looparrowright^k y\, \mathit{in}\, P'\{x'/x\}$ and $[|\,P\sigma\,|] = x'\cdot k(y).[|\,P'\{x'/x\}\,|]$ and we have that $[|\,P\sigma\,|] = x'\cdot k(y).[|\,P'\,|]\{x'/x\}$.
  In addition, $[|\,P\,|]\sigma = (x\cdot k(y).[|\,P'\,|])\sigma = x'\cdot k(y).[|\,P'\,|]\{x'/x\}$.

□

In order to prove the soundness and completeness of the encoding with respect to barbed congruence - a contextual equivalence -, which we proved in Corollary 1.3.44 coincides with early congruence, we build on successive auxiliary results. Note that we will consider as a target language a sub-calculus of the π-calculus with polyadic synchronization without summation (as we just saw cryptographic protocols do not necessarily make use of this construct). We shall consider this calculus with and without cryptographic primitives (encoding the first in the second) because the proof is lighter.

The following lemmas show there is an operational correspondence between the actions of the encoding and those of the process.

[7] The other cases are handled in an analogous way. Note that if e.g. $\sigma = \{x/m\}$ then we would have to perform α-conversion.

[8] The other cases are handled in an analogous way. Note that if e.g. $\sigma = \{y/k\}$ then we would have to perform α-conversion.

Lemma 3.3.3 *If P is a process in the cryptographic π-calculus with polyadic synchronization and $[|P|]$ is its respective encoding, then*[9]*:*

- *if $[|P|] \xrightarrow{\overline{u}y} Q$ then for some P_1, P_2 and some $\tilde{n}$ such that $n(u) \cap \{\tilde{n}\} = \emptyset$, $P \equiv (\nu\tilde{n})(\overline{u}y.P_1|P_2)$ where $[|P|] = (\nu\tilde{n})(\overline{u}y.[|P_1|] \,|\, [|P_2|])$ and $Q \equiv (\nu\tilde{n})([|P_1|] \,|\, [|P_2|])$. We can also have that $P \equiv (\nu\tilde{n})(encrypt\, m \looparrowright^k x\, in\, \overline{u}y.P_1|P_2)$ where $x \neq y$, $[|P|] = (\nu\tilde{n})(\overline{x \cdot k}m|\overline{u}y.[|P_1|] \,|\, [|P_2|])$ and $Q \equiv (\nu\tilde{n})(\overline{x \cdot k}m \,|\, [|P_1|] \,|\, [|P_2|])$.*

- *if $[|P|] \xrightarrow{\overline{u}(y)} Q$ then for some P_1, P_2 and some $\tilde{n}$ such that $n(u) \cap \{\tilde{n}\} = \emptyset$ and $y \neq x$, $P \equiv (\nu\tilde{n}, y)(\overline{u}y.P_1|P_2)$ where $[|P|] = (\nu\tilde{n}, y)(\overline{u}y.[|P_1|] \,|\, [|P_2|])$ and $Q \equiv (\nu\tilde{n})([|P_1|] \,|\, [|P_2|])$. We can also have that $P \equiv (\nu\tilde{n}, y)(encrypt\, m \looparrowright^k x\, in\, \overline{u}y.P_1|P_2)$ where $[|P|] = (\nu\tilde{n}, y, x)(\overline{x \cdot k}m|\overline{u}y.[|P_1|] \,|\, [|P_2|])$ and $Q \equiv (\nu\tilde{n}, x)(\overline{x \cdot k}m \,|\, [|P_1|] \,|\, [|P_2|])$. If $x = y$ we can also have that $P \equiv (\nu\tilde{n})(encrypt\, m \looparrowright^k y\, in\, \overline{u}y.P_1|P_2)$ where $[|P|] = (\nu\tilde{n}, y)(\overline{y \cdot k}m|\overline{u}y.[|P_1|] \,|\, [|P_2|])$ and $Q \equiv (\nu\tilde{n})(\overline{y \cdot k}m \,|\, [|P_1|] \,|\, [|P_2|])$.*

- *if $[|P|] \xrightarrow{u(y)} Q$ then for some P_1, P_2 and some $\tilde{n}$ such that $n(u) \cap \{\tilde{n}\} = \emptyset$, $P \equiv (\nu\tilde{n})(u(y).P_1|P_2)$ where $[|P|] = (\nu\tilde{n})(u(y).[|P_1|] \,|\, [|P_2|])$ and $Q \equiv (\nu\tilde{n})([|P_1|] \,|\, [|P_2|])$. We can also have that $P \equiv (\nu\tilde{n})(encrypt\, m \looparrowright^k x\, in\, u(y).P_1|P_2)$ where $[|P|] = (\nu\tilde{n}, x)(\overline{x \cdot k}m|u(y).[|P_1|] \,|\, [|P_2|])$ and $Q \equiv (\nu\tilde{n}, x)(\overline{x \cdot k}m \,|\, [|P_1|] \,|\, [|P_2|])$. If $u = x \cdot k$, we can also have that $P \equiv (\nu\tilde{n})(decrypt\, x \looparrowright^k y\, in\, P_1|P_2)$, where $[|P|] = (\nu\tilde{n})(x \cdot k(y).[|P_1|] \,|\, [|P_2|])$ and $Q \equiv (\nu\tilde{n})([|P_1|] \,|\, [|P_2|])$.*

PROOF: Follows directly from the definition of processes in the cryptographic π-calculus with polyadic synchronization and from the transition rules.

□

Lemma 3.3.4 *For any process P in the cryptographic π-calculus with polyadic synchronization we have that if $[|\, P \,|] \xrightarrow{\alpha} Q$ then there is a P' such that $P \xrightarrow{\alpha} P'$ and $[|P'|] = Q$.*

PROOF: The proof is done by induction on the inference of the transition of $[|\, P \,|] \xrightarrow{\alpha} Q$.

1. $\alpha = \overline{u}y$. By Lemma 3.3.3 we have that:

   i) $P \equiv (\nu\tilde{n})(\overline{u}y.P_1|P_2)$ in which case $P \xrightarrow{\overline{u}y} P' \equiv (\nu\tilde{n})(P_1|P_2)$ and $[|P'|] = Q$.

   ii) $P \equiv (\nu\tilde{n})(encrypt\, m \looparrowright^k x\, in\, \overline{u}y.P_1|P_2)$ in which case $P \xrightarrow{\overline{u}y} P' \equiv (\nu\tilde{n})(P_1|P_2)$ and $[|P'|] = Q$.

2. $\alpha = \overline{u}(y)$. By Lemma 3.3.3 we have that:

   i) $P \equiv (\nu\tilde{n}, y)(\overline{u}y.P_1|P_2)$ in which case $P \xrightarrow{\overline{u}(y)} P' \equiv (\nu\tilde{n})(P_1|P_2)$ and $[|P'|] = Q$.

   ii) $P \equiv (\nu\tilde{n}, y)(encrypt\, m \looparrowright^k x\, in\, \overline{u}y.P_1|P_2)$ in which case $P \xrightarrow{\overline{u}(y)} P' \equiv (\nu\tilde{n})(encrypt\, m \looparrowright^k x\, in\, P_1|P_2)$ and $[|P'|] = Q$.

   iii) $P \equiv (\nu\tilde{n})(encrypt\, m \looparrowright^k y\, in\, \overline{u}y.P_1|P_2)$ in which case $P \xrightarrow{\overline{u}(y)} P' \equiv (\nu\tilde{n})(\overline{y \cdot k}m|P_1|P_2)$ and $[|P'|] = Q$.

[9] Note that the notion of structurally equivalent processes, denoted by $\equiv$, was presented in Definition 1.3.7.

3. $\alpha = u(y)$. By Lemma 3.3.3 we have that:
   i) $P \equiv (\nu\tilde{n})(u(y).P_1|P_2)$ in which case $P \xrightarrow{u(y)} P' \equiv (\nu\tilde{n})(P_1|P_2)$ and $[|P'|] = Q$.
   ii) $P \equiv (\nu\tilde{n})(\mathit{encrypt}\, m \looparrowright^k x\, \mathit{in}\, u(y).P_1|P_2)$ in which case $P \xrightarrow{u(y)} P' \equiv (\nu\tilde{n})(\mathit{encrypt}\, m \looparrowright^k x\, \mathit{in}\, P_1|P_2)$ and $[|P'|] = Q$.
   iii) $u = x \cdot k$ and $P \equiv (\nu\tilde{n})(\mathit{decrypt}\, x \looparrowright^k y\, \mathit{in}\, P_1|P_2)$ in which case $P \xrightarrow{x \cdot k(y)} P' \equiv (\nu\tilde{n})(P_1|P_2)$ and $[|P'|] = Q$.

4. $\alpha = \tau$. The most relevant cases in which a process can perform a τ action is by application of rules COMM or CLOSE. In the first case, $P|Q \xrightarrow{\tau} P'|Q'\{y/z\}$ if $P \xrightarrow{\overline{u}y} P'$ and $Q \xrightarrow{u(z)} Q'$. By points 1 and 3 we know there exist P_1, P_1', Q_1, Q_1' such that $[|P_1|] = P$, $[|Q_1|] = Q$, $P_1 \xrightarrow{\overline{u}y} P_1'$ where $[|P_1'|] = P'$, and $Q_1 \xrightarrow{u(z)} Q_1'$ where $[|Q_1'|] = Q'$. Thus, $P_1|Q_1 \xrightarrow{\tau} P_1'|Q_1'\{y/z\}$ and by Lemma 3.3.2 $[|P_1'|Q_1'\{y/z\}|] = [|P_1'|] \mid [|Q_1'|]\{y/z\} = P'|Q'\{y/z\}$. The remaining cases follow in a similar manner.

□

Lemma 3.3.5 *For any process P in the cryptographic π-calculus with polyadic synchronization we have that:*

- *if $P \xrightarrow{\overline{u}y} P'$ then for some P_1, P_2 and some $\tilde{n}$ such that $n(u) \cap \{\tilde{n}\} = \emptyset$, $P \equiv (\nu\tilde{n})(\overline{u}y.P_1|P_2)$ or $P \equiv (\nu\tilde{n})(\mathit{encrypt}\, m \looparrowright^k x\, \mathit{in}\, (\overline{u}y.P_1|P_2))$ where $y \neq x$ and $P' \equiv (\nu\tilde{n})(P_1|P_2)$ or $P' \equiv (\nu\tilde{n})(\mathit{encrypt}\, m \looparrowright^k x\, \mathit{in}\, (P_1|P_2))$ respectively.*

- *if $P \xrightarrow{\overline{u}(y)} P'$ then for some P_1, P_2 and some $\tilde{n}$ such that $n(u) \cap \{\tilde{n}, y\} = \emptyset$, $P \equiv (\nu\tilde{n})(\overline{u}y.P_1|P_2)$ or $P \equiv (\nu\tilde{n}, y)(\mathit{encrypt}\, m \looparrowright^k x\, \mathit{in}\, (\overline{u}y.P_1|P_2))$ and $P' \equiv (\nu\tilde{n})(P_1|P_2)$ or $P' \equiv (\nu\tilde{n})(\mathit{encrypt}\, m \looparrowright^k x\, \mathit{in}\, (P_1|P_2))$ respectively. In addition, we can also have that $P \equiv (\nu\tilde{n})(\mathit{encrypt}\, m \looparrowright^k y\, \mathit{in}\, (\overline{u}(y).P_1|P_2))$ and $P' \equiv (\nu\tilde{n})(\overline{x \cdot k}m|P_1|P_2)$.*

- *if $P \xrightarrow{u(y)} P'$ then for some P_1, P_2 and some $\tilde{n}$ such that $n(u) \cap \{\tilde{n}\} = \emptyset$, $P \equiv (\nu\tilde{n})(u(y).P_1|P_2)$ or $P \equiv (\nu\tilde{n})(\mathit{encrypt}\, m \looparrowright^k x\, \mathit{in}\, (u(y).P_1|P_2))$ and $P' \equiv (\nu\tilde{n})(P_1|P_2)$ or $P' \equiv (\nu\tilde{n})(\mathit{encrypt}\, m \looparrowright^k x\, \mathit{in}\, (P_1|P_2))$ respectively. In addition, we can also have that $u = x \cdot k$, $P \equiv (\nu\tilde{n})(\mathit{decrypt}\, x \looparrowright^k y\, \mathit{in}\, (P_1|P_2))$ and $P' \equiv (\nu\tilde{n})(P_1|P_2)$.*

Proof: Follows directly from the definition of processes in the cryptographic π-calculus with polyadic synchronization and from the transition rules.

□

The following Lemma establishes a strong operational correspondence between the actions of a process and the actions of its encoding.

Lemma 3.3.6 *For any process P in the cryptographic π-calculus with polyadic synchronization, if $P \xrightarrow{\alpha} P'$ then $[|P|] \xrightarrow{\alpha} [|P'|]$*

Proof: The proof is done by induction on the inference of the transition $P \xrightarrow{\alpha} P'$.

1. If $\alpha = \overline{u}y$, by Lemma 3.3.5 we have that $P \equiv (\nu\tilde{n})(\overline{u}y.P_1|P_2)$ or $P \equiv (\nu\tilde{n})(encrypt\, m \looparrowright^k x\, in\, (\overline{u}y.P_1|P_2))$ where $y \neq x$ and $P' \equiv (\nu\tilde{n})(P_1|P_2)$ or $P' \equiv (\nu\tilde{n})(encrypt\, m \looparrowright^k x\, in\, (P_1|P_2))$ respectively. In the first case, $[| P |] = (\nu\tilde{n})(\overline{u}y.[|P_1|] \,|\, [|P_2|])$ and $[| P |] \xrightarrow{\overline{u}y} (\nu\tilde{n})([|P_1|] \,|\, [|P_2|]) = [| P' |]$. In the second case, $[| P |] = (\nu\tilde{n}, x)(\overline{x \cdot k}m|\overline{u}y.[|P_1|] \,|\, [|P_2|])$ and $[| P |] \xrightarrow{\overline{u}y} (\nu\tilde{n}, x)(\overline{x \cdot k}m| \,[|P_1|] \,|\, [|P_2|]) = [| P' |]$.

2. If $\alpha = \overline{u}(y)$ the reasoning is analogous, except if $P \equiv (\nu\tilde{n})(encrypt\, m \looparrowright^k y\, in\, (\overline{u}(y).P_1|P_2))$ and $P' \equiv (\nu\tilde{n})(\overline{y \cdot k}m|P_1|P_2)$; then $[| P |] = (\nu\tilde{n}, y)(\overline{y \cdot k}m|\overline{u}y.[| P_1 |] \,|\, [| P_2 |])$ and $[| P |] \xrightarrow{\overline{u}(y)} (\nu\tilde{n})(\overline{y \cdot k}m| \,[| P_1 |] \,|\, [| P_2 |]) = [| P' |]$.

3. If $\alpha = u(y)$, by Lemma 3.3.5 we have that:
   i) $P \equiv (\nu\tilde{n})(u(y).P_1|P_2)$ and $P' \equiv (\nu\tilde{n})(P_1|P_2)$. Then $P = [| P |]$ and $P' = [| P' |]$ so if $P \xrightarrow{u(y)} P'$ so does $[| P |] \xrightarrow{u(y)} [| P' |]$;
   ii) $P \equiv (\nu\tilde{n})(encrypt\, m \looparrowright^k x\, in\, (u(y).P_1|P_2))$ and $P' \equiv (\nu\tilde{n})(encrypt\, m \looparrowright^k x\, in\, (P_1|P_2))$. Then $[| P |] = (\nu\tilde{n}, x)(\overline{x \cdot k}(m)|u(y).[|P_1|] \,|\, [|P_2|])$ and $[| P |] \xrightarrow{u(y)} (\nu\tilde{n}, x)(\overline{x \cdot k}(m)| \,[|P_1|] \,|\, [|P_2|]) = [| P' |]$;
   iii) $P \equiv (\nu\tilde{n})(decrypt\, x \looparrowright^k y\, in\, P_1|P_2)$ and $P' \equiv (\nu\tilde{n})(P_1|P_2)$. Then $P \xrightarrow{x \cdot k(y)} P'$ and since $[| P |] = (\nu\tilde{n})(x \cdot k(y).[|P_1|] \,|\, [|P_2|])$ we have that $[| P |] \xrightarrow{x \cdot k(y)} (\nu\tilde{n})([|P_1|] \,|\, [|P_2|]) = [| P' |]$.

4. If $\alpha = \tau$ many rules may have been applied; we consider only the case when the rule COMM was applied (the others are analogous). Note that $P|Q \xrightarrow{\tau} P'|Q'\{y/z\}$ if $P \xrightarrow{\overline{u}y} P'$ and $Q \xrightarrow{u(z)} Q'$. By points 1 and 3 we have that $[|P|] \xrightarrow{\overline{u}y} [|P'|]$ and $[|Q|] \xrightarrow{u(z)} [|Q'|]$ and by application of rule COMM $[|P|] \,|\, [|Q|] \xrightarrow{\tau} [|P'|] \,|\, [|Q'|]\{y/z\}$ and by Lemma 3.3.2 we have that $[|P'|] \,|\, [|Q'|]\{y/z\} = [|P'|] \,|\, [|Q'\{y/z\}|] = [|P'|Q'\{y/z\}|]$.

□

Lemma 3.3.7 *If* $[| P |] \sim_e [| Q |]$ *then* $P \sim_e Q$

Proof: We prove that $\mathcal{R} = \{(P, Q) : [| P |] \sim_e [| Q |]\}$ is an early bisimulation. We split the proof according to the possible transitions of $[| P |]$

- $\alpha \in \{\overline{u}y, \overline{u}(y)\}$ where $bn(\alpha) \cap fn(P, Q) = \emptyset$. If $P \xrightarrow{\alpha} P'$ then by Lemma 3.3.6 we have that $[| P |] \xrightarrow{\alpha} [| P' |]$. Since by hypothesis $[| P |] \sim_e [| Q |]$ then there is a Q' such that $[| Q |] \xrightarrow{\alpha} Q'$ and by Lemma 3.3.4 we have that there is a Q'' such that $Q \xrightarrow{\alpha} Q''$ where $[| Q'' |] = Q'$. By definition of $\sim_e$ we have that $[| P' |] \sim_e [| Q'' |]$ and therefore $P'\mathcal{R}Q''$.

- $\alpha = u(y)$ where $y \notin fn(P, Q)$. If $P \xrightarrow{\alpha} P'$ then by Lemma 3.3.6 we have that $[| P |] \xrightarrow{\alpha} [| P' |]$. Since by hypothesis $[| P |] \sim_e [| Q |]$ then there is a Q' such that $[| Q |] \xrightarrow{\alpha} Q'$ and by Lemma 3.3.4 we have that there is a Q'' such that $Q \xrightarrow{\alpha} Q''$ where $[| Q'' |] = Q'$. By definition of $\sim_e$ we have that $[| P' |]\{w/y\} \sim_e [| Q'' |]\{w/y\}$ and by application of Lemma 3.3.2 we know that $[| P'\{w/y\}|] \sim_e [| Q''\{w/y\}|]$. Thus, $P'\{w/y\}\mathcal{R}Q''\{w/y\}$.

- $\alpha = \tau$. If $P \xrightarrow{\alpha} P'$ then by Lemma 3.3.6 we have that $[| P |] \xrightarrow{\alpha} [| P' |]$. Since by hypothesis $[| P |] \sim_e [| Q |]$ then there is a Q' such that $[| Q |] \xrightarrow{\alpha} Q'$ and by Lemma 3.3.4 we have that there is a Q'' such that $Q \xrightarrow{\alpha} Q''$ where $[| Q'' |] = Q'$. By Lemma 1.3.8, definition and transitivity of $\sim_e$ we have that $[| P' |] \sim_e [| Q'' |]$ and therefore $P'\mathcal{R}Q''$.

□

Lemma 3.3.8 *If* $[| P |] \sim_{beq} [| Q |]$ *then* $P \sim_{beq} Q$

Proof: Follows directly from Lemma 3.3.7 and Theorem 1.3.43 where it was proven that early bisimulation coincides with barbed equivalence.

□

Theorem 3.3.9 *Soundness*
If $[| P |] \simeq_b [| Q |]$ *then* $P \simeq_b Q$

Proof: If $[| P |] \simeq_b [| Q |]$ then for any substitution σ we have that $[| P |]\sigma \sim_{beq} [| Q |]\sigma$. By Lemma 3.3.2 we then know that $[| P\sigma |] \sim_{beq} [| Q\sigma |]$, and by Lemma 3.3.8 we have that $P\sigma \sim_{beq} Q\sigma$.

□

Lemma 3.3.10 *If* $P \sim_e Q$ *then* $[| P |] \sim_e [| Q |]$

Proof: We prove that $\mathcal{R} = \{([| P |], [| Q |]) : P \sim_e Q\}$ is an early bisimulation. We split the proof according to the possible transitions of P.

- $\alpha \in \{\overline{u}y, \overline{u}(y)\}$ where $bn(\alpha) \cap fn(P, Q) = \emptyset$. If $[| P |] \xrightarrow{\alpha} P'$ then by Lemma 3.3.4 we have that there is a P'' such that $P \xrightarrow{\alpha} P''$ and $[| P'' |] = P'$. Since by hypothesis $P \sim_e Q$ there is a Q' such that $Q \xrightarrow{\alpha} Q'$ and by Lemma 3.3.6 we have that $[| Q |] \xrightarrow{\alpha} [| Q' |]$. By definition of $\sim_e$ we have that $P' \sim_e Q'$ and therefore $[| P' |]\mathcal{R}[| Q' |]$.

- $\alpha = u(y)$ where $y \notin fn(P, Q)$. If $[| P |] \xrightarrow{\alpha} P'$ then by Lemma 3.3.4 we have that there is a P'' such that $P \xrightarrow{\alpha} P''$ and $[| P'' |] = P'$. Since by hypothesis $P \sim_e Q$ there is a Q' such that $Q \xrightarrow{\alpha} Q'$ and by Lemma 3.3.6 we have that $[| Q |] \xrightarrow{\alpha} [| Q' |]$. By definition of $\sim_e$ we have that $P'\{w/y\} \sim_e Q'\{w/y\}$ and therefore $[| P'\{w/y\} |]\mathcal{R}[| Q'\{w/y\} |]$. By Lemma 3.3.2 we have that $[| P' |]\{w/y\}\mathcal{R}[| Q' |]\{w/y\}$.

- $\alpha = \tau$. If $[\![\, P \,]\!] \xrightarrow{\alpha} P'$ then by Lemma 3.3.4 we have that there is a P'' such that $P \xrightarrow{\alpha} P''$ and $[\![P'']\!] = P'$. Since by hypothesis $P \sim_e Q$ there is a Q' such that $Q \xrightarrow{\alpha} Q'$ and by Lemma 3.3.6 we have that $[\![\, Q \,]\!] \xrightarrow{\alpha} [\![\, Q' \,]\!]$. By definition of $\sim_e$ we have that $P' \sim_e Q'$ and by Lemma 1.3.8 and transitivity of $\sim_e$ we have that $[\![\, P' \,]\!] \mathcal{R} [\![\, Q' \,]\!]$.

□

Lemma 3.3.11 *If $P \sim_{beq} Q$ then $[\![\, P \,]\!] \sim_{beq} [\![\, Q \,]\!]$*

Proof: Follows directly from Lemma 3.3.10 and Theorem 1.3.43 where it was proven that early bisimilarity coincides with barbed equivalence.

□

Theorem 3.3.12 *Completeness*
If $P \simeq_b Q$ then $[\![\, P \,]\!] \simeq_b [\![\, Q \,]\!]$

Proof: If $P \simeq_b Q$ then for any substitution σ we have that $P\sigma \sim_{beq} Q\sigma$. By Lemma 3.3.11 then $[\![\, P\sigma \,]\!] \sim_{beq} [\![\, Q\sigma \,]\!]$ and by Lemma 3.3.2 we know that $[\![\, P \,]\!]\sigma \sim_{beq} [\![\, Q \,]\!]\sigma$, i.e., $[\![\, P \,]\!] \simeq_b [\![\, Q \,]\!]$.

□

Chapter 4

Conclusions and future work

We have studied in detail the π-calculus with polyadic synchronization proposed in [10] where channels are vectors of names. We formally defined and compared some notions of bisimilarity and contextual equivalences in the π-calculus with polyadic synchronization, which to our knowledge had not been done until now.

We have studied the spi-calculus originally proposed in [12] as a model of a calculus which is used to reason about security protocols.

We have extended the π-calculus with polyadic synchronization with cryptographic primitives by defining the syntax and operational semantics of the calculus. Following [10] we proposed an encoding of the new constructs for encryption and decryption of messages into the π-calculus with polyadic synchronization. Further, we proved that such an encoding is sound and complete with respect to barbed congruence (which we also proved coincides with early congruence). We therefore concluded that the π-calculus with polyadic synchronization was expressive enough to be used to model security protocols, which strengthens the hypothesis that an encoding of the spi-calculus into the π-calculus with polyadic synchronization is possible. In addition, we could, as future work, study if and how the π-calculus with polyadic synchronization can express properties of cryptographic protocols such as authenticity and secrecy.

Appendix

In the Appendix we prove that the intuitive weak version of late bisimilarity is not an equivalence relation as mentioned in Section 3.2.

Definition 1 *(Intuitive) weak late bisimilarity*
Let $u = x_1 \cdot \ldots \cdot x_k$ where $k \in \mathbb{N}$ and P, $Q \in \mathcal{P}_S$.
A binary symmetric relation $\mathcal{S}$ is a weak late bisimulation *if $P\mathcal{S}Q$ implies:*

- *if $P \xrightarrow{\alpha} P'$ where $\alpha = \overline{u}y, \overline{u}(y)$ or τ and $bn(\alpha) \notin fn(P,Q)$ then there is a Q' such that $Q \overset{\alpha}{\Longrightarrow} Q'$ and $P'\mathcal{S}Q'$.*

- *if $P \xrightarrow{u(y)} P'$ where $y \notin fn(P,Q)$ then there is a Q' such that $Q \overset{u(y)}{\Longrightarrow} Q'$ and for each w, $P'\{w/y\}\mathcal{S}Q'\{w/y\}$.*

Two processes P and Q are weakly late bisimilar *if $P\mathcal{S}Q$ for some weak late bisimulation $\mathcal{S}$.*
Weak late bisimilarity*, written $\approx_l$, is the largest weak late bisimulation.*

The corresponding definition of weak late congruence is as follows.

Definition 2 *Weak late congruence*
Let P, $Q \in \mathcal{P}_S$. The two processes are weakly late congruent*, written $P \approxeq_l Q$, if for all substitutions σ we have that $P\sigma \approx_l Q\sigma$.*

We now resort to a similar example to that given in [4] which shows that in the weak late congruence relation defined above transitivity does not hold.

Example 1 *Let $P_1 = c(a).P' + c(a).(\tau.P' + \tau.\overline{d}d + \tau)$ and $P_2 = c(a).(\tau.P' + \tau.\overline{d}d + \tau)$ and $P_3 = c(a).(\tau.\overline{d}d + \tau)$, where $P' = (\nu z)(z \cdot a|\overline{z \cdot b}).\overline{d}d$. We have that $P_1 \approx_l P_2$ because if $P_1 \xrightarrow{c(a)} P'$ then $P_2 \xrightarrow{c(a)}\xrightarrow{\tau} P'$. In addition, $P_2 \approx_l P_3$ since if $P_2 \xrightarrow{c(a)} \tau.P' + \tau.\overline{d}d + \tau$ then $P_3 \xrightarrow{c(a)} \tau.\overline{d}d + \tau$ and whether we consider the substitution $\{b/a\}$ or $\{c/a\}$ where $c \neq b$, the derivatives are weakly late bisimilar. Nonetheless, $P_1 \not\approx_l P_3$ because if $P_1 \xrightarrow{c(a)} P'$ then P_3 cannot match it. If $P_3 \xrightarrow{c(a)} \tau.\overline{d}d + \tau$ then $(\nu z)(z \cdot a|\overline{z \cdot b}).\overline{d}d \not\approx_l \tau.\overline{d}d + \tau$; if $P_3 \overset{c(a)}{\Longrightarrow} \overline{d}d$ then $(\nu z)(z \cdot a|\overline{z \cdot b}).\overline{d}d \not\approx_l \overline{d}d$; and finally if $P_3 \overset{c(a)}{\Longrightarrow} \mathbf{0}$ then $((\nu z)(z \cdot a|\overline{z \cdot b}).\overline{d}d)\{b/a\} \not\approx_l \mathbf{0}\,\{b/a\}$. Note that $P_1 \approxeq_l P_2$ and $P_2 \approxeq_l P_3$ also since the only relevant substitution would be of the sort $\sigma = \{b/a\}$ or $\sigma = \{a/b\}$ and these are not possible because a is bound in P_1, P_2 and P_3. Since $P_1 \not\approx_l P_3$, we also have that $P_1 \not\approxeq_l P_3$.*

Alternative notions of weak late bisimilarity that are equivalence relations were put forth and we now present one of these (in e.g. [4]). Weak late congruence is defined as before, resorting to the (new) definition of weak late bisimilarity.

Definition 3 *Weak late bisimilarity*
Let $u = x_1 \cdot ... \cdot x_k$ *where* $k \in \mathbb{N}$ *and* P*,* $Q \in \mathcal{P}_S$*.*
A binary symmetric relation $\mathcal{S}$ *is a* weak late bisimulation *if* $P\mathcal{S}Q$ *implies:*

- *if* $P \xrightarrow{\alpha} P'$ *where* $\alpha = \overline{u}y, \overline{u}(y)$ *or* τ *and* $bn(\alpha) \notin fn(P, Q)$ *then there is a* Q' *such that* $Q \overset{\alpha}{\Longrightarrow} Q'$ *and* $P'\mathcal{S}Q'$*.*

- *if* $P \overset{u(y)}{\Longrightarrow} P'$ *where* $y \notin fn(P, Q)$ *then there is a* Q' *such that* $Q \overset{u(y)}{\Longrightarrow} Q'$ *and for each* w*,* $P'\{w/y\}\mathcal{S}Q'\{w/y\}$*.*

Two processes P *and* Q *are* weakly late bisimilar *if* $P\mathcal{S}Q$ *for some weak late bisimulation* $\mathcal{S}$*.*
Weak late bisimilarity*, written* $\approx_l$*, is the largest weak late bisimulation.*

Bibliography

[1] R. Milner, J. Parrow and D. Walker. *A Calculus of Mobile Processes Part I/II.* Technical Report ECS-LFCS-89-85/86, Laboratory for Foundations of Computer Science, University of Edinburgh, June 1989. Published in *Information and Computation* 100:1-77, 1992.

[2] R. Milner, J. Parrow and D. Walker. *Modal Logics for Mobile Processes.* In CONCUR'91; volume 527 of *Lecture Notes in Computer Science*, pp 45-60, Springer, 1991. Available as Report ECS-LFCS-91-136, University of Edinburgh. Published in *Theoretical Computer Science* 114:149-171, 1993.

[3] D. Sangiorgi and D. Walker. *The π-calculus: A Theory of Mobile Processes.* Cambridge University Press 2003 (1st edition 2001).

[4] D. Sangiorgi. *A Theory of Bisimulation for the π-calculus.* In *Best*, pp 127-142. Extended version as Report ECS-LFCS-93-270, University of Edinburgh. Revised version in *Acta Informatica* 33:69-97, 1996.

[5] P. Quaglia. *The π-calculus: notes on labelled semantics* In *Bulletin of the EATCS*, No 68, June 1999. Preliminary version as BRICS Report LS-98-4.

[6] S. Arun-Kumar and M. Hennessy. *An Efficiency Preorder for Processes.* In *Acta Informatica*, 29:737-760, 1992. Previously published as Computer Science Report 90:05, University of Sussex.

[7] R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.

[8] D. Sangiorgi and R. Milner. *The problem of "weak bisimulation up to".* In *CONCUR'92: Concurrency Theory*, volume 630 of *Lecture Notes in Computer Science.* Springer-Verlag, 1992.

[9] M. Hansen, H. Hüttel and J. Kleist. *Bisimulations for asynchronous mobile processes.* In *Proceedings of the Tiblisi Symposium on Language, Logic, and Computation*, 1995. Research paper HCRC/RP-72, Human Communication Research Centre, University of Edinburgh.

[10] M. Carbone and S. Maffeis. *On the expressive power of polyadic synchronization in π-calculus* In *Nordic Journal of Computing*, pp. 70-98, Volume 10, Number 2, Summer 2003.

[11] C. Palamidessi. *Comparing the Expressive Power of the Synchronous and the Asynchronous π-calculi.* In Volume 13, Number 5 (special issue) *The Difference Between Concurrent and Sequential Computation (2)*, pp. 685-719, October 2003.

[12] M. Abadi and A. D. Gordon. *A calculus for cryptographic protocols: The spi-calculus.* In *Fourth ACM Conference on Computer and Communications Security.* ACM Press, 1997.

[13] R. Milner. *A Calculus of Communicating Systems.* In *Lecture Notes in Computer Science*, vol 92, Springer-Verlag, 1980.

[14] D. Sangiorgi. *Lazy functions and mobile processes.* In *Technical Report RR-2515*, INRIA-Sophia Antipolis, 1995.

[15] R. Milner. *The Polyadic π-calculus: a Tutorial.* In *Technical Report ECS-LFCS-91-180*, University of Edinburgh, October 1991.

[16] D. Sangiorgi. *Expressing Mobility in Process Algebras – First-Order and Higher-Order Paradigms.* PhD thesis, LFCS, University of Edinburgh, 1993.

[17] U. Nestmann and B. C. Pierce. *Decoding Choice Encodings.* In *Proceedings of CONCUR'96*, vol 1119 of LNCS, pages 179-194. Springer, 1996.

[18] U. Nestmann. *What is a 'Good' Encoding of Guarded Choice?* In *Proceedings of EXPRESS'97*, vol 7 of ENTCS. Elsevier Science Publishers, 1997.

[19] R. Milner. *Fully abstract models of typed lambda calculus.* In *Theoretical Computer Science.* 1977.

[20] G. D. Plotkin. *LCF as a programming language.* In *Theoretical Computer Science.* 1977.

[21] R. Milner and D. Sangiorgi. *Barbed Bisimulation.* In *Lecture Notes in Computer Science*, Springer-Verlag, 1992.

[22] U. Frendrup, H. Hüttel and J. N. Jensen. *Modal Logics for Cryptographic Processes.* In *Proceedings of EXPRESS'02*, vol 68(2) of ENTCS. 2002.

[23] J. Borgström and U. Nestmann. *On Bisimulations for the Spi Calculus.* Accepted for publication in *Mathematical Structures in Computer Science.* 2004.

[24] M. Boreale, R. De Nicola and R. Pugliese. *Proof Techniques for Cryptographic Processes.* In *SIAM Journal on Computing.* 2002.

[25] M. Baldamus, J. Parrow and B. Victor. *Spi Calculus Translated to π-Calculus Preserving May-Testing.* In *Report 2003-063*, Department of Information Technology, Uppsala University, 2003.