Spi Calculus Translated to π-Calculus Preserving May-Testing

Michael Baldamus, Joachim Parrow, Björn Victor

Department of Information Technology
Uppsala University
Box 337, SE-751 05 Uppsala, Sweden

Technical report 2003-063
December 2003
ISSN 1404-3203

Spi Calculus Translated to π-Calculus Preserving May-Testing *
(Extended Abstract)

Michael Baldamus, Joachim Parrow, Björn Victor
Department of Information Technology, Uppsala University, Sweden

Abstract

We present a concise and natural encoding of the spi-calculus into the more basic π-calculus and establish its correctness with respect to a formal notion of testing. This is particularly relevant for security protocols modelled in spi since the tests can be viewed as adversaries. The translation has been implemented in a prototype tool. As a consequence, protocols can be described in the spi calculus and analysed with the emerging flora of tools already available for π. The translation also entails a more detailed operational understanding of spi since high level constructs like encryption are encoded in a well known lower level. The formal correctness proof is nontrivial and interesting in its own; so called context bisimulations and new techniques for compositionality make the proof simpler and more concise.

1 Introduction

The current proliferation of computer communication services and technologies is accompanied by an equally bewildering plethora of different formal description techniques. There are many different families with different purposes. Some of them, like the π-calculus [13], aim at a basic formalism with few and low-level primitives, applicable as a springboard for more high-level and specialised techniques. As an example the spi calculus [1, 3] is developed as an extension of it with high-level primitives for among other things encryption and decryption. This makes the spi calculus especially appropriate to describe authentication protocols and services. It is natural to ask if the added complexity of the spi calculus is necessary in a formal sense, or if there is a natural encoding in terms of the more basic π-calculus. The contribution of this paper is to exhibit such an encoding together with new proof techniques to establish its properties.

Our encoding is surprisingly natural and concise. The main idea is to represent spi calculus terms as objects with a number of predefined methods. Encryption corresponds to creating an object with a special method for decryption; the object will perform this only when the correct encryption key is presented. This entails a distributed view of encryptions and avoids a global repository of cleartexts and keys in the formal model.

The spi calculus is expressively powerful enough to model not only a large class of authentication protocols but also a large class of potential adversaries trying to break them. Thus a formal notion of correctness of a protocol is that it satisfies tests which themselves are spi calculus terms. Granted, these tests will not be able to model attacks on the fundamental assumptions (for example that ciphertexts can never be decrypted without access to the encryption key and that it is impossible to interfere with the execution of principals), but they do capture the fact that a protocol is correct under those assumptions. Our main technical result of the encoding is that it is faithful to such tests in a formal sense. Let P be a protocol and E a test, both formulated in the spi calculus, and let $[\![\cdot]\!]$ be our encoding from the spi calculus to the π-calculus. We prove that

$$P \text{ satisfies } E \text{ iff } [\![P]\!] \text{ satisfies } [\![E]\!].$$

Thus, analyses carried out on the encodings are relevant for the original spi calculus descriptions. As expected the formal proof of this is nontrivial. We develop techniques based on so called context bisimulations previously defined in higher-order calculi, and new techniques for compositionality make the proof simpler and more concise. To our knowledge, no equally complicated term-enriched variant of the π-calculus has been analysed in this way, with a translation into π that is formally proved adequate.

One implication of our work is that automated analysis of spi calculus descriptions can be conducted by first translating them to the π-calculus. There is today an emerging flora of automatic tools for the π-calculus, and we have implemented the translation in a prototype that works with some of these tools. This opens the opportunity to formally connect the efforts on tools and algorithms related to the

*Work supported by European Union project PROFUNDIS, Contract No. IST-2001-33100.

calculi. Of course much work remains to determine exactly what analyses can be performed effectively and where the translational approach holds advantages.

In the next section we give the formal syntax and semantics of the spi- and π-calculi; the paper is formally self contained although a reader completely unfamiliar with these calculi will probably have difficulty in appreciating the paper. In Section 3 we present and explain the encoding in detail, and the following section contains a detailed sketch of the main technical result. The final section comments on related and future work.

2 The π- and the Spi-Calculus

In this section we present the syntax and operational semantics of the versions of the π-calculus and spi calculus used.

2.1 The Polyadic π-Calculus

We assume an infinite enumerable set $\mathcal{N}_\pi$ of names, ranged over by $a, b, \ldots$. Names represent communication channels, and are also the values passed in communication.

π-calculus agents are ranged over by P, Q, R, and are formed by the following grammar:

$$\begin{array}{rclllllll} P & ::= & \mathbf{0} & \mid & \alpha.P & \mid & P \mid Q & \mid & P + Q \\ & \mid & (\nu\, a)\, P & \mid & [a = b]\, P & \mid & !\, P & & \end{array}$$

$\mathbf{0}$ is the deadlocked process which can perform no actions. $\alpha.P$ can perform the action α and continue as P; $P \mid Q$ acts as P and Q in parallel; $(\nu\, a)\, P$ binds the name a in P and no process outside P knows about a unless P reveals it by communication; $[a = b]\, P$ can proceed as P only if a and b are the same; $!\, P$ represents any number of copies of P in parallel. We write $(\nu\, \widetilde{a})\, P$ for $(\nu\, a_1) \cdots (\nu\, a_k)\, P$ where $\widetilde{a}$ is the vector $a_1 \cdots a_k$, $k \geq 0$.

Prefixes, ranged over by α, are outputs $\overline{a}\langle\widetilde{b}\rangle$ which send the vector of names $\widetilde{b}$ on the channel a, and inputs $a(\widetilde{b})$, where all names in $\widetilde{b}$ are distinct, which receive names on the channel a and substitute them for $\widetilde{b}$ in the prefixed agent.

Names are bound by the ν operator and by the input prefix. We write $\mathrm{bn}(P)$ and $\mathrm{fn}(P)$ for the bound and free names of P, respectively. We assume, from now on, that all bound names are distinct from each other and from all free names. Also, we identify agents that can be alpha-converted into each other. The result of substituting a name a for (all free occurrences of) a name b in an agent P is denoted by $P[b := a]$, with the expected extension to equal-length vectors $\widetilde{a}$ and $\widetilde{b}$ of names, where all names in $\widetilde{b}$ are mutually distinct.

The operational semantics is given in Table 1 (cf. also [16]), where transition labels, ranged over by μ, are inputs $a(\widetilde{b})$, outputs $(\nu\, \widetilde{b}')\, \overline{a}\langle\widetilde{b}\rangle$ (where $\widetilde{b}' \subseteq \widetilde{b}$ are extruded by the output), and the internal action τ.

2.2 The Spi Calculus

The spi calculus is an extension of the π-calculus with primitives for encryption, decryption and hashcodes, and pairs and the natural numbers as basic datatypes. Encryption is assumed to be perfect, i.e., the possibility to break a cipher by crypto-analysis, chance, or brute force is ignored. The focus is not on cryptographic algorithms but on protocols using them.

We assume the infinite enumerable sets $\mathcal{N}_{spi}$, ranged over by $m, n, \ldots$; and $\mathcal{V}_{spi}$, ranged over by $x, y, \ldots$. The set $\mathcal{N}_{spi}$ represents communication channels and atomic encryption keys, and $\mathcal{V}_{spi}$ represents bound variables to be instantiated by communication.

In the spi calculus, the values sent and received are *terms*, ranged over by M, N, K. These are defined by

$$\begin{array}{lll} M ::= n \quad \mid \quad x & & \text{name, variable} \\ \quad \mid \ 0 \quad \mid \quad \mathrm{suc}(M) & & \text{zero, successor} \\ \quad \mid \ (M, N) & & \text{pair} \\ \quad \mid \ \{M\}_N & & \text{symmetric encryption} \\ \quad \mid \ \{\!|M|\!\}_N & & \text{asymmetric encryption} \\ \quad \mid \ M^+ \quad \mid \quad M^- & & \text{private, public key} \\ \quad \mid \ \mathrm{hashc}(M) & & \text{hashcode} \end{array}$$

and the spi calculus agents $\mathcal{A}_{spi}$, ranged over by P, Q, R, are defined by

$$\begin{array}{lll} P ::= \mathbf{0} \quad \mid \quad \alpha.P \quad \mid \quad P \mid Q \quad \mid \quad (\nu\, n)\, P & & \\ \quad \mid \ [M \text{ **is** } N]\, P \quad \mid \quad !\, P & & \\ \quad \mid \ \mathbf{let}\ (x, y) = M\ \mathbf{in}\ P & & \text{(pair splitting)} \\ \quad \mid \ \mathbf{case}\ M\ \mathbf{of}\ 0 : P\ \mathrm{suc}(x) : Q & & \text{(integer case)} \\ \quad \mid \ \mathbf{case}\ M\ \mathbf{of}\ [x]_N\ \mathbf{in}\ P & & \text{(decryption)} \end{array}$$

The first few agent constructions are familiar from the π-calculus. The **let** and **case** constructions are used for projection: pair splitting binds x and y, which must be distinct, to the first and second component of M and proceeds as P (if M is a pair); integer case proceeds as P if M is 0 or binds x to its predecessor and proceeds as Q (if M is a positive natural number); decryption binds x and proceeds as P if M is a term encrypted with a key matching N.

Prefixes in spi calculus are outputs, $\overline{N}\langle M\rangle$, which sends the term M on the channel N, and inputs, $N(x)$, which receives a term on N and substitutes it for x in the prefixed agent. Here the term N must evaluate to a name of a communication channel.

Names are bound by the ν operator, while variables are bound by input prefixes and **case** constructions. We extend $\mathrm{bn}(P)$ and $\mathrm{fn}(P)$ as expected, and write $\mathrm{fv}(P)$ for the free variables of P. We still assume that all bound names

$$P + \mathbf{0} \equiv P \quad P + Q \equiv Q + P \quad (P + Q) + R \equiv P + (Q + R)$$
$$P \mid \mathbf{0} \equiv P \quad P \mid Q \equiv Q \mid P \quad (P \mid Q) \mid R \equiv P \mid (Q \mid R)$$
$$(\nu\, a)\, \mathbf{0} \equiv \mathbf{0} \quad (\nu\, a)\, P \equiv P \ \text{ if } a \notin \mathrm{fn}(P) \quad (\nu\, a)\, (\nu\, b)\, P \equiv (\nu\, b)\, (\nu\, a)\, P \quad (\nu\, a)(P \mid Q) \equiv ((\nu\, a)\, P) \mid Q \ \text{ if } a \notin \mathrm{fn}(Q)$$

$$\frac{P \equiv Q \xrightarrow{\mu} Q \equiv P'}{P \xrightarrow{\mu} P'} \quad a(\widetilde{b}).P \xrightarrow{a(\widetilde{b})} P \quad \overline{a}\langle\widetilde{b}\rangle.P \xrightarrow{\overline{a}\langle\widetilde{b}\rangle} P \quad \frac{P \xrightarrow{\mu} P'}{P + Q \xrightarrow{\mu} P'} \quad \frac{P \xrightarrow{\mu} P'}{P \mid Q \xrightarrow{\mu} P' \mid Q}\ \mathrm{bn}(\mu) \cap \mathrm{fn}(Q) = \emptyset$$

$$\frac{P \xrightarrow{(\nu\, \widetilde{b}')\, \overline{a}\langle\widetilde{b}\rangle} P' \quad Q \xrightarrow{a(\widetilde{c})} Q'}{P \mid Q \xrightarrow{\tau} (\nu\, \widetilde{b}')(P' \mid Q'[\widetilde{c} := \widetilde{b}])}\ \begin{array}{l}|\widetilde{b}| = |\widetilde{c}| \\ \widetilde{b}' \cap \mathrm{fn}(Q) = \emptyset\end{array} \quad \frac{P \xrightarrow{\mu} P'}{(\nu\, a)\, P \xrightarrow{\mu} (\nu\, a)\, P'}\ a \notin \mathrm{n}(\mu) \quad \frac{P \xrightarrow{(\nu\, \widetilde{b}')\, \overline{a}\langle\widetilde{b}\rangle} P'}{(\nu\, c)\, P \xrightarrow{(\nu\, c, \widetilde{b}')\, \overline{a}\langle\widetilde{b}\rangle} P'}\ \begin{array}{l}a \neq c \\ c \in \widetilde{b} \setminus \widetilde{b}'\end{array}$$

$$\frac{P \xrightarrow{\mu} P'}{[a = a]\, P \xrightarrow{\mu} P'} \quad \frac{P \xrightarrow{\mu} P'}{!\, P \xrightarrow{\mu} P' \mid !\, P}\ \mathrm{bn}(\mu) \cap \mathrm{fn}(P) = \emptyset \quad \frac{P \xrightarrow{(\nu\, \widetilde{c}')\, \overline{a}\langle\widetilde{b}\rangle} P_1' \quad P \xrightarrow{a(\widetilde{c})} P_2'}{!\, P \xrightarrow{\tau} ((\nu\, \widetilde{b}')(P_1' \mid P_2'[\widetilde{c} := \widetilde{b}])) \mid !\, P}\ \begin{array}{l}|\widetilde{b}| = |\widetilde{c}| \\ \widetilde{b}' \cap \mathrm{fn}(P) = \emptyset\end{array}$$

Table 1. Structural congruence and SOS clauses for late semantics for π-calculus.

and variables are distinct from each other and from all free names and variables, and we identify agents that can be alpha-converted into each other. Substitutions are extended to terms for variables, $P[\widetilde{x} := \widetilde{M}]$.

The operational semantics of the closed agents of our spi calculus are given by the structural axioms and rules listed in Table 2 (cf. also [3]).

This semantics is modelled on the one for π-calculus in Table 1, where transition labels, ranged over by μ, are $n(x)$, $(\nu\, m)\, \overline{n}\langle M\rangle$ and τ. The only essential new ingredient is that an auxiliary commitment relation is used, intuitively representing evaluation of terms.

3 The Translation

The main idea of the translation from the spi calculus to the π-calculus is to represent spi calculus terms as objects with a number of predefined methods. Encryption, e.g., corresponds to creating an object with a special method for decryption; the object will perform this only when the correct encryption key is presented.

The main complication turns out to lie in representing equality. In the spi calculus checking equality of terms is a primitive operation whereas the π-calculus only has equality of atomic names. Therefore each term needs a particular method to determine if another term is equal; such methods actually make up the bulk of the encoding.

In the translation $\mathcal{N}_{spi}$ and $\mathcal{V}_{spi}$ are both represented by $\mathcal{N}_\pi$. Spi-calculus agents and terms are translated to π-calculus agents where some names (called *reserved* names) are used to signal operations on the encodings of terms.

Let P be a spi-calculus agent containing a spi-calculus term M. In the translation $[\![P]\!]$ of P there will then occur, for some name ℓ restricted in $[\![P]\!]$, the translation $[\![M]\!]_\ell$ of M located at ℓ, meaning that other parts of $[\![P]\!]$ will be able to access the translation of M by using the link ℓ. Over this link a challenge-response protocol is used to perform operations on the term. Three names are passed to the term encoding in the challenge:

c representing the operation to perform, e.g., $\mathbf{id}$ to get the identity of an encoded name, or $\mathbf{match}$ to check for equality to another term.

m representing a parameter to the operation, e.g., the location of the term to compare with in a $\mathbf{match}$ operation;

r being the name of the channel to send a response or result on. The type of response depends on the operation, e.g., for $\mathbf{id}$ the name being encoded is passed over r, while for $\mathbf{match}$ no object is needed – the synchronization on r in itself means the terms did match.

The names of operations are reserved, i.e. we assume they do not appear anywhere in the source term or agent. They are described in Table 3. The names $\mathbf{private}$ and $\mathbf{public}$ are reserved for the types of asymmetric keys, and in addition, the reserved name $\mathbf{void}$ is used as the parameter m for operations which need no parameter, e.g., the $\mathbf{id}$ operation.

An agent (or term) operating on an encoded term typically generates a new name for the result channel, in order to avoid interference from concurrent accesses. For example, to check whether two terms located at ℓ and m are equal, the construction $(\nu\, r)\, \overline{\ell}\langle \mathbf{match}, m, r\rangle.r().P$ could be used, which will continue as P if the terms match.

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

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

$$[M \textbf{ is } M]\, P > P \qquad \textbf{case } \{M\}_N \textbf{ of } [x]_N \textbf{ in } P > P[x := M]$$

$$\textbf{let } (x, y) = (M, N) \textbf{ in } P > P[(x, y) := (M, N)] \qquad \textbf{case } \{\!|M|\!\}_N \textbf{ of } [x]_{N^{-1}} \textbf{ in } P > P[x := M]$$

$$\textbf{case } 0 \textbf{ of } 0 : P\ \mathrm{suc}(x) : Q > P \qquad N = K^+ \text{ or } N = K^- \text{ for some } K,$$

$$\textbf{case } \mathrm{suc}(M) \textbf{ of } 0 : P\ \mathrm{suc}(x) : Q > Q[x := M] \qquad \text{where } (K^-)^{-1} = K^+ \text{ and } (K^+)^{-1} = K^-$$

$$\frac{P \equiv Q \xrightarrow{\mu} Q \equiv P'}{P \xrightarrow{\mu} P'} \quad \frac{P > P' \quad P' \xrightarrow{\mu} P''}{P \xrightarrow{\mu} P''} \quad \begin{array}{l} m(x).P \xrightarrow{m(x)} P \\ \overline{m}\langle M\rangle.P \xrightarrow{\overline{m}\langle M\rangle} P \end{array} \quad \frac{P \xrightarrow{(\nu\, \widetilde{n})\, \overline{m}\langle M\rangle} P' \quad Q \xrightarrow{m(x)} Q'}{P \mid Q \xrightarrow{\tau} (\nu\, \widetilde{n})(P' \mid Q'[x := M])}\ \widetilde{n} \cap \mathrm{fn}(Q) = \emptyset$$

$$\frac{P \xrightarrow{\mu} P'}{P \mid Q \xrightarrow{\mu} P' \mid Q} \quad \frac{P \xrightarrow{\mu} P'}{(\nu\, m)\, P \xrightarrow{\mu} (\nu\, m)\, P'}\ m \notin \mathrm{n}(\mu) \quad \frac{P \xrightarrow{(\nu\, \widetilde{n})\, \overline{m}\langle M\rangle} P'}{(\nu\, k)\, P \xrightarrow{(\nu\, k, \widetilde{n})\, \overline{m}\langle M\rangle} P'}\ \begin{array}{l} m \neq k \\ k \notin \mathrm{fn}(M) \setminus n' \end{array}$$

$$\frac{P \xrightarrow{\mu} P'}{!\, P \xrightarrow{\mu} P' \mid !\, P}\ \mathrm{bn}(\mu) \cap \mathrm{fn}(P) = \emptyset \quad \frac{P \xrightarrow{(\nu\, \widetilde{n})\, \overline{m}\langle M\rangle} P_1' \quad P \xrightarrow{m(x)} P_2'}{!\, P \xrightarrow{\tau} ((\nu\, \widetilde{n})(P_1' \mid P_2'[x := M])) \mid !\, P}\ \widetilde{n} \cap \mathrm{fn}(Q) = \emptyset$$

Table 2. Structural congruence axioms, commitment axioms and SOS clauses for late semantics for the spi calculus.

As a shorthand for $(\nu\, r)\, \overline{\ell}\langle c, m, r\rangle.P$ we often write $\overline{\ell}\langle c, m, \nu\, r\rangle.P$.

The type-orthodox reader may notice that our translation does not give a proper typing for the response channel r, which is sometimes nullary and sometimes unary. Types are out of the scope of this paper, but we believe that introducing a dummy name as object in the nullary case would remedy the situation.

In the presentation of the translation, we use π-calculus abstractions of the form $(\lambda\, x)\, P$ (ranged over by F) and applications, defined by $((\lambda\, x)\, P)(m) = P[x := m]$; we write $(\lambda\, m, n)\, P$ for $(\lambda\, m)\, (\lambda\, n)\, P$ and $F(m, n)$ for $F(m)(n)$. The translation also involves a case analysis on the terms occurring in the outermost operator of spi agents, and subterms occurring in terms. Each term (or subterm) which is not a variable must be recursively translated. (Variables are represented directly by names, which get substituted by term locations in inputs.) We write $M \Rightarrow F$ and $M, N \Rightarrow F$ for the translated spi calculus term(s) M (and N) being used by the π-calculus abstraction F. This is formally defined as follows, where different cases are obtained depending on whether the parameter(s) of F be variable(s) or not.

$$M \Rightarrow F = \begin{cases} (\nu\, m)([\![M]\!]_m \mid F(m)) & \text{if } M \text{ is not a variable} \\ F(M) & \text{if } M \text{ is a variable} \end{cases}$$

$$M, N \Rightarrow F = \begin{cases} (\nu\, m)\, (\nu\, n)([\![M]\!]_m \mid [\![N]\!]_n \mid F(m, n)) & \text{if } M \text{ and } N \text{ are not variables} \\ (\nu\, m)([\![M]\!]_m \mid F(m, N)) & \text{if } N \text{ is a variable but not } M \\ (\nu\, n)(F(M, n) \mid [\![N]\!]_n) & \text{if } M \text{ is a variable but not } N \\ F(M, N) & \text{if both are variables} \end{cases}$$

This notation will be used both in the following section which describes the translation of spi-calculus agents, and the the next which describes the translations of the terms occurring in agents.

3.1 Translating Agents

The parallel composition, replication, and restriction operators involve no terms, and the translation is homomorphic: $[\![(\nu\, n)\, P]\!] = (\nu\, n)[\![P]\!]$, $[\![P \mid Q]\!] = [\![P]\!] \mid [\![Q]\!]$, and $[\![!\, P]\!] = ![\![P]\!]$.

Outputs and inputs use the original subject name as communication channel, but the communicated objects are *links* to encodings of terms.

$$\begin{aligned} [\![\overline{M}\langle N\rangle.P]\!] = M, N \Rightarrow (\lambda\, m, n)\, & \overline{m}\langle \mathbf{id}, \mathbf{void}, \nu\, r\rangle. && (1) \\ & r(i). && (2) \\ & \overline{i}\langle n\rangle. && (3) \\ & [\![P]\!] && (4) \end{aligned}$$

Name	Term	Description
match	*all*	synchronize on r if m matches this term
id	names	respond with the name n being encoded
zero	0	synchronize on r if 0 is the encoded term
pred	$\mathrm{suc}(n)$	respond with the location of n
text	hashcodes	respond with the location of the text hashed
fst	pairs	respond with the location of the first component
scd	pairs	– or second component
type	asymmetric keys	respond with the type of key, **private** or **public**
base	asymmetric keys	respond with the location of k in the key k^+ or k^-
k_match	asymmetric keys	match k against k' for k^+ and k'^-
decrypt	ciphertexts	if m matches the key used for encryption, respond with the location of plaintext
splain	symmetric ciphertexts	respond with location of plaintext
skey	symmetric ciphertexts	respond with location of key
pplain	asymmetric ciphertexts	respond with location of plaintext
pkey	asymmetric ciphertexts	respond with location of key

Table 3. Reserved names for operations on translated terms

Notes: On line 1, ask the output subject for its identity; line 2, wait for a reply (carrying the identity, i.e., the name encoded); line 3, output the link to the output object, and on line 4, continue.

For the input prefix, note that the bound variable x in the spi input is syntactically the same as the name x in the π-calculus input.

$$\begin{aligned}[\![M(x).P]\!] = M \Rightarrow (\lambda\, m)\, & \overline{m}\langle \mathbf{id}, \mathbf{void}, \nu\, r\rangle. \\ & r(i). \\ & i(x). \\ & [\![P]\!]\end{aligned}$$

The remaining operators are dealt with using the appropriate operations on the terms involved. Note that for the integer **case** operator, the two parallel agents in the encoding are mutually exclusive: a term encoding never responds both on **zero** and **pred** operations. Again, the spi variables x and y bound in **let** and **case** are syntactically the same as the names x and y in the π-calculus translations.

$$\begin{aligned}&[\![[M \textbf{ is } N]\, P]\!] = \\ &\qquad M, N \Rightarrow (\lambda\, m, n)\, \overline{m}\langle \mathbf{match}, n, \nu\, r\rangle. && (1)\\ &\qquad\qquad r(). && (2)\\ &\qquad\qquad [\![P]\!]\end{aligned}$$

Notes: On line 1, tell the first term object to match against the second, and on line 2, wait for a reply (indicating that they did match).

$$\begin{aligned}&[\![\textbf{let } (x, y) = M \textbf{ in } P]\!] = \\ &\qquad M \Rightarrow (\lambda\, m)\, \overline{m}\langle \mathbf{fst}, \mathbf{void}, \nu\, r\rangle.r(x). && (1)\\ &\qquad\qquad \overline{m}\langle \mathbf{scd}, \mathbf{void}, \nu\, r\rangle.r(y). && (2)\\ &\qquad\qquad [\![P]\!]\end{aligned}$$

Notes: On line 1, ask for the first sub-object and wait for a link in return, and on line 2, do the same for the second sub-object.

$$\begin{aligned}&[\![\textbf{case } M \textbf{ of } 0 : P \ \mathrm{suc}(x) : Q]\!] = \\ &\qquad M \Rightarrow (\lambda\, m)\, \big(\ \overline{m}\langle \mathbf{zero}, \mathbf{void}, \nu\, r\rangle.r(). && (1)\\ &\qquad\qquad [\![P]\!] \\ &\qquad\qquad \mid \\ &\qquad\qquad \overline{m}\langle \mathbf{pred}, \mathbf{void}, \nu\, r\rangle.r(x). \\ &\qquad\qquad [\![Q]\!] \\ &\qquad\quad \big)\end{aligned}$$

Notes: Ask the object if it is zero (line 1), and in parallel ask for a link to a predecessor object. Only one of the challenges will get a reply.

The decryption operator simply asks the encrypted term M to decrypt using the key N, binding x to the plaintext received:

$$\begin{aligned}&[\![\textbf{case } M \textbf{ of } [x]_N \textbf{ in } P]\!] = \\ &\qquad M, N \Rightarrow (\lambda\, m, n)\, \overline{m}\langle \mathbf{decrypt}, n, \nu\, r\rangle. \\ &\qquad\qquad r(x). \\ &\qquad\qquad [\![P]\!]\end{aligned}$$

3.2 Translating Terms

In spi calculus *terms* are the values sent and received. As mentioned earlier, their translation into π-calculus, $[\![M]\!]_\ell$, is parameterised by ℓ, which is a link for accessing the encoding of M. We continue to use the $M \Rightarrow F$ and $M, N \Rightarrow F$ notations to handle the case analysis on subterms.

Different types of terms handle different operations. If a term encoding is given an operation it does not handle, it will simply ignore the challenge and (possibly) let the "caller" deadlock. This corresponds to a type error in the source term/agent; again, typing is not in the scope of this paper. All term translations are replications, which handle concurrent accesses.

We start by giving the encodings of the two simplest terms: names n and the constant 0, which have no subterms.

$$
\begin{array}{lr}
[\![n]\!]_\ell = !\,\ell(c,m,r). & (1)\\
\quad\big([c=\mathbf{id}]\,\overline{r}\langle n\rangle & (2)\\
\quad\;+[c=\mathbf{match}] & (3)\\
\qquad\overline{m}\langle \mathbf{id},\mathbf{void},\nu\, s\rangle. & (4)\\
\qquad s(x). & (5)\\
\qquad [x=n]\,\overline{r}\langle\rangle & (6)\\
\quad\big) &
\end{array}
$$

Notes: On line 1, replicate in order to handle concurrent requests; on line 2, handle an identity request by returning your true name; on line 3, handle a match request by asking the other end for its identity (line 4), await a reply (line 5), and if it's our identity then reply (line 6).

The encoding of zero is similar but simpler:

$$
\begin{array}{l}
[\![0]\!]_\ell = !\,\ell(c,m,r).\\
\quad\big([c=\mathbf{zero}]\,\overline{r}\langle\rangle\\
\quad\;+[c=\mathbf{match}]\\
\qquad\overline{m}\langle \mathbf{zero},\mathbf{void},r\rangle\\
\quad\big)
\end{array}
$$

The successor, hash code and asymmetric key terms have a single subterm, and use $M \Rightarrow F$ notation.

$$
\begin{array}{l}
[\![\mathrm{suc}(M)]\!]_\ell =\\
\qquad M \Rightarrow (\lambda\, m)\,\mathbf{Unary}(\ell,\mathbf{pred},m)\\
[\![\mathrm{hashc}(M)]\!]_\ell =\\
\qquad M \Rightarrow (\lambda\, m)\,\mathbf{Unary}(\ell,\mathbf{text},m)\\
[\![M^+]\!]_\ell =\\
\qquad M \Rightarrow (\lambda\, m)\,\mathbf{PKey}(\ell,\mathbf{private},m,\mathbf{public})\\
[\![M^-]\!]_\ell =\\
\qquad M \Rightarrow (\lambda\, m)\,\mathbf{PKey}(\ell,\mathbf{public},m,\mathbf{private})
\end{array}
$$

The agent **Unary** handles the simplest unary terms, and is parameterised by the name of the subterm field; it handles requests for the subterm (line 1 below), and uses the agent **Unary_Match** which handles **match** operations for unary terms. To handle **match**, it asks the other object for its corresponding subobject (line 2 below) and tells the subobjects to match (line 3).

$$
\begin{array}{lr}
\mathbf{Unary}(\ell,sub,n) = & \\
\quad !\,\ell(c,m,r). & \\
\quad\big([c=sub]\,\overline{r}\langle n\rangle & (1)\\
\quad\;+[c=\mathbf{match}] & \\
\qquad\mathbf{Unary\_Match}(m,sub,n,r) & \\
\quad\big) &
\end{array}
$$

$$
\begin{array}{lr}
\mathbf{Unary\_Match}(m,sub,n,r) = & \\
\qquad\overline{m}\langle sub,\mathbf{void},\nu\, s\rangle. & (2)\\
\qquad s(x). & \\
\qquad\overline{n}\langle \mathbf{match},x,r\rangle & (3)
\end{array}
$$

Asymmetric keys are encoded using $\mathbf{PKey}$, where the parameters t and u give the type of the key and its complementary type, and n is the *key base*, i.e., the location of M for a key M^+ or M^-.

$$
\begin{array}{lr}
\mathbf{PKey}(\ell,t,n,u) = & \\
\quad !\,\ell(c,m,r). & \\
\quad\big([c=\mathbf{type}]\,\overline{r}\langle t\rangle & (1)\\
\quad\;+[c=\mathbf{base}]\,\overline{r}\langle n\rangle & (2)\\
\quad\;+[c=\mathbf{match}] & (3)\\
\qquad\overline{m}\langle \mathbf{type},\mathbf{void},\nu\, s\rangle. & (4)\\
\qquad s(x). & \\
\qquad [x=t] & (5)\\
\qquad\quad\mathbf{Unary\_Match}(m,\mathbf{base},n,r) & \\
\quad\;+[c=\mathbf{k\_match}] & (6)\\
\qquad\overline{m}\langle \mathbf{type},\mathbf{void},\nu\, s\rangle. & \\
\qquad s(x). & \\
\qquad [x=u] & (7)\\
\qquad\quad\mathbf{Unary\_Match}(m,\mathbf{base},n,r) & \\
\quad\big) &
\end{array}
$$

Notes: Lines 1 and 2 simply return the type and base of the key; on line 3 a match request is handled by asking the other object for its key type (line 4), checking that it's the same (line 5), and then matching the key bases. Line 6 handles a key match request similarly to a match, but now checking that the key types are complementary (line 7).

The remaining terms (pairs and encryptions) have two subterms, and thus use the $M, N \Rightarrow F$ notation.

$$
\begin{array}{lr}
[\![(M,N)]\!]_\ell = & \\
\quad M,N \Rightarrow & \\
\qquad(\lambda\, n_1,n_2)\,!\,\ell(c,m,r). & (1)\\
\qquad\quad\big([c=\mathbf{fst}]\,r\langle n_1\rangle & (2)\\
\qquad\quad\;+[c=\mathbf{scd}]\,r\langle n_2\rangle & (3)\\
\qquad\quad\;+[c=\mathbf{match}] & \\
\qquad\qquad\mathbf{Binary\_Match}(& \\
\qquad\qquad\quad m,\mathbf{fst},n_1,\mathbf{scd},n_2,r & \\
\qquad\qquad) & \\
\qquad\quad\big) &
\end{array}
$$

Notes: Lines 1 and 2 handle requests for subcomponents, while line 3 handles match requests using the **Binary_Match** agent (see below).

$$\llbracket\{M\}_N\rrbracket_\ell = \\ M, N \Rightarrow (\lambda\, m, n) \\ \qquad \mathbf{Cipher}(\ell, \mathbf{splain}, m, \mathbf{skey}, n, \mathbf{match})$$

$$\llbracket\{\!|M|\!\}_N\rrbracket_\ell = \\ M, N \Rightarrow \\ (\lambda\, m, n) \\ \mathbf{Cipher}(\ell, \mathbf{pplain}, m, \mathbf{pkey}, n, \mathbf{k\_match})$$

The **Binary_Match** agent is parameterised by the operations to access the subterms sub_1 and sub_2 of m, and simply "chains" a unary match for each.

$$\begin{aligned}&\mathbf{Binary\_Match}(m, sub_1, n_1, sub_2, n_2, r) = \\ &\qquad (\nu\, s)\,\big(\mathbf{Unary\_Match}(m, sub_1, n_1, s) \\ &\qquad\quad \mid s().\mathbf{Unary\_Match}(m, sub_2, n_2, r) \\ &\qquad\big)\end{aligned}$$

Ciphertexts are created using symmetric or asymmetric ciphers, and use different operations to access plaintext and to match encryption keys.

$$\begin{aligned}&\mathbf{Cipher}(\ell, plaintext, p, key, k, k\_match) = \\ &\qquad !\,\ell(c, m, r). \\ &\qquad \big([c = plaintext]\, r\langle p\rangle && (1)\\ &\qquad\ + [c = key]\, r\langle k\rangle && (2)\\ &\qquad\ + [c = \mathbf{match}] \\ &\qquad\qquad \mathbf{Binary\_Match}(&& (3)\\ &\qquad\qquad\quad m, plaintext, p, key, k, r \\ &\qquad\qquad) \\ &\qquad\ + [c = \mathbf{decrypt}] \\ &\qquad\qquad \overline{k}\langle k\_match, m, \nu\, s\rangle. && (4)\\ &\qquad\qquad s(). \\ &\qquad\qquad r\langle p\rangle && (5)\\ &\qquad\big)\end{aligned}$$

Notes: Lines 1 and 2 handle requests for plaintext and key (i.e. **splain/pplain** and **skey/pkey**); line 3 handles match by matching both plaintext and key; line 4 handles decrypt requests by first key-matching the supplied key against our own (using **match** or **k_match** for symmetric or asymmetric keys, respectively), and returning the plaintext if they match (line 5).

3.2.1 Unique-Plaintext Ciphertexts

Under the assumption that each encryption of a fixed plaintext generates a new ciphertext, which is the case e.g. if random padding of the plaintext is used, the encoding can be optimized slightly, resulting in smaller state spaces: at line 1 below, a match is successful only if matching against the same term.

$$\begin{aligned}&\mathbf{Cipher}(\ell, plaintext, p, key, k, k\_match) = \\ &\qquad !\,\ell(c, m, r). \\ &\qquad \big([c = plaintext]\, r\langle p\rangle \\ &\qquad\ + [c = key]\, r\langle k\rangle \\ &\qquad\ + [c = \mathbf{match}] \\ &\qquad\qquad [m = \ell]\, r\langle\rangle && (1)\\ &\qquad\ + [c = \mathbf{decrypt}] \\ &\qquad\qquad \overline{k}\langle k\_match, m, \nu\, s\rangle. \\ &\qquad\qquad s(). \\ &\qquad\qquad r\langle p\rangle \\ &\qquad\big)\end{aligned}$$

4 Preservation of Spi Calculus May-Testing

Testing in the sense of De Nicola and Hennessy [10] rests on the idea of building an observation scenario for some description framework for concurrent processes by employing the expressive power of that framework itself. This goal is achieved by employing agents as so-called experiments, setting them up in concurrent interaction with agents that are to be tested and letting those experiments emit some signal if and when they have reached any success state. There is a substantial literature about testing over the π-calculus, for example by Boreale and De Nicola [9], and Abadi and Gordon identified may-testing already in [2] as particularly suitable for the spi calculus since it is generally associated with safety properties, the ones that are obviously most interesting in connection with spi. We prove that our translation is adequate with respect to may-testing in the sense that a spi calculus agent P may pass a spi calculus experiment E if and only if its translation, $\llbracket P\rrbracket$, may pass E's translation, $\llbracket E\rrbracket$. We state next the formal definitions that are needed for our purposes and the adequacy theorem itself.

Definition 1

1. An *experiment* is a spi or π-calculus agent that may use a distinguished name \$. We call an action on \$ a *success signal*.

2. A spi or π-calculus agent P *may pass* an experiment E if some sequence of τ-steps of the composed agent $P \mid E$ has a state in which success is signalled. Formally, we denote this property by P $\textsc{may}$ E.

Theorem 2 *Let P and E be a spi calculus agent and experiment, respectively. Then P $\textsc{may}$ E if and only if $\llbracket P\rrbracket$ $\textsc{may}$ $\llbracket E\rrbracket$.*

We give next a largely informal overview of the long and complex proof of this result, relegating the details to the appendix. The result is essentially a consequence of several operational correspondence properties in whose proofs the actual work lies. By far the most difficult one of these properties is concerned with going from operational steps of any agent obtained by translation back to operational steps or commitments of its pre-image. The translation induces sequences of steps on the π-calculus side where there is only a commitment or a single step on the spi calculus side. Concurrent sequences of this kind can be interleaved and they can be composed of interleaved sub-sequences amongst which there is communication. Therefore we get a very complicated correspondence between the states of $[\![P]\!]$ and those of P, where P is any spi agent. The decisive observation, however, is that it is not imperative to use a direct operational correspondence. Instead we use an indirect one, where we rearrange the states of $[\![P]\!]$ along the way so that they stay structurally more similar to those of P. This strategy requires that the rearrangement preserves bisimulation and two crucial properties of the translation are needed for that: First, it is compositional with respect to all static operators, that is, parallel composition, restriction and replication; second, the translations of terms can be viewed as *resources* in the sense of the Replication Lemmas over the π-calculus (see, for example, [16]), and we get suitable agent rearrangements by applying these lemmas.

For the core of the proof of Theorem 2, we need to use operational semantics without structural congruence since arbitrary rearrangements due to it would complicate the compositional arguments as they would have to be distinguished from rearrangements of the kind explained in the previous paragraph. In consequence, we use bisimulation over both the π- and the spi calculus to relate our rearrangements to actual states of spi agents and their translations. To this end, we carry over the notion of context bisimulation from higher-order process algebra [15] to both calculi. That gives us a uniform framework to work with and, moreover, it totally avoids the complications involved in other bisimilarities proposed for the spi calculus (see [7] for a good recent overview). Our immediate purposes are best served by the definitions below; the definitions that we actually use in the proof are equivalent, but are presented somewhat differently to make the proof go smoother.

Definition 3 *Context bisimilarity* on spi calculus agents is defined to be the largest binary relation $\sim_{\rm cxt}$ so that $P \sim_{\rm cxt} Q$ implies:

i.
  i. Whenever $P \xrightarrow{(\nu\,\widetilde{n})\,\overline{m}\langle M\rangle} P'$, then $Q \xrightarrow{(\nu\,\widetilde{n})\,\overline{m}\langle M\rangle} Q'$ for some Q' s.t. $(\nu\,\widetilde{n})(R\{x := M\} \mid P') \sim_{\rm cxt} (\nu\,\widetilde{n})(R\{x := M\} \mid Q')$ for every spi agent expression R in which at most x occurs as a free variable.
  ii. Whenever $P \xrightarrow{m(x)} P'$, then $Q \xrightarrow{m(x)} Q'$ for some Q' s.t. $P'\{x := M\} \sim_{\rm cxt} Q'\{x := M\}$ for every closed term M.
  iii. Whenever $P \xrightarrow{\tau} P'$, then $Q \xrightarrow{\tau} Q'$ for some Q' s.t. $P' \sim_{\rm cxt} Q'$.

ii. Like i.i-i.iii but with Q driving the bisimulation game.

Definition 4 *Context bisimilarity* on π-calculus agents is defined to be the largest binary relation $\sim_{\rm cxt}$ so that $P \sim_{\rm cxt} Q$ implies:

i.
  i. Whenever $P \xrightarrow{(\nu\,\widetilde{b}')\,\overline{a}\langle\widetilde{b}\rangle} P'$, then $Q \xrightarrow{(\nu\,\widetilde{b}')\,\overline{a}\langle\widetilde{b}\rangle} Q'$ for some Q' s.t. $(\nu\,\widetilde{b}')(R\{\widetilde{c} := \widetilde{b}\} \mid P') \sim_{\rm cxt} (\nu\,\widetilde{b}')(R\{\widetilde{c} := \widetilde{b}\} \mid Q')$ for every π agent R, assuming that the names in $\widetilde{c}$ are distinct with $|\widetilde{b}| = |\widetilde{c}|$.
  ii. Whenever $P \xrightarrow{a(\widetilde{c})} P'$, then $Q \xrightarrow{a(\widetilde{c})} Q'$ for some Q' s.t. $P'\{\widetilde{c} := \widetilde{b}\} \sim_{\rm cxt} Q'\{\widetilde{c} := \widetilde{b}\}$ for every $\widetilde{b}$ with $|\widetilde{c}| = |\widetilde{b}|$.
  iii. Whenever $P \xrightarrow{\tau} P'$, then $Q \xrightarrow{\tau} Q'$ for some Q' s.t. $P' \sim_{\rm cxt} Q'$.

ii. Like i.i-i.iii but with Q driving the bisimulation game.

The operational correspondences together establish what could be regarded as a translation-coupled expansion between P and $[\![P]\!]$ for any spi agent P. Conventional expansion, that is, expansion where the translation is not directly built into it, has played a major role in work on translating the asynchronous π- to the πI-calculus, the π-calculus where only private names are mobile [6]. But we need to formulate our expansion also on the basis of what we call the *ancestor relation* (cf. [5]). The ancestor relation seems to be absolutely necessary for us to handle concurrent threads of activity on the π-calculus side were each such thread corresponds to just a single step or commitment on the spi calculus side. We denote it by $\blacktriangleleft$ and some schematic clauses for it are collected in Table 4, where $\Rightarrow$ stands for zero or more τ-labelled transitions, $\stackrel{\mu}{\Rightarrow}$ for zero or more τ-labelled transitions followed by a transition labelled with μ. An essential property of the ancestor relation is that its defining clauses are compositional on the static operators, since that allows us to reason compositionally. The other essential aspect is conveyed by those clauses that are of the form

$$\frac{[\![P]\!] \Rightarrow Q >_{\varphi} \stackrel{\mu}{\Rightarrow}\sim_{\rm cxt} [\![P']\!]}{P \blacktriangleleft Q},$$

where the outermost operator of P is a match, **let**, **case** or prefix. They are to be read as follows: It holds that $P \blacktriangleleft Q$ if

i. Q is equal to $[\![P]\!]$ or an intermediate state in the π-calculus execution trace of P's outermost operator and

ii. if φ holds, then $Q \stackrel{\mu}{\Rightarrow}\sim_{\rm cxt} [\![P']\!]$.

The intuition is that $P \blacktriangleleft Q$ holds if Q is a successor state of $[\![P]\!]$ in which no thread of activity that corresponds to an unguarded occurrence of a match, $\mathbf{let}$, $\mathbf{case}$ or prefix construct in P has been completed.

The central and most difficult part of the proof of Theorem 2 then consists of establishing a "backward" operational correspondence that goes from steps of $[\![P]\!]$ to steps and commitments of P, where P is any spi agent. This operational correspondence is coupled via the ancestor relation and involves context bisimilarity as for term rearrangements. There is also a "forward" operational correspondence that goes from steps P to steps of $[\![P]\!]$, and there are auxiliary operational correspondences that bridge the gap between operational semantics with and without structural congruence.

As a final remark, we note that, while context bisimilarity is new to the spi calculus, it has recently found proof technical application also to Cardelli and Gordon's mobile ambients [12].

5 Related and Future Work

Our translation from the spi to the π-calculus is obviously related to the general question of how to express encryption and other security concepts directly in the π-calculus, foregoing the spi calculus or any other higher-level framework. In the appendix to [2] Abadi and Gordon discuss three different approaches, where these schemes might partly be considered embryonic versions of what we get via our translation on the π-calculus side. In particular the technique of representing a piece of data by an agent that is to be accessed via a dedicated link, which goes back to the early π-calculus literature, is already proposed as useful in [3]. There is, however, no explicit translation from the spi to the π-calculus in [3] and therefore of course also no adequacy result along the lines of Theorem 2. Moreover, only a very limited range of features is discussed with respect to their expressibility in the π-calculus whereas we effectively deal with everything that belongs to the spi calculus as it was originally presented. It can nevertheless already be concluded from [3] that in the way we are using the π-calculus as a kind of assembler language for implementing the spi calculus, we can probably not hope to obtain full abstraction: There will be π-calculus experiments that distinguish the respective translations of any two agents even if they are testing equivalent with respect to spi calculus experiments.

Another idea discussed in [3], and also by Carbone and Maffeis in [8], is to model the communication of encrypted data on the basis of extending the π-calculus by multi-name synchronisation, that is, a channel may consist of what may perhaps be seen as several sub-channels that all take part in any synchronisation on the channel at once. This approach avoids at least to some degree the above-mentioned problem with stronger forms of adequacy than the preservation of may- and must-testing. The downside is of course that the target framework is less fundamental than the original π-calculus and also that there do not seem to be any automated π-calculus verification tools that support multi-name synchronisation in the input.

We have implemented our translation in a prototype that generates input for the Mobility Workbench [17]. With regard to this implementation we note that a π-calculus encoding will be finite-state if the spi calculus agent is finite-control. The reason is roughly that (a) only the term parts are never finite-control and (b) any thread of activity within them is triggered by activity within some agent part and then guaranteed to terminate, where that agent part can be infinite-control only if its pre-image is already such on the spi calculus side. Our prototype can therefore be used to translate both an agent and an experiment, whereupon the Mobility Workbench can check whether the agent may pass the experiment or not. As for work related to that, we are aware of work based on [7] that aims at tools for symbolic bisimulation for the spi calculus. Also, the spi calculus has been considered as an extension of a logic programming implementation of the π-calculus in [14].

As for possible future work, we want to carry our further experiments and case studies with our prototype, and enhance it so that we can make it publicly available. We also want to extend the prototype so as to produce output for backends other than the Mobility Workbench, such as for example for the MIHDA toolkit [11].

Acknowledgement We would like to thank Emilio Tuosto for discussions about an earlier version of the translation presented herein.

References

[1] M. Abadi and A. Gordon. The Spi Calculus. In *Computer and Communications Security*, pages 36–47. ACM, 1997. Conference proceedings.

[2] M. Abadi and A. Gordon. A Calculus for Cryptographic Protocols: The Spi Calculus. Technical Report 149, SRC, Palo Alto, California, 1998.

[3] M. Abadi and A. Gordon. A Calculus for Cryptographic Protocols: The Spi Calculus. *Information and Computation*, 148(1):1–70, 1999.

[4] R. Amadio and M. Dam. Reasoning about Higher-Order Processes. In *Theory and Practice of Software Development*, LNCS 915, pages 202–216. Springer-Verlag, 1995. TAPSOFT '95 conference proceedings.

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

$$\frac{[\![[M\ \mathbf{is}\ N]\,P]\!] \Rightarrow Q >_{M=N} \stackrel{\tau}{\Rightarrow}\sim_{\rm cxt} [\![P]\!]}{[M\ \mathbf{is}\ N]\,P \blacktriangleleft Q} \quad (\blacktriangleleft\text{-MATCH})$$

$$\frac{[\![\mathbf{let}\ (x_1,x_2)=M\ \mathbf{in}\ P]\!] \Rightarrow Q >_{M=(M_1,M_2)} \stackrel{\tau}{\Rightarrow}\sim_{\rm cxt} [\![P[(x_1,x_2):=(M_1,M_2)]]\!]}{\mathbf{let}\ (x_1,x_2)=M\ \mathbf{in}\ P \blacktriangleleft Q} \quad (\blacktriangleleft\text{-PAIR})$$

$$\frac{[\![\mathbf{case}\ 0\ \mathbf{of}\ 0:P_1\ \mathrm{suc}(x):P_2]\!] \Rightarrow Q >_{\rm true} \stackrel{\tau}{\Rightarrow}\sim_{\rm cxt} [\![P_1]\!]}{\mathbf{case}\ 0\ \mathbf{of}\ 0:P_1\ \mathrm{suc}(x):P_2 \blacktriangleleft Q} \quad (\blacktriangleleft\text{-ZERO})$$

$$\frac{[\![\mathbf{case}\ M\ \mathbf{of}\ 0:P_2\ \mathrm{suc}(x):P_2]\!] \Rightarrow Q >_{M=\mathrm{suc}(M_1)} \stackrel{\tau}{\Rightarrow}\sim_{\rm cxt} [\![P_2[x:=M_1]]\!]}{\mathbf{case}\ M\ \mathbf{of}\ 0:P_1\ \mathrm{suc}(x):P_2 \blacktriangleleft Q} \quad (\blacktriangleleft\text{-SUC})$$

$$\frac{[\![\mathbf{case}\ M\ \mathbf{of}\ [x]_K\ \mathbf{in}\ P]\!] \Rightarrow Q >_{\varphi} \stackrel{\tau}{\Rightarrow}\sim_{\rm cxt} [\![P[x:=M_1]]\!]}{\mathbf{case}\ M\ \mathbf{of}\ [x]_K\ \mathbf{in}\ P \blacktriangleleft Q} \quad (\blacktriangleleft\text{-DEC})$$

φ in the above rule stating that (a) $M=\{M_1\}_K$ or (b) $M=\{\!|M_1|\!\}_{N^p}$ and $K=(N^p)^{-1}$ for some N, $p=-,+$

$$\mathbf{0} \blacktriangleleft \mathbf{0} \qquad \frac{P_1 \blacktriangleleft Q_1 \quad P_2 \blacktriangleleft Q_2}{P_1 \mid P_2 \blacktriangleleft Q_1 \mid Q_2} \qquad \frac{P \blacktriangleleft Q}{(\nu\, m)\, P \blacktriangleleft (\nu\, m)\, Q} \qquad !\,P \blacktriangleleft\, !\,[\![P]\!]$$

($\blacktriangleleft$-NIL, $\blacktriangleleft$-PAR, $\blacktriangleleft$-NEW, $\blacktriangleleft$-REP)

Table 4. Selected schematic clauses for the ancestor relation on spi and π-terms.

[6] M. Boreale. On the Expressiveness of Internal Mobility in Name-Passing Calculi. *Theoretical Computer Science*, 195:205–226, 1998.

[7] J. Borgström and U. Nestmann. On Bisimulations for the Spi Calculus. Technical Report IC/2003/34, EPFL I&C, Lausanne, Switzerland, 2003.

[8] M. Carbone and S. Maffeis. On the Expressive Power of Polyadic Synchronisation in π-Calculus. *Nordic Journal of Computing*, 10(2):70–98, 2003.

[9] R. De Nicola and M. Boreale. Testing Equivalences for Mobile Processes. *Information and Computation*, 120:279–303, 1995.

[10] R. De Nicola and M. Hennessy. Testing Equivalences for Processes. *Theoretical Computer Science*, 34:83–133, 1983.

[11] G. Ferrari, U. Montanari, R. Raggi, and E. Tuosto. From Co-Algebraic Specifications to Implementation: The MIHDA Toolkit. In *Formal Methods for Components and Objects*, LNCS. Springer-Verlag, 2003. FMCO '03 symposium proceedings.

[12] M. Merro and F. Zappa Nardelli. Bisimulation Proof Methods for Mobile Ambients. In *Automata, Logic and Programming*, LNCS 2719, pages 584–598. Springer-Verlag, 2003. ICALP '03 proceedings.

[13] R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes, Parts I+II. *Information and Computation*, 100:1–77, 1992.

[14] Ping Yang, C. Ramakrishnan, and S. Smolka. A Logical Encoding of the π-Calculus: Model Checking Mobile Processes Using Tabled Resolution. Available via http://www.cs.sunysb.edu/∼lmc/mmc, 2003.

[15] D. Sangiorgi. Bisimulation in Higher-Order Calculi. *Information and Computation*, 131:141–178, 1996.

[16] D. Sangiorgi and D. Walker. *The π-Calculus: A Theory of Mobile Processes*. Cambridge University Press, 2003.

[17] B. Victor and F. Moller. The Mobility Workbench – A Tool for the π-Calculus. In *Computer Aided Verification*, LNCS 818, pages 428–440. Springer-Verlag, 1994. CAV '94 proceedings.

A Appendix

Theorem 2 is a direct consequence of Lemmas 17, 19, 21 and 23 below, which are about the preservation of may-testing with respect to different operational semantics for the spi and the π-calculus. By far the most difficult one is Lemma 17, as it is the one that directly connects may-testing and the translation. The other ones are much easier since they are not about the translation, but just about individually connecting the testing scenarios used in Lemma 17 to those that arise for the two calculi from the operational semantics laid out in Section 2.

All of Lemmas 17, 19, 21 and 23 are easy consequences of operational correspondence properties in whose proofs the actual work lies. It is in proving the operational correspondences required for Lemma 17 where we employ context bisimilarity. This part of the proof of Theorem 2 is the subject of Subsection A.1, which is titled:

A.1 Preservation of May-Testing in Continuation Passing Style

since we use continuation-passing style operational semantics in it, a technique that has proved advantageous in reasoning about context bisimilarity. Subsection A.2 is about the preservation of may-testing if one goes back from continuation-passing-style to conventional operational semantics and everything is just very briefly put together in Subsection A.3.

A.1.1 Continuation-Passing-Style Operational Semantics

The first occurrence of continuation-passing-style operational semantics for higher-order process calculi was probably in [4]. The idea is that the receiver in a communication is transferred as an abstraction or, to put it differently, as a continuation to the sender, where this abstraction/continuation is applied to the entity to be sent, be that entity a process, a name (tuple), a spi calculus term or whatever. Besides being very well suited for reasoning about context is bisimulation, the advantage is that such an operational semantics requires simpler labels and fewer side conditions on the transitions.

The π-Calculus Case The first thing to put in place for a continuation-passing-style operational semantics for the π-calculus consists of introducing *higher-order π-calculus agent expressions*. These expressions are ranged over by F and the syntactic clause for them is $F ::= \mathrm{X} \mid (\lambda\,\widetilde{a})\,P$, where X is a fixed place holder for abstractions of the form $(\lambda\,\widetilde{a})\,P$. The syntax for ordinary π-agent expressions is extended by the clause $P ::= F\widetilde{a}$. A *higher-order substitution*, $P[\mathrm{X} :=_{\mathrm{ho}} (\lambda\,\widetilde{a})\,Q]$, denotes the result of first substituting $(\lambda\,\widetilde{a})\,Q$ for X in P, obtaining a (transient) expression P', and then replacing every sub-expression in P' that is of the form $((\lambda\,\widetilde{a})\,Q)\widetilde{b}$ by $Q[\widetilde{a} := \widetilde{b}]$ provided that $|\widetilde{a}| = |\widetilde{b}|$. Higher-order substitution is undefined whenever any sub-expression of the form $((\lambda\,\widetilde{a})\,Q)\widetilde{b}$ occurs where $|\widetilde{a}| \neq |\widetilde{b}|$. An *agent* is an expression in which no sub-expression of the form $F\widetilde{a}$ occurs. Transition labels are of the form $\mu ::= a(\widetilde{b}) \mid \overline{a}\langle \mathrm{X}\colon k\rangle \mid \tau$, where $k \in \omega$, $\overline{a}\langle \mathrm{X}\rangle$ abbreviating $\overline{a}\langle \mathrm{X}\colon 1\rangle$ The SOS is defined on agents in such a way that never any undefined higher-order substitution occurs. See Table 5.

The Spi Calculus Case The continuation-passing-style operational semantics for the spi calculus is essentially a re-run of the previous subsection: *Higher-order spi calculus agent expressions* are ranged over by F and the syntactic clause for them is $F ::= \mathrm{X} \mid (\lambda\,x)\,P$, where X is a fixed place holder for abstractions of the form $(\lambda\,x)\,P$. The syntax for (ordinary) spi expressions is extended by the clause $P ::= FM$. A *higher-order substitution*, $P[\mathrm{X} :=_{\mathrm{ho}} (\lambda\,x)\,Q]$, denotes the result of first substituting $(\lambda\,x)\,Q$ for X in P, obtaining a (transient) term P', and then replacing every sub-term in P' that is of the form $((\lambda\,x)\,Q)M$ by $Q[x := M]$. An *agent* is a closed term in which no sub-term of the form FM occurs. Transition labels are of the form $\mu ::= m(x) \mid \overline{m}\langle \mathrm{X}\rangle \mid \tau$. The SOS clauses are given in Table 6. It might be noted that this SOS does not use a commitment relation. That is to ease the task of obtaining an operational correspondence between spi agents and their translations. To relate may-testing under the SOS from Table 6 to may-testing with commitments, we use yet another SOS for the spi calculus (cf. Subsection A.1.5).

A.1.2 A Central Tool: Context Bisimulation in Continuation Passing Style

We continue with a technical definition that helps to make the definition of context bisimilarity over the spi and the π-calculus more concise. This definition is about extending binary relations on agents to terms.

Definition 5

1. Let $\mathcal{R}$ be a binary relation on π-calculus agents. Then we denote by $P\ \mathcal{R}^{\widetilde{x}}\ Q$ the property that $P[\widetilde{x} := \widetilde{m}]\ \mathcal{R}\ Q[\widetilde{x} := \widetilde{m}]$ whenever $|\widetilde{x}| = |\widetilde{m}|$. Also, we denote by $P\ \mathcal{R}^{\mathrm{X}:k}\ Q$ the property that $P[\mathrm{X} :=_{\mathrm{ho}} (\lambda\,\widetilde{x})\,R]\ \mathcal{R}_{\mathrm{cxt}}\ Q[\mathrm{X} :=_{\mathrm{ho}} (\lambda\,\widetilde{x})\,R]$ for every abstraction $(\lambda\,\widetilde{x})\,R$ where $|\widetilde{x}| = k$. We usually abbreviate $P\ \mathcal{R}^{\mathrm{X}:1}\ Q$ by $P\ \mathcal{R}^{\mathrm{X}}\ Q$, we have $\mathrm{obj}(\overline{n}\langle \mathrm{X}{:}k\rangle) = \mathrm{X}{:}k$, $\mathrm{obj}(n(x)) = x$ and $\mathrm{obj}(\tau) = \mathrm{VOID}$, $\mathcal{R}^{\mathrm{VOID}} = \mathcal{R}$, and we usually write $\mathcal{R}^{\mu}$ for $\mathcal{R}^{\mathrm{obj}(\mu)}$.

$$a(\widetilde{b}).P \xrightarrow{a(\widetilde{b})}_{\mathrm{cp}} P \qquad \frac{P \xrightarrow{\mu}_{\mathrm{cp}} P'}{P + Q \xrightarrow{\mu}_{\mathrm{cp}} P'} \qquad \frac{P \xrightarrow{\mu}_{\mathrm{cp}} P'}{P \mid Q \xrightarrow{\mu}_{\mathrm{cp}} P' \mid Q}\ \mathrm{bn}(\mu) \cap \mathrm{fn}(Q) = \emptyset$$

$$\overline{a}\langle\widetilde{b}\rangle.P \xrightarrow{\overline{a}\langle \mathrm{X}:|\widetilde{b}|\rangle}_{\mathrm{cp}} \mathrm{X}\widetilde{b} \mid P$$

$$\frac{P \xrightarrow{\overline{a}\langle \mathrm{X}:k\rangle}_{\mathrm{cp}} P' \quad Q \xrightarrow{a(\widetilde{b})}_{\mathrm{cp}} Q'}{P \mid Q \xrightarrow{\tau}_{\mathrm{cp}} P'[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, \widetilde{b})\, Q']}\ k = |\widetilde{b}\,| \qquad \frac{P \xrightarrow{\mu}_{\mathrm{cp}} P'}{(\nu\, a)\, P \xrightarrow{\mu}_{\mathrm{cp}} (\nu\, a)\, P'}\ a \notin \mathrm{n}(\mu) \qquad \frac{P \xrightarrow{\mu}_{\mathrm{cp}} P'}{[a = a]\, P \xrightarrow{\mu}_{\mathrm{cp}} P'}$$

$$\frac{P \xrightarrow{\mu}_{\mathrm{cp}} P'}{!\, P \xrightarrow{\mu}_{\mathrm{cp}} P' \mid !\, P}\ \mathrm{bn}(\mu) \cap \mathrm{fn}(P) = \emptyset \qquad \frac{P \xrightarrow{\overline{a}\langle \mathrm{X}:k\rangle}_{\mathrm{cp}} P_1' \quad P \xrightarrow{a(\widetilde{b})}_{\mathrm{cp}} P_2'}{!\, P \xrightarrow{\tau}_{\mathrm{cp}} P_1'[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, \widetilde{b})\, P_2'] \mid !\, P}\ k = |\widetilde{b}|$$

Table 5. SOS clauses for continuation-passing-style operational semantics for the polyadic π-calculus, where only one symmetric clause is respectively stated for nondeterministic choice, interleaving and communication.

$$[M \ \mathbf{is}\ M]\, P \xrightarrow{\tau}_{\mathrm{cp}} P \quad \mathbf{let}\ (x_1, x_2) = (M_1, M_2)\ \mathbf{in}\ P \xrightarrow{\tau}_{\mathrm{cp}} P[(x_1, x_2) := (M_1, M_2)]$$

$$\mathbf{case}\ 0\ \mathbf{of}\ 0 : P\ \mathrm{suc}(x) : Q \xrightarrow{\tau}_{\mathrm{cp}} P \quad \mathbf{case}\ \mathrm{suc}(M)\ \mathbf{of}\ 0 : P\ \mathrm{suc}(x) : Q \xrightarrow{\tau}_{\mathrm{cp}} Q[x := M]$$

$$\mathbf{case}\ \{M\}_K\ \mathbf{of}\ [x]_K\ \mathbf{in}\ P \xrightarrow{\tau}_{\mathrm{cp}} P[x := M]$$

$$\mathbf{case}\ \{\!|M|\!\}_{N^p}\ \mathbf{of}\ [x]_{(N^p)^{-1}}\ \mathbf{in}\ P \xrightarrow{\tau}_{\mathrm{cp}} P[x := M]\ \ p = -, +$$

$$m(x).P \xrightarrow{m(x)}_{\mathrm{cp}} P \qquad \frac{P \xrightarrow{\mu}_{\mathrm{cp}} P'}{P \mid Q \xrightarrow{\mu}_{\mathrm{cp}} P' \mid Q} \qquad \frac{P \xrightarrow{\overline{m}\langle \mathrm{X}\rangle}_{\mathrm{cp}} P' \quad Q \xrightarrow{m(x)}_{\mathrm{cp}} Q'}{P \mid Q \xrightarrow{\tau}_{\mathrm{cp}} P'[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, Q']}$$

$$\overline{m}\langle M\rangle.P \xrightarrow{\overline{m}\langle \mathrm{X}\rangle}_{\mathrm{cp}} \mathrm{X}M \mid P$$

$$\frac{P \xrightarrow{\mu}_{\mathrm{cp}} P'}{(\nu\, m)\, P \xrightarrow{\mu}_{\mathrm{cp}} (\nu\, m)\, P'}\ m \notin \mathrm{n}(\mu) \qquad \frac{P \xrightarrow{\mu}_{\mathrm{cp}} P'}{!\, P \xrightarrow{\mu}_{\mathrm{cp}} P' \mid !\, P} \qquad \frac{P \xrightarrow{\overline{m}\langle \mathrm{X}\rangle}_{\mathrm{cp}} P_1' \quad P \xrightarrow{m(x)}_{\mathrm{cp}} P_2'}{!\, P \xrightarrow{\tau}_{\mathrm{cp}} P_1'[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, P_2'] \mid !\, P}$$

Table 6. SOS clauses for continuation-passing-style operational semantics for the spi calculus, where only one symmetric clause is respectively stated for interleaving and communication.

2. Let $\mathcal{R}$ be a binary relation on closed spi calculus agents. Then we denote by $P\ \mathcal{R}^x\ Q$ the property that $P[x := M]\ \mathcal{R}\ Q[x := M]$ for every closed term M. Also, we denote by $P\ \mathcal{R}^{\mathrm{X}}\ Q$ the property that $P[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, R]\ \mathcal{R}_{\mathrm{cxt}}\ Q[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, R]$ for every abstraction $(\lambda\, x)\, R$. Moreover, similarly to the above, we have $\mathrm{obj}(\overline{n}\langle \mathrm{X}\rangle) = \mathrm{X}$, $\mathrm{obj}(n(x)) = x$ and $\mathrm{obj}(\tau) = \textsc{void}$, $\mathcal{R}^{\textsc{void}} = \mathcal{R}$, and we usually write $\mathcal{R}^\mu$ for $\mathcal{R}^{\mathrm{obj}(\mu)}$.

We have now set up things so that we can give a simultaneous definition of context bisimilarity for both the spi and the π-calculus:

Definition 6 *Strong context bisimilarity* is defined to be the largest binary relation $\sim_{\mathrm{cxt}}$ on spi or π-calculus agents so that $P \sim_{\mathrm{cxt}} Q$ implies:

i. Whenever $P \xrightarrow{\mu}_{\mathrm{cp}} P'$, then $Q \xrightarrow{\mu}_{\mathrm{cp}} Q'$ for some Q' so that $P' \sim^{\mu}_{\mathrm{cxt}} Q'$.

ii. Whenever $Q \xrightarrow{\mu}_{\mathrm{cp}} Q'$, then $P \xrightarrow{\mu}_{\mathrm{cp}} P'$ for some P' so that $P' \sim^{\mu}_{\mathrm{cxt}} Q'$.

Remarks:

1. We mostly drop qualifications as "strong" for the rest of this paper, since our present purposes do not require us to consider any weak form of bisimulation.

2. Context bisimilarity on π-agents evidently coincides on input and silent transitions with the familiar notion of late π-calculus bisimilarity. It is the treatment of output transitions where the difference lies: Instead of observing the subject, the object and any extruded names, one observes only the subjects and compares the residuals with respect to all receiving contexts. The universal quantification involved in that renders context bisimilarity not very meaningful as an independent, stand-alone notion. Not being able to observe objects and extruded names in output transitions – at least

not directly – may in many situations be another drawback. As far as our purposes herein are concerned, it is not a problem since for proving the required properties about testing it evidently suffices.

3. As can easily be seen, context bisimilarity is an equivalence.

4. As a side remark, we conjecture that barbed equivalence under the SOS from Table 6 can be characterised as context bisimilarity if one does not use the late version but the early one. The proof is very much along a proof in [16], where barbed equivalence over the π-calculus is characterised in terms of early bisimilarity: One stratifies early context bisimilarity and then one carries out an inductive argument.

The following two lemmas are concerned with some standard properties of context bisimilarity that are needed in the sequel. Analogously to Definition 6, they pertain to both the π- and the spi calculus, including their proofs, which also follow standard lines. That might be taken as supporting evidence for the versatility of context bisimilarity as an auxiliary tool when reasoning about the spi – and also the π- – calculus.

Lemma 7 *Context bisimilarity is preserved by parallel composition and restriction.*

Proof. Let $\mathcal{C}$ be the closure of $\sim_{\mathrm{cxt}}$ under parallel composition and restriction, that is, $\mathcal{C}$ is the smallest binary relation on spi or π-calculus agents so that $\sim_{\mathrm{cxt}} \subseteq \mathcal{C}$, $P_1 \mid P_2 \;\mathcal{C}\; Q_1 \mid Q_2$ whenever $P_i \;\mathcal{C}\; Q_i$, $i = 1, 2$, and $(\nu\, m)\, P \;\mathcal{C}\; (\nu\, m)\, Q$ whenever $P \;\mathcal{C}\; Q$. The first step then consists of showing that $P \;\mathcal{C}\; Q$ implies:

i. Whenever $P \xrightarrow{\mu}_{\mathrm{cp}} P'$, then $Q \xrightarrow{\mu}_{\mathrm{cp}} Q'$ for some Q' so that $P' \;\mathcal{C}^{*\mu}\; Q'$.

ii. Whenever $Q \xrightarrow{\mu}_{\mathrm{cp}} Q'$, then $P \xrightarrow{\mu}_{\mathrm{cp}} P'$ for some P' so that $P' \;\mathcal{C}^{*\mu}\; Q'$.

This step is essentially a straightforward matter of induction on the inference used to establish $P \;\mathcal{C}\; Q$.

The second step consists of showing that $\mathcal{C}^*$ is a context bisimulation, which is straightforward by exploiting Step 1. We may then conclude that $\mathcal{C} \subseteq \sim_{\mathrm{cxt}}$, so $\sim_{\mathrm{cxt}} = \mathcal{C}$ as $\sim_{\mathrm{cxt}} \subseteq \mathcal{C}$ due to the construction of $\mathcal{C}$, so $\sim_{\mathrm{cxt}}$ is preserved by parallel composition and restriction due to the way $\mathcal{C}$ has been constructed. $\square$

We claim that context bisimilarity is in fact not only preserved by parallel composition and restriction but by all operators except input, so that it is a *non-input congruence*. We do not prove this property since it is not required for proving Theorem 2.

Lemma 8 *(Some Basic Laws)*

1. $P \mid Q \sim_{\mathrm{cxt}} Q \mid P$
2. $P \mid (Q \mid R) \sim_{\mathrm{cxt}} (P \mid Q) \mid R$
3. $(\nu\, m)\, (\nu\, n)\, P \sim_{\mathrm{cxt}} (\nu\, n)\, (\nu\, m)\, P$
4. $(\nu\, m)(P \mid Q) \sim_{\mathrm{cxt}} P \mid (\nu\, m)\, Q$ *if* $m \notin \mathrm{fn}(P)$
5. $P \mid !\, P \sim_{\mathrm{cxt}} !\, P$
6. $(\nu\, \mathrm{subj}(\alpha))(P \mid \alpha.Q) \sim_{\mathrm{cxt}} P$ *if* $\mathrm{subj}(\alpha) \notin \mathrm{fn}(P)$

Proof. Properties (1)-(5) can be proved simultaneously and in a way similar to the proof of Lemma 7. More specifically, let $\mathcal{C}$ be the smallest reflexive and symmetric relation on spi or π-calculus agents that satisfies (1)-(5) where $\sim_{\mathrm{cxt}}$ is replaced by $\mathcal{C}$, and that is closed under parallel composition and restriction. The next two steps are then exactly analogous to the two main steps of the proof of Lemma 7, and we may then conclude that $\mathcal{C} \subseteq \sim_{\mathrm{cxt}}$, so that we have established the validity of (1)-(5) due to the way $\mathcal{C}$ has been constructed. $\square$

Next we state the Replication Lemmas for context bisimilarity over the π-calculus – they will be of crucial importance in rearranging the states of any translated spi agent so that we will be able to obtain operational correspondences by compositional reasoning. As a preliminary, we denote by $P\{a = (\widetilde{b})\, Q\}$ the agent $(\nu\, a)(P \mid !\, a(\widetilde{b}).Q)$ whenever a occurs in both P and Q only in output subject position, where this notation binds stronger than any syntactic operator. The agent $!\, a(\widetilde{b}).Q$ is commonly called a *replicated resource*, a notion that obviously fits our term translations, including the side condition on the occurrences of a. This fit is why we can use the Replication Lemmas to such a great extent, since term translations are the main source of potentially concurrent threads of activity on the π-side that correspond to single steps or commitments on the spi side.

Lemma 9 *(Replication Lemmas – cf. [16])*

1. $(\alpha.P)\{a = (\widetilde{b})\, Q\} \sim_{\mathrm{cxt}} \alpha.P\{a = (\widetilde{b})\, Q\}$ *if* $\mathrm{subj}(\alpha) \neq a$
2. $(\overline{a}\langle\widetilde{c}\rangle.P)\{a = (\widetilde{b})\, Q\} \sim_{\mathrm{cxt}} (\overline{a}\langle\widetilde{c}\rangle.(P\{a = (\widetilde{b})\, Q\}))\{a = (\widetilde{b})\, Q\}$
3. $(P_1 \mid P_2)\{a = (\widetilde{b})\, Q\} \sim_{\mathrm{cxt}} P_1\{a = (\widetilde{b})\, Q\} \mid P_2\{a = (\widetilde{b})\, Q\}$
4. $((\nu\, n)\, P)\{a = (\widetilde{b})\, Q\} \sim_{\mathrm{cxt}} (\nu\, n)\, P\{a = (\widetilde{b})\, Q\}$
5. $(!\, P)\{a = (\widetilde{b})\, Q\} \sim_{\mathrm{cxt}} !\, P\{a = (\widetilde{b})\, Q\}$
6. $P\{a = (\widetilde{b})\, Q\} \sim_{\mathrm{cxt}} P$ *if* $a \notin \mathrm{fn}(P)$

Proof. The Replication Lemmas are stated in [16] for replicated resources with one parameter and strong early congruence. The proof in [16] carries over without any problem to replicated resources with an arbitrary number of parameters. Moreover, strong early congruence implies strong late bisimilarity and strong late bisimilarity, in turn, can readily be seen to be a strong context bisimulation chiefly because it is a non-input congruence. □

Our first application of the Replication Lemmas consists of a lemma about how translation and substitution commute. As a preliminary, we extend the translation to agent expressions containing sub-expressions of the form $\mathrm{X}M$: $[\![\mathrm{X}M]\!] = (\nu\,\ell)(\mathrm{X}\ell \mid [\![M]\!]_\ell)$, where ℓ is fresh.

Lemma 10

1. $[\![P[x := M]]\!] \sim_{\mathrm{cxt}} (\nu\,x)([\![P]\!] \mid [\![M]\!]_x)$
2. *If* X *occurs in* P *at most beneath parallel composition and restriction, then* $[\![P[\mathrm{X} :=_{\mathrm{ho}} (\lambda\,x)\,Q]]\!] \sim_{\mathrm{cxt}} [\![P]\!][\mathrm{X} :=_{\mathrm{ho}} (\lambda\,x)[\![Q]\!]]$.

Proof.

1. Structural induction on P, using the Replication Lemmas.
2. Structural induction on P, using (1) to cover the case of $P = \mathrm{X}M$ for some M.

□

A.1.3 Translation-Coupled Operational Correspondences in Continuation Passing Style

We are now able to carry out the central parts of the proof of Theorem 2, which consists of establishing operational correspondences between spi agents and their translations. The first one is about spi terms whose outermost operator is a match, **let**, **case** or prefix construct, stating that to each initial transition of such an agent on the spi calculus side there corresponds an initial sequence of transitions of its translation on the π-calculus side, and conversely. This operational correspondence already holds up to context bisimilarity. As a preliminary we denote by $\Rightarrow_{\mathrm{cp}}$ the reflexive and transitive closure of the binary relation $\xrightarrow{\tau}_{\mathrm{cp}}$ and by $P \overset{\mu}{\Rightarrow}_{\mathrm{cp}} P'$ the property that $P \Rightarrow_{\mathrm{cp}} \xrightarrow{\mu}_{\mathrm{cp}} P'$.

Lemma 11 *Assertions (a) and (b) are respectively equivalent:*

1. a. $[M \textbf{ is } N]\,P \xrightarrow{\tau}_{\mathrm{cp}} P$

   b. $[\![[M \textbf{ is } N]\,P]\!] \overset{\tau}{\Rightarrow}_{\mathrm{cp}} \sim_{\mathrm{cxt}} [\![P]\!]$

   where in both directions it must hold that $M = N$

2. a. $\textbf{let } (x_1, x_2) = M \textbf{ in } P \xrightarrow{\tau}_{\mathrm{cp}} P[(x_1, x_2) := (M_1, M_2)]$

   b. $[\![\textbf{let } (x_1, x_2) = M \textbf{ in } P]\!] \overset{\tau}{\Rightarrow}_{\mathrm{cp}} \sim_{\mathrm{cxt}} [\![P[(x_1, x_2) := (M_1, M_2)]]\!]$

   where in both directions it must hold that $M = (M_1, M_2)$

3. a. $\textbf{case } M \textbf{ of } 0 : P \ \mathrm{suc}(x) : Q \xrightarrow{\tau}_{\mathrm{cp}} P$

   b. $[\![\textbf{case } M \textbf{ of } 0 : P \ \mathrm{suc}(x) : Q]\!] \overset{\tau}{\Rightarrow}_{\mathrm{cp}} \sim_{\mathrm{cxt}} [\![P]\!]$

   where in both directions it must hold that $M = 0$

4. a. $\textbf{case } M \textbf{ of } 0 : P \ \mathrm{suc}(x) : Q \xrightarrow{\tau}_{\mathrm{cp}} Q[x := M_1]$

   b. $[\![\textbf{case } M \textbf{ of } 0 : P \ \mathrm{suc}(x) : Q]\!] \overset{\tau}{\Rightarrow}_{\mathrm{cp}} \sim_{\mathrm{cxt}} [\![Q[x := M_1]]\!]$

   where in both directions it must hold that $M = \mathrm{suc}(M_1)$

5. a. $\textbf{case } M \textbf{ of } [x]_K \textbf{ in } P \xrightarrow{\tau}_{\mathrm{cp}} P[x := M_1]$

   b. $[\![\textbf{case } M \textbf{ of } [x]_K \textbf{ in } P]\!] \overset{\tau}{\Rightarrow}_{\mathrm{cp}} \sim_{\mathrm{cxt}} [\![P[x := M_1]]\!]$

   where in both directions it must hold that $M = \{M_1\}_K$

6. a. $\textbf{case } M \textbf{ of } [x]_K \textbf{ in } P \xrightarrow{\tau}_{\mathrm{cp}} P[x := M_1]$

   b. $[\![\textbf{case } M \textbf{ of } [x]_K \textbf{ in } P]\!] \overset{\tau}{\Rightarrow}_{\mathrm{cp}} \sim_{\mathrm{cxt}} [\![P[x := M_1]]\!]$

   where in both directions it must hold that $M = \{\!|M_1|\!\}_{N^p}$ *with* $K = (N^p)^{-1}$, $p = +, -$

7. a. $M(x).P \xrightarrow{m(x)}_{\mathrm{cp}} P$

   b. $[\![M(x).P]\!] \overset{m(x)}{\Longrightarrow}_{\mathrm{cp}} \sim^{x}_{\mathrm{cxt}} [\![P]\!]$

   where in both directions it must hold that $M = m$

8. a. $\overline{M}\langle N\rangle.P \xrightarrow{\overline{m}\langle \mathrm{X}\rangle}_{\mathrm{cp}} \mathrm{X}N \mid P$

   b. $[\![\overline{M}\langle N\rangle.P]\!] \overset{\overline{m}\langle \mathrm{X}\rangle}{\Longrightarrow}_{\mathrm{cp}} \sim^{\mathrm{X}}_{\mathrm{cxt}} [\![\mathrm{X}N \mid P]\!]$

   where in both directions it must hold that $M = m$

Proof. Essentially straightforward by analysing what π-calculus execution traces after translation are possible for the outermost operator of the spi term under consideration, doing garbage collection by applying Lemma 9(6) and also Lemma 8(6) as well as commuting substitution and translation by applying Lemma 10(1). Lemmas 9(6), 8(6) and 10(1) are together responsible for why $\sim_{\mathrm{cxt}}$ appears in the (b) parts. □

Next we are already able to prove the first main operational correspondence, the one that states that to every transition of a spi calculus agent there corresponds a sequence of transitions of its translation so that the ensuing spi agent is mapped up to context bisimilarity onto the ensuing π-agent. Its proof is relatively straightforward since we do not need to take into account any interleaving of concurrent threads that correspond to single steps on the spi calculus side.

Lemma 12 *Let P be a spi calculus agent. Then $P \xrightarrow{\mu}_{\mathrm{cp}} P'$ implies $[\![P]\!] \stackrel{\mu}{\Rightarrow}_{\mathrm{cp}} \sim^{\mu}_{\mathrm{cxt}} [\![P']\!]$.*

Proof. Transition induction on $P \xrightarrow{\mu}_{\mathrm{cp}} P'$, using properties from above to establish the respective conclusion. Sample cases:

- Suppose $P = P_1 \mid P_2$, $\mu = \tau$ and $P \xrightarrow{\mu}_{\mathrm{cp}} P'$ has been established from $P_1 \xrightarrow{\overline{m}\langle \mathrm{X}\rangle}_{\mathrm{cp}} P_1'$ and $P_2 \xrightarrow{m(x)}_{\mathrm{cp}} P_2'$, where $P' = P_1'[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, P_2']$.
  Then, by induction: $[\![P_1]\!] \xRightarrow{\overline{m}\langle \mathrm{X}\rangle}_{\mathrm{cp}} P_1^{\#} \sim^{\mathrm{X}}_{\mathrm{cxt}} [\![P_1']\!]$ for some $P_1^{\#}$ and $[\![P_2]\!] \xRightarrow{m(x)}_{\mathrm{cp}} P_2^{\#} \sim^{x}_{\mathrm{cxt}} [\![P_2']\!]$ for some $P_2^{\#}$.
  Then $[\![P]\!] \stackrel{\tau}{\Rightarrow}_{\mathrm{cp}} P_1^{\#}[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, P_2^{\#}]$ and it remains to show $P_1^{\#}[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, P_2^{\#}] \sim_{\mathrm{cxt}} [\![P_1'[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, P_2']]\!]$. This property, however, follows from $P_1^{\#} \sim^{\mathrm{X}}_{\mathrm{cxt}} [\![P_1']\!]$, $P_2^{\#} \sim^{x}_{\mathrm{cxt}} [\![P_2']\!]$, Lemma 7, the transitivity of $\sim_{\mathrm{cxt}}$ and Lemma 10(2).
- Suppose $P = \mathbf{case}\ (M_1, M_2)\ \mathbf{of}\ (x_1, x_2)\ \mathbf{in}\ P_1$ and $P \xrightarrow{\mu}_{\mathrm{cp}} P'$ is of the form $P \xrightarrow{\tau}_{\mathrm{cp}} P_1[(x_1, x_2) := (M_1, M_2)]$. Then the desired conclusion follows from Lemma 11(2).

□

For the above-explained reasons, a *backward operational correspondence*, that is, an operational correspondence that constitutes a converse of Lemma 12 seems to be definitely much harder to come by. The cornerstone of our solution is what we call the *ancestor relation*, which is a specific binary relation on spi and π–calculus agents. We denote it by $\blacktriangleleft$ and it is defined by the clauses in Table 7. Those clauses there that are of the form

$$\frac{[\![P]\!] \Rightarrow_{\mathrm{cp}} Q >_{\varphi} \stackrel{\mu}{\Rightarrow}_{\mathrm{cp}} \sim^{\mu}_{\mathrm{cxt}} [\![P']\!]}{P \blacktriangleleft Q},$$

where the outermost operator of P is a match, **let**, **case** or prefix, are to be read as follows: It holds that $P \blacktriangleleft Q$ if

i. Q is equal to $[\![P]\!]$ or an intermediate state in the π-calculus execution trace of P's outermost operator and

ii. if φ holds, then $Q \stackrel{\mu}{\Rightarrow}_{\mathrm{cp}} \sim^{\mu}_{\mathrm{cxt}} [\![P']\!]$.

The intuition is that $P \blacktriangleleft Q$ holds if Q is a successor state of $[\![P]\!]$ in which no thread of activity that corresponds to an unguarded occurrence of a match, **let**, **case** or prefix construct in P has been completed.

The next two lemmas are about auxiliary properties of the ancestor relation. Lemma 13 states that every spi calculus agent is an ancestor of its translation; Lemma 14 is needed to show that every external action of a descendant of a spi agent P corresponds to an external action of P itself.

Lemma 13 *For every spi calculus agent P it holds that $P \blacktriangleleft [\![P]\!]$.*

Proof. Let P be a spi calculus agent. The lemma can be proven by straightforward structural induction on P, where the base case occurs if $P = \mathbf{0}$ or if the outermost construct of P is a replication, match, **let**, **case** or prefix. The first two of these sub-cases are immediate by $\blacktriangleleft$-NIL, the property that $[\![\mathbf{0}]\!] = \mathbf{0}$ and $\blacktriangleleft$-REP; the other sub-cases are very much alike, so we consider only the one of $P = \overline{M}\langle N\rangle.P_1$ as a sample case. In this case, if $M \neq m$ for any m, then the premise of $\blacktriangleleft$-OUT is trivially satisfied for $Q = [\![P]\!]$, so $P \blacktriangleleft [\![P]\!]$. If, on the other hand, $M = m$ for some m, then $P \xrightarrow{\overline{m}\langle \mathrm{X}\rangle}_{\mathrm{cp}} \mathrm{X}N \mid P_1$, so we can apply Lemma 11 to deduce $[\![P]\!] \xRightarrow{\overline{m}\langle \mathrm{X}\rangle}_{\mathrm{cp}} \sim^{\mathrm{X}}_{\mathrm{cxt}} [\![\mathrm{X}N \mid P_1]\!]$, so the premise of $\blacktriangleleft$-OUT is satisfied for $Q = [\![P]\!]$, so $P \blacktriangleleft [\![P]\!]$. □

Lemma 14 *Whenever $P \blacktriangleleft Q$ and (a) $Q \xrightarrow{\overline{a}\langle \mathrm{X}:k\rangle}_{\mathrm{cp}} Q'$ or (b) $Q \xrightarrow{a(\widetilde{b})}_{\mathrm{cp}} Q'$, then $a = m$ for some m and $|\widetilde{b}| = 1$ in case (b).*

Proof. Suppose $P \blacktriangleleft Q$ and (a) $Q \xrightarrow{\overline{a}\langle \mathrm{X}:k\rangle}_{\mathrm{cp}} Q'$ or (b) $Q \xrightarrow{a(\widetilde{b})}_{\mathrm{cp}} Q'$. Then the desired conclusion can be shown by straightforward induction on the inference of $P \blacktriangleleft Q$. The key is that, in the base cases, our translation ensures that any external action that Q can perform corresponds to an external action of P. □

Whenever case (b) above occurs, we may assume, without loss of generality, that $\widetilde{b} = x$ for some x.

We are now almost ready for proving backward operational correspondence. The only two preliminaries that are still left are a slight variant of Definition 5(2) and a notational convention that allows us to condense the statement of the lemma into just one clause.

Definition 15 Let $\mathcal{R}$ be a binary relation on spi calculus agents. Then we denote by $P\ \langle\mathcal{R}\rangle^{x}\ Q$ the property that $P[x := M]\ \mathcal{R}\ Q[x := M]$ for every closed term M, and by $P\ \langle\mathcal{R}\rangle^{\mathrm{X}}\ Q$ the property that $P[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, R]\ \mathcal{R}\ Q[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, S]$ whenever $R\ \langle\mathcal{R}\rangle^{x}\ S$. Moreover, similarly as before, we have $\langle\mathcal{R}\rangle^{\mathrm{VOID}} = \mathcal{R}$, and we usually write $\langle\mathcal{R}\rangle^{\mu}$ for $\langle\mathcal{R}\rangle^{\mathrm{obj}(\mu)}$. We call $\langle\mathcal{R}\rangle^{\xi}$, where $\xi \in \{\mathrm{X}, x, \mathrm{VOID}\}$, a *universal closure* of $\mathcal{R}$.

$$\frac{[\![[M \textbf{ is } N]\, P]\!] \Rightarrow_{\mathrm{cp}} Q >_{M=N} \overset{\tau}{\Rightarrow}_{\mathrm{cp}} \sim^{\tau}_{\mathrm{cxt}} [\![P]\!]}{[M \textbf{ is } N]\, P \blacktriangleleft Q} \qquad (\blacktriangleleft\text{-MATCH})$$

$$\frac{[\![\textbf{let } (x_1, x_2) = M \textbf{ in } P]\!] \Rightarrow_{\mathrm{cp}} Q >_{M=(M_1,M_2)} \overset{\tau}{\Rightarrow}_{\mathrm{cp}} \sim^{\tau}_{\mathrm{cxt}} [\![P[(x_1,x_2) := (M_1,M_2)]]\!]}{\textbf{let } (x_1, x_2) = M \textbf{ in } P \blacktriangleleft Q} \qquad (\blacktriangleleft\text{-PAIR})$$

$$\frac{[\![\textbf{case } 0 \textbf{ of } 0 : P_1 \ \mathrm{suc}(x) : P_2]\!] \Rightarrow_{\mathrm{cp}} Q >_{\mathrm{true}} \overset{\tau}{\Rightarrow}_{\mathrm{cp}} \sim^{\tau}_{\mathrm{cxt}} [\![P_1]\!]}{\textbf{case } 0 \textbf{ of } 0 : P_1 \ \mathrm{suc}(x) : P_2 \blacktriangleleft Q} \qquad (\blacktriangleleft\text{-ZERO})$$

$$\frac{[\![\textbf{case } M \textbf{ of } 0 : P_2 \ \mathrm{suc}(x) : P_2]\!] \Rightarrow_{\mathrm{cp}} Q >_{M=\mathrm{suc}(M_1)} \overset{\tau}{\Rightarrow}_{\mathrm{cp}} \sim^{\tau}_{\mathrm{cxt}} [\![P_2[x := M_1]]\!]}{\textbf{case } M \textbf{ of } 0 : P_1 \ \mathrm{suc}(x) : P_2 \blacktriangleleft Q} \qquad (\blacktriangleleft\text{-SUC})$$

$$\frac{[\![\textbf{case } M \textbf{ of } [x]_K \textbf{ in } P]\!] \Rightarrow_{\mathrm{cp}} Q >_{\varphi} \overset{\tau}{\Rightarrow}_{\mathrm{cp}} \sim^{\tau}_{\mathrm{cxt}} [\![P[x := M_1]]\!]}{\textbf{case } M \textbf{ of } [x]_K \textbf{ in } P \blacktriangleleft Q} \qquad (\blacktriangleleft\text{-DEC})$$

φ in the above rule stating that (a) $M = \{M_1\}_K$ or (b) $M = \{\!|M_1|\!\}_{N^p}$ and $K = (N^p)^{-1}$ for some $N, p = -, +$

$$\frac{[\![M(x).P]\!] \Rightarrow_{\mathrm{cp}} Q >_{M=m} \overset{\overline{m}(x)}{\Longrightarrow}_{\mathrm{cp}} \sim^{x}_{\mathrm{cxt}} [\![P]\!]}{M(x).P \blacktriangleleft Q} \qquad (\blacktriangleleft\text{-IN})$$

$$\frac{[\![\overline{M}\langle N\rangle.P]\!] \Rightarrow_{\mathrm{cp}} Q >_{M=m} \overset{\overline{m}\langle \mathrm{X}\rangle}{\Longrightarrow}_{\mathrm{cp}} \sim^{\mathrm{X}}_{\mathrm{cxt}} [\![\mathrm{X}N \mid P]\!]}{\overline{M}\langle N\rangle.P \blacktriangleleft Q} \qquad (\blacktriangleleft\text{-OUT})$$

$$\mathbf{0} \blacktriangleleft \mathbf{0} \qquad \frac{P_1 \blacktriangleleft Q_1 \quad P_2 \blacktriangleleft Q_2}{P_1 \mid P_2 \blacktriangleleft Q_1 \mid Q_2} \qquad \frac{P \blacktriangleleft Q}{(\nu\, m)\, P \blacktriangleleft (\nu\, m)\, Q} \qquad !\,P \blacktriangleleft\, !\,[\![P]\!]$$

($\blacktriangleleft$-NIL, $\blacktriangleleft$-PAR, $\blacktriangleleft$-NEW, $\blacktriangleleft$-REP)

Table 7. Defining clauses for the ancestor relation on spi and π-terms.

The notational convention is such: Given any spi calculus agents P, P' and some continuation passing label μ, we denote by $P \xrightarrow{\widehat{\mu}}_{\mathrm{cp}} P'$ the property that $P \xrightarrow{\mu}_{\mathrm{cp}} P'$ if $\mu \neq \tau$ and $P = P'$ or $P \xrightarrow{\mu}_{\mathrm{cp}} P'$ if $\mu = \tau$.

The backward operational correspondence, then, is by far the most non-standard part of the proof of Theorem 2. For this reason, we spell out its proof in some detail. It is carried out by transition induction but covering the individual sub-cases requires relatively long arguments, using many of the properties introduced earlier.

Lemma 16 *Suppose $P \blacktriangleleft Q$. Then $Q \xrightarrow{\mu}_{\mathrm{cp}} Q'$ implies that there is a P' so that $P \xrightarrow{\widehat{\mu}}_{\mathrm{cp}} P' \langle \sim_{\mathrm{cxt}} \blacktriangleleft \sim_{\mathrm{cxt}} \rangle^{\mu} Q'$.*

Proof. Suppose $P \blacktriangleleft Q$ and $Q \xrightarrow{\mu}_{\mathrm{cp}} Q'$. We show the desired conclusion by induction on the measure $\mathrm{sdepth}(P)$, where $\mathrm{sdepth}(op(P_1, \ldots, P_k))$ is 0 if op is not $\mathbf{0}$, $|$, $(\nu\, m)$ for some m or !, and $1 + \max(\mathrm{sdepth}(P_1), \ldots, \mathrm{sdepth}(P_k))$ otherwise. We organise the proof along how P is composed at the topmost level

- $\underline{P = \mathbf{0}}$ (a sub-case of the overall base case): In this case $P \blacktriangleleft Q$ must have been inferred with via $\blacktriangleleft$-NIL from $Q = \mathbf{0}$, whence the case is vacuous.
- $\underline{P = [M \textbf{ is } N]\, P_1}$ (a sub-case of the overall base case): In this case $P \blacktriangleleft Q$ must have been inferred via $\blacktriangleleft$-MATCH from (i) $[\![[M \textbf{ is } N]\, P_1]\!] \Rightarrow_{\mathrm{cp}} Q$ and (ii) if $M = N$, then $Q \overset{\tau}{\Rightarrow}_{\mathrm{cp}} R \sim^{\tau}_{\mathrm{cxt}} [\![P_1]\!]$ for some R.
  We observe that $[\![[M = N]\, P_1]\!] \Rightarrow_{\mathrm{cp}} Q'$.
  Moreover, if $Q' \neq R$, which means that $\mu = \tau$, then it must hold that, if $M = N$, Q' must eventually evolve to R just as Q, so $P \blacktriangleleft Q'$, so $\blacktriangleleft$-MATCH entails $P \blacktriangleleft Q'$. Setting $P' = P$, by reflexivity and $\mu = \tau$, we thus have $P' \langle \sim_{\mathrm{cxt}} \blacktriangleleft \sim_{\mathrm{cxt}} \rangle^{\mu} Q'$. Obviously we then have also $P \xrightarrow{\widehat{\mu}}_{\mathrm{cp}} P'$.
  Finally, if $Q' = R$, which also means that $\mu = \tau$, then it must hold that $M = N$ since there are no other circumstances under which any descendant of $[\![P]\!]$ can evolve to a state that is related via $\sim_{\mathrm{cxt}}$ to $[\![P_1]\!]$.
  Hence, again $P \xrightarrow{\widehat{\mu}}_{\mathrm{cp}} P'$, this time setting $P' = P_1$. Moreover, $P' \blacktriangleleft [\![P']\!]$ because of Lemma 13, so we have also $P' \langle \sim_{\mathrm{cxt}} \blacktriangleleft \sim_{\mathrm{cxt}} \rangle^{\mu} Q'$.
- $\underline{P = \textbf{let } (x_1, x_2) = M \textbf{ in } P_1}$, $\underline{P = \textbf{case } M \textbf{ of } 0 : P_1 \ \mathrm{suc}(x) : P_2}$, or $\underline{P = \textbf{case } M \textbf{ of } [x]_K \textbf{ in } P}$ (sub-cases of the overall base case): These cases can be treated very similarly to the previous one.
- $\underline{P = M(x).P_1}$ (a sub-case of the overall base case): In this case $P \blacktriangleleft Q$ must have been inferred via $\blacktriangleleft$-(in)

from (i) $[\![M(x).P_1]\!] \Rightarrow_{\mathrm{cp}} Q$ and (ii) if $M = m$ for some m, then $Q \xRightarrow{m(x)}_{\mathrm{cp}} R \sim^x_{\mathrm{cxt}} [\![P_1]\!]$ for some R.
This case is largely a rerun of the first one too except for the last step, where more effort is required. It seems to be sensible to spell out the entire case to make that clear.
We observe that $[\![M(x).P_1]\!] \Rightarrow_{\mathrm{cp}} Q'$.
Moreover, if $Q' \neq R$, which means that $\mu = \tau$, then it must hold that, if $M = m$ for some m, Q' must eventually evolve to R just as Q, so $P \blacktriangleleft Q'$, so $\blacktriangleleft$-IN entails $P \blacktriangleleft Q'$. Setting $P' = P$, by reflexivity and $\mu = \tau$, we thus have $P' \langle \sim_{\mathrm{cxt}} \blacktriangleleft \sim_{\mathrm{cxt}} \rangle^{\mu} Q'$. Obviously we then have also $P \xrightarrow{\widehat{\mu}}_{\mathrm{cp}} P'$.
Finally, if $Q' = R$, then it must hold that $M = m$ and $\mu = m(x)$ for some m since there are no other circumstances under which any descendant of $[\![P]\!]$ can evolve to a state that is related via $\sim^x_{\mathrm{cxt}}$ to $[\![P_1]\!]$. Hence, again $P \xrightarrow{\widehat{\mu}}_{\mathrm{cp}} P'$, this time setting $P' = P_1$, and it remains to show $P' \langle \sim_{\mathrm{cxt}} \blacktriangleleft \sim_{\mathrm{cxt}} \rangle^{\mu} Q'$ or, in other words, $P_1 \langle \sim_{\mathrm{cxt}} \blacktriangleleft \sim_{\mathrm{cxt}} \rangle^{x} Q'$.
So let M be closed. Then, by reflexivity of $\sim_{\mathrm{cxt}}$ and Lemma 13: $P_1[x := M] \sim_{\mathrm{cxt}} \blacktriangleleft [\![P_1[x := M]]\!]$.
Moreover, by Lemma 10: $[\![P_1[x := M]]\!] \sim_{\mathrm{cxt}} (\nu\,\ell)([\![P_1]\!][x := \ell] \mid [\![M]\!]_\ell)$ and, further, $[\![P_1]\!][x := \ell] \sim_{\mathrm{cxt}} Q_1[x := \ell]$ because of $P_1 \sim^x_{\mathrm{cxt}} Q'$, where ℓ is fresh.
Then, by Lemma 7: $(\nu\,\ell)([\![P_1]\!][x := \ell] \mid [\![M]\!]_\ell) \sim_{\mathrm{cxt}} (\nu\,\ell)(Q_1[x := \ell] \mid [\![M]\!]_\ell)$, so $[\![P_1[x := M]]\!] \sim_{\mathrm{cxt}} (\nu\,\ell)(Q_1[x := \ell] \mid [\![M]\!]_\ell)$ by transitivity, so $P_1 \langle \sim_{\mathrm{cxt}} \blacktriangleleft \sim_{\mathrm{cxt}} \rangle^{x} Q'$ as required.

- $P = \overline{M}\langle N\rangle.P_1$ (a sub-case of the overall base case): In this case $P \blacktriangleleft Q$ must have been inferred via $\blacktriangleleft$-OUT from (i) $[\![\overline{M}(N).P_1]\!] \Rightarrow_{\mathrm{cp}} Q$ and (ii) if $M = m$ for some m, then $Q \xRightarrow{\overline{m}(\mathrm{X})}_{\mathrm{cp}} R \sim^{\mathrm{X}}_{\mathrm{cxt}} [\![P_1]\!]$ for some R.
This case is very similar to the previous one up to a point where one has to conclude the case by showing $\mathrm{X}N \mid P_1 \langle \sim_{\mathrm{cxt}} \blacktriangleleft \sim_{\mathrm{cxt}} \rangle^{\mathrm{X}} Q'$ under the assumption of $[\![\mathrm{X}N \mid P_1]\!] \sim^{\mathrm{X}}_{\mathrm{cxt}} Q'$.
So suppose $R \langle \sim_{\mathrm{cxt}} \blacktriangleleft \sim_{\mathrm{cxt}} \rangle^{x} S$. Then , by definition,
$$R[x := N] \sim_{\mathrm{cxt}} T \blacktriangleleft U \sim_{\mathrm{cxt}} (\nu\,\ell)(S[x := \ell] \mid [\![N]\!]_\ell)$$
for some T and some U, where ℓ is fresh. Hence, by Lemma 7, since Lemma 13 entails $P_1 \blacktriangleleft [\![P_1]\!]$ and because of $\blacktriangleleft$-PAR:
$$\begin{aligned} &R[x := N] \mid P_1 \\ &\quad \sim_{\mathrm{cxt}} T \mid P_1 \blacktriangleleft U \mid [\![P_1]\!] \sim_{\mathrm{cxt}} \\ &\qquad (\nu\,\ell)(S[x := \ell] \mid [\![N]\!]_\ell) \mid [\![P_1]\!]. \end{aligned}$$
On the left hand side of that, by definition:
$$(\mathrm{X}N \mid P_1)[\mathrm{X} :=_{\mathrm{ho}} (\lambda\,x)\,R] = R[x := N] \mid P_1;$$
on the right hand side, also by definition:
$$\begin{aligned} &(\nu\,\ell)(S[x := \ell] \mid [\![N]\!]_\ell) \mid [\![P_1]\!] \\ &\quad = (((\nu\,\ell)(\mathrm{X}\ell \mid [\![N]\!]_\ell)) \mid [\![P_1]\!])[\mathrm{X} :=_{\mathrm{ho}} (\lambda\,x)\,S] \\ &\quad = [\![\mathrm{X}N \mid P_1]\!][\mathrm{X} :=_{\mathrm{ho}} (\lambda\,x)\,S]. \end{aligned}$$
Further, by $[\![\mathrm{X}N \mid P_1]\!] \sim^{\mathrm{X}}_{\mathrm{cxt}} Q'$:
$$[\![\mathrm{X}N \mid P_1]\!][\mathrm{X} :=_{\mathrm{ho}} (\lambda\,x)\,S] \sim_{\mathrm{cxt}} Q'[\mathrm{X} :=_{\mathrm{ho}} (\lambda\,x)\,S].$$
Hence, by transitivity, $\mathrm{X}N \mid P_1 \langle \sim_{\mathrm{cxt}} \blacktriangleleft \sim_{\mathrm{cxt}} \rangle^{\mathrm{X}} Q'$ as required.

- $P = P_1 \mid P_2$ (a sub-case of the overall inductive case): In this case $P \blacktriangleleft Q$ must have been inferred via $\blacktriangleleft$-PAR from $P_i \blacktriangleleft Q_i$, $i = 1, 2$, where $Q = Q_1 \mid Q_2$. Then there are three further sub-cases:
  - $Q \xrightarrow{\mu}_{\mathrm{cp}} Q'$ has been inferred from $Q_1 \xrightarrow{\mu}_{\mathrm{cp}} Q'_1$ for some Q'_1 , where $Q' = Q'_1 \mid Q_2$:
  By induction, there is a P'_1 so that $P_1 \xrightarrow{\widehat{\mu}}_{\mathrm{cp}} P'_1$ and $P'_1 \langle \sim_{\mathrm{cxt}} \blacktriangleleft \sim_{\mathrm{cxt}} \rangle^{\mu} Q'_1$. Setting $P' = P'_1 \mid P_2$ we thus have $P \xrightarrow{\widehat{\mu}}_{\mathrm{cp}} P'$.
  As for the property $P' \langle \sim_{\mathrm{cxt}} \blacktriangleleft \sim_{\mathrm{cxt}} \rangle^{\mu} Q'$, we consider the case of $\mu = \overline{m}\langle \mathrm{X}\rangle$ for some m, as the other ones are similar.
  So suppose $R \langle \sim_{\mathrm{cxt}} \blacktriangleleft \sim_{\mathrm{cxt}} \rangle^{x} S$. Then, first of all, $P'_1[\mathrm{X} :=_{\mathrm{ho}} (\lambda\,x)\,R] \sim_{\mathrm{cxt}} T \blacktriangleleft U \sim_{\mathrm{cxt}} Q'_1[\mathrm{X} :=_{\mathrm{ho}} (\lambda\,x)\,S]$ for some T and some U.
  Hence, by Lemma 7 and $\blacktriangleleft$-PAR:
  $$\begin{aligned} &P'_1[\mathrm{X} :=_{\mathrm{ho}} (\lambda\,x)\,R] \mid P_2 \\ &\quad \blacktriangleleft T \mid Q_2 \sim_{\mathrm{cxt}} \\ &\qquad Q'_1[\mathrm{X} :=_{\mathrm{ho}} (\lambda\,x)\,S] \mid Q_2. \end{aligned}$$
  Further, $P'[\mathrm{X} :=_{\mathrm{ho}} (\lambda\,x)\,R] = P'_1[\mathrm{X} :=_{\mathrm{ho}} (\lambda\,x)\,R] \mid P_2$ and $Q'_1[\mathrm{X} :=_{\mathrm{ho}} (\lambda\,x)\,S] \mid Q_2 = Q'[\mathrm{X} :=_{\mathrm{ho}} (\lambda\,x)\,S]$, so $P' \langle \sim_{\mathrm{cxt}} \blacktriangleleft \sim_{\mathrm{cxt}} \rangle^{\mu} Q'$ as required.
  - $Q \xrightarrow{\mu}_{\mathrm{cp}} Q'$ has been inferred from $Q_1 \xrightarrow{\overline{a}\langle \mathrm{X}:k\rangle}_{\mathrm{cp}} Q'_1$ for some Q'_1 and $Q_2 \xrightarrow{a(\widetilde{b})}_{\mathrm{cp}} Q'_2$ for some Q'_2, $k = |\widetilde{b}|$ and $Q' = Q'_1[\mathrm{X} :=_{\mathrm{ho}} (\lambda\,\widetilde{b})\,Q'_2]$ and $\mu = \tau$:
  Then, by Lemma 14, $a = m$ for some m, $k = 1$ and $\widetilde{b} = x$ for some x.
  Hence, by induction, there are P'_i, $i = 1, 2$, so that $P_1 \xrightarrow{\overline{m}\langle \mathrm{X}\rangle}_{\mathrm{cp}} P'_1$, $P'_1 \langle \sim_{\mathrm{cxt}} \blacktriangleleft \sim_{\mathrm{cxt}} \rangle^{\mathrm{X}} Q'_1$, $P_2 \xrightarrow{m(x)}_{\mathrm{cp}} P'_2$ and $P'_2 \langle \sim_{\mathrm{cxt}} \blacktriangleleft \sim_{\mathrm{cxt}} \rangle^{x} Q'_2$.

Setting $P' = P'_1[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, P'_2]$, we thus have $P \xrightarrow{\widehat{\mu}} P'$ and also $P' \,\langle\sim_{\mathrm{cxt}}\blacktriangleleft\sim_{\mathrm{cxt}}\rangle^{\mu}\, Q'$.

- $\underline{P = (\nu\, m)\, P_1}$ (a sub-case of the overall inductive case): In this case $P \blacktriangleleft Q$ must have been inferred via $\blacktriangleleft$-NEW from $P_1 \blacktriangleleft Q_1$, where $Q = (\nu\, m)\, Q_1$.
  Hence, $Q \xrightarrow{\mu}_{\mathrm{cp}} Q'$ must have been inferred from $Q_1 \xrightarrow{\mu}_{\mathrm{cp}} Q'_1$ for some Q'_1, where $Q' = (\nu\, m)\, Q'_1$ and $m \notin \mathrm{n}(\mu)$.
  Hence, by induction, there is a P'_1 so that $P_1 \xrightarrow{\widehat{\mu}}_{\mathrm{cp}} P'_1$ and $P'_1 \,\langle\sim_{\mathrm{cxt}}\blacktriangleleft\sim_{\mathrm{cxt}}\rangle^{\mu}\, Q'_1$. Setting $P' = P'_1 \mid P_2$ we thus have $P \xrightarrow{\widehat{\mu}}_{\mathrm{cp}} P'$.
  As for the property $P' \,\langle\sim_{\mathrm{cxt}}\blacktriangleleft\sim_{\mathrm{cxt}}\rangle^{\mu}\, Q'$, we consider the case of $\mu = \overline{m}\langle \mathrm{X}\rangle$ for some m, as the ones are similar.
  So suppose $R \,\langle\sim_{\mathrm{cxt}}\blacktriangleleft\sim_{\mathrm{cxt}}\rangle^{x}\, S$. Then, first of all, $P'_1[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, R] \sim_{\mathrm{cxt}} T \blacktriangleleft U \sim_{\mathrm{cxt}} Q'_1[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, S]$ for some T and some U.
  Hence, by Lemma 7 and $\blacktriangleleft$-NEW:
  $$\begin{aligned}&(\nu\, m)\, P'_1[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, R]\\ &\quad\sim_{\mathrm{cxt}} (\nu\, m)\, T \blacktriangleleft (\nu\, m)\, U \sim_{\mathrm{cxt}}\\ &\qquad\qquad (\nu\, m)\, Q'_1[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, S].\end{aligned}$$
  Further, by the convention about bound names, $P'[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, R] = (\nu\, m)\, P'_1[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, R]$ and $(\nu\, m)\, Q'_1[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, S] = Q'[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, S]$, so $P' \,\langle\sim_{\mathrm{cxt}}\blacktriangleleft\sim_{\mathrm{cxt}}\rangle^{\mu}\, Q'$ as required.

- $\underline{P = !\, P_1}$ (a sub-case of the overall inductive case): In this case $P \blacktriangleleft Q$ must have been inferred from $Q = !\, [\![P_1]\!]$. Then there are two further sub-cases:
  - $Q \xrightarrow{\mu}_{\mathrm{cp}} Q'$ has been inferred from $[\![P_1]\!] \xrightarrow{\mu}_{\mathrm{cp}} Q^{\#}$ for some $Q^{\#}$, where $Q' = Q^{\#} \mid !\, [\![P_1]\!]$:
    First of all, because Lemma 13 entails $P_1 \blacktriangleleft [\![P_1]\!]$, we can apply the induction hypothesis, obtaining a P'_1 so that $P_1 \xrightarrow{\widehat{\mu}}_{\mathrm{cp}} P'_1$ and $P'_1 \,\langle\sim_{\mathrm{cxt}}\blacktriangleleft\sim_{\mathrm{cxt}}\rangle^{\mu}\, Q^{\#}$.
    * If $\mu = \tau$ and $P_1 \xrightarrow{\widehat{\mu}}_{\mathrm{cp}} P'_1$ is a non-transition, that is, it holds that $P_1 = P'_1$, then we set $P' = P$, so that $P \xrightarrow{\widehat{\mu}}_{\mathrm{cp}} P'$.
      As for the property $P' \,\langle\sim_{\mathrm{cxt}}\blacktriangleleft\sim_{\mathrm{cxt}}\rangle^{\mu}\, Q'$, we observe that $P'_1 \sim_{\mathrm{cxt}}\blacktriangleleft\sim_{\mathrm{cxt}} Q^{\#}$ and $P_1 = P'_1$ together entail that $P_1 \sim_{\mathrm{cxt}} R \blacktriangleleft S \sim_{\mathrm{cxt}} Q^{\#}$ for some R and some S.
      Hence, by Lemma 7, since Lemma 13 entails $!\, P_1 \blacktriangleleft !\, [\![P_1]\!]$ and by $\blacktriangleleft$-PAR:
      $$\begin{aligned}&P_1 \mid !\, P_1\\ &\quad\sim_{\mathrm{cxt}} R \mid !\, P_1 \blacktriangleleft S \mid !\, [\![P_1]\!] \sim_{\mathrm{cxt}}\\ &\qquad\qquad Q^{\#} \mid !\, [\![P_1]\!].\end{aligned}$$
      Further, by $P' = P$ and Lemma 8(5), it holds that $P' \sim_{\mathrm{cxt}} P_1 \mid !\, P_1$, so $P' \sim_{\mathrm{cxt}} R \mid !\, P_1$ by transitivity. Still further, we have $Q^{\#} \mid !\, [\![P_1]\!] = Q'$, so $P' \,\langle\sim_{\mathrm{cxt}}\blacktriangleleft\sim_{\mathrm{cxt}}\rangle^{\mu}\, Q'$ as required.
    * If $P_1 \xrightarrow{\mu}_{\mathrm{cp}} P'_1$, then we set $P' = P'_1 \mid P$, so that $P \xrightarrow{\widehat{\mu}}_{\mathrm{cp}} P'$.
      As for the property $P' \,\langle\sim_{\mathrm{cxt}}\blacktriangleleft\sim_{\mathrm{cxt}}\rangle^{\mu}\, Q'$, we consider the case of $\mu = \overline{m}\langle \mathrm{X}\rangle$ for some m, as the other ones are similar.
      So suppose $R \,\langle\sim_{\mathrm{cxt}}\blacktriangleleft\sim_{\mathrm{cxt}}\rangle^{x}\, S$. Then, first of all, $P'_1[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, R] \sim_{\mathrm{cxt}} T \blacktriangleleft U \sim_{\mathrm{cxt}} Q^{\#}[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, S]$ for some T and some U.
      Hence, by Lemma 7, since Lemma 13 entails $!\, P_1 \blacktriangleleft !\, [\![P_1]\!]$ and by $\blacktriangleleft$-PAR:
      $$\begin{aligned}&P'_1[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, R] \mid !\, P_1\\ &\quad\sim_{\mathrm{cxt}} T \mid !\, P_1 \blacktriangleleft U \mid !\, [\![P_1]\!] \sim_{\mathrm{cxt}}\\ &\qquad\qquad Q^{\#}[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, S] \mid !\, [\![P_1]\!].\end{aligned}$$
      Further, $P'[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, R] = P'_1[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, R] \mid !\, P_1$ and $Q^{\#}[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, S] \mid !\, [\![P_1]\!] = Q'[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, S]$, so $P' \,\langle\sim_{\mathrm{cxt}}\blacktriangleleft\sim_{\mathrm{cxt}}\rangle^{\mu}\, Q'$ as required.
  - $Q \xrightarrow{\mu}_{\mathrm{cp}} Q'$ has been inferred from $[\![P_1]\!] \xrightarrow{\overline{a}\langle \mathrm{X}:k\rangle}_{\mathrm{cp}} R_1$ for some R_1 and $[\![P_1]\!] \xrightarrow{a(\widetilde{b})}_{\mathrm{cp}} R_2$ for some R_2, where $k = |\widetilde{b}|$, $Q' = R_1[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, R_2] \mid !\, [\![P_1]\!]$ and $\mu = \tau$:
    First of all, by Lemma 14, $a = m$ for some m, $k = 1$ and $\widetilde{b} = x$ for some x.
    Hence, because of the property $P_1 \blacktriangleleft [\![P_1]\!]$, we can apply the induction hypothesis, obtaining a P'_1 and a P'_2 so that $P_1 \xrightarrow{\overline{m}\langle \mathrm{X}\rangle}_{\mathrm{cp}} P'_1$, $P'_1 \,\langle\sim_{\mathrm{cxt}}\blacktriangleleft\sim_{\mathrm{cxt}}\rangle^{\mathrm{X}}\, R_1$, $P_2 \xrightarrow{m(x)}_{\mathrm{cp}} P'_2$ and $P'_2 \,\langle\sim_{\mathrm{cxt}}\blacktriangleleft\sim_{\mathrm{cxt}}\rangle^{x}\, R_2$.
    We set $P' = P'_1[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, P'_2] \mid P$, so that $P \xrightarrow{\widehat{\mu}}_{\mathrm{cp}} P'$.
    As for the property $P' \,\langle\sim_{\mathrm{cxt}}\blacktriangleleft\sim_{\mathrm{cxt}}\rangle^{\mu}\, Q'$, we observe that $P'_1[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, P'_2] \sim_{\mathrm{cxt}} S \blacktriangleleft T \sim_{\mathrm{cxt}} R_1[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, R_2]$ for some S and some T Hence, by Lemma 7, since Lemma 13 entails $!\, P_1 \blacktriangleleft !\, [\![P_1]\!]$ and by $\blacktriangleleft$-PAR:
    $$\begin{aligned}&P'_1[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, P'_2] \mid !\, P_1\\ &\quad\sim_{\mathrm{cxt}} S \mid !\, P_1 \blacktriangleleft T \mid !\, [\![P_1]\!] \sim_{\mathrm{cxt}}\\ &\qquad\qquad R_1[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, R_2] \mid !\, [\![P_1]\!].\end{aligned}$$
    Further, $P' = P'_1[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, P'_2] \mid !\, P_1$ and $R_1[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, R_2] \mid !\, [\![P_1]\!] = Q'$, so $P' \,\langle\sim_{\mathrm{cxt}}\blacktriangleleft\sim_{\mathrm{cxt}}\rangle^{\mu}\, Q'$ as required.

□

A.1.4 Preservation of May-Testing in Continuation Passing Style

We denote by $P \ \textsc{may}\ E$ the property that any agent P may pass any experiment E over the same calculus with respect to the appropriate continuation-passing-style operational semantics. Lemmas 12 and 16 then allow us to conclude:

Lemma 17 *Let P and E be a spi calculus agent or experiment, respectively. Then $P \ \textsc{may}_{\mathrm{cp}}\ E$ if and only if $[\![P]\!] \ \textsc{may}_{\mathrm{cp}}\ [\![E]\!]$.*

Proof. Straightforward from Lemmas 12 and 16. □

A.1.5 Equivalence of May-Testing in Continuation Passing Style with and without Commitments

What is now left of the proof of Theorem 2 is to connect the testing scenarios treated in Lemma 17 with the testing scenarios over the operational semantics for the π- and the spi calculus introduced in Section 2. We proceed in three steps, where none of them involves the translation. For this reason, all of them are much simpler than establishing the validity of Lemma 17.

The first step is concerned with may-testing over the spi-calculus, namely with proving that continuation-passing-style operational semantics with and without commitments yield the same results in that. The axioms and clauses for the operational semantics with commitments are gathered in Table 8.

The sought-after equivalence proof is again essentially a matter of establishing a suitable operational correspondence. More specifically, we use a binary relation on spi agents, which is given as follows:

$$P >^{\bullet} P$$

$$\frac{P >^{\bullet} Q \xrightarrow{\tau}_{\mathrm{cp}} Q'}{P >^{\bullet} Q'} \quad \begin{array}{l} Q = [M \text{ is } N] \ldots, \\ Q = \textbf{let} \ \ldots \ \text{or} \\ Q = \textbf{case} \ \ldots \end{array}$$

$$\frac{P_1 >^{\bullet} Q_1 \quad P_2 >^{\bullet} Q_2}{P_1 \mid P_2 >^{\bullet} Q_1 \mid Q_2}$$

$$\frac{P >^{\bullet} Q}{(\nu\, m)\, P >^{\bullet} (\nu\, m)\, Q}$$

Then the operational correspondence can again be seen as a kind of expansion:

Lemma 18 *Suppose $P >^{\bullet} Q$. Then:*

1. *Whenever $P \xrightarrow{\mu}_{\mathrm{cp}'} P'$, then $Q \overset{\mu}{\Longrightarrow}_{\mathrm{cp}} Q'$ for some Q' so that $P' \ \langle >^{\bullet} \rangle^{\mu}\ Q'$.*
2. *Whenever $Q \xrightarrow{\mu}_{\mathrm{cp}} Q'$, then $P \xrightarrow{\widehat{\mu}}_{\mathrm{cp}'} P'$ for some P' so that $P' \ \langle >^{\bullet} \rangle^{\mu}\ Q'$.*

Proof. Both (1) and (2) can be shown essentially by straightforward induction on the length of the inference used to establish $P >^{\bullet} Q$, exploiting the universal closures very much like in the proof of Lemma 16 to treat those sub-cases where $P >^{\bullet} Q$ has been inferred from $P_i >^{\bullet} Q_i$, $i = 1, 2$, $P = P_1 \mid P_2$ and $Q_1 \mid Q_2 = Q$, and where $P \xrightarrow{\mu}_{\mathrm{cp}'} P'$ / $Q \xrightarrow{\mu}_{\mathrm{cp}} Q'$ is the outcome of a communication between P_1 and P_2 / Q_1 and Q_2 with $\mu = \tau$. The only somewhat delicate sub-case occurs in proving (2), namely if $P >^{\bullet} Q$ has been inferred from $P >^{\bullet} Q^{\#} \xrightarrow{\tau}_{\mathrm{cp}} Q$, where $Q^{\#} = [M \text{ is } N] \ldots$, $Q^{\#} = \textbf{let} \ \ldots$ or $Q^{\#} = \textbf{case} \ \ldots$. If that same condition holds with $Q^{\#}$ replaced by Q, then we must have $\mu = \tau$ and we we can instantiate the same rule as the one used to establish $P >^{\bullet} Q$ to infer $P >^{\bullet} Q'$, in which case $P' = P$ with $P' \ \langle >^{\bullet} \rangle^{\mu}\ Q'$; otherwise, because of the side condition on $Q^{\#}$, $P >^{\bullet} Q^{\#}$ must have been inferred either by instantiating the same rule as for $P >^{\bullet} Q$ or from $P = Q^{\#}$ and the first clause for $>^{\bullet}$. Thus, we can carry out an auxiliary induction on the length of the inference used to establish $P >^{\bullet} Q^{\#}$ to obtain a sequence of the form $P = Q^{\#}_0 \xrightarrow{\tau}_{\mathrm{cp}} \ldots \xrightarrow{\tau}_{\mathrm{cp}} Q^{\#}_k = Q^{\#}$, $k \geq 1$, so that $Q^{\#}_i = [M_i \text{ is } N_i] \ldots$, $Q^{\#}_i = \textbf{let} \ \ldots$ or $Q^{\#}_i = \textbf{case} \ \ldots$ (and $P >^{\bullet} Q^{\#}_i$) for all $i \in \{0, \ldots, k\}$. Thus, it must hold that $P = Q^{\#}_0 > \ldots > Q^{\#}_k$ and, at the same time, the side condition on $Q^{\#}$ entails $Q^{\#} > Q$. Thus, by moving backward along $P = Q^{\#}_0, \ldots, Q^{\#}_k = Q^{\#}$, we can infer $P \xrightarrow{\mu}_{\mathrm{cp}'} Q'$. In this case $P' = Q'$ and it remains to show $P' \ \langle >^{\bullet} \rangle^{\mu}\ Q'$. If $\mu = \tau$, then $\langle >^{\bullet} \rangle^{\mu} = \langle >^{\bullet} \rangle^{\text{VOID}} = >^{\bullet}$, whence $P' \ \langle >^{\bullet} \rangle^{\mu}\ Q'$ follows from the first clause for $>^{\bullet}$; if $\mu = m(x)$, then $P' = Q'$ implies $P'[x := M] = Q'[x := M]$ for every closed term M, so $P'[x := M] >^{\bullet} Q'[x := M]$ for every closed M, so $P' \ \langle >^{\bullet} \rangle^{\mu}\ Q'$; if $\mu = \overline{m}\langle \mathrm{X} \rangle$, then the occurrence of X in P'/Q' lies beneath zero or more instances of parallel composition and/or restriction, so we can carry out a straightforward structural induction to show that $P'[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, R] >^{\bullet} Q'[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, Q]$ whenever $R \ \langle >^{\bullet} \rangle^{x}\ S$, so again $P' \ \langle >^{\bullet} \rangle^{\mu}\ Q'$. □

Lemma 19 *Let P and E be a spi calculus agent or experiment, respectively. Then $P \ \textsc{may}_{\mathrm{cp}'}\ E$ if and only if $P \ \textsc{may}_{\mathrm{cp}}\ E$.*

Proof. Straightforward from Lemma 18. □

A.2 From Continuation Passing back to Conventional Style

The last two steps are the most simple ones, being concerned with establishing the equivalence of may-testing

$$[M \textbf{ is } M]\, P > P \quad \textbf{let } (x_1, x_2) = (M_1, M_2) \textbf{ in } P > P[(x_1, x_2) := (M_1, M_2)]$$

$$\textbf{case } 0 \textbf{ of } 0 : P\ \mathrm{suc}(x) : Q > P \quad \textbf{case } \mathrm{suc}(M) \textbf{ of } 0 : P\ \mathrm{suc}(x) : Q > Q[x := M]$$

$$\textbf{case } \{M\}_K \textbf{ of } [x]_K \textbf{ in } P > P[x := M]$$

$$\textbf{case } \{\!\!\{M\}\!\!\}_{N^p} \textbf{ of } [x]_{(N^p)^{-1}} \textbf{ in } P > P[x := M] \;\; p = -, +$$

$$\frac{P > P' \quad P' \xrightarrow{\mu}_{\mathrm{cp}'} P''}{P \xrightarrow{\mu}_{\mathrm{cp}'} P''} \quad m(x).P \xrightarrow{m(x)}_{\mathrm{cp}'} P \quad \overline{m}\langle M\rangle.P \xrightarrow{\overline{m}\langle \mathrm{X}\rangle}_{\mathrm{cp}'} \mathrm{X}M \mid P$$

$$\frac{P \xrightarrow{\mu}_{\mathrm{cp}'} P'}{P \mid Q \xrightarrow{\mu}_{\mathrm{cp}'} P' \mid Q} \quad \frac{P \xrightarrow{\overline{m}\langle \mathrm{X}\rangle}_{\mathrm{cp}'} P' \quad Q \xrightarrow{m(x)}_{\mathrm{cp}'} Q'}{P \mid Q \xrightarrow{\tau}_{\mathrm{cp}'} P'[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, Q']}$$

$$\frac{P \xrightarrow{\mu}_{\mathrm{cp}'} P'}{(\nu\, m)\, P \xrightarrow{\mu}_{\mathrm{cp}'} (\nu\, m)\, P'}\; m \notin \mathrm{n}(\mu) \quad \frac{P \xrightarrow{\mu}_{\mathrm{cp}'} P'}{!\, P \xrightarrow{\mu}_{\mathrm{cp}'} P' \mid !\, P} \quad \frac{P \xrightarrow{\overline{m}\langle \mathrm{X}\rangle}_{\mathrm{cp}'} P_1' \quad P \xrightarrow{m(x)}_{\mathrm{cp}'} P_2'}{!\, P \xrightarrow{\tau}_{\mathrm{cp}'} P_1'[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, P_2'] \mid !\, P}$$

Table 8. Axioms and SOS clauses for continuation-passing-style operational semantics with commitments for the spi calculus, where only one symmetric clause is respectively stated for interleaving and communication.

over the spi or π-calculus with standard operational semantics as introduced in Section 2 or with continuation-passing-style operational semantics as laid out in Tables 5 or 8, respectively. Thanks to context bisimilarity, these steps are also very much alike, whence we shall comment on them together: In both cases we first introduce a labelled *extraction relation*, by which we get hold of the ordinary label and the ordinary residual that correspond to any continuation-passing-style output transition. On this basis, it is possible to formulate and prove a suitable operational correspondence whereupon the desired equivalence is straightforward. When going from continuation-passing-style to ordinary operational semantics, the operational correspondence is partly up to structural congruence; in the converse direction, it is partly up to context bisimilarity. Also, as for output transitions, we show on the fly that a certain closure property with respect to closed abstractions holds. The operational correspondences are yet again proved by transition induction and the just-mentioned closure properties are instrumental in covering τ-steps that have been inferred from visible transitions using communication rules.

A.2.1 The Spi Calculus Side

The extraction relation on spi calculus agents is defined as follows:

$$\mathrm{X}M \xrightarrow{(\nu)\, M}_{\mathrm{x}} \mathbf{0}$$

$$\frac{P \xrightarrow{(\nu\, \widetilde{m})\, M}_{\mathrm{x}} P'}{P \mid Q \xrightarrow{(\nu\, \widetilde{m})\, M}_{\mathrm{x}} P' \mid Q}\; \widetilde{m} \cap \mathrm{fn}(Q) = \emptyset$$

$$\frac{P \xrightarrow{(\nu\, \widetilde{m})\, M}_{\mathrm{x}} P'}{(\nu\, n)\, P \xrightarrow{(\nu\, \widetilde{m})\, M}_{\mathrm{x}} (\nu\, n)\, P'}\; n \notin \mathrm{n}(M)$$

$$\frac{P \xrightarrow{(\nu\, \widetilde{m})\, M}_{\mathrm{x}} P'}{(\nu\, n)\, P \xrightarrow{(\nu\, n, \widetilde{m})\, M}_{\mathrm{x}} P'}\; n \in \mathrm{n}(M) \setminus \widetilde{m}$$

Whenever $P \xrightarrow{(\nu\, \widetilde{m})\, M}_{\mathrm{x}} P'$, then we fix a P' with this property, denoting it by $\mathrm{fx}(P)$.

Lemma 20 *Let P be a spi calculus agent. Then:*

1. i. *Whenever $P \xrightarrow{m(x)}_{\mathrm{cp}'} P'$, then $P \xrightarrow{m(x)} P'$.*
   ii. *Whenever $P \xrightarrow{\overline{m}\langle \mathrm{X}\rangle}_{\mathrm{cp}'} P'$, where $P' \xrightarrow{(\nu\, \widetilde{n})\, M}_{\mathrm{x}} \mathrm{fx}(P')$, then $P \xrightarrow{(\nu\, \widetilde{n})\, \overline{m}\langle M\rangle} \mathrm{fx}(P')$, where $P'[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, Q] \equiv \mathrm{fx}(P')[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, Q]$ for every closed abstractions $(\lambda\, x)\, Q$.*
   iii. *Whenever $P \xrightarrow{\tau}_{\mathrm{cp}'} P'$ then $P \xrightarrow{\tau}\equiv P'$.*
2. i. *Whenever $P \xrightarrow{m(x)} P'$, then $P \xrightarrow{m(x)}_{\mathrm{cp}'} P'$.*
   ii. *Whenever $P \xrightarrow{(\nu\, \widetilde{n})\, \overline{m}\langle M\rangle} P'$, then $P \xrightarrow{\overline{m}\langle \mathrm{X}\rangle} (\nu\, n)(\mathrm{X}M \mid P')$, where $P'[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, Q] \equiv (\nu\, n)(\mathrm{X}M \,|\, P')[\mathrm{X} :=_{\mathrm{ho}} (\lambda\, x)\, Q]$ for every closed abstractions $(\lambda\, x)\, Q$.*
   iii. *Whenever $P \xrightarrow{\tau}_{\mathrm{cp}'} P'$ then $P \xrightarrow{\tau}\sim_{\mathrm{cxt}} P'$.*

Proof. Straightforward transition induction. □

Lemma 21 *Let P and E be a spi calculus agent or experiment, respectively. Then P $\textsc{may}_{\mathrm{cp}'}$ E if and only if P $\textsc{may}_{\mathrm{cp}}$ E.*

Proof. Straightforward from Lemma 20. □

A.2.2 The π-Calculus Side

The extraction relation on π-calculus agents is defined as follows:

$$\mathrm{X}\widetilde{a} \xrightarrow{(\nu)\,\widetilde{a}}_{\mathrm{x}} \mathbf{0}$$

$$\frac{P \xrightarrow{(\nu\,\widetilde{a}')\,\widetilde{a}}_{\mathrm{x}} P'}{P \mid Q \xrightarrow{(\nu\,\widetilde{a}')\,\widetilde{a}}_{\mathrm{x}} P' \mid Q}\ \widetilde{a}' \cap \mathrm{fn}(Q) = \emptyset$$

$$\frac{P \xrightarrow{(\nu\,\widetilde{a}')\,\widetilde{a}}_{\mathrm{x}} P'}{(\nu\, b)\, P \xrightarrow{(\nu\,\widetilde{a}')\,\widetilde{a}}_{\mathrm{x}} (\nu\, b)\, P'}\ b \notin \widetilde{a}$$

$$\frac{P \xrightarrow{(\nu\,\widetilde{a}')\,\widetilde{a}}_{\mathrm{x}} P'}{(\nu\, b)\, P \xrightarrow{(\nu\, b, \widetilde{a}')\,\widetilde{a}}_{\mathrm{x}} P'}\ b \in \widetilde{a} \setminus \widetilde{a}'$$

Analogously to the situation in subsection A.2.2, whenever $P \xrightarrow{(\nu\,\widetilde{a}')\,\widetilde{a}}_{\mathrm{x}} P'$, then we fix a P' with this property, denoting it by $\mathrm{fx}(P)$.

Lemma 22 *Let P be a π-calculus agent. Then:*

1. i. *Whenever $P \xrightarrow{a(b)}_{\mathrm{cp}} P'$, then $P \xrightarrow{a(b)} P'$.*

   ii. *Whenever $P \xrightarrow{\overline{a}\langle \mathrm{X}\rangle}_{\mathrm{cp}} P'$, where $P' \xrightarrow{(\nu\,\widetilde{b}')\,\widetilde{b}}_{\mathrm{x}} \mathrm{fx}(P')$, then $P \xrightarrow{(\nu\,\widetilde{b}')\,\overline{a}\langle\widetilde{b}\rangle} \mathrm{fx}(P')$, where $P'[\mathrm{X} :=_{\mathrm{ho}} (\lambda\,\widetilde{c})\,Q] \equiv \mathrm{fx}(P')[\mathrm{X} :=_{\mathrm{ho}} (\lambda\,\widetilde{c})\,Q]$ for all closed abstractions $(\lambda\,\widetilde{c})\,Q$ with $|\widetilde{b}| = |\widetilde{c}|$.*

   iii. *Whenever $P \xrightarrow{\tau}_{\mathrm{cp}} P'$ then $P \xrightarrow{\tau}\equiv P'$.*

2. i. *Whenever $P \xrightarrow{a(b)} P'$, then $P \xrightarrow{a(b)}_{\mathrm{cp}} P'$.*

   ii. *Whenever $P \xrightarrow{(\nu\,\widetilde{b}')\,\overline{a}\langle\widetilde{b}\rangle} P'$, then $P \xrightarrow{\overline{a}\langle \mathrm{X}\rangle} (\nu\,\widetilde{b}')(\mathrm{X}\widetilde{b} \mid P')$, where $P'[\mathrm{X} :=_{\mathrm{ho}} (\lambda\,x)\,Q] \equiv (\nu\,\widetilde{b}')(\mathrm{X}\widetilde{b} \mid P')[\mathrm{X} :=_{\mathrm{ho}} (\lambda\,x)\,Q]$ for all closed abstractions $(\lambda\,x)\,Q$ with $|\widetilde{b}| = |\widetilde{c}|$.*

   iii. *Whenever $P \xrightarrow{\tau}_{\mathrm{cp}} P'$ then $P \xrightarrow{\tau}\sim_{\mathrm{cxt}} P'$.*

Proof. Straightforward transition induction. □

Lemma 23 *Let P and E be π-calculus agent or experiment, respectively. Then P $\textsc{may}$ E if and only if P $\textsc{may}_{\mathrm{cp}}$ Q.*

Proof. Straightforward from Lemma 22. □

A.3 Concluding the Proof of Theorem 2

Proof of Theorem 2. As a direct consequence of Lemmas 17, 19, 21 and 23. □

Recent technical reports from the Department of Information Technology

2003-045 Sven-Olof Nyström: *A Polyvariant Type Analysis for Erlang*

2003-046 Martin Karlsson: *A Power-Efficient Alternative to Highly Associative Caches*

2003-047 Jimmy Flink: *Simuleringsmotor för tågtrafik med stöd för experimentell konfiguration*

2003-048 Timour Katchaounov and Tore Risch: *Interface Capabilities for Query Processing in Peer Mediator Systems*

2003-049 Martin Nilsson: *A Parallel Shared Memory Implementation of the Fast Multipole Method for Electromagnetics*

2003-050 Alexandre David: *Hierarchical Modeling and Analysis of Timed Systems*

2003-051 Pavel Krcal and Wang Yi: *Decidable and Undecidable Problems in Schedulability Analysis Using Timed Automata*

2003-052 Magnus Svärd and Jan Nordström: *Well Posed Boundary Conditions for the Navier-Stokes Equations*

2003-053 Erik Bängtsson and Maya Neytcheva: *Approaches to Reduce the Computational Cost when Solving Linear Systems of Equations Arising in Boundary Element Method Discretizations*

2003-054 Martin Nilsson: *Stability of the Fast Multipole Method for Helmholtz Equation in Three Dimensions*

2003-055 Martin Nilsson: *Rapid Solution of Parameter-Dependent Linear Systems for Electromagnetic Problems in the Frequency Domain*

2003-056 Parosh Aziz Abdulla, Johann Deneux, Pritha Mahata, and Aletta Nylén: *Forward Reachability Analysis of Timed Petri Nets*

2003-057 Erik Berg: *Low-Overhead Spatial and Temporal Data Locality Analysis*

2003-058 Erik Berg: *StatCache: A Probabilistic Approach to Efficient and Accurate Data Locality Analysis*

2003-059 Jonas Persson and Lina von Sydow: *Pricing European Multi-asset Options Using a Space-time Adaptive FD-method*

2003-060 Pierre Flener: *Realism in Project-Based Software Engineering Courses: Rewards, Risks, and Recommendations*

2003-061 Lars Ferm and Per Lötstedt: *Space-Time Adaptive Solution of First Order PDEs*

2003-062 Emilio Tuosto, Björn Victor, and Kidane Yemane: *Polyadic History-Dependent Automata for the Fusion Calculus*

UPPSALA
UNIVERSITET

December 2003
ISSN 1404-3203
http://www.it.uu.se/