PSI-CALCULI: A FRAMEWORK FOR MOBILE PROCESSES WITH NOMINAL DATA AND LOGIC

JESPER BENGTSON, MAGNUS JOHANSSON, JOACHIM PARROW, AND BJÖRN VICTOR

Department of Information Technology, Uppsala University, Sweden
e-mail address: jesper.bengtson@it.uu.se

Department of Information Technology, Uppsala University, Sweden
e-mail address: magnus.johansson@it.uu.se

Department of Information Technology, Uppsala University, Sweden
e-mail address: joachim.parrow@it.uu.se

Department of Information Technology, Uppsala University, Sweden
e-mail address: bjorn.victor@it.uu.se

Abstract. A psi-calculus is an extension of the pi-calculus with nominal data types for data structures and for logical assertions and conditions. These can be transmitted between processes and their names can be statically scoped as in the standard pi-calculus. Psi-calculi can capture the same phenomena as other proposed extensions of the pi-calculus such as the applied pi-calculus, the spi-calculus, the fusion calculus, the concurrent constraint pi-calculus, and calculi with polyadic communication channels or pattern matching. Psi-calculi can be even more general, for example by allowing structured channels, higher-order formalisms such as the lambda calculus for data structures, and predicate logic for assertions.

We provide ample comparisons to related calculi and discuss a few significant applications. Our labelled operational semantics and definition of bisimulation is straightforward, without a structural congruence. We establish minimal requirements on the nominal data and logic in order to prove general algebraic properties of psi-calculi, all of which have been checked in the interactive theorem prover Isabelle. We are the first to formulate a truly compositional labelled operational semantics for calculi of this calibre. Expressiveness and therefore modelling convenience significantly exceeds that of other formalisms, while the purity of the semantics is on par with the original pi-calculus.

Received by the editors December 30, 2009.
1998 ACM Subject Classification: Currently unknown.
Key words and phrases: pi-calculus, nominal sets, bisimulation, operational semantics.

LOGICAL METHODS IN COMPUTER SCIENCE DOI:10.2168/LMCS-???

1. Introduction

The pi-calculus [MPW92] has a multitude of extensions where higher-level data structures and operations on them are given as primitive. To mention only two there are the spi-calculus by Abadi and Gordon [AG99] focusing on cryptographic primitives, and the applied pi-calculus of Abadi and Fournet [AF01] where agents can introduce statically scoped aliases of names for data, used e.g. to express how knowledge of an encryption is restricted. It is also parametrised by an arbitrary signature for expressing data and an equation system for expressing data equalities. The impact of these enriched calculi is considerable with hundreds of papers applying or developing the formalisms. As Abadi and Fournet rightly observe there is a tradeoff between "purity", meaning the simplicity and elegance of the original pi-calculus, and modelling convenience; expressing complicated schemes in the original pi-calculus can simply become too gruesome and error prone.

But the modelling convenience of many high-level primitives comes at a price. The theory of the formalism may instead become gruesome and error prone, and it can be difficult to assess the effects of modifications to it. For example, the semantics of the applied pi-calculus is defined using two levels of processes (pure and extended), an inductively defined reduction relation, an algebraically defined structural congruence, and (in order to achieve compositionality) a notion of a barb and an explicit quantification over contexts. The authors therefore propose a more tractable semantics based on inductively defined labelled transitions, but as we show in Section 3.1 below it turns out to be non-compositional and in fact does not agree with the reduction semantics. For other examples, the labelled semantics of the concurrent constraint pi-calculus by Buscemi and Montanari [BM08] turns out to be non-compositional, as we show in Section 3.2, and our own earlier attempt to extend the pi-calculus [JPVB08] where the requirements for the scope extension law to hold are incomplete (see Section 2.6). The fact that such mistakes can go unnoticed for years indicates the complexity of the proofs.

Our contribution in this paper is to define psi-calculi: a framework where a range of calculi can be formulated with a lean and symmetric semantics, and where proofs can be conducted using straightforward induction without resorting to a structural congruence or explicit quantification of contexts. We claim to be the first to formulate such truly compositional labelled operational semantics for calculi of this calibre. Psi-calculi accommodate not only the examples mentioned above, but also extensions such as the pi-calculus with polyadic synchronisation [CM03], fusion [WG05], and concurrent constraints [BM07].

The main idea is that a psi-calculus is obtained by extending the basic untyped pi-calculus with three parameters. The first is a set of data terms which can function as both communication channels and communicated objects. The second is a set of conditions, for use in conditional constructs such as **if** statements. The third is a set of assertions, used to express e.g. constraints or aliases, which can resolve the conditions. These sets need not be disjoint, and one of our main results is to identify minimal requirements on them. They turn out to be quite general and natural.

Psi-calculi go beyond our previous work on extended pi-calculi [JPVB08] since we allow arbitrary assertions (and not only declarations of aliases), and arbitrary conditions (and not only equality tests). Also, we base our exposition on nominal data types and these accommodate e.g. alpha-equivalence classes of terms with binders. For example, we can use a higher-order logic for assertions and conditions, and higher-order formalisms such as the lambda calculus for data terms and channels. Last but not least the formalisation

is leaner and more symmetric. Thus we get the best of two worlds: expressiveness and therefore modelling convenience significantly exceeds that of the applied pi-calculus, while the "purity" of the semantics is on par with the original pi-calculus.

The straightforward definitions make our proofs suitable for checking in a theorem prover. We have implemented our framework in Isabelle [NPW02] using its nominal data type package [Urb08], also known as Nominal Isabelle, and proved all results in Section 5 [BP09]. This gives us absolute certainty of general results for a large class of calculi — at least to the point of the current state of the art for machine checked proofs.

In the next section we give the basic definitions of the syntax and semantics of psi-calculi. In Section 3 we relate to other work and demonstrate the expressiveness by showing how a variety of calculi can be formulated. Section 4 contains more substantial examples on frequency hopping spread spectrum, multiple local services with a common global name, and cryptographic mechanisms including the Diffie-Hellman key agreement protocol. In Section 5 we introduce a notion of bisimilarity, establish the expected algebraic results about it, and demonstrate the proof of the most difficult parts. In Section 6 we discuss the full formalisation and implementation in Isabelle. Finally Section 7 concludes with ideas for further work.

2. Definitions

2.1. **Nominal data types.** We base psi-calculi on nominal data types. A reader unfamiliar with these need not fear: we shall provide what little background is needed and be generous with examples. A traditional data type can be built from a signature of constant symbols, functions symbols, etc. A nominal data type is more general, for example it can also contain binders and identify alpha-variants of terms. Formally a nominal data type is not required to be built in any particular way; the only requirements are related to the treatment of the atomic symbols called names as explained below.

As usual we assume a countably infinite set of atomic *names* $\mathcal{N}$ ranged over by $a, \ldots, z$. Intuitively, names will represent the symbols that can be statically scoped, and also represent symbols acting as variables in the sense that they can be subjected to substitution. A typed calculus would distinguish names of different kinds but our account will be untyped. A typing may certainly contribute to clarity of expressions but it is not necessary for our results.

A *nominal set* [Pit03, GP01] is a set equipped with *name swapping* functions written $(a\ b)$, for any names a, b. An intuition is that for any member X it holds that $(a\ b) \cdot X$ is X with a replaced by b and b replaced by a. Formally, a name swapping is any function satisfying certain natural axioms such as $(a\ b) \cdot ((a\ b) \cdot X) = X$. One main point of this is that even though we have not defined any particular syntax we can define what it means for a name to "occur" in an element: it is simply that it can be affected by swappings. The names occurring in this way in an element X constitute the *support* of X, written $\mathrm{n}(X)$. We write $a\#X$, pronounced "a is fresh for X", for $a \notin \mathrm{n}(X)$. In an inductively defined data type without binders we will have $a\#X$ if a does not occur syntactically in X. In for example the lambda calculus where alpha-equivalent terms are identified (i.e. the elements are alpha-equivalence classes of terms) the support corresponds to the free names. If A is a set of names we write $A\#X$ to mean $\forall a \in A\ .\ a\#X$.

We require all elements to have finite support, i.e., $\mathrm{n}(X)$ is finite for all X. It follows that for any X there are infinitely many a such that $a\#X$. Some elements will have empty support, a prime example is the identity function in the lambda calculus, or a term of a traditional data type not containing any names. Such elements satisfy *equivariance*, $(a\ b)\cdot X = X$ for all a, b. A function f is equivariant if $(a\ b)\cdot f(X) = f((a\ b)\cdot X)$ holds for all X, and similarly for functions and relations of any arity. Intuitively, this means that all names are treated equally.

A *nominal data type* is a nominal set together with a set of equivariant functions on it. In particular we require a substitution function, which intuitively substitutes elements for names. If X is an element of a data type, $\tilde{a}$ is a sequence of names without duplicates and $\tilde{Y}$ is an equally long sequence of elements, the *substitution* $X[\tilde{a} := \tilde{Y}]$ is an element of the same data type as X. In a traditional data type substitution can be thought of as replacing all occurrences of names $\tilde{a}$ by $\tilde{Y}$. In a calculus with binders it can be thought of as replacing the free names, alpha-converting any binders to avoid capture. Formally, we define substitution as any equivariant function satisfying the substitution laws, e.g. $z\#(X, \tilde{Y}) \Rightarrow z\#X[\tilde{a} := \tilde{Y}]$. The full list is given in Section 6.

The main point of using nominal data types is that we obtain a general framework, allowing many different instantiations. Our only requirements are on the notions of support, name swapping, and substitution. This corresponds precisely to the essential ingredients for data transmitted between agents. Since names can be statically scoped and data sent into and out of scope boundaries, it must be possible to discern exactly what names are contained in what data items, and this is just the role of the support. In case a data element intrudes a scope, the scoped name needs to be alpha converted to avoid clashes, and name swapping can achieve precisely this. When a term is received in a communication between agents it must replace all occurrences of the placeholder in the input construct, in other words, the placeholder is substituted by the term.

Since these are the only things we assume about data terms we can handle data types that are not inductively defined, such as equivalences classes and sets defined by comprehension or co-induction. Examples include higher-order data types such as the lambda calculus, and even agents of a psi-calculus. As long as it satisfies the axioms of a nominal data type it can be used in our framework. Similarly, the notions of conditions, i.e., the tests on data that agents can perform during their execution, and assertions, i.e. the facts that can be used to resolve conditions, are formulated as nominal data types. This means that logics with binders and even higher-order logics can be used. Moreover, alpha-variants of terms can be formally equated by taking the quotient of terms under alpha equality, thereby facilitating the formalism and proofs.

2.2. **Terms, conditions, and assertions.** Formally, a psi-calculus is defined by instantiating three nominal data types and four operators:

Definition 1 (Psi-calculus parameters)**.** A psi-calculus requires the three (not necessarily disjoint) nominal data types:

T	the (data) terms, ranged over by M, N
C	the conditions, ranged over by φ
A	the assertions, ranged over by Ψ

and the four equivariant operators:

$\dot{\leftrightarrow}$: $\mathbf{T} \times \mathbf{T} \to \mathbf{C}$	Channel Equivalence
$\otimes : \mathbf{A} \times \mathbf{A} \to \mathbf{A}$	Composition
$\mathbf{1} : \mathbf{A}$	Unit
$\vdash \subseteq \mathbf{A} \times \mathbf{C}$	Entailment

The binary functions above will be written in infix. Thus, if M and N are terms then $M \mathrel{\dot{\leftrightarrow}} N$ is a condition, pronounced "M and N are channel equivalent" and if Ψ and Ψ' are assertions then so is $\Psi \otimes \Psi'$. Also we write $\Psi \vdash \varphi$, pronounced "Ψ entails φ", for $(\Psi, \varphi) \in \vdash$.

The data terms are used to represent all kinds of data, including communication channels. Intuitively, two agents can communicate if one sends and the other receives along the same channel. This is why we require a condition $M \mathrel{\dot{\leftrightarrow}} N$ to say that M and N represent the same communication channel. For example, in the pi-calculus $\dot{\leftrightarrow}$ would be just identity of names.

The assertions will be used to declare information necessary to resolve the conditions. Assertions can be contained in agents and represent constraints; they can contain names and thereby be syntactically scoped and represent information known only to the agents within that scope. The operator $\otimes$ on assertions will, intuitively, be used to represent conjunction of the information in the assertions. The assertion $\mathbf{1}$ is the unit for $\otimes$.

The intuition of entailment is that $\Psi \vdash \varphi$ means that given the information in Ψ, it is possible to infer φ. We say that two assertions are equivalent if they entail the same conditions:

Definition 2 (assertion equivalence)**.** Two assertions are *equivalent*, written $\Psi \simeq \Psi'$, if for all φ we have that $\Psi \vdash \varphi \Leftrightarrow \Psi' \vdash \varphi$.

We can now formulate our requisites on valid psi-calculus parameters:

Definition 3 (Requisites on valid psi-calculus parameters)**.**

Channel Symmetry:	$\Psi \vdash M \mathrel{\dot{\leftrightarrow}} N \implies \Psi \vdash N \mathrel{\dot{\leftrightarrow}} M$
Channel Transitivity:	$\Psi \vdash M \mathrel{\dot{\leftrightarrow}} N \wedge \Psi \vdash N \mathrel{\dot{\leftrightarrow}} L \implies \Psi \vdash M \mathrel{\dot{\leftrightarrow}} L$
Compositionality:	$\Psi \simeq \Psi' \implies \Psi \otimes \Psi'' \simeq \Psi' \otimes \Psi''$
Identity:	$\Psi \otimes \mathbf{1} \simeq \Psi$
Associativity:	$(\Psi \otimes \Psi') \otimes \Psi'' \simeq \Psi \otimes (\Psi' \otimes \Psi'')$
Commutativity:	$\Psi \otimes \Psi' \simeq \Psi' \otimes \Psi$

Our requisites on a psi-calculus is that the channel equivalence is a partial equivalence relation, that $\otimes$ is compositional, and that the equivalence classes of assertions form an abelian monoid. These requisites turn out to be strictly minimal for our results in Section 5 to hold. Note that channel equivalence is not required to be reflexive. Thus it is possible to have data terms that are not channel equivalent to anything at all, meaning that they cannot be used as channels.

As an example, we can choose data terms inductively generated by some signature, assertions and conditions to be elements of a first-order logic with equality over these terms, entailment to be logical implication, $\otimes$ to be conjunction and $\mathbf{1}$ to be TRUE. To verify that this indeed is a psi-calculus one needs to check the requirements on a nominal data type (meaning the notions of swapping, support and substitution are defined and fulfills the required properties), and that the requisites in Definition 3 hold.

2.3. **Frames.** Assertions can contain information about names, and names can be scoped using the familiar pi-calculus operator ν. For example, in a cryptography application an assertion Ψ could be that the a datum represents the encoding of a message using a key k. This Ψ can occur under the scope of νk, to signify that the key is known only locally. In order to admit this in a general way we use the notion of a frame, first introduced by Abadi and Fournet [AF01]. Basically, a frame is just an assertion with additional information about which names are scoped. The example above where Ψ occurs under the scope of k will be written $(\nu k)\Psi$, to signify a frame consisting of the assertion Ψ where the name k is local.

In the following $\tilde{a}$ means a finite (possibly empty) sequence of names, $a_1, \ldots, a_n$. The empty sequence is written ϵ and the concatenation of $\tilde{a}$ and $\tilde{b}$ is written $\tilde{a}\tilde{b}$. When occurring as an operand of a set operator, $\tilde{a}$ means the corresponding set of names $\{a_1, \ldots, a_n\}$. We also use sequences of terms, conditions, assertions etc. in the same way.

Definition 4 (Frame)**.** A *frame* is of the form $(\nu\widetilde{b})\Psi$ where $\widetilde{b}$ is a sequence of names that bind into the assertion Ψ. We identify alpha variants of frames. [1]

We use F, G to range over frames. Since we identify alpha variants we can always choose the bound names freely.

Notational conventions: We write just Ψ for $(\nu\epsilon)\Psi$ when there is no risk of confusing a frame with an assertion, and $\otimes$ to mean composition on frames defined by $(\nu\widetilde{b}_1)\Psi_1 \otimes (\nu\widetilde{b}_2)\Psi_2 = (\nu\widetilde{b}_1\widetilde{b}_2)\Psi_1 \otimes \Psi_2$ where $\widetilde{b}_1 \# \widetilde{b}_2, \Psi_2$ and vice versa. We write $(\nu c)((\nu\widetilde{b})\Psi)$ to mean $(\nu c\widetilde{b})\Psi$.

Intuitively a condition is entailed by a frame if it is entailed by the assertion and does not contain any names bound by the frame. Two frames are equivalent if they entail the same conditions:

Definition 5 (Equivalence of frames)**.** We define $F \vdash \varphi$ to mean that there exists an alpha variant $(\nu\widetilde{b})\Psi$ of F such that $\widetilde{b}\#\varphi$ and $\Psi \vdash \varphi$. We also define $F \simeq G$ to mean that for all φ it holds that $F \vdash \varphi$ iff $G \vdash \varphi$.

Continuing the example of first-order logic with equality, assume that the term $\mathsf{enc}(M, k)$ represents the result of encoding message M with key k. Let Ψ be the assertion $C = \mathsf{enc}(M, k)$, stating that the ciphertext C is the result of encoding M by k. If an agent contains this assertion the environment of the agent will be able to use it to resolve tests on the data, in particular to infer that $C = \mathsf{enc}(M, k)$. In other words, if the environment receives C it can test if this is the encryption of M. In order to restrict access to the key k it can be enclosed in a scope νk. The environment of the agent will then have access to the frame $(\nu k)\Psi$ rather than Ψ itself. This frame is much less informative, for example it does *not* hold that $(\nu k)\Psi \vdash C = \mathsf{enc}(M, k)$. Here great care has to be made to formulate the class of allowed conditions. If these only contain equivalence tests of terms, $(\nu k)\Psi$ will entail nothing but tautologies and be equivalent to $\mathbf{1}$. But if quantifiers are allowed in the conditions, then by existential introduction $\Psi \vdash \exists k.C = \mathsf{enc}(M, k)$, and since this condition has no free k we get $(\nu k)\Psi \vdash \exists k.C = \mathsf{enc}(M, k)$. In other words the environment will learn that C is the encryption of M for some key k. We shall return to examples related to cryptography in Section 3.1.

[1] In some presentations frames have been written just as pairs $\langle\widetilde{b}, \Psi\rangle$. The notation in this paper better conveys the idea that the names bind into the assertion, at the slight risk of confusing frames with agents. Formally, we establish frames and agents as separate types, although a valid intuition is to regard a frame as a special kind of agent, containg only scoping and assertions. This is the view taken in [AF01].

Most of the properties of assertions carry over to frames. Channel symmetry and channel transitivity, identity, associativity and commutativity all hold, but compositionality in general does not. In other words, there are psi-instances with frames F, G, H where $F \simeq G$ but not $F \otimes H \simeq G \otimes H$. An example is if there are assertions Ψ, Ψ' and Ψ_a for all names a, conditions φ' and φ_a for all names a, and where the entailment relation satisfies $\Psi_a \vdash \varphi_a$ and $\Psi' \vdash \varphi'$. Suppose composition is defined such that $\Psi \otimes \Psi = \Psi$ and all other compositions yield Ψ'. By adding a unit element this satisfies all requirements on a psi-calculus. In particular $\otimes$ is trivially compositional because no two different assertions are equivalent. Also $(\nu a)\Psi_a \simeq \Psi$, but $\Psi \otimes (\nu a)\Psi_a \not\simeq \Psi \otimes \Psi$ since $\Psi \otimes \Psi_a = \Psi' \vdash \varphi'$.

2.4. **Agents.**

Definition 6 (psi-calculus agents)**.** Given valid psi-calculus parameters as in Definitions 1 and 3, the psi-calculus *agents*, ranged over by $P, Q, \ldots$, are of the following forms.

$\overline{M}\, N \,.\, P$	Output
$\underline{M}(\lambda \widetilde{x})N \,.\, P$	Input
case $\varphi_1 : P_1 \,[]\, \cdots \,[]\, \varphi_n : P_n$	Case
$(\nu a)P$	Restriction
$P \mid Q$	Parallel
$!P$	Replication
$(\!\mid \Psi \mid\!)$	Assertion

In the Input $\underline{M}(\lambda \widetilde{x})N.P$ we require that $\widetilde{x} \subseteq \mathrm{n}(N)$ is a sequence without duplicates, and the names $\widetilde{x}$ bind occurrences in both N and P. Restriction binds a in P. We identify alpha equivalent agents. An assertion is *guarded* if it is a subterm of an Input or Output. In a replication $!P$ there may be no unguarded assertions in P, and in **case** $\varphi_1 : P_1 \,[]\, \cdots \,[]\, \varphi_n : P_n$ there may be no unguarded assertion in any P_i.

In the Output and Input forms M is called the subject and N the object. Output and Input are similar to those in the pi-calculus, but arbitrary terms can function as both subjects and objects. In the input $\underline{M}(\lambda \widetilde{x})N.P$ the intuition is that there is a pattern matching where the pattern $(\lambda \widetilde{x})N$ can match any term obtained by instantiating $\widetilde{x}$, e.g., $\underline{M}(\lambda x, y)f(x, y).P$ can only communicate with an output $\overline{M}\, f(N_1, N_2)$ for some data terms N_1, N_2. This can be thought of as a generalisation of the polyadic pi-calculus where the patterns are just tuples of names. Another significant extension is that we allow arbitrary data terms also as communication channels. Thus it is possible to include functions that create channels.

The **case** construct as expected works by behaving as one of the P_i for which the corresponding φ_i is true. **case** $\varphi_1 : P_1 \,[]\, \cdots \,[]\, \varphi_n : P_n$ is sometimes abbreviated as **case** $\widetilde{\varphi} : \widetilde{P}$, or if $n = 1$ as **if** φ_1 **then** P_1. In psi-calculi where a condition $\top$ exists such that $\Psi \vdash \top$ for all Ψ we write $P + Q$ to mean **case** $\top : P \,[]\, \top : Q$.

Input subjects are underlined to facilitate parsing of complicated expressions; in simple cases we often omit the underline. In the traditional pi-calculus terms are just names and its input construct $a(x) \,.\, P$ can be represented as $\underline{a}(\lambda x)x.P$. In some of the examples to follow we shall use the simpler notation $a(x) \,.\, P$ for this input form.

Some further notational conventions: We define the inert agent $\mathbf{0}$ as $(\!\mid \mathbf{1} \mid\!)$, and in some examples we omit a trailing $\mathbf{0}$, writing just $\overline{M}\, N$ for $\overline{M}\, N \,.\, \mathbf{0}$. If the object of an Output is a long term we enclose it in brackets $\langle\, \rangle$ to make it easier to parse.

For a simple example, the pi-calculus [MPW92] can be represented as a psi-calculus where the only data terms are names, the only assertion is $\mathbf{1}$, and the conditions are equality tests on names. Formally:

$$\begin{array}{lcl} \mathbf{T} & = & \mathcal{N} \\ \mathbf{C} & = & \{a = b : a, b \in \mathbf{T}\} \cup \{a \leftrightarrow b : a, b \in \mathbf{T}\} \\ \mathbf{A} & = & \{\mathbf{1}\} \\ \otimes & = & \lambda\Psi_1, \Psi_2.\, \mathbf{1} \\ \vdash & = & \{(\mathbf{1}, a = a) : a \in \mathcal{N}\} \cup \{(\mathbf{1}, a \leftrightarrow a) : a \in \mathcal{N}\} \end{array}$$

Here we can let $\top$ be $a = a$ and can thus represent pi-calculus sum through **case**. We obtain the polyadic pi-calculus by adding the tupling symbols t_n for tuples of arity n to $\mathbf{T}$., i.e. $\mathbf{T} = \mathcal{N} \cup \{\mathrm{t}_n(a_1, \ldots, a_n) : \ a_1, \ldots, a_n \in \mathcal{N}\}$. The polyadic output is to simply output the corresponding tuple of object names, and the polyadic input $a(b_1, \ldots, b_n) \,.\, P$ is represented by a pattern matching $\underline{a}(\lambda b_1, \ldots, b_n)\mathtt{t}_n(b_1, \ldots, b_n) \,.\, P$. Strictly speaking this allows tuples also in subject position in agents, but as we shall see such prefixes will not give rise to any transition, since in this psi-calculus $M \leftrightarrow M$ is only entailed when M is a name, i.e., only names are channels.

In a psi-calculus the channels can be arbitrary terms. This means that it is possible to introduce functions on channels (e.g., if M is a channel then so is $f(M)$). It also means that a channel can contain more than one name. An extension of this kind is explored by Carbone and Maffeis [CM03] in the so called pi-calculus with polyadic synchronisation, ${}^e\pi$. Here action subjects are tuples of names, and it is demonstrated that this allows a gradual enabling of communication by opening the scope of names in a subject, results in simple representations of localities and cryptography, and gives a strictly greater expressiveness than standard pi-calculus. We can represent ${}^e\pi$ by using tuples of names in subject position. The only modification to the representation of the polyadic pi-calculus is to extend $\vdash$ to $\vdash = \{(\mathbf{1}, M \leftrightarrow M) : \ M \in \mathbf{T}\}$, and to remove the conditions of type $M = N$ (since they can be encoded in ${}^e\pi$).

The data terms can be also be drawn from a higher-order formalisms. It is thus possible to transmit functions between agents. For example, let $\mathbf{T}$ be the lambda calculus, containing abstractions $\boldsymbol{\lambda}x.M$ and applications MN. In the parallel composition $\overline{a}\,\langle \boldsymbol{\lambda}x.M \rangle \,.\, P \mid a(z) \,.\, \overline{b}\,\langle zN \rangle \,.\, Q$ the left hand component transmits the function $\boldsymbol{\lambda}x.M$ to the right, where the application of it to N is transmitted along b. Reduction would be represented as a binary predicate over lambda terms and could be tested in psi-calculus conditions (the reduction rules would be part of the definition of entailment). In this sense psi can resemble a higher-order calculus. It is even possible to let the terms be the psi-calculus agents themselves. An agent transmitted as a term cannot directly communicate with the agent that sent or received it, but there is a possibility of indirect interaction through the entailment relation. This area we leave for further study.

2.5. **Operational semantics.** In this section we define an inductive transition relation on agents. In particular it establishes what transitions are possible from a parallel composition $P|Q$. In the standard pi-calculus the transitions from a parallel composition can be uniquely determined by the transitions from its components, but in psi-calculi the situation is more complex. Here the assertions contained in P can affect the conditions tested in Q and vice versa. For this reason we introduce the notion of the *frame of an agent* as the combination

of its top level assertions, retaining all the binders. It is precisely this that can affect a parallel agent.

Definition 7 (Frame of an agent)**.** The *frame* $\mathcal{F}(P)$ *of an agent* P is defined inductively as follows:

$$\begin{aligned}&\mathcal{F}(\underline{M}(\lambda\widetilde{x})N.P) = \mathcal{F}(\overline{M}\ N.P) = \\ &\qquad\qquad \mathcal{F}(\mathbf{case}\ \widetilde{\varphi} : \widetilde{P}) = \mathcal{F}(!P) = \mathbf{1}\\ &\mathcal{F}((\!|\Psi|\!)) = \Psi\\ &\mathcal{F}(P \mid Q) = \mathcal{F}(P) \otimes \mathcal{F}(Q)\\ &\mathcal{F}((\nu b)P) = (\nu b)\mathcal{F}(P)\end{aligned}$$

For a simple example, if $a\#\Psi_1$:

$$\mathcal{F}((\!|\Psi_1|\!) \mid (\nu a)((\!|\Psi_2|\!) \mid \overline{M}\ N.(\!|\Psi_3|\!)) \quad = \quad (\nu a)(\Psi_1 \otimes \Psi_2)$$

Here Ψ_3 occurs under a prefix and is therefore not on top level. An agent where all assertions are guarded thus has the frame $\mathbf{1}$. In the following we often write $(\nu\widetilde{b_P})\Psi_P$ for $\mathcal{F}(P)$, but note that this is not a unique representation since frames are identified up to alpha equivalence.

The actions α that agents can perform are of three kinds: output actions, input actions of the early kind, meaning that the input action contains the received object, and the silent action τ. The operational semantics will consist of transitions of the form $\Psi \rhd P \xrightarrow{\alpha} P'$. This transition intuitively means that P can perform an action α leading to P', in an environment that asserts Ψ.

Definition 8 (Actions)**.** The *actions* ranged over by α, β are of the following three kinds:

$\overline{M}\ (\nu\tilde{a})N$	Output, where $\tilde{a} \subseteq \mathrm{n}(N)$
$\underline{M}\ N$	Input
τ	Silent

For actions we refer to M as the *subject* and N as the *object*. We define $\mathrm{bn}(\overline{M}\ (\nu\tilde{a})N) = \tilde{a}$, and $\mathrm{bn}(\alpha) = \emptyset$ if α is an input or τ. We also define $\mathrm{n}(\tau) = \emptyset$ and $\mathrm{n}(\alpha) = \mathrm{n}(N) \cup \mathrm{n}(M)$ if α is an output or input. As in the pi-calculus, the output $\overline{M}\ (\nu\tilde{a})N$ represents an action sending N along M and opening the scopes of the names $\tilde{a}$. Note in particular that the support of this action includes $\tilde{a}$. Thus $\overline{M}\ (\nu a)a$ and $\overline{M}\ (\nu b)b$ are different actions.

Definition 9 (Transitions)**.** A *transition* is of the kind $\Psi \rhd P \xrightarrow{\alpha} P'$, meaning that when the environment contains the assertion Ψ the agent P can do an α to become P'. The transitions are defined inductively in Table 1. We write $P \xrightarrow{\alpha} P'$ to mean $\mathbf{1} \rhd P \xrightarrow{\alpha} P'$.

Both agents and frames are identified by alpha equivalence. This means that we can choose the bound names fresh in the premise of a rule. In a transition the names in $\mathrm{bn}(\alpha)$ count as binding into both the action object and the derivative, and transitions are identified up to alpha equivalence. This means that the bound names can be chosen fresh, substituting each occurrence in both the object and the derivative. This is the reason why $\mathrm{bn}(\alpha)$ is in the support of the output action: otherwise it could be alpha-converted in the action alone. Also, for the side conditions in Scope and Open it is important that $\mathrm{bn}(\alpha) \subseteq \mathrm{n}(\alpha)$. The freshness conditions on the involved frames will ensure that if a name is bound in one agent its representative in a frame is distinct from names in parallel agents, and also (in Par)

$$\textsc{In}\ \frac{\Psi \vdash M \leftrightarrow K}{\Psi \rhd \underline{M}(\lambda\widetilde{y})N.P \xrightarrow{\underline{K}\,N[\widetilde{y}:=\widetilde{L}]} P[\widetilde{y} := \widetilde{L}]} \qquad \textsc{Out}\ \frac{\Psi \vdash M \leftrightarrow K}{\Psi \rhd \overline{M}\,N.P \xrightarrow{\overline{K}\,N} P}$$

$$\textsc{Case}\ \frac{\Psi \rhd P_i \xrightarrow{\alpha} P' \qquad \Psi \vdash \varphi_i}{\Psi \rhd \mathbf{case}\ \widetilde{\varphi} : \widetilde{P} \xrightarrow{\alpha} P'}$$

$$\textsc{Com}\ \frac{\Psi_Q \otimes \Psi \rhd P \xrightarrow{\overline{M}\,(\nu\widetilde{a})N} P' \qquad \Psi_P \otimes \Psi \rhd Q \xrightarrow{\underline{K}\,N} Q' \qquad \Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K}{\Psi \rhd P \mid Q \xrightarrow{\tau} (\nu\widetilde{a})(P' \mid Q')}\ \widetilde{a}\#Q$$

$$\textsc{Par}\ \frac{\Psi_Q \otimes \Psi \rhd P \xrightarrow{\alpha} P'}{\Psi \rhd P|Q \xrightarrow{\alpha} P'|Q}\ \mathrm{bn}(\alpha)\#Q \qquad \textsc{Scope}\ \frac{\Psi \rhd P \xrightarrow{\alpha} P'}{\Psi \rhd (\nu b)P \xrightarrow{\alpha} (\nu b)P'}\ b\#\alpha, \Psi$$

$$\textsc{Open}\ \frac{\Psi \rhd P \xrightarrow{\overline{M}\,(\nu\widetilde{a})N} P'}{\Psi \rhd (\nu b)P \xrightarrow{\overline{M}\,(\nu\widetilde{a}\cup\{b\})N} P'}\ \begin{array}{l} b\#\widetilde{a}, \Psi, M \\ b \in \mathrm{n}(N) \end{array} \qquad \textsc{Rep}\ \frac{\Psi \rhd P \mid !P \xrightarrow{\alpha} P'}{\Psi \rhd !P \xrightarrow{\alpha} P'}$$

Table 1: Operational semantics. Symmetric versions of Com and Par are elided. In the rule Com we assume that $\mathcal{F}(P) = (\nu\widetilde{b}_P)\Psi_P$ and $\mathcal{F}(Q) = (\nu\widetilde{b}_Q)\Psi_Q$ where $\widetilde{b}_P$ is fresh for all of $\Psi, \widetilde{b}_Q, Q, M$ and P, and that $\widetilde{b}_Q$ is correspondingly fresh. In the rule Par we assume that $\mathcal{F}(Q) = (\nu\widetilde{b}_Q)\Psi_Q$ where $\widetilde{b}_Q$ is fresh for Ψ, P and α. In Open the expression $\nu\tilde{a} \cup \{b\}$ means the sequence $\tilde{a}$ with b inserted anywhere.

that it does not occur on the transition label. We defer a more precise account of this to Section 6.

The environmental assertions $\Psi \rhd \cdots$ in Table 1 express the effect that the environment has on the agent: enabling conditions in Case, giving rise to action subjects in In and Out and enabling interactions in Com. Thus Ψ never changes between hypothesis and conclusion except for the parallel operator, where an agent is part of the environment for another agent. In a derivation tree for a transition, the assertion will therefore increase towards the leafs by application of Par and Com. If all environmental assertions are erased and channel equivalence replaced by identity we get the standard laws of the pi-calculus enriched with data structures.

In comparison to the applied pi-calculus and the concurrent constraint pi calculus one main novelty is the inclusion of environmental assertions in the rules. They are necessary to make our semantics compositional, i.e., the effect of the environment on an agent is wholly captured by the semantics. In contrast, the labelled transitions of the applied and the concurrent constraint pi-calculi must rely on an auxiliary structural congruence, containing axioms such as scope extension $(\nu a)(P \mid Q) \equiv (\nu a)P \mid Q$ if $a\#Q$. With our semantics such laws are derived rather than postulated. The advantage of our approach is that proofs of metatheoretical results such as compositionality are much simpler since there is only the one inductive definition of transitions.

2.6. **Illustrative examples.** For a simple example of a transition, suppose for an assertion Ψ and condition φ that $\Psi \vdash \varphi$. Assume that

$$\forall \Psi'.\Psi' \rhd Q \xrightarrow{\alpha} Q'$$

i.e., Q has an action α regardless of the environment. Then by the Case rule we get

$$\Psi \rhd \textbf{if } \varphi \textbf{ then } Q \xrightarrow{\alpha} Q'$$

i.e., **if** φ **then** Q has the same transition if the environment is Ψ. Since $\mathcal{F}(\llparenthesis \Psi \rrparenthesis) = \Psi$ and $\Psi \otimes \mathbf{1} = \Psi$ we get by Par that

$$\mathbf{1} \rhd \llparenthesis \Psi \rrparenthesis \mid \textbf{if } \varphi \textbf{ then } Q \xrightarrow{\alpha} \llparenthesis \Psi \rrparenthesis \mid Q'$$

Data terms may also represent communication channels and here the channel equivalence comes into play. For example, in a polyadic pi-calculus the terms include tuples and projection functions with the usual equalities, for example $\pi_1(\mathrm{t}_2(a,b)) = a$. If these terms can represent channels then they must represent the *same* channel, consequently we must have $\Psi \vdash \pi_1(\mathrm{t}_2(a,b)) \leftrightarrow a$ for all Ψ. As an example,

$$\overline{a}\, N \,.\, P \mid \underline{\pi_1(\mathrm{t}_2(a,b))}\,(y) \,.\, Q \xrightarrow{\tau} P \mid Q[y := N]$$

Agents such as $\underline{\pi_1(\mathrm{t}_2(a,b))}\,(y) \,.\, Q$ can arise naturally if tuples of channels are transmitted as objects. For example, an agent that receives a pair of channels along c and then inputs along the first of them is written $\underline{c}(x) \,.\, \underline{\pi_1(x)}(y) \,.\, Q$. When put in parallel with an agent that sends $\mathrm{t}_2(a,b)$ along c it will have a transition leading to the agent where x is substituted by $\mathrm{t}_2(a,b)$, i.e. $\underline{\pi_1(\mathrm{t}_2(a,b))}\,(y) \,.\, Q$.

The semantics makes no particular provision for an equality of terms in object position. Thus, the agents $\overline{c}\, a \,.\, P$ and $\overline{c}\, \pi_1(\mathrm{t}_2(a,b)) \,.\, P$ have different transitions, and correspond to sending out the unevaluated "texts" a and $\pi_1(\mathrm{t}_2(a,b))$ respectively. To represent agents which send evaluated "values" we can do as in the applied pi-calculus where assertions declare equivalence of terms and agents send freshly generated aliases, e.g.

$$(\nu z)(\overline{c}\, z \,.\, P \mid \llparenthesis z = \pi_1(\mathrm{t}_2(a,b)) \rrparenthesis)$$

This agent has the same transition as $(\nu z)(\overline{c}\, z \,.\, P \mid \llparenthesis z = a \rrparenthesis)$. Any agent receiving the z will not be able to distinguish if z is a or $\mathrm{t}_2(a,b))$ since these terms are equated by all assertions. Also, if a and b are scoped as in

$$(\nu a, b, z)(\overline{c}\, z \,.\, P \mid \llparenthesis z = \pi_1(\mathrm{t}_2(a,b)) \rrparenthesis)$$

then their scopes will *not* open as a consequence of the output. In the applied pi-calculus this is the only form of communication and it is not possible to directly transmit data structures containing channel names, like the name tuples of the polyadic pi-calculus above. In psi-calculi these communication possibilities can coexist.

The main technical issue in the semantics is the treatment of scoping, as illustrated by the following example where the terms are just names. The intuition is that there is a communication channel available to all agents, and agents can declare any name to represent it through an assertion. The assertions are thus sets of names, and any name occurring in the assertion can be used as the subject of an action. Any two names in the assertion are deemed channel equivalent. Formally,

$$\begin{array}{lcl}\mathbf{T} & = & \mathcal{N} \\ \mathbf{C} & = & \{a \leftrightarrows b : a, b \in \mathbf{T}\} \\ \mathbf{A} & = & \mathcal{P}_{\text{fin}}(\mathcal{N}) \\ \otimes & = & \cup \\ \mathbf{1} & = & \emptyset \\ \vdash & = & \{(\Psi, a \leftrightarrow b) : a, b \in \Psi\}\end{array}$$

Omitting the action and prefix objects we get

$$\{a, b\} \rhd \overline{a}\,.\,\mathbf{0} \xrightarrow{\overline{a}} \mathbf{0}$$

and also

$$\{a, b\} \rhd \overline{a}\,.\,\mathbf{0} \xrightarrow{\overline{b}} \mathbf{0}$$

By the Par rule we have

$$\emptyset \rhd \overline{a}\,.\,\mathbf{0} \mid (\!|\{a, b\}|\!) \xrightarrow{\overline{a}} \mathbf{0} \mid (\!|\{a, b\}|\!)$$

and

$$\emptyset \rhd \overline{a}\,.\,\mathbf{0} \mid (\!|\{a, b\}|\!) \xrightarrow{\overline{b}} \mathbf{0} \mid (\!|\{a, b\}|\!)$$

Applying a restriction we get

$$\emptyset \rhd (\nu a)(\overline{a}\,.\,\mathbf{0} \mid (\!|\{a, b\}|\!)) \xrightarrow{\overline{b}} (\nu a)(\mathbf{0} \mid (\!|\{a, b\}|\!))$$

but no corresponding action with subject a because of the side condition on Scope. Thus, a communication through Com can be inferred from

$$(\nu a)(\overline{a}\,.\,\mathbf{0} \mid (\!|\{a, b\}|\!)) \mid b\,.\,\mathbf{0}$$

but *not* from

$$(\nu a)(\overline{a}\,.\,\mathbf{0} \mid (\!|\{a, b\}|\!)) \mid a\,.\,\mathbf{0}$$

This instance of a psi-calculus also illustrates two features of the semantics: firstly that channel equivalence is used in all three rules In, Out and Com, and secondly that assertions rather than frames represent the environment. Both issues are related to the law of scope extension, i.e., if $a\#Q$ then the agents $(\nu a)(P \mid Q)$ and $((\nu a)P) \mid Q$ should have the same transitions. Elaborating the example above and noting that $\{a\} \cup \{b\} \vdash a \leftrightarrows b$, we get that

$$(\nu a, b)((\!|\{a\}|\!) \mid (\!|\{b\}|\!) \mid \overline{a}\,.\,\mathbf{0} \mid b\,.\,\mathbf{0})$$

has an internal communication. By scope extension this agent should have the same transitions as $P \mid Q$ where

$$P = (\nu a)((\!|\{a\}|\!) \mid \overline{a}\,.\,\mathbf{0}) \qquad Q = (\nu b)((\!|\{b\}|\!) \mid b\,.\,\mathbf{0})$$

Here P and Q are alpha equivalent, and $\mathcal{F}(P) = (\nu a)\{a\}$ and $\mathcal{F}(Q) = (\nu b)\{b\}$ represent the same frame. Since they will be composed below we choose different representatives for the bound names. A communication from $P \mid Q$ is inferred by Com and the premises

1. $\{a\} \rhd P \xrightarrow{\overline{b}} (\nu a)((\!|\{b\}|\!) \mid \mathbf{0})$
   (derived using $\{a\} \otimes \{b\} = \{a, b\} \vdash a \leftrightarrows b$ in Out)
2. $\{b\} \rhd Q \xrightarrow{a} (\nu a)((\!|\{a\}|\!) \mid \mathbf{0})$
   (derived using $\{b\} \otimes \{a\} = \{a, b\} \vdash a \leftrightarrows b$ in In)
3. $\{a\} \otimes \{b\} = \{a, b\} \vdash a \leftrightarrows b$

Note how the action subjects are derived by the assertions in both cases to not clash with the binders, and that channel equivalence is necessary in all three rules.

The same example demonstrates why transitions in Table 1 are defined with assertions and not frames, for whereas $\{a, b\} \vdash a \leftrightarrow b$ the corresponding result cannot be obtained from the frames of the agents. We have that $\mathcal{F}(Q) \otimes \{a\} = (\nu b)\{a, b\} \nvdash a \leftrightarrow b$, so that frame is not useful for deriving a transition from P. Our earlier attempt [JPVB08] erroneously used frames rather than assertions, and this means that scope extension does not hold unless a further condition is imposed on the entailment relation to eliminate this kind of example.

Finally, a more complicated variant of the example motivates the requisite that $\leftrightarrow$ must be transitive. Consider any psi-calculus where Ψ_1 and Ψ_2 such that $\Psi_1 \otimes \Psi_2 \vdash a \leftrightarrow b$ and $\Psi_1 \otimes \Psi_2 \vdash b \leftrightarrow c$. Then the agent

$$(\nu a, b, c)(\langle\!|\Psi_1|\!\rangle \mid \langle\!|\Psi_2|\!\rangle \mid \overline{a}\,.\,\mathbf{0} \mid c\,.\,\mathbf{0})$$

has an internal communication using b. If $c\#\Psi_1$ and $a, b\#\Psi_2$, by scope extension the agent should behave similarly as

$$(\nu a, b)(\langle\!|\Psi_1|\!\rangle \mid \overline{a}\,.\,\mathbf{0}) \mid (\nu c)(\langle\!|\Psi_2|\!\rangle \mid c\,.\,\mathbf{0})$$

Clearly the left hand component cannot act with a subject a or b, so the only possibility to derive an internal communication is that it can act with subject c given assertion Ψ_2, i.e. that

$$\Psi_2 \rhd (\nu a, b)(\langle\!|\Psi_1|\!\rangle \mid \overline{a}\,.\,\mathbf{0}) \xrightarrow{\overline{c}} \cdots$$

This requires that $\Psi_1 \otimes \Psi_2 \vdash a \leftrightarrow c$.

3. Expressiveness and related calculi

In this section we explore the expressiveness of psi-calculi, mainly in comparison to other process calculi. In Section 2 we have already seen how to represent the pi-calculus as a psi-calculus.

3.1. **Calculi for cryptography.** Psi-calculi can express a variety of cryptographic operations on data. The main idea was illustrated in Section 2.3, using assertions to define relations between ciphertext and plaintext. Here we make the description more precise. Let the assertion "$C = \mathsf{enc}(M, k)$" mean that encrypting the message M with the key k results in the ciphertext C, and let "$M = \mathsf{dec}(C, k)$" mean that decrypting C with key k yields M. Entailment contains equations relating encryption and decryption such as $\forall M, k.\, \mathsf{dec}(\mathsf{enc}(M, k), k) = M$. The point is that a secure key can be represented by a bound name: it is unguessable outside its scope. An example agent $\overline{a}C\,.\,(\nu k)(\langle\!|C = \mathsf{enc}(M, k)|\!\rangle \mid P)$ outputs a term C and asserts that it is the encryption of M using the bound k as key, without opening the scope of k. Therefore an agent receiving C can resolve the condition $\mathsf{dec}(C, k) = M$ only after receiving this k in a communication. Technically this is because of the freshness conditions in the Par rule in Table 1 where $\widetilde{b}_Q$ is assumed fresh for P: this means that to apply the rule, P cannot use any name bound in the frame of Q.

This closely resembles the situation in the applied pi-calculus [AF01]. By contrast, in the spi-calculus [AG99] encrypted messages such as $\mathsf{enc}(M, k)$ are transmitted directly. Consider an example spi-calculus process

$$P = (\nu k, m)\overline{a}\langle\mathsf{enc}(m, k)\rangle\,.\,P' \tag{3.1}$$

where $P' = b(x)\,.\,\mathbf{if}\ x = m\ \mathbf{then}\ \overline{c}$. Here P sends a fresh name m encrypted with a fresh key k to the environment, and then receives a value x. Assuming perfect encryption, the environment cannot know m or k, so P' cannot receive m along b, and the output on c will never be possible. However, in the spi-calculus the transition $P \xrightarrow{(\nu k,m)\overline{a}\langle \mathsf{enc}(m,k)\rangle} P'$ opens the scopes of k and m, so here scoping does not correspond to restriction of knowledge. A reasonable equivalence must explicitly keep track of which names are known, leading to several complex bisimulation definitions (see [BN05] for an overview).

The applied pi-calculus is parametrised by a signature Σ for data terms and an equational theory $\vdash_\Sigma$ over Σ, and more importantly introduces *active substitutions* $\{^M/_x\}$ of data terms for variables. These can be introduced by the inferred structural rule $(\nu x)(\{^M/_x\} \mid A) \equiv A[x := M]$. There are names a, b, c distinct from variables x, y, z where only variables can be substituted, and a simple type system to distinguish names and variables of channel type from other terms of base type. Only names of channel type can be used as communication channels. Structured data terms cannot be sent directly, instead an *alias* variable such as x must be used, and the term itself does not occur on the transition label. We have $P \equiv Q$ for P above in (3.1), where

$$Q = (\nu x, k, m)(\{^{\mathsf{enc}(m,k)}/_x\} \mid \overline{a}x\,.\,P') \tag{3.2}$$

Here $Q \xrightarrow{\overline{a}\,(\nu x)x} (\nu k, m)(\{^{\mathsf{enc}(m,k)}/_x\} \mid P')$ and only the alias of the encryption (its "value") appears on the label; the scope of k and m is not opened and in this sense they are still confidential to the environment. However, the labelled semantics does not allow sending structured data terms where the scope *should* be opened, such as a tuple of names in the polyadic pi-calculus.

The labelled semantics for applied pi turns out to be non-compositional. Consider the closed (extended) applied pi-calculus agents

$$A = (\nu a)(\{^a/_x\} \mid x.b.\mathbf{0}) \qquad B = (\nu a)(\{^a/_x\} \mid \mathbf{0}) \tag{3.3}$$

where we omit the objects of the prefixes. They have the same frame and no transitions, and are thus semantically equivalent. But a context can contain x and can therefore use the active substitution to communicate with A. Formally, let $R = \overline{x}.\mathbf{0}$ and $\Downarrow b$ the usual weak observation or barb. We have by scope extension that $A|R \equiv (\nu a)(\{^a/_x\} \mid x.b.\mathbf{0} \mid \overline{x}.0) \Downarrow b$, but it is not the case that $B|R \Downarrow b$. Therefore, *no* observational equivalence that is preserved by all contexts and satisfies scope extension can be captured by the labelled semantics. In this, Theorem 1 of [AF01] is incorrect; the labelled and observational equivalences do in fact not coincide, nor is labelled equivalence a congruence. This is relevant for other papers that use or develop the labelled semantics, e.g. [GLPT07, KR05, DKR07, CRZ07].

Possible fixes are to disallow aliases for channel names, to be satisfied with compositionality for closed contexts, or to allow variables in action subjects. The consequences are difficult to assess, and our proposed solution is to instead define a psi-calculus representing the corresponding applied pi-calculus as follows. Assertions are finite sets of active substitutions, $\otimes$ is union, and entailment deduces equality under $\vdash_\Sigma$ and application of all relevant active substitutions.

Requirements on the assertions in applied pi are that they can only contain one active substitution per variable, that the active substitutions are non-circular, that they do not occur under a replication etc. To stay as close to the applied pi-calculus as possible we inherit these restrictions and only consider agents that satisfy them. This allows us to write

$M(\Psi^*)$ for the term resulting from the fixpoint of the substitutions in Ψ. We write $v(M)$ for the free variables of M, and Chan for the set of names of channel type.

$$
\begin{array}{lcl}
\mathbf{T} & = & \text{the set of terms defined by } \Sigma \\
\mathbf{A} & = & \mathcal{P}_{\text{fin}}(\{\{^{M}/_{x}\} : M \in \mathbf{T}, x \text{ variable}\}) \\
\mathbf{C} & = & \{M = N, \neg(M = N), M \dot{\leftrightarrow} N : M, N \in \mathbf{T}\} \\
\mathbf{1} & = & \emptyset \\
\otimes & = & \cup \\
\Psi \vdash M = N \text{ if } & & \vdash_{\Sigma} M(\Psi^*) = N(\Psi^*) \\
\Psi \vdash \neg(M = N) \text{ if } & & v(M(\Psi^*)) \cup v(N(\Psi^*)) = \emptyset \\
& & \wedge \neg(\Psi \vdash M = N) \\
\Psi \vdash M \dot{\leftrightarrow} N \text{ if } & & \Psi \vdash M = N \wedge \exists c \in \mathsf{Chan} : \Psi \vdash M = c
\end{array}
$$

Terms, assertions and conditions are as for the applied pi-calculus except for the condition $\neg(M = N)$ which is needed to represent the **if** $M = N$ **then** P **else** Q construct of applied pi as **case** $M = N : P \,[]\, \neg(M = N) : Q$ in psi. As in applied pi, the terms compared for inequality need to be ground. Channel equivalence $M \dot{\leftrightarrow} N$ requires that there is a channel name equal to both M and N.

The resulting psi-calculus differs from the applied pi-calculus in some ways, the most important being that in psi, $\overline{a}M \,.\, P \xrightarrow{\overline{a}M} P$ where M is a structured term, corresponds to sending the cleartext of M directly. This is not possible in the applied pi-calculus. In order to transmit M in the applied pi-calculus the structural rule $(\nu x)(\{^{M}/_{x}\} \mid A) \equiv A[x := M]$ must be used and an alias x for M be sent. To send an alias in this way in psi it must be introduced explicitly, as in $(\!|\{^{M}/_{x}\}|\!) \mid \overline{a}x \,.\, P$, and this agent is *not* the same as $\overline{a}M.P$.

Therefore, although the agents P and Q above (in (3.1) and (3.2)) are the same in the applied pi-calculus, the psi counterparts of the agents are different. In psi, P in (3.1) represents an agent that emits the clear text "$\mathsf{enc}(m, k)$". Any agent that receives this will immediately learn both m and k, and any scope of k will be opened in the process. This kind of agent can only indirectly be represented in the applied pi-calculus, by sending the restricted names separately one at a time. In contrast, the psi counterpart of (3.2) is $Q = (\nu x, k, m)((\!|\{^{\mathsf{enc}(m,k)}/_{x}\}|\!) \mid \overline{a}x \,.\, P')$ and defines Q to emit an alias for $\mathsf{enc}(m, k)$. As in the applied pi-calculus since k is scoped a recipient will not learn m. If the same recipient later receives k, an alias u for the message m can be constructed as $(\!|\{^{\mathsf{dec}(x,k)}/_{u}\}|\!)$.

Similarly, the agents R_1 and R_2 below are different in the psi-calculus semantics, but the corresponding agents in applied pi are equivalent.

$$
\begin{array}{lcl}
R_1 & = & (\nu x, k, m)((\!|\{^{\mathsf{enc}(m,k)}/_{x}\}|\!) \mid \overline{a}x \,.\, \overline{a}x \,.\, P') \\
R_2 & = & (\nu x, k, m)((\!|\{^{\mathsf{enc}(m,k)}/_{x}\}|\!) \mid \overline{a}x \,.\, (\nu y)((\!|\{^{x}/_{y}\}|\!) \mid \overline{a}y \,.\, P'))
\end{array}
$$

In the applied pi-calculus, a new alias for a term can always be introduced "on-the-fly", and it is impossible to tell R_1 and R_2 apart – they are structurally equivalent. The psi-calculus approach gives the possibility to discern the two agents, similarly to how the same ciphertext bitstring sent twice can be identified even if the plaintext cannot be recovered. To avoid this, a new alias needs to be explicitly introduced for each transmission, mimicking a probabilistic crypto where different ciphertext bitstrings correspond to the same plaintext and key.

Thus in psi, communication objects can range from literal data terms to indirect references, giving the user of the calculus the possibility to choose the appropriate form.

Another difference between the calculi is illustrated by the agent A of the compositionality counterexample (3.3): Its counterpart P_A in psi is $(\nu a)(\{\!|\{^a\!/\!_x\}|\!\} \mid x.b.\mathbf{0}) \xrightarrow{x} (\nu a)(\{\!|\{^a\!/\!_x\}|\!\} \mid b.\mathbf{0})$ and is not equivalent to $(\nu a)(\{\!|\{^a\!/\!_x\}|\!\} \mid \mathbf{0})$; indeed also $P_A \mid \overline{x}.\mathbf{0} \xrightarrow{\tau}\xrightarrow{b}$ in our labelled semantics.

3.2. **Fusion and concurrent constraints.** In the *explicit fusion calculus* by Wischik and Gardner, pi-F [WG05], the names in object position are fused by a communication, as in $a\tilde{b}\,.\,P \mid \overline{a}\tilde{d}\,.\,Q \xrightarrow{\tau} \{\tilde{b} = \tilde{d}\} \mid P \mid Q$ where $\{\tilde{b} = \tilde{d}\}$ (for $\tilde{b}$ and $\tilde{d}$ of equal length) is a *fusion* which allows us to treat each $b_i \in \tilde{b}$ as equivalent to $d_i \in \tilde{d}$. The explicit fusion calculus has a simple formulation as a psi-calculus:

$$\begin{array}{lcl}
\mathbf{T} & = & \mathcal{N} \\
\mathbf{C} & = & \{a = b : a, b \in \mathbf{T}\} \cup \{a \dot{\leftrightarrow} b : a, b \in \mathbf{T}\} \\
\mathbf{A} & = & \{\, \{a_1 = b_1, \ldots, a_n = b_n\} : a_i \in \mathcal{N}, b_i \in \mathcal{N}\} \\
\otimes & = & \cup \\
\mathbf{1} & = & \emptyset \\
\Psi \vdash a = b \text{ if } a = b \in \textsc{eq}(\Psi) & & \\
\Psi \vdash a \dot{\leftrightarrow} b \text{ if } \Psi \vdash a = b & &
\end{array}$$

where $\textsc{eq}(\Psi)$ is the equivalence closure of Ψ (i.e. transitive, symmetric and reflexive closure). Thus terms are names, assertions are name fusions, and the entailment relation deduces equality between names based on fusion assertions treated as equivalence relations. We can represent the pi-F input $a\tilde{b}\,.\,P$ as $a(\tilde{c})\,.\,(\{\!|\{\tilde{b} = \tilde{c}\}|\!\} \mid P)$ where $\tilde{c}\#a\tilde{b}\,.\,P$. This is reminiscent of Merro's encoding of fusion into asynchronous pi-calculus, using equators [Mer98]. For example, the fusion calculus communications

$$ab\,.\,\overline{c}c\,.\,P \mid \overline{a}c\,.\,bd\,.\,Q \xrightarrow{\tau} \{b = c\} \mid \overline{c}c\,.\,P \mid bd\,.\,Q \xrightarrow{\tau} \{b = c\} \mid \{c = d\} \mid P \mid Q$$

is expressed as:

$$\begin{array}{ll}
 & a(e)\,.\,\big(\{\!|\{b = e\}|\!\} \mid \overline{c}c\,.\,P\big) \;\mid\; \overline{a}c\,.\,b(x)\,.\,\big(\{\!|\{x = d\}|\!\} \mid Q\big) \\
\xrightarrow{\tau} & \big(\{\!|\{b = e\}|\!\} \mid \overline{c}c\,.\,P\big)[e := c] \;\mid\; b(x)\,.\,\big(\{\!|\{x = d\}|\!\} \mid Q\big) \\
= & \{\!|\{b = c\}|\!\} \mid \overline{c}c\,.\,P \;\mid\; b(x)\,.\,\big(\{\!|\{x = d\}|\!\} \mid Q\big) \\
\xrightarrow{\tau} & \{\!|\{b = c\}|\!\} \mid P \;\mid\; \big(\{\!|\{x = d\}|\!\} \mid Q\big)[x := c] \\
= & \{\!|\{b = c\}|\!\} \mid P \;\mid\; \{\!|\{c = d\}|\!\} \mid Q
\end{array}$$

The *concurrent constraint pi-calculus* (CC-Pi) [BM08] can be seen as a (monadic) pi-F calculus with **ask** and **tell** statements, parametrised by a constraint system in the form of a *named c-semiring*. Such a constraint system contains names, name fusion constraints and a name hiding operator ν, and supports general constraint semirings, e.g. Herbrand constraints. The semantics is given by a structural congruence and a reduction relation. There is also a labelled operational semantics, but it is in fact not compositional. Consider the CC-Pi agents

$$P = (\nu b, x)(x = b \mid \overline{a}\,x\,.\,b\,.\,c) \qquad Q = (\nu b, x)(x = b \mid \overline{a}\,x)$$

(where insignificant objects are omitted). They have the same constraint store and the same transitions in all constraint contexts. However, they do not have the same transitions in all process contexts: a parallel context $R = a(y).\overline{y}$ tells the difference:

$$P \mid R \xrightarrow{\tau}\xrightarrow{\tau} (\nu b)(x = b \mid x = y \mid c) \xrightarrow{c}$$

while $Q \mid R$ of course has no such c transition. Thus Theorem 1 of [BM08] is incorrect: open bisimilarity is not a congruence.

The labelled semantics of CC-Pi has a curious asymmetry between the rule for prefixes and the rule for communication: in the first case, the constraint store cannot affect the label induced by the prefix, while in the communication case, the constraint store judges whether the subjects should be considered the same, enabling the communication. The psi-calculi have no such asymmetry: the assertions (corresponding to the store) allow the subject to be rewritten in the prefix rules and the subjects in COM are compared using the assertions (see Section 2.6 for a discussion). A possible fix for CC-Pi would involve allowing the store to rewrite terms, thus also subjects in prefixes [Bus09].

A psi-calculus corresponding to CC-Pi with semiring $\mathcal{C}$ extends the psi-calculus for pi-F as follows:

$$\begin{array}{rcl} \mathbf{T} = \mathbf{A} & = & \mathcal{C} \\ \mathbf{C} & = & \mathcal{C} \cup \{a \leftrightarrow b : a, b \in \mathcal{N}\} \\ \otimes & = & \text{The similarly notated operator } \otimes \text{ in CC-Pi} \\ \mathbf{1} & = & 1 \\ \Psi \vdash \varphi & \text{if} & \begin{cases} \Psi \preceq \varphi & \text{if } \Psi, \varphi \in \mathcal{C} \\ \Psi \vdash a = b \wedge a, b \in \mathcal{N} & \text{if } \varphi = a \leftrightarrow b \end{cases} \end{array}$$

Thus terms, conditions and assertions are defined by the carrier of the named c-semiring (which by definition includes names and name fusions); $\mathbf{1}$ is the unit element, $\otimes = \otimes$, entailment is the CC-Pi partial order $\preceq$ and in addition $\Psi \vdash a \leftrightarrow b$ if $\Psi \vdash a = b$ and both a and b are names. We extend a monadic version of the pi-F formulation above by representing **ask** $\varphi \,.\, P$ as **if** φ **then** P and **tell** $\Psi \,.\, P$ as $(\!|\Psi|\!) \mid P$. We avoid recursion (it can be encoded using replication).

There are a couple of ways in which this psi-calculus differs from CC-Pi, apart from the source of the non-compositionality mentioned above. Most prominently, in the semantics of CC-Pi the fusions resulting from communication are required to be consistent with the store (as defined by the constraint system). In contrast our semantics will allow transitions that lead to an inconsistent store. In general both of these approaches have been used in concurrent constraint systems. It appears not possible to integrate a consistency check in a psi-calculus communication without changing our COM-rule.

An example of how a CC-Pi communication is represented in psi:

In CC-Pi:

$$\begin{array}{rcl} P & = & \Psi \mid \overline{a}b \,.\, Q \mid cd \,.\, R \xrightarrow{\tau} \Psi \otimes (b = d) \mid Q \mid R \\ & & \quad \text{if } \Psi \vdash a = c \text{ and } \Psi \otimes (b = d) \text{ consistent} \end{array}$$

In psi:

$$\begin{array}{rcl} P & = & (\!|\Psi|\!) \mid \overline{a}\, b \,.\, Q \mid c(x) \,.\, ((\!|x = d|\!) \mid R) \\ & \xrightarrow{\tau} & (\!|\Psi|\!) \mid (\!|b = d|\!) \mid Q \mid R \qquad \text{if } \Psi \vdash a = c \end{array}$$

Psi-calculi go beyond most concurrent constraint systems in two ways. Firstly, we allow arbitrary logics, even higher-order ones. Secondly, we allow constraints and conditions to be data terms, which means an agent can transmit and receive these. For example, assume that $\mathtt{c}$ is a constraint and that $\mathtt{f}$ is a function from assertions to assertions. Then in the agent $\overline{a}\,\mathtt{c} \,.\, P \mid a(z) \,.\, ((\!|\mathtt{f}(z)|\!) \mid Q) \xrightarrow{\tau} P \mid (\!|\mathtt{f}(\mathtt{c})|\!) \mid Q$ the left hand agent sends a the constraint $\mathtt{c}$ to the right, and $\mathtt{f}$ is applied to it. Similarly, if $\mathtt{p}$ is a unary predicate, in the

agent $\overline{a}\ \mathtt{p}\,.\,P \mid a(z)\,.\,\mathbf{if}\ z(x)\ \mathbf{then}\ Q \xrightarrow{\tau} P \mid \mathbf{if}\ \mathtt{p}(x)\ \mathbf{then}\ Q$ the left hand agent sends the predicate to the right, which applies it to x.

4. Applications

In this section we will look at a few applications of psi-calculi, some of which have been described before in other formalisms, and some which are novel.

4.1. **Structured terms as channels.**

Calculi with channels that can carry complex data are common, but in most cases the terms that represent channels are very simple, usually only a single name. We here give some examples where they have structure, and thus may contain more than one name.

4.1.1. *Frequency hopping spread spectrum.*

Wireless communication over a constant radio frequency has a number of drawbacks. In a hostile environment a radio can be tuned in to the correct frequency and monitor the communication which is also vulnerable to jamming. A solution to these problems is to jump quickly between different frequencies in a scheme called frequency hopping spread spectrum (FHSS), first patented in 1942 [MA42]. To eavesdrop it would then be necessary to match both the order of the frequencies and the pace of switching. Jamming is also made more difficult since the available power would have to be distributed over many frequencies.

We will here show how this is modelled in a psi-calculus. It is assumed that the initiator of the communication and the receiver share an algorithm used to calculate the next frequency. The procedure starts by the initiator sending a communication request over some predetermined frequency. The receiver then sends a seed back to the initiator and both use it to calculate the sequence of frequencies to be used. Then the initiator synchronises over the first calculated frequency to verify that it got the right sequence. The communication then proceeds and both parties change frequencies accordingly.

We will now look at the psi-calculus used to model this frequency hopping algorithm. We let terms represent radio frequencies and use the unary function $\mathsf{nextFreq}(M)$ to represent the algorithm for calculating the next frequency, given the previous frequency M. This psi-calculus has no assertions other than unit.

$$\begin{array}{lcl}\mathbf{T} & = & \mathcal{N} \cup \{\mathsf{nextFreq}(M) : M \in \mathbf{T}\}\\ \mathbf{C} & = & \{M \leftrightarrows N : M, N \in \mathbf{T}\}\\ \mathbf{A} & = & \{\mathbf{1}\}\\ \otimes & = & \lambda\Psi_1, \Psi_2.\ \mathbf{1}\\ \mathbf{1} \vdash M \leftrightarrows M & & \end{array}$$

Let $X_{in,out}$ be an arbitrary agent that communicates with the environment via the channels *in* and *out*. This agent will be wrapped in contexts that will let it do FHSS in a transparent way: from the agent's point of view it will only communicate over the local channels *in* and *out*. The agent *FHSS* that implements frequency hopping looks like:

$$FHSS = !\,\underline{fh}(freq)\,.\,\begin{pmatrix}\underline{out}(y)\,.\,\overline{freq}\langle y\rangle\,.\,\overline{fh}\langle\mathsf{nextFreq}(freq)\rangle\\ +\\ \underline{freq}(y)\,.\,\overline{in}\langle y\rangle\,.\,\overline{fh}\langle\mathsf{nextFreq}(freq)\rangle\end{pmatrix}$$

This agent can be thought of as a function fh that will take a frequency and then either wait for something to be received from the local channel out to send over this frequency, or to receive something over this frequency and forward it to the local channel in. It will then calculate the next frequency and start over.

The behaviour when the agent $X_{in,out}$ acts as initiator is modeled as a context where the initiating sequence starts by sending a synchronisation message $sync$ over a predetermined control channel ctl, and then waits for a seed from that channel. It then starts the frequency hopping algorithm with the seed and sends a synchronisation message over the first frequency, and behaves as $X_{in,out}$. It is assumed that $seed\#X_{in,out}$.

$$I[X_{in,out}] = \overline{ctl}\langle sync\rangle \,.\, \underline{ctl}(seed) \,.\, \overline{fh}\langle seed\rangle \,.\, \overline{out}\langle sync\rangle \,.\, X_{in,out} \mid FHSS$$

The behaviour when the agent $X_{in,out}$ acts as a receiver is modeled similarily: the receiver listens to the control channel ctl and sends back a seed. Then it starts the frequency hopping algorithm with this seed and waits for a synchronisation message. The receiver then behaves as $X_{in,out}$. It is assumed that $x, seed, s\#X_{in,out}$.

$$R[X_{in,out}] = \underline{ctl}(s) \,.\, (\nu seed)\overline{ctl}\langle seed\rangle \,.\, \overline{fh}\langle seed\rangle \,.\, \underline{in}(x) \,.\, X_{in,out} \mid FHSS$$

The full system where $X_{in,out}$ may behave as either a receiver or initiator is then modeled as

$$FH[X_{in,out}] = (\nu fh, in, out)\,(I[X_{in,out}] + R[X_{in,out}])$$

where it is assumed that $fh\#X_{in,out}$.

Let us look at a few transitions of the receiver. First the receiver gets a request to do frequency hopping over the control channel:

$$\begin{aligned} &FH[X_{in,out}] \xrightarrow{\underline{ctl}\ sync} \\ &\quad (\nu fh, in, out)\left((\nu seed)\overline{ctl}\langle seed\rangle \,.\, \overline{fh}\langle seed\rangle \,.\, \underline{in}(x) \,.\, X_{in,out} \mid FHSS\right) \end{aligned}$$

It then sends the seed to the initiator and starts the frequency hopping using this seed:

$$\begin{aligned} &\xrightarrow{\overline{ctl}\ (\nu seed)\langle seed\rangle} (\nu fh, in, out)\left(\overline{fh}\langle seed\rangle \,.\, \underline{in}(x) \,.\, X_{in,out} \mid FHSS\right) \\ &\xrightarrow{\tau} (\nu fh, in, out)\left(\begin{array}{l} \underline{in}(x) \,.\, X_{in,out} \\ \mid \left(\begin{array}{l} \underline{out}(y) \,.\, \overline{seed}\langle y\rangle \,.\, \overline{fh}\langle \mathsf{nextFreq}(seed)\rangle \\ + \\ \underline{seed}(y) \,.\, \overline{in}\langle y\rangle \,.\, \overline{fh}\langle \mathsf{nextFreq}(seed)\rangle \end{array}\right) \mid FHSS \end{array}\right) \end{aligned}$$

At this point the initiator will send the $sync$ message:

$$\begin{aligned} &\xrightarrow{\underline{seed}\ sync} \\ &\quad (\nu fh, in, out)\left(\underline{in}(x) \,.\, X_{in,out} \mid \overline{in}\langle sync\rangle \,.\, \overline{fh}\langle \mathsf{nextFreq}(seed)\rangle \mid FHSS\right) \\ &\xrightarrow{\tau} (\nu fh, in, out)\left(X_{in,out} \mid \overline{fh}\langle \mathsf{nextFreq}(seed)\rangle \mid FHSS\right) \end{aligned}$$

After another τ-transition the agent is ready to communicate over the next frequency:

$$\xrightarrow{\tau} (\nu fh, in, out)\left(\begin{array}{l} X_{in,out} \\ \mid \left(\begin{array}{l} \underline{out}(y) \,.\, \overline{\mathsf{nextFreq}(seed)}\langle y\rangle \,.\, \overline{fh}\langle \mathsf{nextFreq}(\mathsf{nextFreq}(seed))\rangle \\ + \\ \underline{\mathsf{nextFreq}(seed)}(y) \,.\, \overline{in}\langle y\rangle \,.\, \overline{fh}\langle \mathsf{nextFreq}(\mathsf{nextFreq}(seed))\rangle \end{array}\right) \mid FHSS \end{array}\right)$$

This example could easily be made more complex by adding relevant error checking (e.g. the receiver could check that the synchronisation message is correct), but even in this form it illustrates the use of structured channels.

4.1.2. *Local services.* A common scenario is that different servers implement the same kind of functionality known under some globally known name. HTTP servers are examples of this where the service provided is normally available on IP port 80. Here the name of the service (port 80) is shared among the different servers. The general problem is that there is a service known under a global name, but available from servers with different names. This problem is treated in depth in [CS01] where the authors invent a new calculus for this purpose. Here we show how the same problem can be solved using an instance of psi-calculi.

The instance used is basically the same as for polyadic pi-calculus as presented in Section 2.4 augmented with terms of form $a@b$ and the entailment $\mathbf{1} \vdash a@b \dot{\leftrightarrow} a@b$, where a and b are names. This gives the possibility to scope a part of a channel term, e.g $(\nu b)(\overline{a@b}\, c.P)$, as in [CM03].

$$\begin{array}{lcl}
\mathbf{T} & = & \{a@b : a, b \in \mathcal{N}\} \cup \\
 & & \{\mathrm{t}_2(M, N) : M, N \in \mathbf{T}\} \cup \mathcal{N} \\
\mathbf{C} & = & \{M \dot{\leftrightarrow} N : M, N \in \mathbf{T}\} \\
\mathbf{A} & = & \{\mathbf{1}\} \\
\otimes & = & \lambda\Psi_1, \Psi_2.\, \mathbf{1} \\
\mathbf{1} \vdash a \dot{\leftrightarrow} a & & \forall a \in \mathcal{N} \\
\mathbf{1} \vdash a@b \dot{\leftrightarrow} a@b & & \forall a, b \in \mathcal{N}
\end{array}$$

The following example is adapted from [CS01]. Assume there are globally known names *finger* and *daytime* which refer to resources located at some server. Different servers have different local information, but this information is accessed through the same globally known names. This can be modelled as

$$\begin{array}{rcl}
Server & = & !\,\underline{server}(\mathrm{t}_2(service, replyc)) \,.\, (\nu a) \left(\begin{array}{ll} & \overline{service@a}\langle replyc\rangle \,.\, \mathbf{0} \\ | & Finger(a) \\ | & Daytime(a) \end{array} \right) \\
Finger(a) & = & \underline{finger@a}(replyc) \,.\, \overline{replyc}\langle UserList\rangle \,.\, \mathbf{0} \\
Daytime(a) & = & \underline{daytime@a}(replyc) \,.\, \overline{replyc}\langle Date\rangle \,.\, \mathbf{0}
\end{array}$$

where *UserList* and *Date* are some terms containing the requested information. The exact nature of these terms is unimportant for this example.

The server listens to incoming requests on channel *server* and receives two names. The first name is the requested service, and the second is the reply channel. It will then do an internal communication with the particular service daemon. There is no risk of interference since a locally scoped name is part of the service channel. The result of the request is then forwarded along the reply channel.

$$\begin{array}{rl}
Server & \xrightarrow{\underline{server}\,\mathrm{t}_2(finger, c)} Server \mid (\nu a) \left(\begin{array}{ll} & \overline{finger@a}\langle c\rangle \,.\, \mathbf{0} \\ | & Finger(a) \\ | & Daytime(a) \end{array} \right) \\
 & \xrightarrow{\tau} Server \mid (\nu a) \left(\; \mathbf{0} \mid \overline{c}\langle UserList\rangle \,.\, \mathbf{0} \mid Daytime(a) \; \right) \\
 & \xrightarrow{\overline{c}\langle UserList\rangle} Server \mid (\nu a) \left(\; \mathbf{0} \mid \mathbf{0} \mid Daytime(a) \; \right)
\end{array}$$

Since any transitions from $Daytime(a)$ are prevented by the restriction, the final derivative will behave like $Server$.

4.2. **Cryptography.** In this section we give a sequence of examples from cryptography, culminating with a model of the Diffie-Hellman key agreement protocol. Our exposition is quite similar to the applied pi-calculus as presented in [AF01], and we will use a psi-calculus that mimics this closely. The main point is that psi-calculi can express these cryptographic examples in an equally concise way, and within a leaner and more symmetric formalism.

The psi-calculus instance we use for the examples below can be seen as a simplification of the one in Section 3.1. To construct this psi-calculus we assume an inductively defined set of terms $\mathbf{T}$ using a signature Σ, and an equational theory $\vdash_\Sigma$ which let us infer equations $M = N$ where M and N are terms. Exactly how this theory works is unimportant for this presentation. We let an assertion in $\mathbf{A}$ be a finite set of equations between terms and names, and often elide the set brackets in agents, e.g. writing $(\!|M = x|\!)$ instead of $(\!|\{M = x\}|\!)$. In such an assertion we call x an *alias* of M. The conditions are just equations $M = N$ and channel equivalences $M \leftrightarrows N$. Entailment is defined such that $\Psi \vdash M = N$ and $\Psi \vdash M \leftrightarrows N$ both hold if $M = N$ can be inferred from the equations in Ψ using the equational theory $\vdash_\Sigma$. The unit $\mathbf{1}$ is $\emptyset$ and composition $\otimes$ of assertions is union of sets.

We start by looking at how one-way hashing is modelled. The signature contains the unary symbol $\mathsf{hash}(x)$ which has no equations. The only equation on hash that is true is $\mathsf{hash}(M) = \mathsf{hash}(M)$, and this means that the hash function is collision free. The following example shows one agent sending a message M together with a hashing x of the message and a secret name s to another agent. The second agent will only forward M if it is properly hashed.

$$(\nu s)((\!|\mathsf{hash}(\mathsf{t}_2(s, M)) = x|\!) \mid \overline{a}\langle \mathsf{t}_2(M, x)\rangle \mid \underline{a}(y) \,.\, \mathbf{if}\ \mathsf{hash}(\mathsf{t}_2(s, \pi_1(y))) = \pi_2(y)\ \mathbf{then}\ \overline{b}\langle \pi_1(y)\rangle)$$

To model symmetric cryptography, the signature is extended as in Section 3.1: we add the binary symbols $\mathsf{enc}(x, y)$ and $\mathsf{dec}(x, y)$ together with the equation $\mathsf{dec}(\mathsf{enc}(x, y), y) = x$. The following agent sends a message M encrypted with the secret key k, without revealing the plaintext or key.

$$(\nu k, x)((\!|\mathsf{enc}(M, k) = x|\!) \mid \overline{a}\, x) \xrightarrow{\overline{a}\,(\nu x)x} \ldots$$

Asymmetric encryption is modelled by adding two new unary symbols $\mathsf{pk}(s)$ and $\mathsf{sk}(s)$ which generate the public and secret keys from a common seed value, and the equation $\mathsf{dec}(\mathsf{enc}(x, \mathsf{pk}(k)), \mathsf{sk}(k)) = x$. The following agent sends the public key on channel a, receives a message along channel b, decrypts it with the secret key, and sends the decrypted message along channel c:

$$(\nu s, x)((\!|\mathsf{pk}(s) = x|\!) \mid \overline{a}\, x \mid \underline{b}(y).((\!|\mathsf{dec}(y, \mathsf{sk}(s)) = z|\!)) \mid \overline{c}\, z)$$

Non-deterministic crypto is modelled by using a ternary version of the symbol $\mathsf{enc}(x, y, z)$ where the third argument is some salt, together with the equation $\mathsf{dec}(\mathsf{enc}(x, \mathsf{pk}(k), z), \mathsf{sk}(k)) = x$. Consider the following agent:

$$\underline{a}(x).\big((\nu m, y)((\!|\mathsf{enc}(M, x, m) = y|\!) \mid \overline{b}\, y) \mid (\nu n, z)((\!|\mathsf{enc}(M, x, n) = z|\!)) \mid \overline{c}\, z\big)$$

An observer of this agent cannot tell whether y and z are encryptions of the same message or not, because of the unique salt.

Digital signatures are modelled by adding the binary symbol $\mathsf{sign}(x, y)$, the ternary symbol $\mathsf{check}(x, y, z)$, the constant symbol ok, and the equation $\mathsf{check}(x, \mathsf{sign}(x, \mathsf{sk}(k)), \mathsf{pk}(k)) =$

ok. The following agent sends a signed message along a, then the parallel component receives it and checks the signature. If it is ok it is then forwarded.

$$(\nu s, z)((\!|\mathsf{pk}(s) = y|\!) \mid (\!|\mathsf{sign}(M, \mathsf{sk}(s)) = z|\!) \mid \overline{a}\,\mathsf{t}_2(M, z)) \\ \mid \underline{a}(x).\mathbf{if}\ \mathsf{check}(\pi_1(x), \pi_2(x), y) = \mathsf{ok}\ \mathbf{then}\ \overline{b}\,\pi_1(x)$$

The Diffie-Hellman protocol [DH76] is used to establish a shared secret between two principals who do not necessarily share any secrets beforehand. This is done by exchanging messages over a public channel.

We let Σ include $\mathsf{f}(x, y)$ and $\mathsf{g}(x)$, and the equation system includes $\mathsf{f}(x, \mathsf{g}(y)) = \mathsf{f}(y, \mathsf{g}(x))$, but no other equations on f and g. The first principal P creates a secret n_P and sends an alias x_P of $\mathsf{g}(n_P)$ to the other principal Q, and Q does likewise. Then P can create the term $\mathsf{f}(n_P, x_Q)$ and Q can create the term $\mathsf{f}(n_Q, x_P)$. Using the equations above these two terms are equivalent and the shared secret has been established. Concretely f and g are functions in a multiplicative group modulo a large prime, but here we ignore the number theory.

Let P_{k_P} and Q_{k_Q} be two agents that will share a secret key and will use the names k_P and k_Q, respectively, to refer to it. The Diffie-Hellman key agreement is modelled as two symmetric contexts $DH_{01}[\cdot]$ and $DH_{10}[\cdot]$ in which the agents are placed. The context $DH_{01}[X_k]$ is defined as

$$DH_{01}[X_k] = (\nu n, x, a_{01}, a_{10})((\!|\mathsf{g}(n) = x|\!) \mid \overline{a_{01}}\,x \mid \underline{a_{10}}(z).(\nu k)((\!|\mathsf{f}(n, z) = k|\!) \mid X_k))$$

where $n, x, a_{01}, a_{10} \# X_k$ and k occurs in X_k as a a name that refers to a key. The context $DH_{10}[X_k]$ is defined in the same way but with a_{10} and a_{01} swapped.

The agents P_{k_P} and Q_{k_Q} agree on the secret by placing them in the contexts: $DH_{01}[P_{k_P}]$ and $DH_{10}[Q_{k_Q}]$. The key agreement will then do two internal transitions:

$$DH_{01}[P_{k_P}] \mid DH_{10}[Q_{k_Q}] \xrightarrow{\tau} \xrightarrow{\tau} (\nu x_P, x_Q)(P' \mid Q')$$

where

$$\begin{aligned} P' &= (\nu n_P, a_{01}, a_{10})((\!|\mathsf{g}(n_P) = x_P|\!) \mid (\nu k_P)((\!|\mathsf{f}(n_P, x_Q) = k_P|\!) \mid P_{k_P})) \\ Q' &= (\nu n_Q, a_{01}, a_{10})((\!|\mathsf{g}(n_Q) = x_Q|\!) \mid (\nu k_Q)((\!|\mathsf{f}(n_Q, x_P) = k_Q|\!) \mid Q_{k_Q})) \end{aligned}$$

The x and n from the context have been alpha-converted to the variants with subscripts to avoid clashes.

Since the agents are communicating over a public channel the messages may be intercepted by a passive attacker which then forwards them unmodified. In presence of such an attacker the agents evolve to $P' \mid Q'$ where the lack of binders for x_P and x_Q represent that the hostile environment now has access to these values. We show that this does not break the protocol.

As a specification for this protocol we put $P_{k_P} \mid Q_{k_Q}$ in a context where they already share a secret, here represented by the name k': $S = (\nu k_P, k_Q, k')((\!|k' = k_P|\!) \mid (\!|k' = k_Q|\!) \mid P_{k_P} \mid Q_{k_Q})$. We then show that $P' \mid Q'$ and S behave the same, denoted $P' \mid Q' \sim S$. The precise meaning of $\sim$ is given in Section 5, but for this particular example it is sufficient to think of $\sim$ as equivalence of the frames of S and $P' \mid Q'$ according to Definition 5. This equivalence is closed under parallel composition (if P and Q behave the same, then so will $P \mid R$ and $Q \mid R$ for any agent R) and restriction (if P and Q behave the same, then so will $(\nu a)P$ and $(\nu a)Q$, for any a).

We have that

$$(\nu n_P, a_P)((\!|\mathsf{g}(n_P) = x_P|\!) \mid (\!|\mathsf{f}(n_P, x_Q) = k_P|\!)) \mid (\nu n_Q, a_Q)((\!|\mathsf{g}(n_Q) = x_Q|\!) \mid (\!|\mathsf{f}(n_Q, x_P) = k_Q|\!))$$
$$\sim$$
$$(\nu k')((\!|k' = k_P|\!) \mid (\!|k' = k_Q|\!))$$

The reason is that the only condition entailed on both sides is $k_P = k_Q$, no equalities can be entailed on x_P and x_Q. Since $\sim$ is closed under parallel composition we can add the agents:

$$(\nu n_P, a_P)((\!|\mathsf{g}(n_P) = x_P|\!) \mid (\!|\mathsf{f}(n_P, x_Q) = k_P|\!)) \mid (\nu n_Q, a_Q)((\!|\mathsf{g}(n_Q) = x_Q|\!) \mid (\!|\mathsf{f}(n_Q, x_P) = k_Q|\!)) \mid P_{k_P} \mid Q_{k_Q}$$
$$\sim$$
$$(\nu k')((\!|k' = k_P|\!) \mid (\!|k' = k_Q|\!)) \mid P_{k_P} \mid Q_{k_Q}$$

Since $\sim$ is closed under the restriction operator:

$$(\nu k_P, k_Q)((\nu n_P, a_P)((\!|\mathsf{g}(n_P) = x_P|\!) \mid (\!|\mathsf{f}(n_P, x_Q) = k_P|\!)) \mid (\nu n_Q, a_Q)((\!|\mathsf{g}(n_Q) = x_Q|\!) \mid (\!|\mathsf{f}(n_Q, x_P) = k_Q|\!)) \mid P_{k_P} \mid Q_{k_Q})$$
$$\sim$$
$$(\nu k_P, k_Q)((\nu k')((\!|k' = k_P|\!) \mid (\!|k' = k_Q|\!)) \mid P_{k_P} \mid Q_{k_Q})$$

Finally, by the structural laws of Theorem 16 in Section 5.2:

$$P' \mid Q' \sim S.$$

□

5. Bisimilarity

In this section we define a notion of strong bisimilarity on agents and prove that it satisfies the expected algebraic laws and substitutive properties. The results hold for any psi-calculus and gives us confidence in the semantic definitions.

5.1. **Definition.** In the standard pi-calculus the notion of strong bisimulation is used to formalise the intuition that two agents "behave in the same way"; it is defined as a symmetric binary relation $\mathcal{R}$ satisfying the simulation property: $\mathcal{R}(P, Q)$ implies that for α such that $\mathrm{bn}(\alpha)\#Q$,

$$P \xrightarrow{\alpha} P' \Longrightarrow Q \xrightarrow{\alpha} Q' \wedge \mathcal{R}(P', Q')$$

For a psi-calculus we in addition need to take the assertions into consideration. The behaviour of an agent is always taken with respect to an environmental assertion, and therefore a bisimulation must include the assertion to represent this. In other words, we get a ternary relation $\mathcal{R}(\Psi, P, Q)$, saying that P and Q behave in the same way when the environment asserts Ψ. Because of this two additional issues arise. The first is that the agents can affect their environment through their frames (and not only by performing actions), and this must be represented in the definition of bisimulation. The second is that the environment (represented by Ψ in $\mathcal{R}(\Psi, P, Q)$) can change, and for P and Q to be bisimilar they must

continue to be related after such changes. This leads to the following definition of strong bisimulation.

Definition 10 (Bisimulation)**.** A *bisimulation* $\mathcal{R}$ is a ternary relation between assertions and pairs of agents such that $\mathcal{R}(\Psi, P, Q)$ implies all of

(1) Static equivalence: $\Psi \otimes \mathcal{F}(P) \simeq \Psi \otimes \mathcal{F}(Q)$
(2) Symmetry: $\mathcal{R}(\Psi, Q, P)$
(3) Extension of arbitrary assertion:
$\forall \Psi'.\ \mathcal{R}(\Psi \otimes \Psi', P, Q)$
(4) Simulation: for all α, P' such that $\mathrm{bn}(\alpha) \# \Psi, Q$ there exists a Q' such that

$$\Psi \rhd P \xrightarrow{\alpha} P' \Longrightarrow \Psi \rhd Q \xrightarrow{\alpha} Q' \wedge \mathcal{R}(\Psi, P', Q')$$

We define $P \mathrel{\dot{\sim}_\Psi} Q$ to mean that there exists a bisimulation $\mathcal{R}$ such that $\mathcal{R}(\Psi, P, Q)$, and write $\dot{\sim}$ for $\dot{\sim}_{\mathbf{1}}$.

Clauses 2 and 4 are familiar from the pi-calculus. Clause 1 captures the fact that the related agents have exactly the same influence on the environment through their frames, namely that when they add to the existing environment (Ψ) then exactly the same conditions are entailed. Clause 3 means that when the environment changes (by adding a new assertion Ψ') the agents are still related. An example may clarify the role of this clause. Let β be a prefix and let φ be any non-trivial condition, and consider

$$P = \beta.\beta.\mathbf{0} + \beta.\mathbf{0} + \beta.\ \mathbf{if}\ \varphi\ \mathbf{then}\ \beta.\mathbf{0}$$
$$Q = \beta.\beta.\mathbf{0} + \beta.\mathbf{0}$$

P can nondeterministically choose between three branches and Q between the two first of them. Here P and Q are not bisimilar. If P performs an action corresponding to its third case, reaching the agent $P' = \mathbf{if}\ \varphi\ \mathbf{then}\ \beta.\mathbf{0}$, there is no way that Q can simulate since neither $Q' = \mathbf{0}$ nor $Q' = \beta.\mathbf{0}$ is equivalent to P' in all environments. In fact, any reasonable variant of bisimulation that equates P and Q will not be preserved by parallel. To see this, let T be $\gamma.(\!|\Psi|\!)$, where γ is any prefix and Ψ an assertion that entails φ. Then the transition $P \mid T \xrightarrow{\beta} P' \mid T$ cannot be simulated by $Q|T$, since $P'|T$ can only do an action γ followed by an action β, whereas $\beta.\mathbf{0}|T$ can do β immediately, and $\mathbf{0}|T$ can do no β at all. This demonstrates why clause 3, extension of arbitrary assertion, is necessary: it says that after each step all possible extensions of the assertion must be considered. If we would merely require this at top level, i.e. remove clause 3 and instead require $\forall \Psi.\mathcal{R}(\Psi, P, Q)$ in the definition of $P \mathrel{\dot{\sim}} Q$, the extensions would not recur; as a consequence P and Q in the example would be equivalent, and the equivalence would not be preserved by parallel.

For another example, consider

$$R = \mathbf{if}\ \varphi\ \mathbf{then}\ \beta\,.\,\mathbf{if}\ \varphi\ \mathbf{then}\ \beta.\mathbf{0} \qquad S = \mathbf{if}\ \varphi\ \mathbf{then}\ \beta.\beta.\mathbf{0}$$

In R the condition φ is checked twice. In general R and S are not equivalent. To see this, let Ψ and Ψ' be such that $\Psi \vdash \varphi$ and $\Psi \otimes \Psi' \nvdash \varphi$. We then have that $\Psi \rhd R \xrightarrow{\beta} \mathbf{if}\ \varphi\ \mathbf{then}\ \beta.\mathbf{0}$ and it cannot be simulated by $\Psi \rhd S \xrightarrow{\beta} \beta.\mathbf{0}$ because of the recurring clause of extension of arbitrary assertion: $\mathbf{if}\ \varphi\ \mathbf{then}\ \beta.\mathbf{0}$ has no transition in the environment $\Psi \otimes \Psi'$. However, if the entailment relation satisfies weakening, i.e. $\Psi \vdash \varphi \Rightarrow \Psi \otimes \Psi' \vdash \varphi$, we get the intuitive result that R and S are bisimilar. Weakening is a quite natural requirement, intuitively it says that no assertion can "undo" any entailments. This also demonstrates why we rejected

the smaller and simpler definition of $\dot{\sim}$ as the largest relation satisfying

$$\forall \Psi. \Psi \rhd P \xrightarrow{\beta} P' \Longrightarrow \Psi \rhd Q \xrightarrow{\beta} Q' \wedge P' \dot{\sim} Q'$$

The difference is that here bisimulation recurringly requires to hold for *all* assertions, not only for those that are extensions of the ones passed so far. This would have the unintuitive effect of making R and S in the example above non-bisimilar, even if weakening holds.

Interestingly, there is an alternative way to define bisimulation as a *binary* relation preserved by parallel contexts.

Definition 11 (Context bisimulation)**.** A *context bisimulation* $\mathcal{R}$ is a binary relation on agents such that $\mathcal{R}(P, Q)$ implies all of

(1) Static equivalence: $\mathcal{F}(P) \simeq \mathcal{F}(Q)$
(2) Symmetry: $\mathcal{R}(Q, P)$
(3) Extension of contextual assertion:
$\forall \Psi.\ \mathcal{R}((\!|\Psi|\!) \mid P,\ (\!|\Psi|\!) \mid Q)$
(4) Simulation: for all α, P' such that $\mathrm{bn}(\alpha)\#Q$ there exists a Q' such that

$$\mathbf{1} \rhd P \xrightarrow{\alpha} P' \quad \Longrightarrow \quad \mathbf{1} \rhd Q \xrightarrow{\alpha} Q' \wedge \mathcal{R}(P', Q')$$

We define $P \dot{\sim}^c Q$ to mean that there exists a context bisimulation $\mathcal{R}$ such that $\mathcal{R}(P, Q)$.

Such a definition is more in line with standard contextual bisimulations, and also the way bisimulation is defined in the applied pi-calculus. The drawback is that it relies on an operator in the calculus (parallel) for its definition. For conducting proofs our experience is that Definition 10 is preferable. We have shown that these bisimilarities coincide, i.e., the definitions result in the same bisimulation equivalence:

Theorem 12 (Bisimulation and context bisimulation coincide)**.** $\dot{\sim} = \dot{\sim}^c$

5.2. **Algebraic properties.** Our results are that bisimilarity is preserved by the operators in the expected way, and also satisfies the expected structural algebraic laws.

Theorem 13. For all Ψ:

(1) $P \dot{\sim}_\Psi Q \Longrightarrow P \mid R \dot{\sim}_\Psi Q \mid R$.
(2) $P \dot{\sim}_\Psi Q \Longrightarrow (\nu a)P \dot{\sim}_\Psi (\nu a)Q$.
(3) $P \dot{\sim}_\Psi Q \Longrightarrow !P \dot{\sim}_\Psi !Q$.
(4) $\forall i. P_i \dot{\sim}_\Psi Q_i \Longrightarrow \mathbf{case}\ \widetilde{\varphi} : \widetilde{P} \dot{\sim}_\Psi \mathbf{case}\ \widetilde{\varphi} : \widetilde{Q}$.
(5) $P \dot{\sim}_\Psi Q \Longrightarrow \overline{M}\, N.P \dot{\sim}_\Psi \overline{M}\, N.Q$.
(6) $(\forall \widetilde{L}.\ P[\widetilde{a} := \widetilde{L}] \dot{\sim}_\Psi Q[\widetilde{a} := \widetilde{L}]) \Longrightarrow$
$\underline{M}(\lambda \widetilde{a})N.P \dot{\sim}_\Psi \underline{M}(\lambda \widetilde{a})N.Q$

Definition 14. $P \sim_\Psi Q$ means that for all $\widetilde{x}, \widetilde{M}$ it holds $P[\widetilde{x} := \widetilde{M}] \dot{\sim}_\Psi Q[\widetilde{x} := \widetilde{M}]$, and we write $P \sim Q$ for $P \sim_{\mathbf{1}} Q$.

Theorem 15. $\sim_\Psi$ is a congruence for all Ψ.

Theorem 16.

$$\begin{array}{rcll}
P & \sim & P \mid \mathbf{0} & \\
P \mid (Q \mid R) & \sim & (P \mid Q) \mid R & \\
P \mid Q & \sim & Q \mid P & \\
(\nu a)\mathbf{0} & \sim & \mathbf{0} & \\
P \mid (\nu a)Q & \sim & (\nu a)(P \mid Q) & \text{if } a\#P \\
\overline{M}\,N.(\nu a)P & \sim & (\nu a)\overline{M}\,N.P & \text{if } a\#M, N \\
\underline{M}(\lambda\widetilde{x})N.(\nu a)P & \sim & (\nu a)\underline{M}(\lambda\widetilde{x})(N).P & \text{if } a\#\widetilde{x}, M, N \\
\mathbf{case}\ \widetilde{\varphi} : \widetilde{(\nu a)P} & \sim & (\nu a)\mathbf{case}\ \widetilde{\varphi} : \widetilde{P} & \text{if } a\#\widetilde{\varphi} \\
(\nu a)(\nu b)P & \sim & (\nu b)(\nu a)P & \\
!P & \sim & P \mid !P &
\end{array}$$

The most awkward part of the proofs is for Theorem 13(1), and historically this is the proof that most often fails in calculi of this complexity; the intricate correspondences between parallel processes and their assertions are hard to get completely right. We give an outline of the proof, and in detail cover the simulation case where the parallel processes communicate with each other.

We pick the candidate relation $\mathcal{R} = \{(\Psi, (\nu\widetilde{a})(P \mid R), (\nu\widetilde{a})(Q \mid R)) : P \mathrel{\dot{\sim}_{\Psi\otimes\Psi_R}} Q\}$ where $\widetilde{a}\#\Psi$, and prove that $\mathcal{R}$ is a bisimulation. Moreover we assume that $\widetilde{b_P}\#\widetilde{b_Q}, Q, \widetilde{b_R}, R, \Psi$, and $\widetilde{b_R}\#P, Q, \Psi$, or, in other words, that bound names are distinct from all free names and other bound names. Formally the proof is conducted by an induction on the length of $\widetilde{a}$. The induction steps are straightforward, so we focus on the base case. The agent $P \mid R$ can operate either by P or R doing individual actions, or by P and R communicating, where we cover the latter case, as it is the most involved.

In this case we have, by the Com rule, that P does an input transition ($\Psi\otimes\Psi_R \rhd P \xrightarrow{\underline{M}\,N} P'$), R does an output transition ($\Psi\otimes\Psi_P \rhd R \xrightarrow{\overline{K}\,(\nu\widetilde{a})N} R'$), and that the subjects of the transitions are channel equivalent ($\Psi\otimes\Psi_P\otimes\Psi_R \vdash M \leftrightarrow K$). The resulting communication between P and R is thus $\Psi \rhd P \mid R \xrightarrow{\tau} (\nu\widetilde{a})(P' \mid R')$.

To complete this step of the proof we need to find a Q' such that $\Psi \rhd Q \mid R \xrightarrow{\tau} (\nu\widetilde{a})(Q' \mid R')$, and $(\Psi,\ (\nu\widetilde{a})(P' \mid R'),\ (\nu\widetilde{a})(Q' \mid R')) \in \mathcal{R}$.

The presence of assertions in the transitions complicates the proof. We know that $P \mathrel{\dot{\sim}_{\Psi}} Q$, and hence by Definition 10(3) that $P \mathrel{\dot{\sim}_{\Psi\otimes\Psi_R}} Q$. Since $\Psi\otimes\Psi_R \rhd P \xrightarrow{\underline{M}\,N} P'$, we can obtain a Q' such that $\Psi\otimes\Psi_R \rhd Q \xrightarrow{\underline{M}\,N} Q'$ and $P' \mathrel{\dot{\sim}_{\Psi\otimes\Psi_R}} Q'$. However, this transition cannot communicate with $\Psi\otimes\Psi_P \rhd R \xrightarrow{\overline{K}\,(\nu\widetilde{a})N} R'$, since that transition is derived by the assertion $\Psi\otimes\Psi_P$, and not $\Psi\otimes\Psi_Q$. Moreover, M and K are channel equivalent by the assertion $\Psi\otimes\Psi_P\otimes\Psi_R$, and not $\Psi\otimes\Psi_Q\otimes\Psi_R$, which would be needed to derive the desired communication. In order to complete the proof, we need a lemma which switches the occurrences of Ψ_P to Ψ_Q in the transition of R, as well as in the channel equality.

Once a communication has been derived, we must prove that the corresponding derivatives $(\nu\widetilde{a})(P' \mid R')$, and $(\nu\widetilde{a})(Q' \mid R')$ are in the candidate relation $\mathcal{R}$. From the definition of $\mathcal{R}$ we get that this holds if $P' \mathrel{\dot{\sim}_{\Psi\otimes\Psi_{R'}}} Q'$, but we only know that $P' \mathrel{\dot{\sim}_{\Psi\otimes\Psi_R}} Q'$. In order to complete the proof, P' and Q' must be bisimilar in the assertion $\Psi\otimes\Psi_{R'}$, and not only in $\Psi\otimes\Psi_R$.

We provide lemmas which will address both of these obstacles in turn, after which this proof will be concluded. Lemma 19 simultaneously changes the assertion deriving the

transition for R, and the channel equality, and Lemmas 20 and 21 ensure that the derivatives of the communicating agents are in the candidate relation $\mathcal{R}$. Lemmas 17 and 18 are two generally applicable lemmas used to prove Lemma 19.

First, we define $\text{subj}(\overline{M}\,(\nu\widetilde{a})N) = M$ and similarly for input actions, and in all lemmas we tacitly assume $\mathcal{F}(P) = (\nu\widetilde{b}_P)\Psi_P$, where $\widetilde{b}_P\#P$, for any agent P.

The first lemma shows that given a finite set of names B that are fresh for P we can find a term M channel equivalent to the subject of an action from P whose names are fresh for B.

Lemma 17 (Find equivalent term)**.**

$$\begin{array}{rl} & B \subseteq \mathcal{N} \wedge B \text{ finite } \wedge B\#P \\ \wedge & \Psi \rhd P \xrightarrow{\alpha} P' \text{ where } \alpha \neq \tau \\ \wedge & \widetilde{b}_P\#\Psi, P, \text{subj}(\alpha), B \\ \Longrightarrow & \exists M \quad . \quad B\#M \\ & \qquad \wedge\, \Psi\otimes\Psi_P \vdash M \leftrightarrow \text{subj}(\alpha) \end{array}$$

Proof. A straightforward induction on the length of the derivation of the transition. In the base case we choose M as the prefix in the agent. □

The next lemma shows that given a transition we can find another transition whose subject is channel equivalent to the subject of the original transition and that leads to the same derivative as the original transition.

Lemma 18 (Rewrite subject)**.**

$$\begin{array}{rl} & \Psi \rhd P \xrightarrow{\overline{M}\,(\nu\widetilde{a})N} P' \\ \wedge & \Psi\otimes\Psi_P \vdash K \leftrightarrow M \\ \wedge & \widetilde{b}_P\#\Psi, P, K, M \\ \Longrightarrow & \Psi \rhd P \xrightarrow{\overline{K}\,(\nu\widetilde{a})N} P' \end{array}$$

The symmetric lemma where P does an input is omitted.

Proof. A straightforward induction on the length of the derivation of the transition. □

We can now prove the lemma which allows us to simultaneously switch the assertions deriving a transition, as well as channel equality in a communication. This lemma looks a bit intimidating and the proof details can safely be skipped at a first reading. It says that if P and Q are bisimilar and P can communicate with R via the channel K, then there exists a channel K' such that Q can communicate with R via K'.

Lemma 19 (Switching)**.**

$$\begin{array}{rl} & P \mathrel{\dot{\sim}}_{\Psi\otimes\Psi_R} Q \\ \wedge & \Psi\otimes\Psi_R \rhd P \xrightarrow{\underline{M}\,N} P' \\ \wedge & \Psi\otimes\Psi_P \rhd R \xrightarrow{\overline{K}\,(\nu\widetilde{a})N} R' \\ \wedge & \Psi\otimes\Psi_P\otimes\Psi_R \vdash K \mathrel{\dot{\leftrightarrow}} M \\ \wedge & \widetilde{b}_R\#\widetilde{b}_P, \widetilde{b}_Q, \Psi, P, Q, R, K \\ \wedge & \widetilde{b}_Q\#\Psi, R, P, Q, M \\ \wedge & \widetilde{b}_P\#R, M, \Psi \\ \Longrightarrow & \exists K'.\Psi\otimes\Psi_Q \rhd R \xrightarrow{\overline{K'}\,(\nu\widetilde{a})N} R' \\ \wedge & \Psi\otimes\Psi_Q\otimes\Psi_R \vdash K' \mathrel{\dot{\leftrightarrow}} M \\ \wedge & \widetilde{b}_R\#K' \end{array}$$

There is also a symmetric lemma where R does an input.

Proof. By induction on the length of the derivation of the transition from R. We only look at one base case and one induction step here. The other cases are similar.

Out: In this case $R = \overline{K_s}\,N.R'$ for some term K_s, and the transition is derived like this:

$$\textsc{Out}\ \frac{\Psi\otimes\Psi_P \vdash K_s \mathrel{\dot{\leftrightarrow}} K}{\Psi\otimes\Psi_P \rhd \overline{K_s}\,N.R' \xrightarrow{\overline{K}\,N} R'}$$

Since $\widetilde{b}_P\#\Psi, R$ we get that $\Psi\otimes\mathcal{F}(P) \vdash K_s \mathrel{\dot{\leftrightarrow}} K_s$. This in turn gives us that $\Psi\otimes\mathcal{F}(Q) \vdash K_s \mathrel{\dot{\leftrightarrow}} K_s$, which means that $\Psi\otimes\Psi_Q \vdash K_s \mathrel{\dot{\leftrightarrow}} K_s$. We then establish the first conjunct by:

$$\textsc{Out}\ \frac{\Psi\otimes\Psi_Q \vdash K_s \mathrel{\dot{\leftrightarrow}} K_s}{\Psi\otimes\Psi_Q \rhd \overline{K_s}\,N.R' \xrightarrow{\overline{K_s}\,N} R'}$$

For the second conjunct, we have that $\Psi\otimes\Psi_P \vdash K_s \mathrel{\dot{\leftrightarrow}} K$ and that $\Psi\otimes\Psi_P\otimes\mathbf{1} \vdash K \mathrel{\dot{\leftrightarrow}} M$ (since in this case Ψ_R is $\mathbf{1}$). Identity and transitivity then gives us that $\Psi\otimes\Psi_P \vdash K_s \mathrel{\dot{\leftrightarrow}} M$. Since $\widetilde{b}_P\#R, M$ we have that $\Psi\otimes\mathcal{F}(P) \vdash K_s \mathrel{\dot{\leftrightarrow}} M$ and since P and Q are bisimilar we also have that $\Psi\otimes\mathcal{F}(Q) \vdash K_s \mathrel{\dot{\leftrightarrow}} M$. We finally get $\Psi\otimes\Psi_Q \vdash K_s \mathrel{\dot{\leftrightarrow}} M$. The third conjunct is trivial since $\widetilde{b}_R$ is empty.

Scope: In this case $R = (\nu b)R'$ for some name b and the transition is derived like this:

$$\textsc{Scope}\ \frac{\Psi\otimes\Psi_P \rhd R' \xrightarrow{\overline{K}\,(\nu\widetilde{a})N} R''}{\Psi\otimes\Psi_P \rhd (\nu b)R' \xrightarrow{\overline{K}\,(\nu\widetilde{a})N} (\nu b)R''}\ b\#\overline{K}\,(\nu\widetilde{a})N, \Psi$$

Let $b\#\widetilde{b}_P, \widetilde{b}_Q, P, Q$. Note that by definition we have that $\Psi_{(\nu b)R'} = \Psi_{R'}$. We also have that $\widetilde{b}_{(\nu b)R'}\#\widetilde{b}_P, \widetilde{b}_Q, \Psi, P, Q, (\nu b)R', K \Longrightarrow \widetilde{b}_{R'}\#\widetilde{b}_P, \widetilde{b}_Q, \Psi, P, Q, R', K$ and

that $\widetilde{b}_{(\nu b)R'}\#\widetilde{b}_P, \widetilde{b}_Q \wedge \widetilde{b}_P, \widetilde{b}_Q\#(\nu b)R' \wedge b\#\widetilde{b}_P, \widetilde{b}_Q \Longrightarrow \widetilde{b}_P, \widetilde{b}_Q\#R'$. From the induction hypothesis we then get that $\Psi\otimes\Psi_Q \rhd R' \xrightarrow{\overline{K'}\,(\nu\widetilde{a})N} R''$, $\Psi\otimes\Psi_Q\otimes\Psi_{R'} \vdash M \leftrightarrows K'$, and that $\widetilde{b}_{R'}\#K'$.

From the fact that P and Q are bisimilar we get that $\Psi\otimes\Psi_{(\nu b)R'} \rhd Q \xrightarrow{\underline{M}\,N} Q'$. Let $B = \{b\} \cup \widetilde{b}_{R'}$. By Lemma 17 we learn that there exists a term K'' such that $\Psi\otimes\Psi_{(\nu b)R'}\otimes\Psi_Q \vdash K'' \leftrightarrow M$, fulfilling the second obligation, and that $B\#K''$. This gives us that $\widetilde{b}_{R'}, b\#K''$. By transitivity we then get that $\Psi\otimes\Psi_{(\nu b)R'}\otimes\Psi_Q \vdash K' \leftrightarrow K''$. We now use Lemma 18 to get that $\Psi\otimes\Psi_Q \rhd R' \xrightarrow{\overline{K''}\,(\nu\widetilde{a})N} R''$. Finally we do the following derivation:

$$\textsc{Scope}\ \frac{\Psi\otimes\Psi_Q \rhd R' \xrightarrow{\overline{K''}\,(\nu\widetilde{a})N} R''}{\Psi\otimes\Psi_Q \rhd (\nu b)R' \xrightarrow{\overline{K''}\,(\nu\widetilde{a})N} (\nu b)R''}\ b\#\overline{K''}\ (\nu\widetilde{a})N, \Psi$$

That $\widetilde{b}_{(\nu b)R'}\#K''$ follows from $B\#K''$.

□

The following lemma proves that any when an agent performs a transition, its frame is extended with a new assertion (Ψ' below):

Lemma 20. If $\Psi \rhd R \xrightarrow{\underline{M}\,N} R'$ and $\widetilde{b}_R\#R, N, C$ where C is a set of names, then $\exists\Psi', \widetilde{b}_{R'}, \Psi_{R'}$ such that $\mathcal{F}(R') = (\nu\widetilde{b}_{R'})\Psi_{R'} \wedge \Psi_R\otimes\Psi' \simeq \Psi_{R'} \wedge \widetilde{b}_{R'}\#C, R'$.

Proof. A straightforward induction on the length of the derivation of the transition. □

Finally, we need a lemma which allows us to switch the environment for a bisimulation for an equivalent one.

Lemma 21. If $\Psi \rhd P \mathrel{\dot\sim} Q$ and $\Psi \simeq \Psi'$ then $\Psi' \rhd P \mathrel{\dot\sim} Q$

Proof. The candidate relation for the bisimulation is $\mathcal{R} = \{(\Psi',\ P,\ Q) : \Psi \rhd P \mathrel{\dot\sim} Q \wedge \Psi \simeq \Psi'\}$. The four cases are proven separately.

Case 1: Follows from the fact that $\otimes$ is compositional, where the bound names of the frames of P and Q are alpha-converted not to clash with Ψ'.

Case 2: $\mathcal{S}$ is trivially symmetric, since $\mathrel{\dot\sim}$ and $\simeq$ are symmetric.

Case 3: Follows from the fact that $\otimes$ is compositional.

Case 4: From the definition of $\mathrel{\dot\sim}$ and the transition $\Psi \rhd P \xrightarrow{\alpha} P'$, we obtain a Q'. s.t. $\Psi \rhd Q \xrightarrow{\alpha} Q'$ and $\Psi \rhd P' \mathrel{\dot\sim} Q'$. By induction on the derivation of this transition, and the fact that $\Psi \simeq \Psi'$, we get that $\Psi' \rhd Q \xrightarrow{\alpha} Q'$. Moreover, since $\Psi \rhd P' \mathrel{\dot\sim} Q'$ and $\Psi \simeq \Psi'$ we have that $(\Psi',\ P',\ Q') \in \mathcal{S}$.

□

With these lemmas in place we complete the proof of Theorem 13(1) commenced at the beginning of this section. The case we are proving is when $P \mid R$ performs a communication. We must find a corresponding transition from $Q \mid R$ such that the derivatives remain in the candidate relation $\mathcal{R}$. The agents P and Q can communicate using the following derivation.

$$\textsc{Com}\ \frac{\Psi_R \otimes \Psi \rhd P \xrightarrow{\overline{M}\,(\nu \widetilde{a})N} P' \qquad \Psi_P \otimes \Psi \rhd R \xrightarrow{K\,N} R' \qquad \Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K}{\Psi \rhd P \mid R \xrightarrow{\tau} (\nu \widetilde{a})(P' \mid R')}\ \widetilde{a} \# R$$

Our goal is to replace P with Q in the premises so that we can derive the simulating transition. Let $\mathcal{F}(Q) = (\nu \widetilde{b}_Q)\Psi_Q$ be such that $\widetilde{b}_Q \# P, \widetilde{b}_R, R, \Psi, M$.

We use Lemma 19, to obtain $\Psi_Q \otimes \Psi \rhd R \xrightarrow{K'\,N} R'$ and $\Psi \otimes \Psi_Q \otimes \Psi_R \vdash M \leftrightarrow K'$. Since P and Q are bisimilar we have that $\Psi \otimes \Psi_R \rhd Q \xrightarrow{\overline{M}\,(\nu \widetilde{a})N} Q'$. We then derive the following:

$$\textsc{Com}\ \frac{\Psi_R \otimes \Psi \rhd Q \xrightarrow{\overline{M}\,(\nu \widetilde{a})N} Q' \qquad \Psi_Q \otimes \Psi \rhd R \xrightarrow{K'\,N} R' \qquad \Psi \otimes \Psi_Q \otimes \Psi_R \vdash M \leftrightarrow K'}{\Psi \rhd Q \mid R \xrightarrow{\tau} (\nu \widetilde{a})(Q' \mid R')}\ \widetilde{a} \# R$$

We know that $P' \mathrel{\dot{\sim}}_{\Psi \otimes \Psi_R} Q'$ and by clause 3 in the definition of bisimulation (extension of arbitrary assertion) that $P' \mathrel{\dot{\sim}}_{\Psi \otimes \Psi_R \otimes \Psi'} Q'$ for any Ψ'. By Lemma 20 we know that there exists a Ψ'' such that $\Psi_R \otimes \Psi'' \simeq \Psi_{R'}$, so in particular, using Lemma 21, we have that $P' \mathrel{\dot{\sim}}_{\Psi \otimes \Psi_{R'}} Q'$ We then conclude that $(\Psi, (\nu \widetilde{a})(P' \mid R'), (\nu \widetilde{a})(Q' \mid R')) \in \mathcal{R}$.

□

The proofs of theorems 12–16 follow a similar pattern, using induction over the lengths of the derivations of the transitions. The part we have just shown is the most challenging.

6. Formalisation in Isabelle

As the complexities of calculi increase, the proofs become complicated and therefore more error prone. In Section 3 we discussed how both the applied pi-calculus and the concurrent constraint pi-calculus have turned out to be non-compositional. This hints at the complexity of the proofs and the difficulty of getting them right. Our proofs for psi-calculi are also sometimes long and intricate. For example, the proof sketch of Theorem 13(1), described in the previous section, is substantially more complicated than its corresponding proof for the pi-calculus. However, we emphasize that the proof is not substantially different in structure: it is just a set of properties of transitions, all established by induction over the the definition of the semantics. In this, psi-calculi are simpler then many other calculi that rely on stratified definitions of the semantics with devices such as a structural congruence.

In order to ensure that proofs are correct, automated and interactive proof assistants or theorem provers can be used to formally verify the proofs with the aid of a computer. We have completely formalised all results in Section 5 in the interactive theorem prover Isabelle.

This is pioneering work. To the best of our knowledge, no calculus of this complexity has ever been formalised in a theorem prover. In previous work [BP07] we formalised a substantial part of the pi-calculus meta-theory in Isabelle. This section will cover the main additions to formalise the framework for psi-calculi. A more in depth exposition can be found in [BP09].

6.1. **Alpha-equivalence.** The main difficulty with formalising any process algebra in a theorem prover is to reason about alpha-equivalence in a convenient way. When conducting manual proofs on paper this notion is often glossed over, and statements such as "we assume any bound name under consideration to be sufficiently fresh" are commonplace. For machine checked proofs this poses a problem. Exactly what does it mean for a bound name to be sufficiently fresh?

We use Nominal Isabelle [Urb08] to formalise datatypes with binders, and to reason about them up to alpha-equivalence; in other words, all our proofs deal with alpha equivalence classes rather than with particular representatives. In the standard nominal way alpha variants of agents are identified , so e.g. $(\nu a)P = (\nu b)((a\ b) \cdot P)$, when $b\#P$, and similarly for names bound in the input construct. Formally, name swapping on agents distributes over all constructors, and substitution on agents avoids captures by binders through alpha-conversion in the usual way. In that way Nominal Isabelle provides an alpha-equivalence class of agents where the support of P is the union of the supports of the components of P, removing the bound names. This corresponds to the names with a free occurrence in P.

Frames contain binders and we reason about their alpha equivalence classes in the same way. Also, transitions contain binders. Consider the output transition $\Psi \rhd P \xrightarrow{\overline{M}\,(\nu\tilde{a})N} P'$. To be completely formal, as described in [BP07], $\tilde{a}$ is a binding occurrence with a scope that contains both N and P'. We accomplish this by creating a datatype containing both an action and the derivative process as follows.

Definition 22 (Residuals)**.** A *residual* with the action α and the derivative P', is written $\alpha \prec P'$.

Thus we have the following three forms of residuals:

$\overline{M}\,(\nu\tilde{x})N \prec P'$	Output
$\underline{M}\,N \prec P'$	Input
$\tau \prec P'$	Silent

In the Output residual, $\tilde{x}$ binds into both N and P'. In this way we get a nominal datatype of residuals where name swapping just distributes to its components and the support is the free names. A transition is then simply a pair consisting of an agent and a residual. Again, Nominal Isabelle allows us to reason about alpha equivalence classes of transitions. Typically a property of transitions is established by induction, with one case for each rule. This means that we assume the property of the premise of the rule, and must establish it for the conclusion. Since we work with alpha equivalence classes it is enough to establish the property for one representative of the alpha equivalence class. This formalises the principle that we may always pick bound names fresh.

Datatypes for agents, frames and transitions in Nominal Isabelle require sequences of binders, e.g. in the input prefix and in the output action. Nominal Isabelle only supports single binders, and we have therefore created infrastructure to reason about arbitrarily long binding sequences. It is important to reason about them as atomic objects, otherwise there

would be a constant need for inductive proofs of the length of these sequences. When alpha-converting a binding sequence, we generate a name permutation p which when applied to the sequence makes it sufficiently fresh. The same permutation is then applied to everything under the scope of the binders, for example:

$$\underline{M}(\lambda\tilde{x})N.P = \underline{M}(\lambda p \cdot \tilde{x})(p \cdot N).(p \cdot P) \qquad \text{if } p \subseteq \tilde{x} \times (p \cdot \tilde{x}) \text{ and } (p \cdot \tilde{x}) \# (N,\ P)$$

The side condition of this alpha-conversion looks a bit intimidating, but intuitively p swaps members of the original binding sequence to other names such that the resulting binding sequence meets the desired freshness constraints. This style of alpha-conversion was first introduced by Urban and Berghofer, although to the best of our knowledge it is still unpublished. We cover it more extensively in [BP09].

6.2. **Formalising parametric calculi.** The framework for psi-calculi is a parametric formalism. A psi-calculus agent consists of terms, assertions and conditions. This is modelled in Isabelle by creating a polymorphic datatype with three type variables. A psi-calculus agent will thus have the type $(\alpha,\ \beta,\ \gamma)$ `psi`, where α, β, and γ represents terms, assertions, and conditions respectively. All members of these types need to have finite support.

Isabelle has excellent facilities for a parametric style of reasoning through the use of locales [Bal03]. Locales allow us to postulate which functions must exist for the parameters, and which assumptions must hold on them. The entire proof structure of the meta theory is then built using the provided locale parameters. When creating a psi-calculus instance, the functions must be provided and the assumptions must be proven. Once this is done, all meta-theoretical proofs will be guaranteed to hold for the new instance.

One requirement is a substitution function which substitutes names for terms, in our terms, assertions and conditions. To this end, a locale is created with a substitution function of type $\delta \rightarrow$ `name list` $\rightarrow \alpha$ `list` $\rightarrow \delta$, where the type α will be what we use for terms, and the type δ can be any of the three nominal sets. The locale contains the following constraints:

Equivariance:	$p \cdot (X[\tilde{x} := \tilde{T}]) = (p \cdot X)[(p \cdot \tilde{x}) := (p \cdot \tilde{T})]$
Freshness:	if $a\#X[\tilde{x} := \tilde{T}]$ and $a\#\tilde{x}$ then $a\#X$
	if $a\#X$ and $a\#\tilde{T}$ then $a\#X[\tilde{x} := \tilde{T}]$
	if $\tilde{x} \subseteq \mathrm{n}(X)$ and $a\#X[\tilde{x} := \tilde{T}]$ then $a\#\tilde{T}$
	if $\tilde{x}\#\tilde{X}$ then $M[\tilde{x} := \tilde{T}] = X$
	if $\tilde{x}\#\tilde{y}$ and $\tilde{y}\#\tilde{T}$ then $X[\tilde{x}\tilde{y} := \tilde{T}\tilde{U}] = (X[\tilde{x} := \tilde{T}])[\tilde{y} := \tilde{U}]$
Alpha-equivalence:	if $p \subseteq \tilde{x} \times (p \cdot \tilde{x})$ and $(p \cdot \tilde{x})\#X$ then
	$X[\tilde{x} := \tilde{T}] = (p \cdot M)[(p \cdot \tilde{x}) := \tilde{T}]$

The constraints on this locale are straightforward. As all functions in any nominal formalisation, substitution must be equivariant. The second set of constraints cover which freshness conditions can be derived from the arguments of a substitution, depending on if it appears in the assumptions or the conclusion of a goal. Finally, it must be possible to alpha-convert a substitution. Intuitively this means that the vector being substituted is switched to one which is sufficiently fresh. Consider the Input rule.

$$\textsc{In}\ \frac{\Psi \vdash M \leftrightarrow K}{\Psi \rhd \underline{M}(\lambda\widetilde{y})N.P \xrightarrow{\underline{K}\ N[\widetilde{y}:=\widetilde{L}]} P[\widetilde{y} := \widetilde{L}]}$$

If a proof requires the input agent to be alpha-converted to $\underline{M}(\lambda p \cdot \widetilde{y})(p \cdot N).(p \cdot P)$ such that $p \cdot \tilde{y}$ is sufficiently fresh, it is necessary to convert $N[\widetilde{y} := \widetilde{L}]$ to $(p \cdot N)[(p \cdot \widetilde{y}) := \widetilde{L}]$, and $P[\widetilde{y} := \widetilde{L}]$ to $(p \cdot P)[(p \cdot \widetilde{y}) := \widetilde{L}]$, to still be able to derive the input transition. The last constraint accomplishes this. Additionally we require that the vectors $\tilde{x}$ and $\tilde{T}$, and $\tilde{y}$ and $\tilde{U}$, have equal length. This locale is then instantiated three times: for terms, assertions and conditions respectively.

The nominal morphisms in Definition 1 are modeled in a locale which specifies their existence and equivariance properties. Inside this locale we also define equivalence for assertions and frames and provide an infrastructure for reasoning about equivalence. This locale is then extended with the requisites in Definition 3.

Finally, the substitution locale is combined with the locale for equivalence to form an environment in which the rest of the theories can be proven. The locales offer a very intuitive way of reasoning about parametric systems, and without them this formalisation would have been very hard.

6.3. **Results and experiences.** Formalising the proofs for psi-calculi in parallel to its development has turned out to be invaluable, and we would certainly not have finished successfully without it. Throughout the development we have uncountable times stumbled over slightly incorrect definitions and not quite correct lemmas, prompting frequent changes in the framework. For example, our mistake in [JPVB08] mentioned in Section 2.6 was found during proof mechanisation and would probably not have been found at all without it; at that time we had completed a manual "proof" which turned out incorrect. The Isabelle formalisation gives us a high degree of confidence in the proven theorems, and equally important, it gives us a repository of proofs and proof strategies that can be re-used when some detail needs to change. Finding out which ramifications a change has on the proofs is quick and straight forward. With manual proofs, changing a detail would mean the boring and dangerously error prone process of going over each proof by hand.

As just one example, in a previous version, the Case rule looked as follows:

$$\textsc{Old-Case}\ \frac{\Psi \vdash \varphi_i}{\Psi \rhd \mathbf{case}\ \widetilde{\varphi} : \widetilde{P} \xrightarrow{\tau} P_i}$$

In this rule, the choice of which branch to take in a **case** statement yields an internal action, after which the process P evaluates as usual. An implication is that the requirement that P is guarded can be omitted. We initially adopted this rule since it admits simpler induction proofs. At a quite late stage we decided to change it to the present rule, since this more closely resembles what is used in similar calculi. The change prompted a rework of the entire proof tree from the semantics and up. The total effort was approximately eight hours, after which we had complete certainty that the new rule does not cause any problems.

Currently we have formally proven theorems 12–16 in this paper, including all supporting lemmas. The entire implementation in Isabelle is about 18000 lines of Isabelle code. It includes infrastructure for smooth treatment of binding sequences, and it has developed gradually over the last two years. The total effort for the present framework is hard to assess, since it has followed us through many failed attempts and false starts. Once in place the marginal effort of formalising more results is manageable. As an example, the total

effort in proving Theorem 12, which was one of the last things we formalised, was less than one day.

7. Conclusion and Further Work

We have defined a framework for mobile process calculi, parametrised on nominal types for data terms and for a logic to express assertions and conditions. The expressiveness surpasses the most advanced competing calculi. The semantics is a single inductive definition, which means that proofs are comparatively easy. We have fully formalised the framework in the interactive theorem prover Isabelle, which gives us full confidence in our results on bisimulation and provides a readily available infrastructure for conducting proofs of many instances and variants.

In a companion paper [JVP09] we have developed a symbolic semantics and bisimulation equivalence, and proved full abstraction wrt. $\sim$. This kind of semantics is essential for reducing the state space explosion when exploring transitions and comparing for equivalence, making it ideal for use in automated tools.

Ongoing work includes an analysis of weak bisimulation equivalence, where τ actions are considered unobservable. Our preliminary results indicate that the presence of assertions significantly complicates the definitions, in contrast to the situation with strong bisimulation. Interestingly, for psi-calculi that satisfy weakening (i.e. $\Psi \vdash \varphi \Longrightarrow \Psi \otimes \Psi' \vdash \varphi$) the definitions can be greatly simplified. We also investigate a barbed equivalence and determine what kind of observations are needed for full abstraction.

We intend to explore typed psi-calculi. One idea is to find out what properties the type system must have in order for the usual theorems such as subject reduction to hold.

References

[AF01] Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *Proceedings of POPL '01*, pages 104–115. ACM, January 2001.

[AG99] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The Spi calculus. *Journal of Information and Computation*, 148(1):1–70, 1999.

[Bal03] Clemens Ballarin. Locales and locale expressions in isabelle/isar. In *TYPES*, pages 34–50, 2003.

[BM07] Maria Grazia Buscemi and Ugo Montanari. CC-Pi: A constraint-based language for specifying service level agreements. In Rocco De Nicola, editor, *Proceedings of ESOP 2007*, volume 4421 of *LNCS*, pages 18–32. Springer, 2007.

[BM08] Maria Grazia Buscemi and Ugo Montanari. Open bisimulation for the concurrent constraint pi-calculus. In Sophia Drossopoulou, editor, *Proceedings of ESOP 2008*, volume 4960 of *LNCS*, pages 254–268. Springer, 2008.

[BN05] Johannes Borgström and Uwe Nestmann. On bisimulations for the spi calculus. *Mathematical Structures in Computer Science*, 15(03):487–552, 2005.

[BP07] Jesper Bengtson and Joachim Parrow. Formalising the pi-calculus using nominal logic. In *Proceedings of FoSSaCS 2007*, volume 4423 of *LNCS*, pages 63–77. Springer, 2007.

[BP09] Jesper Bengtson and Joachim Parrow. Psi-calculi in Isabelle. In Stefan Berghofer, Tobias Nipkow, Christian Urban, and Makarius Wenzel, editors, *Proc. of TPHOLs 2009*, volume 5674 of *LNCS*, pages 99–114. Springer, August 2009.

[Bus09] Maria Grazia Buscemi. Personal communication, January 2009.

[CM03] Marco Carbone and Sergio Maffeis. On the expressive power of polyadic synchronisation in π-calculus. *Nordic Journal of Computing*, 10(2):70–98, 2003.

[CRZ07] Véronique Cortier, Michaël Rusinowitch, and Eugen Zalinescu. Relating two standard notions of secrecy. *Logical Methods in Computer Science*, 3(3), 2007.

[CS01] Tom Chothia and Ian Stark. A distributed calculus with local areas of communication. In *HLCL '00: Proceedings of the 4th International Workshop on High-Level Concurrent Languages*, volume 41.2 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 2001.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[DKR07] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Symbolic bisimulation for the applied pi-calculus. In V. Arvind and Sanjiva Prasad, editors, *Proc. of FSTTCS'07*, volume 4855 of *LNCS*, pages 133–145. Springer, December 2007.

[GLPT07] J. Goubault-Larrecq, C. Palamidessi, and A. Troina. A probabilistic applied pi-calculus. In *Proc. of APLAS'07*, volume 4807 of *LNCS*, pages 175–190. Springer, 2007.

[GP01] Murdoch Gabbay and Andrew Pitts. A new approach to abstract syntax with variable binding. *Formal Aspects of Computing*, 13:341–363, 2001.

[JPVB08] Magnus Johansson, Joachim Parrow, Björn Victor, and Jesper Bengtson. Extended pi-calculi. In *Proceedings of ICALP 2008*, volume 5126 of *LNCS*, pages 87–98. Springer, July 2008.

[JVP09] Magnus Johansson, Björn Victor, and Joachim Parrow. A fully abstract symbolic semantics for psi-calculi. In *Proc. 6th Workshop on Structural Operational Semantics: SOS 2009*, Electronic Proceedings in Theoretical Computer Science, 2009. To appear.

[KR05] Steve Kremer and Mark D. Ryan. Analysis of an electronic voting protocol in the applied pi-calculus. In Mooly Sagiv, editor, *Proc. of ESOP'05*, volume 3444 of *LNCS*, pages 186–200. Springer, April 2005.

[MA42] H.K. Markey and G. Antheil. Secret communication system. United States Patent 2,292,387, 1942.

[Mer98] Massimo Merro. On the expressiveness of chi, update, and fusion calculi. In Catuscia Palamidessi and Ilaria Castellani, editors, *Proceedings of EXPRESS '98*, volume 16.2 of *ENTCS*. Elsevier Science Publishers, 1998.

[MPW92] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, part I/II. *Journal of Information and Computation*, 100:1–77, September 1992.

[NPW02] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL: a Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.

[Pit03] A. M. Pitts. Nominal logic, a first order theory of names and binding. *Information and Computation*, 186:165–193, 2003.

[Urb08] Christian Urban. Nominal techniques in Isabelle/HOL. *Journal of Automated Reasoning*, 40(4):327–356, May 2008.

[UT05] Christian Urban and Christine Tasson. Nominal techniques in Isabelle/HOL. In Robert Nieuwenhuis, editor, *CADE*, volume 3632 of *Lecture Notes in Computer Science*, pages 38–53. Springer, 2005.

[WG05] Lucian Wischik and Philippa Gardner. Explicit fusions. *Theoretical Computer Science*, 304(3):606–630, 2005.