Higher-order psi-calculi

JOACHIM PARROW, JOHANNES BORGSTRÖM, PALLE RAABJERG and JOHANNES ÅMAN POHJOLA

Mathematical Structures in Computer Science / *FirstView* Article / June 2013, pp 1 - 37
DOI: 10.1017/S0960129513000170, Published online: 24 June 2013

Math. Struct. in Comp. Science: page 1 of 37.
doi:10.1017/S0960129513000170

Higher-order psi-calculi[†]

JOACHIM PARROW, JOHANNES BORGSTRÖM,
PALLE RAABJERG and JOHANNES ÅMAN POHJOLA

Department of Information Technology, Uppsala University, Uppsala, Sweden
Email: joachim.parrow@it.uu.se;johannes.borgstrom@it.uu.se;
palle.raabjerg@it.uu.se;johannes.aman-pohjola@it.uu.se.

Received 26 January 2011; revised 21 January 2013

In earlier work we explored the expressiveness and algebraic theory Psi-calculi, which form a parametric framework for extensions of the pi-calculus. In the current paper we consider higher-order psi-calculi through a technically surprisingly simple extension of the framework, and show how an arbitrary psi-calculus can be lifted to its higher-order counterpart in a canonical way. We illustrate this with examples and establish an algebraic theory of higher-order psi-calculi. The formal results are obtained by extending our proof repositories in Isabelle/Nominal.

ROBIN MILNER – IN MEMORIAM

Robin Milner pioneered developments in process algebras, higher-order formalisms, and interactive theorem provers. We hope he would have been pleased to see the different strands of his work combined in this way.

1. Introduction

Psi-calculi form a parametric framework for extensions of the pi-calculus to accommodate applications with complex data structures and high-level logics in a single general and parametric framework with machine-checked proofs. In earlier papers (Bengtson *et al.* 2009; Bengtson and Parrow 2009; Johansson *et al.* 2010; Bengtson *et al.* 2010), we have shown how psi-calculi can capture a range of phenomena such as cryptography and concurrent constraints, investigated strong and weak bisimulation, and provided a symbolic semantics. We claim that the theoretical development is more robust than in other calculi of comparable complexity since we use a single inductive definition in the semantics and because we have checked most results in the theorem prover Isabelle/Nominal (Urban 2008).

In the current paper, we extend the framework to include higher-order agents, that is, agents that can send agents as objects in communication. As an example of a traditional higher-order communication, the process $\overline{a}P\,.\,Q$ sends the process P along a and then continues as Q. A recipient looks like $a(X)\,.\,(R \mid X)$, receiving a process P and continuing

[†] This work was partly supported by the Swedish Research Council grant UPMARC.

as $R \mid P$, thus $\overline{a}P \,.\, Q \mid a(X) \,.\, (R \mid X)$ has a transition leading to $Q \mid (R \mid P)$. Higher-order computational paradigms date back to the lambda-calculus, and many different formalisms are based on it. The first to study higher-order communication within a process calculus was probably Thomsen (Thomsen 1989; Thomsen 1993), and the area has been thoroughly investigated by Sangiorgi and others (Sangiorgi 1993; Sangiorgi 1996; Sangiorgi 2001; Jeffrey and Rathke 2005; Lanese *et al.* 2008; Damengeon *et al.* 2009; Lanese *et al.* 2010). There are several important problems related to type systems, to encoding higher-order behaviour using an ordinary calculus and to the precise definition of bisimulation $\sim$. To appreciate the latter, consider an agent bisimulating $\overline{a}P \,.\, Q$. The normal definition would require the same action $\overline{a}P$ leading to an agent that bisimulates Q. In some circumstances this is too strong a requirement. For example, if we assume $P \sim P'$, it is reasonable to let $\overline{a}P \,.\, Q \sim \overline{a}P' \,.\, Q$ even though they have different actions since the only thing a recipient can do with the received object is to execute it, and here bisimilar agents are indistinguishable.

1.1. *Psi-calculi*

In the following we assume familiarity with the basic ideas of process algebras based on the pi-calculus, and explain psi-calculi using a few simple examples. In a psi-calculus, there are data terms $M, N, \ldots$, and we write $\overline{M}N \,.\, P$ to represent an agent sending the term N along the channel M (which is also a data term), and continuing as the agent P. We write $\underline{K}(\lambda\widetilde{x})L \,.\, Q$ to represent an agent that can input along the channel K, receiving some object matching the pattern $\lambda\widetilde{x}L$. These two agents can interact under two conditions:

(1) The two channels must be *channel equivalent*, as defined by the channel equivalence predicate $M \leftrightarrow K$.
(2) N must match the pattern, that is, $N = L[\widetilde{x} := \widetilde{T}]$ for some sequence of terms $\widetilde{T}$.

The receiving agent then continues as $Q[\widetilde{x} := \widetilde{T}]$.

Formally, a *transition* is of kind $\Psi \rhd P \xrightarrow{\alpha} P'$, meaning that when the environment contains the *assertion* Ψ the agent P can do an action α to become P'. An assertion embodies a collection of facts, which resolve, among other things, the channel equivalence predicate $\leftrightarrow$. To continue the example, we will have

$$\Psi \rhd \overline{M}N \,.\, P \mid \underline{K}(\lambda\widetilde{x})L \,.\, Q \xrightarrow{\tau} P \mid Q[\widetilde{x} := \widetilde{T}]$$

exactly when $N = L[\widetilde{x} := \widetilde{T}]$ and $\Psi \vdash M \leftrightarrow K$. The latter says that the assertion Ψ entails that M and K represent the same channel. In this way we can introduce an equational theory over a data structure for channels. Assertions are also used to resolve the *conditions* φ in the **if** construct: we have that

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

if $\Psi \vdash \varphi$ and $\Psi \rhd P \xrightarrow{\alpha} P'$. In order to represent concurrent constraints and local knowledge, assertions can be used as agents: the agent $(\!|\Psi|\!)$ stands for an agent that

asserts Ψ to its environment. For example, in

$$P \mid (\nu a)((\!|\Psi|\!) \mid Q),$$

the agent Q uses all entailments provided by Ψ, while P only uses those that do not contain the name a.

Assertions and conditions can, in general, form any logical theory. Also, the data terms can be drawn from an arbitrary set. One of our major contributions has been to pinpoint the precise requirements on the data terms and logic for a calculus to be useful in the sense that the natural formulation of bisimulation satisfies the expected algebraic laws. It turns out that it is necessary to view the terms and logics as *nominal*. This means that there is a distinguished set of names, and for each term a well-defined notion of *support*, which corresponds intuitively to the names occurring in the term. Functions and relations must be equivariant, meaning that they treat all names equally. The logic must have a binary operator to combine assertions, corresponding to the parallel composition of processes, which must satisfy the axioms of an abelian monoid. Channel equivalence must be symmetric and transitive. In order to define the semantics of an input construct, there must be a function to substitute terms for names, but it does not matter exactly what a substitution actually does to a term. These are all quite general requirements, so psi-calculi accommodate a wide variety of extensions of the pi-calculus.

1.2. *Higher-order psi-calculi*

In one sense it is possible to have a naive higher-order psi-calculus without amending any of the definitions. Data can be *any* set satisfying the requirements mentioned above. In particular, we may include the agents among the data terms. Thus, the higher-order output and input exemplified above are already present. What is lacking is a construct to execute a received agent. A higher-order calculus usually includes the agent variables like X among the process constructors, making it possible to write, for example, $a(X).(X \mid R)$, which can receive any agent P and continue as $P \mid R$.

The route we shall take in this paper is more general and admits definitions of behaviours as recursive expressions without the need to include a new syntactic category of process variables and higher-order substitution. Instead, we introduce the notion of a *clause* $M \Leftarrow P$, meaning that the data term M can be used as a handle to invoke the behaviour of P in the agent **run** M. A sender can transmit the handle M in an ordinary output $\overline{a}M$, and a recipient can receive and run it as in $a(x).(\mathbf{run}\ x \mid R)$.

Just like conditions, clauses are entailed by assertions. In this way we can use scoping to get local definitions of behaviour. For example, let $\{M_b \Leftarrow R\}$ be an assertion entailing $M_b \Leftarrow R$ where b is in the support of M_b. Then, in

$$P \mid (\nu b)(Q \mid (\!|\{M_b \Leftarrow R\}|\!))$$

the agent Q can use the clause, but P cannot since it is out of the scope of b.

Formally, the clauses do not represent an extension of the psi framework since they can be included among the conditions. The only formal extension is the new agent form *invocation* **run** M to invoke an agent represented by M, with the corresponding rule of

action

$$\textsc{Invocation}\ \frac{\Psi \vdash M \Leftarrow P \qquad \Psi \rhd P \xrightarrow{\alpha} P'}{\Psi \rhd \mathbf{run}\ M \xrightarrow{\alpha} P'}$$

In this way, we can perform higher-order communication. In place of

$$\overline{a}P\,.\,Q \mid a(X)\,.\,(X \mid R),$$

we write

$$(\nu b)(\overline{a}M_b\,.\,Q \mid (\!|\{M_b \Leftarrow P\}|\!)) \mid a(x)\,.\,(\mathbf{run}\ x \mid R).$$

Until the left-hand component interacts along a, the scope of b prohibits the environment from using the clause. After the interaction, this scope is extruded, and the recipient can use M_b to invoke the received process. For example, if we let $P = \alpha\,.\,P'$, the communication results in a τ-transition, which can then be followed by an invocation:

$$\begin{array}{lll} (\nu b)(\overline{a}M_b\,.\,Q \mid (\!|\{M_b \Leftarrow \alpha\,.\,P'\}|\!)) & \mid & a(x)\,.\,(\mathbf{run}\ x \mid R) \quad \xrightarrow{\tau} \\ (\nu b)(Q \mid (\!|\{M_b \Leftarrow \alpha\,.\,P'\}|\!)) & \mid & \mathbf{run}\ M_b \mid R) \quad \xrightarrow{\alpha} \\ (\nu b)(Q \mid (\!|\{M_b \Leftarrow \alpha\,.\,P'\}|\!)) & \mid & P' \mid R). \end{array}$$

In this way, we do not send the agent itself, but rather a way to make it appear. This is reminiscent of the encoding of higher-order calculi into their first-order counterparts:

$$(\nu b)\overline{a}b\,.\,(Q \mid !\,b\,.\,P) \mid a(x)\,.\,R.$$

Here the trigger b is used in a normal communication to activate P. A purely syntactic difference is that in this encoding, the invocation will trigger an execution of P in the place from which it was sent, whereas in higher-order psi-calculi, the invocation rule means that P will execute in the place where it is invoked. Therefore, when M_b is a handle for P, its support must include that of P, which ensures that scope extrusions are enforced when a name in the support of P is restricted and M_b is sent out of its scope.

Our work differs from previous work on higher-order calculi in one important respect. Existing work (or at least the work we know of) explores fundamental constructions in extremely parsimonious calculi to determine exactly what can be encoded with the higher-order paradigm, or exactly how it can be encoded. Our aim, by contrast, is to extend a very rich framework, which already contains arbitrarily advanced data types, with a higher-order construct that facilitates the natural representation of applications.

1.3. *Structure of the paper*

In the next section, we recapitulate the definitions of psi-calculi from Bengtson *et al.* (2009) and Bengtson *et al.* (2011). We give all necessary definitions to make the paper formally self contained, and refer to our earlier work for motivation and intuition. In Section 3, we present the smooth extensions to higher-order psi-calculi, namely, the clauses and the invocation rule. This provides a general framework and admits many different languages for expressing the clauses. As an example, we show how to express process abstractions, and how we can construct a canonical higher-order calculus from a

first-order one by just adding a higher-order component to the assertions. In Section 4, we explore the algebraic theory of bisimulation. We inherit the definitions verbatim from first-order psi-calculi, so all properties will still hold. Moreover, we show that Sum and Replication can be directly represented through higher-order constructs. We explore a slightly amended bisimulation definition, which is more natural in a higher-order context. All proofs of all theorems presented in this paper have been formalised in the interactive theorem prover Isabelle, and we comment briefly on our experiences. Finally, we present a comparison of alternative bisimulations and give our conclusions with some ideas for further work.

2. Psi-calculi

This section recapitulates the relevant parts of Bengtson *et al.* (2009) and Bengtson *et al.* (2011).

We assume a countably infinite set of atomic *names* $\mathcal{N}$ ranged over by $a, \ldots, z$. A *nominal set* (Pitts 2003; Gabbay and Pitts 2001) is a set equipped with *name swapping* functions written $(a\ b)$, for any names a, b. An intuition here is that for any member X, we have $(a\ b) \cdot X$ 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 $(a\ b) \cdot ((a\ b) \cdot X) = X$. An important 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 $a \notin \mathrm{n}(X)$. If A is a set or a sequence of names, we write $A\#X$ to mean $\forall a \in A\ .\ a\#X$. We require all elements to have finite support, that is, $\mathrm{n}(X)$ is finite for all X. 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.

In the following, we write $\tilde{a}$ to mean a finite sequence of distinct 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 other nominal sets in the same way, except that we then do not require that all elements in the sequence are pairwise different. We use A and B to range over finite sets of names.

A *nominal datatype* is a nominal set together with a set of equivariant functions on it. In particular, we shall consider substitution functions that substitute elements for names. If X is an element of a datatype, the *substitution* $X[\tilde{a} := \tilde{Y}]$ is an element of the same datatype as X. Substitution is required to satisfy a kind of alpha-conversion law: if $\tilde{b}\#X, \tilde{a}$, then $X[\tilde{a} := \tilde{T}] = ((\tilde{b}\ \tilde{a}) \cdot X)[\tilde{b} := \tilde{T}]$, where it is implicit that $\tilde{a}$ and $\tilde{b}$ have the same length and $(\tilde{a}\ \tilde{b})$ swaps each element of $\tilde{a}$ with the corresponding element of $\tilde{b}$. The name preservation law

$$\tilde{a} \subseteq \mathrm{n}(N) \wedge b \in \mathrm{n}(\tilde{M}) \implies b \in \mathrm{n}(N[\tilde{a} := \tilde{M}])$$

will be important for some substitutions. Apart from these laws, we do not require any particular behaviour of substitution.

Formally, a psi-calculus is defined by instantiating three nominal datatypes and four operators.

Definition 2.1 (psi-calculus parameters). A psi-calculus requires the three (not necessarily disjoint) nominal datatypes

$$\begin{array}{l}\mathbf{T} \text{ the (data) terms, ranged over by } M, N\\ \mathbf{C} \text{ the conditions, ranged over by } \varphi\\ \mathbf{A} \text{ the assertions, ranged over by } \Psi\end{array}$$

and the four equivariant operators

$$\begin{array}{ll}\dot{\leftrightarrow} : \mathbf{T} \times \mathbf{T} \to \mathbf{C} & \text{Channel Equivalence}\\ \otimes : \mathbf{A} \times \mathbf{A} \to \mathbf{A} & \text{Composition}\\ \mathbf{1} : \mathbf{A} & \text{Unit}\\ \vdash \subseteq \mathbf{A} \times \mathbf{C} & \text{Entailment}\end{array}$$

together with substitution functions $[\widetilde{a} := \widetilde{M}]$, substituting terms for names, on each of **T**, **C** and **A**, where the substitution function on **T** satisfies name preservation.

The binary functions above will be written 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 $\Psi \otimes \Psi'$. We also write $\Psi \vdash \varphi$, pronounced 'Ψ entails φ', for $(\Psi, \varphi) \in \vdash$.

Definition 2.2 (assertion equivalence). Two assertions are *equivalent*, written $\Psi \simeq \Psi'$, if for all φ, we have

$$\Psi \vdash \varphi \Leftrightarrow \Psi' \vdash \varphi.$$

The requirements for valid psi-calculus parameters are as follows.

Definition 2.3 (requisites for valid psi-calculus parameters).

$$\begin{array}{lr}\text{Channel Symmetry:} & \Psi \vdash M \mathrel{\dot{\leftrightarrow}} N \Longrightarrow \Psi \vdash N \mathrel{\dot{\leftrightarrow}} M\\ \text{Channel Transitivity:} & \Psi \vdash M \mathrel{\dot{\leftrightarrow}} N \wedge \Psi \vdash N \mathrel{\dot{\leftrightarrow}} L \Longrightarrow \Psi \vdash M \mathrel{\dot{\leftrightarrow}} L\\ \text{Compositionality:} & \Psi \simeq \Psi' \Longrightarrow \Psi \otimes \Psi'' \simeq \Psi' \otimes \Psi''\\ \text{Identity:} & \Psi \otimes \mathbf{1} \simeq \Psi\\ \text{Associativity:} & (\Psi \otimes \Psi') \otimes \Psi'' \simeq \Psi \otimes (\Psi' \otimes \Psi'')\\ \text{Commutativity:} & \Psi \otimes \Psi' \simeq \Psi' \otimes \Psi.\end{array}$$

Our requirements for a psi-calculus are that the channel equivalence is a partial equivalence relation, that $\otimes$ is compositional and that the equivalence classes of assertions form an abelian monoid.

Definition 2.4 (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 the alpha variants of frames.

We use F and G to range over frames. Since we identify alpha variants, we can choose the bound names arbitrarily. We will just write 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

$$(\nu\tilde{b}_1)\Psi_1 \otimes (\nu\tilde{b}_2)\Psi_2 = (\nu\tilde{b}_1\tilde{b}_2)\Psi_1 \otimes \Psi_2$$

where $\tilde{b}_1 \# \tilde{b}_2, \Psi_2$, and *vice versa*. We will also write $(\nu c)((\nu\tilde{b})\Psi)$ to mean $(\nu c\tilde{b})\Psi$.

Definition 2.5. We define $F \vdash \varphi$ to mean that there exists an alpha variant $(\nu\tilde{b})\Psi$ of F such that $\tilde{b}\#\varphi$ and $\Psi \vdash \varphi$. We also define $F \simeq G$ to mean that for all φ, we have $F \vdash \varphi$ if and only if $G \vdash \varphi$.

Definition 2.6 (psi-calculus agents). Given valid psi-calculus parameters as in Definitions 2.1 and 2.3, the psi-calculus *agents* $\mathbf{P}$, ranged over by $P, Q, \ldots$, are of the following forms.

$\mathbf{0}$	Nil
$\overline{M}N.P$	Output
$\underline{M}(\lambda\tilde{x})N.P$	Input
$\mathbf{case}\ \varphi_1 : P_1\ []\ \cdots\ []\ \varphi_n : P_n$	Case
$(\nu a)P$	Restriction
$P \mid Q$	Parallel
$!P$	Replication
$(\!\|\Psi\|\!)$	Assertion.

Restriction binds a in P and Input binds $\tilde{x}$ in both N and P. We identify alpha equivalent agents. An assertion is *guarded* if it is a subterm of an Input or Output. An agent is *assertion guarded* if it contains no unguarded assertions. An agent is *well formed* if:

— in $\underline{M}(\lambda\tilde{x})N.P$ we have $\tilde{x} \subseteq \mathrm{n}(N)$ is a sequence without duplicates;
— the agent P in a replication $!P$ is assertion guarded; and
— the agents P_i in

$$\mathbf{case}\ \varphi_1 : P_1\ []\ \cdots\ []\ \varphi_n : P_n$$

are assertion guarded.

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 $\underline{M}(\lambda\tilde{x})N.P$, the intuition is that the pattern $(\lambda\tilde{x})N$ can match any term obtained by instantiating $\tilde{x}$, for example, $\underline{M}(\lambda x, y)f(x, y).P$ 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 (distinct, bound) names. Another significant extension is that we also allow arbitrary data terms as communication channels. Thus, it is possible to include functions that create channels.

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

Input subjects are underlined to facilitate parsing of complicated expressions, but in simple cases we will 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 later in the paper we shall use the simpler notation $a(x).P$ for this input form, and we will also sometimes omit a trailing $\mathbf{0}$, writing just $\overline{M}N$ for $\overline{M}N.\mathbf{0}$.

In the standard pi-calculus, the transitions from a parallel composition $P \mid Q$ 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 2.7 (frame of an agent). The *frame* $\mathcal{F}(P)$ *of an agent* P is defined inductively as follows:

$$\begin{aligned}
\mathcal{F}(\mathbf{0}) &= \mathbf{1}\\
\mathcal{F}(\underline{M}(\lambda\tilde{x})N.P) &= \mathbf{1}\\
\mathcal{F}(\overline{M}N.P) &= \mathbf{1}\\
\mathcal{F}(\mathbf{case}\ \tilde{\varphi} : \tilde{P}) &= \mathbf{1}\\
\mathcal{F}(!P) &= \mathbf{1}\\
\mathcal{F}((\!|\Psi|\!)) &= \Psi\\
\mathcal{F}(P \mid Q) &= \mathcal{F}(P) \otimes \mathcal{F}(Q)\\
\mathcal{F}((\nu b)P) &= (\nu b)\mathcal{F}(P).
\end{aligned}$$

Hence, an agent where all assertions are guarded has a frame equivalent to $\mathbf{1}$. In the following, we will often write $(\nu\tilde{b}_P)\Psi_P$ 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, input and the silent action τ. The input actions are of the early kind, meaning that they contain the object received. The operational semantics consists of transitions of the form $\Psi \rhd P \xrightarrow{\alpha} P'$. Intuitively, this transition means that P can perform an action α leading to P' in an environment that asserts Ψ.

Definition 2.8 (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

$$\begin{aligned}
\mathrm{bn}(\overline{M}(\nu\tilde{a})N) &= \tilde{a}\\
\mathrm{bn}(\alpha) &= \varnothing \qquad \text{if } \alpha \text{ is an input or } \tau.
\end{aligned}$$

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

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

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

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

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

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 the rule Open*, the expression* $\tilde{a} \cup \{b\}$ *means the sequence* $\tilde{a}$ *with* b *inserted anywhere.*

We also define

$$\begin{aligned} \mathrm{n}(\tau) &= \varnothing \\ \mathrm{n}(\alpha) &= \mathrm{n}(N) \cup \mathrm{n}(M) \qquad \text{if } \alpha \text{ is an output or input.} \end{aligned}$$

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 2.9 (transitions). The transitions are defined inductively in Table 1. We write $P \xrightarrow{\alpha} P'$ to mean $\mathbf{1} \rhd P \xrightarrow{\alpha} P'$. The substitution in In is defined by induction on agents, using substitution on terms, assertions and conditions for the base cases and avoiding captures through alpha-conversion in the standard way.

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 since otherwise it could be alpha-converted in the action alone. Also, for the side conditions in Scope and Open, it is important that $\mathrm{bn}(\alpha) \subseteq \mathrm{n}(\alpha)$. In rules Par and Com, 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 that it does not occur on the transition label.

3. Higher-order psi-calculi

We now proceed to formalise the extension to higher-order psi-calculi described in the introduction. Technically, this means adopting a specific form of assertion and condition, and extending the framework with a construct **run** M.

3.1. *Basic definitions*

In a *higher-order* psi-calculus, we use one particular nominal datatype of *clauses*:

$$\mathbf{Cl} = \{M \Leftarrow P \ : \ M \in \mathbf{T} \wedge P \in \mathbf{P} \wedge \mathrm{n}(M) \supseteq \mathrm{n}(P) \wedge P \text{ assertion guarded}\},$$

and the entailment relation is extended to

$$\vdash \subseteq \mathbf{A} \times (\mathbf{C} \uplus \mathbf{Cl}),$$

where we write

$$\Psi \vdash \varphi \quad \text{for} \quad \Psi \vdash (0, \varphi)$$

and

$$\Psi \vdash M \Leftarrow P \quad \text{for} \quad \Psi \vdash (1, M \Leftarrow P).$$

We amend the definition of assertion equivalence to mean that the assertions entail the same conditions *and clauses*. This extension is not formally necessary since we could instead adjoin **Cl** to the conditions, but calling $M \Leftarrow P$ a 'condition' is a misnomer we want to avoid.

Definition 3.1 (higher-order agents). The higher-order agents in a psi-calculus extend those of an ordinary calculus with one new kind of agent:

$$\mathbf{run}\ M \quad \text{Invoke an agent for which } M \text{ is a handle.}$$

We define $\mathcal{F}(\mathbf{run}\ M)$ to be $\mathbf{1}$.

Finally, there is the following new transition rule.

Definition 3.2 (higher-order transitions). The transitions in a higher-order psi-calculus are those that can be derived from the rules in Table 1 plus the additional rule

$$\textsc{Invocation}\ \frac{\Psi \vdash M \Leftarrow P \qquad \Psi \rhd P \xrightarrow{\alpha} P'}{\Psi \rhd \mathbf{run}\ M \xrightarrow{\alpha} P'}$$

We are free to choose any language we want for the assertions as long as the requirements in Definition 2.3 hold. We will now consider a few simple examples for a language where assertions are finite sets of clauses and composition $\otimes$ corresponds to union.

A higher-order communication is simply an instance of ordinary communication inferred with the Com rule. As an example, if $P \Leftarrow P$ is entailed by all assertions, that is, an agent is always a handle for itself, then

$$\overline{a}P \,.\, Q \mid a(x) \,.\, (\mathbf{run}\ x \mid R) \xrightarrow{\tau} Q \mid \mathbf{run}\ P \mid R[x := P].$$

This corresponds to sending the program code. A recipient can both execute it and use it as data. For example, R can be **if** $x = P'$ **then** …, checking if the received P is syntactically the same as some other agent P'. To prevent the latter (using it as data), we instead send a handle M to represent P:

$$\begin{array}{l}\overline{a}M\,.\,Q \mid (\!|\{M \Leftarrow P\}|\!) \mid a(x)\,.\,(\mathbf{run}\ x \mid R) \quad \xrightarrow{\tau} \\ \qquad Q \mid (\!|\{M \Leftarrow P\}|\!) \mid \qquad (\mathbf{run}\ M \mid R[x := M]).\end{array}$$

In Section 3.3, we shall define canonical higher-order calculi in which receiving a handle M means that the code of P cannot be directly inspected: all that can be done with the process P is to execute it. Thus, our semantics gives a uniform way to capture both a direct higher-order communication, where the recipient gets access to the code, and an indirect one where the recipient only receives the possibility of executing it. This is different from all existing higher-order semantics known to us, and reminiscent of the way encryption is represented in psi-calculi in Bengtson *et al.* (2011).

For another example, we will assume that there are shared private names between a process P being sent and its original environment Q:

$$(\nu b)\overline{a}M\,.\,(Q \mid (\!|\{M \Leftarrow P\}|\!)) \xrightarrow{\alpha} Q \mid (\!|\{M \Leftarrow P\}|\!).$$

If $b \in \mathrm{n}(P)$, we also have $b \in \mathrm{n}(M)$, so b is extruded whenever M is sent, that is,

$$\alpha = \overline{a}(\nu b)M.$$

This means that wherever M is received, the shared link b to Q will still work.

As an example of an invocation, consider the following transition:

$$\begin{array}{lcll}\mathbf{1} \rhd (\nu b)(Q \mid (\!|\{M_b \Leftarrow \alpha\,.\,P\}|\!) & \mid & (\nu c)(\mathbf{run}\ M_b \mid R)) & \xrightarrow{\alpha} \\ \quad (\nu b)(Q \mid (\!|\{M_b \Leftarrow \alpha\,.\,P\}|\!) & \mid & (\nu c)(P \mid R)).\end{array}$$

A derivation of this transition uses the Invocation rule

$$\frac{\{M_b \Leftarrow \alpha\,.\,P\} \vdash M_b \Leftarrow \alpha\,.\,P \qquad \{M_b \Leftarrow \alpha\,.\,P\} \rhd \alpha\,.\,P \xrightarrow{\alpha} P}{\{M_b \Leftarrow \alpha\,.\,P\} \rhd \mathbf{run}\ M_b \xrightarrow{\alpha} P}$$

Using Par and Scope, we get

$$\{M_b \Leftarrow \alpha\,.\,P\} \rhd (\nu c)(\mathbf{run}\ M_b \mid R) \xrightarrow{\alpha} (\nu c)(P \mid R).$$

The conditions on Scope require both $c\#\alpha$ and $c\#\{M_b \Leftarrow \alpha\,.\,P\}$, where the latter implies $c\#P$. Using Par, we have

$$\mathbf{1} \rhd (\!|\{M_b \Leftarrow \alpha\,.\,P\}|\!) \mid (\nu c)(\mathbf{run}\ M_b \mid R) \xrightarrow{\alpha} (\!|\{M_b \Leftarrow \alpha\,.\,P\}|\!) \mid (\nu c)(P \mid R),$$

and, finally, using Par and Scope, we again get the desired transition.

3.1.1. *Example: Representing non-determinism.* Since the same handle can be used to invoke different agents, we can represent non-determinism. Instead of $P + Q$, we can choose $a\#P, Q$ and write

$$(\nu a)(\mathbf{run}\ M_a \mid (\!|\{M_a \Leftarrow P, M_a \Leftarrow Q\}|\!)).$$

We can represent the **case** construct by a unary **if then** as follows. In place of

$$\mathbf{case}\ \varphi_1 : P_1\ [] \cdots []\ \varphi_n : P_n,$$

we write (choosing $a\#P_i, \varphi_i$)

$$(\nu a)(\mathbf{run}\ M_a \mid (\!|\{M_a \Leftarrow \mathbf{if}\ \varphi_1\ \mathbf{then}\ P_1, \cdots, M_a \Leftarrow \mathbf{if}\ \varphi_n\ \mathbf{then}\ P_n\}|\!)).$$

An intuitive reason that this works is that an invocation only occurs when a transition happens.

3.1.2. *Representing fixpoints and replication.* Some versions of CCS and similar calculi use a special fixpoint operator $\mathbf{fix}\, X\,.\,P$, where X is an agent variable, with the rule of action

$$\textsc{Fix}\ \frac{P[X := \mathbf{fix}\, X\,.\,P] \xrightarrow{\alpha} P'}{\mathbf{fix}\, X\,.\,P \xrightarrow{\alpha} P'}$$

The substitution in the premise is of a higher-order kind, replacing an agent variable by an agent. We can represent this as follows. Let the agent variable X be represented by a term M_a with support $\mathrm{n}(P) \cup \{a\}$ where $a\#P$. Then $\mathbf{fix}\, X\,.\,P$ behaves exactly as

$$(\nu a)(\mathbf{run}\ M_a \mid (\!|\{M_a \Leftarrow P[X := \mathbf{run}\ M_a]\}|\!)).$$

In this way, replication $!P$ can be seen as the fixpoint $\mathbf{fix}\, X\,.\,P|X$, and replication can be represented as

$$(\nu a)(\mathbf{run}\ M_a \mid (\!|\{M_a \Leftarrow P \mid \mathbf{run}\ M_a\}|\!)),$$

which is reminiscent of the encoding of replication in the higher-order pi-calculus. In Section 4.2, we shall formulate the precise conditions on higher-order psi-calculi where these encodings are possible.

3.2. *Process abstractions and parameters*

For a higher-order psi-calculus to be useful, there should be a high-level language for expressing clauses. This can be achieved by choosing the psi-calculus parameters in a suitable way, and without any further extension of our framework.

The following provides an example of such a language, which accommodates process abstractions and application in the standard way. It assumes a binary operator on terms $\bullet\langle\bullet\rangle$; in other words, if M and N are terms, then so is $M\langle N\rangle$.

Definition 3.3.

A *parametrised clause* is of the form $M(\lambda\tilde{x})N \Leftarrow P$, with $\tilde{x}$ binding in N and P. The corresponding definition of entailment is

$$M(\lambda\tilde{x})N \Leftarrow P \in \Psi \quad \Longrightarrow \quad \Psi \vdash M\langle N[\tilde{x} := \tilde{L}]\rangle \Leftarrow P[\tilde{x} := \tilde{L}]$$

for all $\tilde{L}$ of the same length as $\tilde{x}$ such that

$$\mathrm{n}(M\langle N[\tilde{x} := \tilde{L}]\rangle) \supseteq \mathrm{n}(P[\tilde{x} := \tilde{L}]).$$

With parametrised clauses, we can formulate recursive behaviour in a convenient way since an invocation of M can be present in P. Consider, for example, the definitions for an agent enacting a stack. The parameter of the stack is its current content, represented as a list, and its behaviour is given by the two parametrised clauses

$$\textsc{Stack}(\lambda x)x \Leftarrow \underline{\mathit{Push}}(\lambda y)y \,.\, \mathbf{run}\ \textsc{Stack}\langle \mathrm{cons}(y, x)\rangle$$
$$\textsc{Stack}(\lambda x, y)\mathrm{cons}(x, y) \Leftarrow \overline{\mathit{Pop}}\, x \,.\, \mathbf{run}\ \textsc{Stack}\langle y\rangle.$$

We use different fonts here to distinguish different kinds of terms; formally, this has no consequence but it makes the agents easier to read. Stack, *Push* and *Pop* are just terms, the first representing a handle and the other communication channels. To satisfy the condition on the names in clauses, the support of *Push* and *Pop* must either be added to the formal parameter in the clauses of Stack or to the support of the term Stack itself. Finally, $\mathrm{cons}(M, N)$ is a term representing the usual list constructor.

Note that a non-empty stack matches both clauses. As an example, let Ψ contain these two parametrised clauses and let nil be a term representing the empty list. For $x = \mathrm{nil}$ we get

$$\Psi \vdash \textsc{Stack}\langle \mathrm{nil}\rangle \Leftarrow \underline{\mathit{Push}}(\lambda y)y \,.\, \mathbf{run}\ \textsc{Stack}\langle \mathrm{cons}(y, \mathrm{nil})\rangle,$$

and thus

$$\Psi \rhd \mathbf{run}\ \textsc{Stack}\langle \mathrm{nil}\rangle \xrightarrow{\underline{\mathit{Push}}\, M} \mathbf{run}\ \textsc{Stack}\langle \mathrm{cons}(M, \mathrm{nil})\rangle,$$

and this agent can continue in two different ways: one is

$$\Psi \rhd \mathbf{run}\ \textsc{Stack}\langle \mathrm{cons}(M, \mathrm{nil})\rangle \xrightarrow{\underline{\mathit{Push}}\, M'} \mathbf{run}\ \textsc{Stack}\langle \mathrm{cons}(M', \mathrm{cons}(M, \mathrm{nil}))\rangle,$$

and the other is, using the second clause with $x = M$ and $y = \mathrm{nil}$,

$$\Psi \rhd \mathbf{run}\ \textsc{Stack}\langle \mathrm{cons}(M, \mathrm{nil})\rangle \xrightarrow{\overline{\mathit{Pop}}\, M} \mathbf{run}\ \textsc{Stack}\langle \mathrm{nil}\rangle.$$

This kind of recursion is often a very convenient way to model iterative behaviour. The earliest process algebras, such as CCS, use it extensively in applications. We say that a clause $M \Leftarrow P$ is *universal* if $\Psi \vdash M \Leftarrow P$ for all Ψ. In order to represent recursion in the CCS way, it is enough to consider universal clauses. In higher-order psi-calculi, we can additionally use local definitions since they reside in assertions where their names can be given local scope. In this way, we gain the possibility of transmitting the agents by sending the handles like Stack. We can represent a 'stack factory' which repeatedly sends out the handle to recipients as $!\overline{a}\,\textsc{Stack} \,.\, \mathbf{0}$. Each recipient will get its own stack, which will develop independently of other copies. As formulated here, all stacks will use the same channels *Push* and *Pop*, but private channels can be achieved by including their names in the formal parameters of the clauses:

$$\textsc{Stack}(\lambda i, o, x)i, o, x \Leftarrow \underline{i}(\lambda y)y \,.\, \mathbf{run}\ \textsc{Stack}\langle i, o, \mathrm{cons}(y, x)\rangle$$
$$\textsc{Stack}(\lambda i, o, x, y)i, o, \mathrm{cons}(x, y) \Leftarrow \overline{o}x \,.\, \mathbf{run}\ \textsc{Stack}\langle i, o, y\rangle.$$

Here each recipient must supply the terms to use for input and output channels as formal parameters when invoking Stack. An alternative is to let each Stack carry those terms and in an initial interaction reveal them to the recipient.

$$\textsc{StackStart} \Leftarrow \overline{c}\langle \mathit{Push}, \mathit{Pop}\rangle \,.\, \mathbf{run}\ \textsc{Stack}\langle (\mathit{Push}, \mathit{Pop}, \mathrm{nil})\rangle.$$

Here the support of *Push* and *Pop*, call it $\tilde{b}$, must be included in the support of StackStart. A recipient of StackStart must begin by receiving, along c, the terms for interacting with the stack. In the stack factory, there is then a choice of where to bind $\tilde{b}$. For example,

$$(\nu\tilde{b})\,!\overline{a}\,\textsc{StackStart}\,.\,\mathbf{0}$$

represents a stack factory that produces stacks all working on the *same* private channels, whereas

$$!(\nu\tilde{b})\overline{a}\,\textsc{StackStart}\,.\,\mathbf{0}$$

represents a factory producing stacks all working on *different* private channels.

3.3. *Canonical higher-order instances*

Given an arbitrary first-order psi-calculus $\mathcal{C}$, we will now show how to lift it systematically to a higher-order psi-calculus $\mathcal{H}(\mathcal{C})$. In earlier work (Bengtson *et al.* 2009), we demonstrated psi-calculi corresponding to the pi-calculus, the polyadic pi-calculus and explicit fusions, and we have also given calculi that capture the same phenomena as the applied pi-calculus and concurrent constraints. Out of these, only the pi-calculus has previously been given in a higher-order variant; our aim here is to lift all of them at the same time.

The main idea is to build $\mathcal{H}(\mathcal{C})$ by starting from $\mathcal{C}$ and adding the parametrised clauses described above. An assertion of $\mathcal{H}(\mathcal{C})$ is thus a pair, where the first component is an assertion in $\mathcal{C}$ and the second is a finite set of parametrised clauses. Composition of assertions is defined component-wise, with identity element $(\mathbf{1}, \varnothing)$. Finally, we define a notion of substitution on sets of process abstractions, which we do point-wise and capture-avoiding using the substitution functions of $\mathcal{C}$.

Parametrised clauses use a binary function on terms $\bullet\langle\bullet\rangle : \mathbf{T}\times\mathbf{T}\to\mathbf{T}$. We could choose this function to be standard pairing, if present in the term language, but our result holds for any such equivariant function.

Definition 3.4 (canonical higher-order psi-calculi). Let a psi-calculus $\mathcal{C}$ be defined by the parameters $\mathbf{T}, \mathbf{C}, \mathbf{A}, \dot{\leftrightarrow}, \otimes, \mathbf{1}, \vdash$. Let $\mathbf{S}$ be the set of finite sets of parametrised clauses as defined above. The canonical higher-order psi-calculus $\mathcal{H}(\mathcal{C})$ extends $\mathcal{C}$ by adding the **run** M agent and its semantic rule, and is defined by the parameters

$$\mathbf{T}_{\mathcal{H}}, \mathbf{C}_{\mathcal{H}}, \mathbf{A}_{\mathcal{H}}, \dot{\leftrightarrow}_{\mathcal{H}}, \otimes_{\mathcal{H}}, \mathbf{1}_{\mathcal{H}}, \vdash_{\mathcal{H}},$$

where

$$\begin{aligned}
\mathbf{T}_{\mathcal{H}} &= \mathbf{T}\\
\mathbf{C}_{\mathcal{H}} &= \mathbf{C}\\
\mathbf{A}_{\mathcal{H}} &= \mathbf{A}\times\mathbf{S}\\
\dot{\leftrightarrow}_{\mathcal{H}} &= \dot{\leftrightarrow}\\
(\Psi_1, S_1)\otimes_{\mathcal{H}}(\Psi_2, S_2) &= (\Psi_1\otimes\Psi_2,\ S_1\cup S_2)\\
\mathbf{1}_{\mathcal{H}} &= (\mathbf{1}, \varnothing)\\
(\Psi, S)\vdash_{\mathcal{H}}\varphi \text{ if } &\ \Psi\vdash\varphi \quad \text{for } \varphi\in\mathbf{C}\\
(\Psi, S)\vdash_{\mathcal{H}} M \Leftarrow P \text{ if } &\ \exists\widetilde{L}, K, \tilde{x}, N, Q.\ \ \mathrm{n}(M)\supseteq\mathrm{n}(P)\wedge(K(\lambda\tilde{x})N \Leftarrow Q)\in S\\
&\qquad\wedge M = K\langle N[\tilde{x}:=\tilde{L}]\rangle\wedge P = Q[\tilde{x}:=\tilde{L}].
\end{aligned}$$

For substitution, assuming $\tilde{x}\#\tilde{y}, \tilde{L}$, we define

$$(M(\lambda\tilde{x})N \Leftarrow P)[\tilde{y} := \tilde{L}]$$

to be

$$M[\tilde{y} := \tilde{L}](\lambda\tilde{x})N[\tilde{y} := \tilde{L}] \Leftarrow P[\tilde{y} := \tilde{L}]$$

and

$$(\Psi, S)[\tilde{x} := \tilde{L}]$$

to be

$$(\Psi[\tilde{x} := \tilde{L}], \{X[\tilde{x} := \tilde{L}] \mid X \in S\}).$$

As a simple example, we will construct a canonical higher-order psi-calculus corresponding to the higher-order pi-calculus. A psi-calculus corresponding to the pi-calculus was presented in Bengtson *et al.* (2009). Here the terms are just names, so lifting would yield a calculus of limited use: in any clause $a \Leftarrow P$, we require $\mathrm{n}(a) \supseteq \mathrm{n}(P)$, so only agents with singleton sorts can be invoked. An extension to admit invocation of arbitrary agents is to let the terms include tuples of names. Because of the requirement of closure under substitution of terms for names, these tuples must then be nested. This yields the psi-calculus **Tup**.

Definition 3.5 (the psi-calculus Tup).

$$\begin{aligned} \mathbf{T} &\stackrel{\text{def}}{=} \mathcal{N} \cup \{\tilde{M} \ : \ \forall i.M_i \in \mathbf{T}\} \\ \mathbf{C} &\stackrel{\text{def}}{=} \{M = N \ : \ M, N \in \mathbf{T}\} \\ \mathbf{A} &\stackrel{\text{def}}{=} \{\mathbf{1}\} \\ M \leftrightarrow N &\stackrel{\text{def}}{=} M = N \\ \vdash &\stackrel{\text{def}}{=} \{(\mathbf{1}, M, M) \ : \ M \in \mathbf{T}\}. \end{aligned}$$

We define $M\langle N\rangle$ as the pair M, N, and gain a canonical higher-order pi-calculus as $\mathcal{H}(\mathbf{Tup})$. As a simple example, let

$$S = \{M(\lambda\tilde{x})\tilde{x} \Leftarrow P\}$$

with

$$(\mathbf{1}, S) \rhd P[\tilde{x} := \tilde{L}] \xrightarrow{\alpha} P'.$$

We can then use M to invoke P with parameters $\tilde{L}$ as follows:

$$(\mathbf{1}, \varnothing) \rhd \mathbf{run}\ M\langle\tilde{L}\rangle \mid (\!|\mathbf{1}, S|\!) \xrightarrow{\alpha} P' \mid (\!|\mathbf{1}, S|\!).$$

Theorem 3.6. For all $\mathcal{C}$ and $\bullet\langle\bullet\rangle$, $\mathcal{H}(\mathcal{C})$ is a higher-order psi-calculus.

The theorem amounts to showing that $\mathcal{H}(\mathcal{C})$ satisfies the requirement on the substitution function we explained informally in Section 2 and set out formally in Bengtson *et al.* (2011), and the requisites on the entailment relation in Definition 2.3. The proof has been verified in Isabelle, where the challenge was more related to getting the nominal data type constructions correct than expressing the proof strategy.

4. Algebraic theory

In this section, we establish the expected algebraic properties of bisimilarity and then investigate the representations of Sum and Replication. We then define an alternative definition of bisimulation for higher-order communication and establish that it enjoys the same properties. The informal proof ideas for the most challenging part, that higher-order bisimilarity is preserved by parallel composition, are explained in some detail. All proofs have been formally checked in the interactive theorem prover Isabelle, and we briefly comment on our experiences in doing so.

4.1. *Bisimulation*

We begin by recollecting the definition from Bengtson *et al.* (2009), which also provides examples and intuitions.

Definition 4.1 (bisimulation). A *strong bisimulation* $\mathcal{R}$ is a ternary relation between assertions and pairs of agents such that $(\Psi, P, Q) \in \mathcal{R}$ implies all of:

(1) *Static equivalence*:

$$\Psi \otimes \mathcal{F}(P) \simeq \Psi \otimes \mathcal{F}(Q).$$

(2) *Symmetry*:

$$(\Psi, Q, P) \in \mathcal{R}.$$

(3) *Extension of arbitrary assertion*:

$$\forall \Psi'.\ (\Psi \otimes \Psi', P, Q) \in \mathcal{R}$$

(4) *Simulation*:
For all α, P' such that

$$\mathrm{bn}(\alpha) \# \Psi, Q,$$

there exists a Q' such that

$$\text{if } \Psi \rhd P \xrightarrow{\alpha} P' \text{ then } \Psi \rhd Q \xrightarrow{\alpha} Q' \wedge (\Psi, P', Q') \in \mathcal{R}.$$

If $\mathcal{R}$ is a ternary relation between assertions and pairs of agents, we will sometimes write $\Psi \rhd P\ \mathcal{R}\ Q$ for $(\Psi, P, Q) \in \mathcal{R}$. We define $\Psi \rhd P \mathrel{\dot{\sim}} Q$ to mean that there exists a strong bisimulation $\mathcal{R}$ such that $\Psi \rhd P\ \mathcal{R}\ Q$, and write $P \mathrel{\dot{\sim}} Q$ for $\mathbf{1} \rhd P \mathrel{\dot{\sim}} Q$.

Exactly the same definition applies for higher-order psi-calculi, where frame equivalence means that two frames entail the same conditions *and clauses*.

In the following, we restrict attention to well-formed agents. The following compositionality properties of strong bisimilarity for a higher-order calculus are the same as those previously established for psi-calculi.

Theorem 4.2. For all Ψ:

$$\Psi \rhd P \dot{\sim} Q \Longrightarrow \Psi \rhd P \mid R \dot{\sim} Q \mid R \tag{1}$$

$$\Psi \rhd P \dot{\sim} Q \Longrightarrow \Psi \rhd (\nu a)P \dot{\sim} (\nu a)Q \qquad \text{if } a\#\Psi. \tag{2}$$

$$\Psi \rhd P \dot{\sim} Q \Longrightarrow \Psi \rhd !P \dot{\sim} !Q \tag{3}$$

$$\forall i.\Psi \rhd P_i \dot{\sim} Q_i \Longrightarrow \Psi \rhd \mathbf{case}\ \widetilde{\varphi} : \widetilde{P} \dot{\sim} \mathbf{case}\ \widetilde{\varphi} : \widetilde{Q} \tag{4}$$

$$\Psi \rhd P \dot{\sim} Q \Longrightarrow \Psi \rhd \overline{M}N.P \dot{\sim} \overline{M}N.Q. \tag{5}$$

$$(\forall \widetilde{L}.\ \Psi \rhd P[\widetilde{a} := \widetilde{L}] \dot{\sim} Q[\widetilde{a} := \widetilde{L}]) \Longrightarrow \Psi \rhd \underline{M}(\lambda\widetilde{a})N.P \dot{\sim} \underline{M}(\lambda\widetilde{a})N.Q \qquad \text{if } \widetilde{a}\#\Psi. \tag{6}$$

We say that a relation on agents is a *congruence* if it is preserved by all operators, that is, as in Theorem 4.2, and additionally by input. Strong bisimilarity is not a congruence since it is not preserved by input. As in similar situations, we get a congruence by closing under all possible substitutions.

Definition 4.3. $\Psi \rhd P \sim Q$ means that for all sequences σ of substitutions, we have

$$\Psi \rhd P\sigma \dot{\sim} Q\sigma,$$

and we write $P \sim Q$ for $\mathbf{1} \rhd P \sim Q$.

Theorem 4.4. For every Ψ, the binary relation $\{(P, Q)\ :\ \Psi \rhd P \sim Q\}$ is a congruence.

The usual structural laws hold for strong congruence.

Theorem 4.5. $\sim$ satisfies the following structural laws:

$$P \sim P \mid \mathbf{0}$$

$$P \mid (Q \mid R) \sim (P \mid Q) \mid R$$

$$P \mid Q \sim Q \mid P$$

$$(\nu a)\mathbf{0} \sim \mathbf{0}$$

$$P \mid (\nu a)Q \sim (\nu a)(P \mid Q) \qquad \text{if } a\#P$$

$$\overline{M}N.(\nu a)P \sim (\nu a)\overline{M}N.P \qquad \text{if } a\#M, N$$

$$\underline{M}(\lambda\widetilde{x})N.(\nu a)P \sim (\nu a)\underline{M}(\lambda\widetilde{x})(N).P \qquad \text{if } a\#\widetilde{x}, M, N$$

$$\mathbf{case}\ \widetilde{\varphi} : \widetilde{(\nu a)P} \sim (\nu a)\mathbf{case}\ \widetilde{\varphi} : \widetilde{P} \qquad \text{if } a\#\widetilde{\varphi}$$

$$(\nu a)(\nu b)P \sim (\nu b)(\nu a)P$$

$$!P \sim P \mid !P$$

These results are all concerned with strong bisimulation. The corresponding results for weak bisimulation also hold, but we shall not set them out explicitly here.

Theorem 4.6. All results on the algebraic properties of weak bisimulation as defined and presented in Bengtson *et al.* (2010) also hold in higher-order psi-calculi.

The proof ideas for all results in this subsection are similar to our previously published results for (non-higher-order) psi-calculi, and the formal proofs in Isabelle required very little modification.

4.2. *Encoding operators*

In this section, we will formalise the ideas from Section 3.1.2 and establish when the operators Replication, Sum and n-ary **case** can be encoded. Recapitulating the idea of the encoding of replication $!P$ as

$$(\nu a)(\mathbf{run}\ M_a \mid (\!|\{M_a \Leftarrow P \mid \mathbf{run}\ M_a\}|\!)).$$

we immediately see that it needs an assertion $\{M_a \Leftarrow P \mid \mathbf{run}\ M_a\}$, which, intuitively, entails the clause $M_a \Leftarrow P \mid \mathbf{run}\ M_a$ and nothing else. We call such an assertion a *characteristic assertion* for M_a and $P \mid \mathbf{run}\ M_a$, and in the corresponding encoding of **case** we need characteristic assertions for sequences of agents $\widetilde{P}$ with a common handle. The full definition is as follows.

Definition 4.7. In a higher-order calculus, for a finite sequence of agents $\widetilde{P} = P_1, \ldots, P_n$ and term M with $\mathrm{n}(\widetilde{P}) \subseteq \mathrm{n}(M)$, the assertion $\Psi^{M \Leftarrow \widetilde{P}}$ is *characteristic* for M and $\widetilde{P}$ if the following hold for all agents Q, assertions Ψ and clauses and conditions ξ:

$$\Psi \vdash M \Leftarrow Q \quad \text{implies} \quad \mathrm{n}(M) \subseteq \mathrm{n}(\Psi). \tag{1}$$

$$\Psi \otimes \Psi^{M \Leftarrow \widetilde{P}} \vdash \xi \quad \text{iff} \quad (\xi = M \Leftarrow P_i \ \vee \ \Psi \vdash \xi). \tag{2}$$

$$\mathrm{n}\left(\Psi^{M \Leftarrow P}\right) \ = \ \mathrm{n}(M). \tag{3}$$

The first of these requirements is a general requirement on the calculus and makes sure that an environment cannot bestow additional invocation possibilities to the handles used in the encodings. For example, if we had $\mathbf{1} \vdash M_a \Leftarrow Q$, which would violate the requirement, then clearly $(\nu a)\mathbf{run}\ M_a \cdots$ can enact Q. In other words, our encoding of $!P$ could also enact Q. Requirement (1) excludes this possibility since $a \in \mathrm{n}(M_a)$ and $a \notin \mathrm{n}(\mathbf{1}) = \varnothing$. The second requirement means that the characteristic assertion only has the effect of entailing its own clauses, no matter how it is combined with other assertions. The third requirement ensures that the characteristic assertion does not invent names that do not occur in its handle.

Fortunately, characteristic assertions exist in most canonical higher-order calculi. We need to restrict attention to calculi with a *unit term* $() \in \mathbf{T}$ such that $\mathrm{n}(()) = \varnothing$, and where the pairing function satisfies

$$M\langle N\rangle = M'\langle N'\rangle \implies M = M',$$

and we have for all $T \in \mathbf{T} \cup \mathbf{A} \cup \mathbf{C}$, $T[\varepsilon := \varepsilon] = T$. The reason is technical: in a canonical calculus, we use parametrised clauses, where the handles must be treated as distinct, and in situations where no parameter is actually needed, we use $()$ as a dummy and communications give rise to empty substitutions. In assertions, we then write $M \Leftarrow P$ for the parametrised clause $M(\lambda\varepsilon)() \Leftarrow P$, and in processes $\mathbf{run}\ M$ for the invocation $\mathbf{run}\ M()$.

Theorem 4.8. In a canonical higher-order calculus with unit term, pairing and empty substitution as above, if $\mathrm{n}(\widetilde{P}) \subseteq \mathrm{n}(M)$ and $\widetilde{P} \neq \varepsilon$, the assertion

$$(\mathbf{1}, \{M \Leftarrow P_i \ : \ P_i \in \widetilde{P}\})$$

is characteristic for M and $\widetilde{P}$.

The following formal theorems of the encodings hold for arbitrary higher-order calculi, and are particularly relevant for canonical calculi where characteristic assertions can be expressed easily.

Theorem 4.9. In a higher-order calculus with the $+$ operator (that is, where there exists a condition $\top$ – *cf.* the discussion following Definition 2.6), for all assertion guarded P, Q and names $a\#P, Q$ and terms M with $\mathrm{n}(P, Q, a) \subseteq \mathrm{n}(M)$ and assertions $\Psi^{M \Leftarrow P,Q}$ characteristic for M and P, Q, we have

$$P + Q \dot{\sim} (\nu a)(\mathbf{run}\ M \mid (\!|\Psi^{M \Leftarrow P,Q}|\!)).$$

Theorem 4.10. In a higher-order calculus, for all assertion guarded $\tilde{P} = P_1, \ldots, P_n$, conditions $\tilde{\varphi} = \varphi_1, \ldots, \varphi_n$, names $a\#\widetilde{P}, \widetilde{\varphi}$ and terms M with $\mathrm{n}(\widetilde{P}, \widetilde{\varphi}, a) \subseteq \mathrm{n}(M)$ and assertions

$$\Psi^{M \Leftarrow \mathbf{if}\ \varphi_1\ \mathbf{then}\ P_1, \ldots, \mathbf{if}\ \varphi_n\ \mathbf{then}\ P_n}$$

characteristic for M and the agent

$$\mathbf{if}\ \varphi_1\ \mathbf{then}\ P_1, \ldots, \mathbf{if}\ \varphi_n\ \mathbf{then}\ P_n,$$

we have

$$\mathbf{case}\ \ \varphi_1 \ : \ P_1 \ [] \ \cdots \ [] \ \varphi_n \ : \ P_n \ \dot{\sim} \ (\nu a)(\mathbf{run}\ \ M \ \mid \ (\!|\Psi^{M \Leftarrow (\mathbf{if}\ \varphi_1\ \mathbf{then}\ P_1), \ldots, (\mathbf{if}\ \varphi_n\ \mathbf{then}\ P_n)}|\!)).$$

Theorem 4.11. In a higher-order calculus, for all assertion guarded P, names $a\#P$ and terms M with $\mathrm{n}(P, a) \subseteq \mathrm{n}(M)$ and assertions $\Psi^{M \Leftarrow P \mid \mathbf{run}\ M}$ characteristic for M and $P \mid \mathbf{run}\ M$, we have

$$!P \dot{\sim} (\nu a)(\mathbf{run}\ M \mid (\!|\Psi^{M \Leftarrow P \mid \mathbf{run}\ M}|\!)).$$

As an example of the encoding of Replication, consider a transition from

$$(\nu a)(\mathbf{run}\ M \mid (\!|\Psi^{M \Leftarrow P \mid \mathbf{run}\ M}|\!)).$$

This can only be by invocation where $P \mid \mathbf{run}\ M$ has a transition leading to $P' \mid \mathbf{run}\ M$ and results in

$$(\nu a)(P' \mid \mathbf{run}\ M \mid (\!|\Psi^{M \Leftarrow P \mid \mathbf{run}\ M}|\!)).$$

Using Theorem 4.5 and $a\#P'$, we can rewrite this as

$$P' \mid (\nu a)(\mathbf{run}\ M \mid (\!|\Psi^{M \Leftarrow P \mid \mathbf{run}\ M}|\!)).$$

In other words, the transition corresponds precisely to the transition of $!P$ derived from $P \mid !P$.

Clearly, for these theorems to be applicable, there must exist terms M with large enough support to represent handles. This is the case for, for example, $\mathcal{H}(\mathbf{Tup})$ in Section 3.3, which has terms with arbitrarily large finite support.

4.3. *Higher-order bisimulation*

The standard notion of bisimilarity is often found unsatisfactory in higher-order process calculi because it requires actions to match exactly: an action $\overline{a}P$ must be simulated by an identical action. Therefore, if $P \neq P'$, we will have $\overline{a}P.\mathbf{0} \not\sim \overline{a}P'.\mathbf{0}$ even if $P \sim P'$, which spoils the claim for $\sim$ to be a congruence in the ordinary sense of the word.

In psi-calculi, the data terms can be anything, even processes, but here the distinction between $\overline{a}P.\mathbf{0}$ and $\overline{a}P'.\mathbf{0}$ is necessary since the semantics allows a recipient to use the received process in a variety of ways. For example, there are psi-calculi where it is possible to receive a process and test whether it is syntactically equal to another process, as in $a(x).\mathbf{if}\ x = Q\ \mathbf{then}\ \ldots$, or to subject it to pattern matching in order to find its outermost operator – this corresponds to inspecting the process code.

In a higher-order process calculus, we can instead transmit the possibility of invoking a process, as in $(\nu b)\overline{a}M_b.(\!|\{M_b \Leftarrow P\}|\!)$. A recipient of M_b has no other use for this handle than to invoke P. Therefore, if $P \mathrel{\dot{\sim}} P'$, it is reasonable to expect the two processes

$$\begin{aligned} Q &= (\nu b)\overline{a}M_b.(\!|\{M_b \Leftarrow P\}|\!) \\ Q' &= (\nu b)\overline{a}M_b.(\!|\{M_b \Leftarrow P'\}|\!) \end{aligned}$$

to be bisimilar since it should not matter which of P or P' is invoked. But with the current definition of bisimilarity, $Q \not\mathrel{\dot{\sim}} Q'$. Consider a transition from Q that opens the scope of b. The resulting agent is simply $(\!|\{M_b \Leftarrow P\}|\!)$. The corresponding transition from Q' leads to $(\!|\{M_b \Leftarrow P'\}|\!)$. These are not bisimilar since they are not statically equivalent: $\{M_b \Leftarrow P\} \not\simeq \{M_b \Leftarrow P'\}$ since they do not entail exactly the same clauses.

This suggests that a slightly relaxed version of bisimilarity is more appropriate, where we weaken static equivalence to require bisimilar (rather than identical) entailed clauses.

Definition 4.12 (HO-bisimulation). A *strong HO-bisimulation* $\mathcal{R}$ is a ternary relation between assertions and pairs of agents such that $(\Psi, P, Q) \in \mathcal{R}$ implies all of:

(1) *Static equivalence*:

(a) $\forall \varphi \in \mathbf{C}.\quad \Psi \otimes \mathcal{F}(P) \vdash \varphi \Rightarrow \Psi \otimes \mathcal{F}(Q) \vdash \varphi$

(b) $\forall (M \Leftarrow P') \in \mathbf{Cl}.$

$$\Psi \otimes \mathcal{F}(P) \vdash M \Leftarrow P' \Rightarrow\ \exists Q'.\ \Psi \otimes \mathcal{F}(Q) \vdash M \Leftarrow Q' \wedge (\mathbf{1}, P', Q') \in \mathcal{R}.$$

(2) *Symmetry*:

$$(\Psi, Q, P) \in \mathcal{R}.$$

(3) *Extension of arbitrary assertion*:

$$\forall \Psi'.\ (\Psi \otimes \Psi', P, Q) \in \mathcal{R}.$$

(4) *Simulation*:
For all α, P' such that $\mathrm{bn}(\alpha)\#\Psi, Q$, there exists a Q' such that

$$\text{if } \Psi \rhd P \xrightarrow{\alpha} P' \text{ then } \Psi \rhd Q \xrightarrow{\alpha} Q' \wedge (\Psi, P', Q') \in \mathcal{R}.$$

We define $\Psi \rhd P \mathrel{\dot{\sim}}^{\text{HO}} Q$ to mean that there exists a strong HO-bisimulation $\mathcal{R}$ such that $\Psi \rhd P \,\mathcal{R}\, Q$, and write $P \mathrel{\dot{\sim}}^{\text{HO}} Q$ for $\mathbf{1} \rhd P \mathrel{\dot{\sim}}^{\text{HO}} Q$.

The only difference between bisimulation and HO-bisimulation is in Clause (1), which here is split into different requirements for conditions and clauses.

Theorem 4.13. In a higher-order psi-calculus, for all assertion guarded P, Q and terms M with $\mathrm{n}(P, Q) \subseteq \mathrm{n}(M)$ with characteristic assertions $\Psi^{M \Leftarrow P}$ and $\Psi^{M \Leftarrow Q}$, we have

$$P \mathrel{\dot{\sim}}^{\text{HO}} Q \Rightarrow (\!|\Psi^{M \Leftarrow P}|\!) \mathrel{\dot{\sim}}^{\text{HO}} (\!|\Psi^{M \Leftarrow Q}|\!).$$

The proof boils down to showing that

$$\{(\Psi, (\!|\Psi^{M \Leftarrow P}|\!), (\!|\Psi^{M \Leftarrow Q}|\!)) : \Psi \in \mathbf{A}, P \mathrel{\dot{\sim}}^{\text{HO}}_{\Psi} Q\} \cup \mathrel{\dot{\sim}}^{\text{HO}}$$

is a HO-bisimulation. The only non-trivial part is static equivalence. In order to prove this, we use Definition 4.7(2), which says that $\Psi \otimes \Psi^{M \Leftarrow P} \vdash \xi$ if and only if $(\xi = M \Leftarrow P \quad \vee \quad \Psi \vdash \xi)$. The proof holds for all calculi with characteristic assertions, and, in particular, it holds for canonical calculi by Theorem 4.8.

In the rest of this section, we will study the algebraic properties of HO-bisimulation in arbitrary calculi (not only canonical ones). The original bisimulation is still a valid proof technique.

Theorem 4.14. $\Psi \rhd P \mathrel{\dot{\sim}} Q \implies \Psi \rhd P \mathrel{\dot{\sim}}^{\text{HO}} Q$.

The proof is that $\mathrel{\dot{\sim}}$ is a HO-bisimulation: take $Q' = P'$ in Clause (1b). Thus, we immediately get a set of useful algebraic laws.

Corollary 4.15. $\mathrel{\dot{\sim}}^{\text{HO}}$ satisfies all the structural laws of Theorem 4.5.

HO-bisimulation is compositional in the same way as ordinary bisimulation:

Theorem 4.16. For all Ψ, we have:

$$\Psi \rhd P \mathrel{\dot{\sim}}^{\text{HO}} Q \Longrightarrow \Psi \rhd P \mid R \mathrel{\dot{\sim}}^{\text{HO}} Q \mid R. \tag{1}$$

$$\Psi \rhd P \mathrel{\dot{\sim}}^{\text{HO}} Q \Longrightarrow \Psi \rhd (\nu a)P \mathrel{\dot{\sim}}^{\text{HO}} (\nu a)Q \qquad \text{if } a\#\Psi \tag{2}$$

$$\Psi \rhd P \mathrel{\dot{\sim}}^{\text{HO}} Q \Longrightarrow \Psi \rhd !P \mathrel{\dot{\sim}}^{\text{HO}} !Q \qquad \text{if guarded}(P, Q) \tag{3}$$

$$\forall i.\Psi \rhd P_i \mathrel{\dot{\sim}}^{\text{HO}} Q_i \Longrightarrow \Psi \rhd \mathbf{case}\ \widetilde{\varphi} : \widetilde{P} \mathrel{\dot{\sim}}^{\text{HO}} \mathbf{case}\ \widetilde{\varphi} : \widetilde{Q} \qquad \text{if guarded}(\widetilde{P}, \widetilde{Q}) \tag{4}$$

$$\Psi \rhd P \mathrel{\dot{\sim}}^{\text{HO}} Q \Longrightarrow \Psi \rhd \overline{M}N.P \mathrel{\dot{\sim}}^{\text{HO}} \overline{M}N.Q. \tag{5}$$

$$(\forall \widetilde{L}.\ \Psi \rhd P[\widetilde{a} := \widetilde{L}] \mathrel{\dot{\sim}}^{\text{HO}} Q[\widetilde{a} := \widetilde{L}]) \tag{6}$$

$$\Longrightarrow \Psi \rhd \underline{M}(\lambda \widetilde{a})N.P \mathrel{\dot{\sim}}^{\text{HO}} \underline{M}(\lambda \widetilde{a})N.Q \qquad \text{if } \widetilde{a}\#\Psi.$$

Combining Theorems 4.16 and 4.13, we get the desired result for our motivating example: in a canonical higher-order psi calculus, we have

$$P \mathrel{\dot{\sim}}^{\text{HO}} P' \;\Rightarrow\; (\nu b)\overline{a}M_b\,.\,(\!|\Psi^{M_b \Leftarrow P}|\!) \mathrel{\dot{\sim}}^{\text{HO}} (\nu b)\overline{a}M_b\,.\,(\!|\Psi^{M_b \Leftarrow P'}|\!).$$

We can characterise higher-order bisimulation congruence in the usual way as follows.

Definition 4.17. We have

$$\Psi \rhd P \sim^{\text{HO}} Q$$

if and only if for all sequences σ of substitutions, we have

$$\Psi \rhd P\sigma \mathrel{\dot{\sim}}^{\text{HO}} Q\sigma.$$

We write $P \sim^{\text{HO}} Q$ for $\mathbf{1} \rhd P \sim^{\text{HO}} Q$.

Theorem 4.18. For every Ψ, the binary relation $\{(P, Q) \;:\; \Psi \rhd P \sim^{\text{HO}} Q\}$ is a congruence.

Theorem 4.19. $\sim^{\text{HO}}$ satisfies the following structural laws:

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

4.4. *Informal proofs*

Most of the proofs closely follow the corresponding results in the original psi-calculi. In this section, we present the most challenging part, where new proof ideas are needed for Theorem 4.16 (1), to show that higher-order bisimilarity is preserved by parallel. A major complication is that the Invocation rule can be used multiple times during the derivation of a transition. Another complication is that the relation $\{(P \mid R, Q \mid R) : P \mathrel{\dot{\sim}}^{\text{HO}} Q\}$ is no longer a bisimulation since if P and Q are different, their assertions can enable different invocations in R, so a transition from R leads to agents outside the relation. In the proof, we therefore work with bisimulation up to transitivity (Sangiorgi 1998). For technical reasons, in the proofs, we additionally parametrise the transitive closure on a set of names that must not appear in processes.

The proof of compositionality for ordinary bisimulation is described in some detail in Bengtson *et al.* (2011) and Johansson (2010), which should also be referred to for

motivating examples and a discussion of the proof structure. Here we will focus on the main differences arising for the higher-order case, including the use of up-to techniques.

Definition 4.20 (up-to techniques). We define *up to union with HO-bisimilarity* (U), *up to restriction* (R) and *up to transitivity* (T) by induction as follows:

$$\begin{aligned} U(\mathcal{R}) &:= \mathcal{R} \cup \dot{\sim}^{\text{HO}} \\ R(\mathcal{R}) &:= \{(\Psi, (\nu\widetilde{a})P, (\nu\widetilde{a})Q) : \widetilde{a}\#\Psi \wedge (\Psi, P, Q) \in \mathcal{R}\} \\ T(\mathcal{R}) &:= \mathcal{R} \cup \{(\Psi, P, R) : (\Psi, P, Q) \in T(\mathcal{R}) \wedge (\Psi, Q, R) \in T(\mathcal{R})\}. \end{aligned}$$

A HO-bisimulation up to S is defined as a HO-bisimulation, except that the derivatives after a simulation step or an invocation should be related by $S(\mathcal{R})$ instead of $\mathcal{R}$.

Definition 4.21 (HO-bisimulation up-to). If S is a function from ternary relations to ternary relations, then $\mathcal{R}$ is a bisimulation up to S if $\mathcal{R}$ satisfies Definition 4.12 with $S(\mathcal{R})$ substituted for $\mathcal{R}$ in clauses (1b) and (4).

The up-to techniques of Definition 4.20 are sound.

Theorem 4.22. If $\mathcal{R}$ is a HO-bisimulation up to $T \circ U \circ R$, then $\mathcal{R} \subseteq \dot{\sim}^{\text{HO}}$.

Proof. The proof is standard. We have if $\mathcal{R}$ is a HO-bisimulation up to $T \circ U \circ R$, then $T(U(R(\mathcal{R})))$ is a HO-bisimulation and $\mathcal{R} \subseteq T(U(R(\mathcal{R})))$. □

The inductive proofs of the following technical lemmas often require a strengthening of the notion of transitivity by parametrising on a finite set of names that are fresh for the processes under consideration and therefore must be avoided.

Definition 4.23 (name-avoiding transitivity). If $\mathcal{R}$ is a ternary relation, then $T_{\text{a}}(\mathcal{R})$ is inductively defined as follows:

$$\begin{aligned} T_{\text{a}}(\mathcal{R}) := \; & \{(B, \Psi, P, Q) : B\#P, Q \wedge \Psi \rhd P \; \mathcal{R} \; Q\} \cup \\ & \{(B, \Psi, P, R) : (B, \Psi, P, Q) \in T_{\text{a}}(\mathcal{R}) \wedge (B, \Psi, Q, R) \in T_{\text{a}}(\mathcal{R}))\}. \end{aligned}$$

If $\mathcal{R}' = T_{\text{a}}(\mathcal{R})$, we write $\Psi \rhd_B P \; \mathcal{R}' \; Q$ for $(B, \Psi, P, Q) \in \mathcal{R}'$.
We write $F \rhd P \; \mathcal{R} \; Q$ if $F = (\nu\widetilde{x})\Psi$ such that $\Psi \rhd P \; \mathcal{R} \; Q$ and $\widetilde{x}\#P, Q$.

Note that $\Psi \rhd P \; (T(\mathcal{R})) \; Q$ if and only if $\Psi \rhd_\varnothing P \; (T_{\text{a}}(\mathcal{R})) \; Q$.
In the remainder of the proof, we will work with the candidate relation $\mathcal{S}$ defined by

$$\mathcal{S} := \{(\Psi, P|R, Q|R) \mid \Psi \otimes \mathcal{F}(R) \rhd P \; \dot{\sim}^{\text{HO}} \; Q\}.$$

We will first show that $\mathcal{S}$ is a HO-bisimulation up to $T \circ U \circ R$. The compositionality of higher-order bisimulation then follows using the soundness of the up-to techniques. We write $\overline{\mathcal{S}}$ for $T_{\text{a}}(U(R(\mathcal{S})))$. The proof begins by showing some closure properties of $\overline{\mathcal{S}}$ that are used in the induction cases of the main lemmas. We then recall some technical lemmas from Bengtson *et al.* (2011) about the choice of subjects in transitions. The main lemmas (Lemma 4.28 and 4.29) concern the simulation case of the definition of HO-bisimilarity, in particular, transitions of R and communications between P and R, respectively.

The following closure properties of $\overline{\mathcal{S}}$ hold. Intuitively, $\overline{\mathcal{S}}$ is:

— a congruence with respect to parallel composition and restriction;
— preserved by bisimilarity;
— monotonic in B and Ψ modulo $\simeq$.

Lemma 4.24. If $\Psi \rhd_B P \;\overline{\mathcal{S}}\; Q$, then:

(1) If

$$\Psi \simeq \Psi' \otimes \Psi_R$$
$$\mathcal{F}(R) = (\nu \widetilde{b}_R)\Psi_R$$
$$\widetilde{b}_R \subseteq B$$
$$B \# R$$
$$\widetilde{b}_R \# \Psi', R,$$

then

$$\Psi' \rhd_{B \setminus \widetilde{b}_R} (P \mid R) \;\overline{\mathcal{S}}\; (Q \mid R).$$

(2) If $a \# \Psi$, then

$$\Psi \rhd_B (\nu a)P \;\overline{\mathcal{S}}\; (\nu a)Q.$$

(3) If

$$\Psi \rhd P' \mathrel{\dot{\sim}} P$$
$$\Psi \rhd Q \mathrel{\dot{\sim}} Q'$$
$$B \# P', Q',$$

then

$$\Psi \rhd_B P' \;\overline{\mathcal{S}}\; Q'.$$

(4) $\Psi \otimes \Psi' \rhd_B P \;\overline{\mathcal{S}}\; Q$.
(5) $\Psi \rhd_{B \setminus B'} P \;\overline{\mathcal{S}}\; Q$.
(6) If $\Psi \simeq \Psi'$, then

$$\Psi' \rhd_B P \;\overline{\mathcal{S}}\; Q.$$

Proof. The proof is by induction on the definition of T_{a}. □

We now recall three lemmas used in the compositionality proof for first-order bisimilarity (Bengtson *et al.* 2011). In their statements we assume that $\mathcal{F}(P) = (\nu \widetilde{b}_P)\Psi_P$. These lemmas have been proved to hold for higher-order psi-calculi too. The first of them states that when performing a non-tau transition, the frame of the process grows such that the bound names in the frame and the action can be chosen fresh for an arbitrary finite set of names B.

Lemma 4.25 (frame grows when doing transitions).

(1) If

$$\Psi \rhd P \xrightarrow{M\,N} P'$$

$$\widetilde{b}_P \# P, N, B,$$

then

$$\exists \Psi', \widetilde{b}_{P'}, \Psi_{P'}$$

such that

$$\mathcal{F}(P') = (\nu \widetilde{b}_{P'})\Psi_{P'}$$

$$\Psi_P \otimes \Psi' \simeq \Psi_{P'}$$

$$\widetilde{b}_{P'} \# B, P'.$$

(2) If

$$\Psi \rhd P \xrightarrow{\overline{M}\,(\nu \widetilde{a})N} P'$$

$$\widetilde{b}_P \# P, \widetilde{a}, B$$

$$\widetilde{a} \# P, B,$$

then

$$\exists p, \Psi', \widetilde{b}_{P'}, \Psi_{P'}$$

such that

$$\mathcal{F}(P') = (\nu \widetilde{b}_{P'})\Psi_{P'}$$

$$\Psi_{P'} \simeq (p \cdot \Psi_P) \otimes \Psi'$$

$$\widetilde{b}_{P'} \# B, P', N, \widetilde{a}$$

$$(p \cdot \widetilde{a}) \# B, P', N, \widetilde{b}_{P'}$$

$$p \subseteq \widetilde{a} \times (p \cdot \widetilde{a}).$$

The second lemma states that given a non-tau transition of P and a finite set of names B that are fresh for P, we can find a term K that is channel-equivalent to the subject of the transition such that B is fresh for K.

Lemma 4.26 (find fresh subject).

$$B \# P \wedge$$

$$\Psi \rhd P \xrightarrow{\alpha} P' \text{ (where } \alpha \neq \tau) \wedge$$

$$\widetilde{b}_P \# \Psi, P, \mathrm{subj}(\alpha), B \implies \exists K.\, B \# K \wedge \Psi \otimes \Psi_P \vdash K \leftrightarrow \mathrm{subj}(\alpha).$$

The third lemma states that if a process P performs a non-tau transition, and K is channel-equivalent to the subject of the transition, then P can perform the same transition with K as subject.

Lemma 4.27 (subject rewriting).

$$\begin{aligned}\Psi \rhd P \xrightarrow{\alpha} P' \wedge \\ \Psi_P \otimes \Psi \vdash K \leftrightarrow M \wedge \\ \widetilde{b}_P \# \Psi, P, K, M \implies \Psi \rhd P \xrightarrow{\alpha'} P'\end{aligned}$$

when

$$\begin{aligned}\alpha &= \overline{M}\,(\nu \widetilde{a})N \\ \alpha' &= \overline{K}\,(\nu \widetilde{a})N\end{aligned}$$

or

$$\begin{aligned}\alpha &= \underline{M}\,N \\ \alpha' &= \underline{K}\,N.\end{aligned}$$

We can now show our main technical lemma, which, intuitively, states that if P and Q are bisimilar in the environment of R, and R makes a transition in the environment of P, then R can make the same transition in the environment of Q, leading to $\overline{\mathcal{S}}$-related derivatives. The proof makes use of a set B of names that are required to be fresh, which grows in the induction case. A similar lemma applies in first-order psi-calculi (Bengtson *et al.* 2011), where the derivatives are always syntactically equal (not just related by $\overline{\mathcal{S}}$).

Lemma 4.28 (frame switching lemma).

$$\begin{aligned}\Psi \otimes \Psi_R \rhd P \mathrel{\dot{\sim}}^{\text{HO}} Q \wedge \\ \Psi \otimes \Psi_P \rhd R \xrightarrow{\alpha} R_P \wedge \\ \mathcal{F}(P) = (\nu \widetilde{b}_P)\Psi_P \wedge \\ \mathcal{F}(Q) = (\nu \widetilde{b}_Q)\Psi_Q \wedge \\ \mathcal{F}(R) = (\nu \widetilde{b}_R)\Psi_R \wedge \\ \widetilde{b}_P \# \alpha, \Psi, R \wedge \\ \widetilde{b}_Q \# \alpha, \Psi, R \wedge \\ \widetilde{b}_R \# \alpha, \widetilde{b}_P, \widetilde{b}_Q, \Psi, P, Q, R \wedge \\ \text{bn}(\alpha) \# \Psi, P, Q, R \wedge \\ B \# \Psi, P, Q, R, \text{obj}(\alpha), R_P, \widetilde{b}_R \implies \exists R_Q . \Psi \otimes \Psi_Q \rhd R \xrightarrow{\alpha} R_Q \\ \wedge\ \Psi \rhd_B (P|R_P)\ \overline{\mathcal{S}}\ (Q|R_Q)\end{aligned}$$

Proof. The proof is by induction on the derivation of the transition of R. The base cases are as in Bengtson *et al.* (2011), and we will only show some of the interesting induction cases here:

— Inv

In this case, $R = \mathbf{run}\ M$ and $\mathcal{F}(R) = \mathbf{1}$ and the transition is derived like

$$\textsc{Invocation}\ \frac{\Psi \otimes \Psi_P \vdash M \Leftarrow R_1 \qquad \Psi \otimes \Psi_P \rhd R_1 \xrightarrow{\alpha} R_P}{\Psi \otimes \Psi_P \rhd \mathbf{run}\ M \xrightarrow{\alpha} R_P}$$

By induction, there is R' such that

$$\Psi \otimes \Psi_Q \rhd R_1 \xrightarrow{\alpha} R'$$
$$\Psi \rhd_B (P|R_P)\ \overline{\mathcal{S}}\ (Q|R').$$

Since

$$\Psi \otimes \mathbf{1} \rhd P \dot{\sim}^{\text{HO}} Q,$$

there is R_2 such that

$$\Psi \otimes \Psi_Q \vdash M \Leftarrow R_2$$
$$\mathbf{1} \rhd R_1 \dot{\sim}^{\text{HO}} R_2.$$

Hence,

$$\Psi \otimes \Psi_Q \rhd R_1 \dot{\sim}^{\text{HO}} R_2,$$

so there is R_Q such that

$$\Psi \otimes \Psi_Q \rhd R_2 \xrightarrow{\alpha} R_Q$$
$$\Psi \otimes \Psi_Q \rhd R' \dot{\sim}^{\text{HO}} R_Q.$$

By the definition of $\mathcal{S}$, we then get

$$\Psi \rhd (Q|R')\ \mathcal{S}\ (Q|R_Q).$$

Since

$$B\#R, \text{obj}(\alpha),$$

we get $B\#R_Q$, so

$$\Psi \rhd_B (Q|R')\ \overline{\mathcal{S}}\ (Q|R_Q).$$

By transitivity, we then get

$$\Psi \rhd_B (P|R_P)\ \overline{\mathcal{S}}\ (Q|R_Q).$$

— Scope

In this case,

$$\mathcal{F}((\nu b)R) = (\nu b)\mathcal{F}(R)$$

where

$$\mathcal{F}(R) = (\nu \widetilde{b_R})\Psi_R,$$

so

$$\mathcal{F}((\nu b)R) = (\nu b\widetilde{b}_R)\Psi_R.$$

We assume that $b\#\widetilde{b}_R$. Since $b\widetilde{b}_R\#(\nu b)R$, we then have $\widetilde{b}_R\#R$. The transition is derived like

$$\textsc{Scope}\ \frac{\Psi \otimes \Psi_P \rhd R \xrightarrow{\alpha} R_P}{\Psi \otimes \Psi_P \rhd (\nu b)R \xrightarrow{\alpha} (\nu b)R_P}\ b\#\alpha, \Psi \otimes \Psi_P$$

By induction, we get that there exists R_Q such that

$$\Psi \otimes \Psi_Q \rhd R \xrightarrow{\alpha} R_Q$$
$$\Psi \rhd_{B\cup\{b\}} (P|R_P)\ \overline{\mathcal{S}}\ (Q|R_Q).$$

Using Scope,

$$\Psi \otimes \Psi_Q \rhd (\nu b)R \xrightarrow{\alpha} (\nu b)R_Q.$$

Using Lemma 4.24, we then get

$$\Psi \rhd_B (P|(\nu b)R_P)\ \overline{\mathcal{S}}\ (Q|(\nu b)R_Q).$$

— Par

In this case,

$$\mathcal{F}(R_1 \mid R_2) = (\nu \widetilde{b}_{R_1}\widetilde{b}_{R_2})\Psi_{R_1}\otimes\Psi_{R_2}$$

with

$$\widetilde{b}_{R_1}\#\widetilde{b}_{R_2}, \Psi_{R_2}$$
$$\widetilde{b}_{R_2}\#\widetilde{b}_{R_1}, \Psi_{R_1}.$$

The transition is derived like

$$\textsc{Par}\ \frac{\Psi_{R_2}\otimes\Psi\otimes\Psi_P \rhd R_1 \xrightarrow{\alpha} R_P}{\Psi\otimes\Psi_P \rhd R_1 \mid R_2 \xrightarrow{\alpha} R_P \mid R_2}\ \mathrm{bn}(\alpha)\#R_2$$

We know that

$$\widetilde{b}_P\#\Psi, R_1 \mid R_2,$$

and that

$$\widetilde{b}_{R_1}\widetilde{b}_{R_2}\#R_1 \mid R_2, \widetilde{b}_P,$$

so we also have

$$\widetilde{b}_P\#\Psi_{R_2}\otimes\Psi, R_1$$
$$\widetilde{b}_{R_1}\#\Psi_{R_2}\otimes\Psi, R_1.$$

By induction, we get R_Q such that

$$\Psi_{R_2}\otimes\Psi\otimes\Psi_Q \rhd R_1 \xrightarrow{\alpha} R_Q$$
$$\Psi_{R_2} \otimes \Psi \rhd_{B\cup\widetilde{b}_{R_2}} (P|R_P)\ \overline{\mathcal{S}}\ (Q|R_Q).$$

We then derive

$$\textsc{Par}\ \frac{\Psi_{R_2}\otimes\Psi\otimes\Psi_Q \rhd R_1 \xrightarrow{\alpha} R_Q}{\Psi\otimes\Psi_Q \rhd R_1 \mid R_2 \xrightarrow{\alpha} R_Q \mid R_2}\ \mathrm{bn}(\alpha)\#R_2$$

Finally, using Lemma 4.24, we get

$$\Psi \rhd_B (P|R_P|R_2)\ \overline{\mathcal{S}}\ (Q|R_Q|R_2).$$

— Com

In this case,

$$\mathcal{F}(R_1 \mid R_2) = (\nu\widetilde{b}_{R_1}\widetilde{b}_{R_2})\Psi_{R_1}\otimes\Psi_{R_2}$$

with

$$\widetilde{b}_{R_1}\#\widetilde{b}_{R_2}, \Psi_{R_2},$$

and *vice versa*. The transition is derived like

$$\textsc{Com}\ \frac{\begin{array}{c}\Psi_{R_2}\otimes\Psi\otimes\Psi_P \rhd R_1 \xrightarrow{\overline{M}(\nu\widetilde{a})N} R_{P1} \\ \Psi_{R_1}\otimes\Psi\otimes\Psi_P \rhd R_2 \xrightarrow{K\,N} R_{P2} \qquad \Psi\otimes\Psi_P\otimes\Psi_{R_1}\otimes\Psi_{R_2} \vdash M \leftrightarrow K\end{array}}{\Psi\otimes\Psi_P \rhd R_1 \mid R_2 \xrightarrow{\tau} (\nu\widetilde{a})(R_{P1} \mid R_{P2})}$$

We assume that $\widetilde{b}_P\#\widetilde{a}$ (otherwise α-convert $\widetilde{a}$ as necessary). Since $\widetilde{b}_P\#R_1 \mid R_2$, we get $\widetilde{b}_P\#N$. However, we cannot use the induction hypothesis directly since we do not know that $\widetilde{b}_P\#M$ and $\widetilde{b}_P\#K$, respectively.
Let $B_1 = \widetilde{b}_P \cup \widetilde{b}_{R_2}$. We have

$$\widetilde{b}_{R_1}\#\Psi_{R_2}, \Psi, \Psi_P, R_1, M, B'.$$

By Lemma 4.26, we get that there exists M' such that $B_1\#M'$ and

$$\Psi\otimes\Psi_P\otimes\Psi_{R_1}\otimes\Psi_{R_2} \vdash M \leftrightarrow M'.$$

Similarly, by applying Lemma 4.26 to the transition of R_2, we get K' such that $\widetilde{b}_P\widetilde{b}_{R_1}\#K'$ and

$$\Psi\otimes\Psi_P\otimes\Psi_{R_1}\otimes\Psi_{R_2} \vdash K \leftrightarrow K'.$$

By symmetry and transitivity of $\leftrightarrow$, we then get

$$\Psi\otimes\Psi_P\otimes\Psi_{R_1}\otimes\Psi_{R_2} \vdash M' \leftrightarrow K'.$$

By Lemma 4.27, we get

$$\Psi_{R_2}\otimes\Psi\otimes\Psi_P \rhd R_1 \xrightarrow{\overline{K'}(\nu\widetilde{a})N} R_{P1}$$

$$\Psi_{R_1}\otimes\Psi\otimes\Psi_P \rhd R_2 \xrightarrow{M'\,N} R_{P2}.$$

By induction, we get

$$\Psi_{R_2}\otimes\Psi\otimes\Psi_Q \rhd R_1 \xrightarrow{\overline{K'}(\nu\widetilde{a})N} R_{Q1}$$

$$\Psi_{R_1}\otimes\Psi\otimes\Psi_Q \rhd R_2 \xrightarrow{M'\,N} R_{Q2}$$

such that

$$\Psi_{R_2} \otimes \Psi \rhd_{B \cup \widetilde{b}_{R_2}} (P|R_{P1})\ \overline{\mathcal{S}}\ (Q|R_{Q1})$$
$$\Psi_{R_1} \otimes \Psi \rhd_{B \cup \widetilde{b}_{R_1}} (P|R_{P2})\ \overline{\mathcal{S}}\ (Q|R_{Q2}).$$

Since $\widetilde{b}_P \# K', M'$, we get

$$\Psi \otimes \mathcal{F}(P) \otimes \Psi_{R_1} \otimes \Psi_{R_2} \vdash M' \dot{\leftrightarrow} K'.$$

From

$$\Psi \otimes \Psi_{R_1} \otimes \Psi_{R_2} \rhd P \dot{\sim}^{\text{HO}} Q,$$

we then get

$$\Psi \otimes \mathcal{F}(Q) \otimes \Psi_{R_1} \otimes \Psi_{R_2} \vdash M' \dot{\leftrightarrow} K'.$$

Finally, we get

$$\Psi \otimes \Psi_Q \otimes \Psi_{R_1} \otimes \Psi_{R_2} \vdash M' \dot{\leftrightarrow} K',$$

which allows the derivation

$$\textsc{Com}\ \frac{\Psi_{R_2} \otimes \Psi \otimes \Psi_Q \rhd R_1 \xrightarrow{\overline{K'}(\nu \widetilde{a})N} R_{Q1} \qquad \Psi_{R_1} \otimes \Psi \otimes \Psi_Q \rhd R_2 \xrightarrow{M'\,N} R_{Q2} \qquad \Psi \otimes \Psi_Q \otimes \Psi_{R_1} \otimes \Psi_{R_2} \vdash M' \dot{\leftrightarrow} K'}{\Psi \otimes \Psi_Q \rhd R_1 \mid R_2 \xrightarrow{\tau} (\nu \widetilde{a})(R_{Q1} \mid R_{Q2})}$$

We now assume

$$\mathcal{F}(R_{P2}) = (\nu \widetilde{b}_{R_{P2}})\Psi_{R_{P2}}$$
$$\mathcal{F}(R_{Q1}) = (\nu \widetilde{b}_{R_{Q1}})\Psi_{R_{Q1}}$$

with

$$\widetilde{b}_{R_{Q1}} \# \widetilde{b}_{R_{P2}}$$
$$\widetilde{b}_{R_{Q1}} \widetilde{b}_{R_{P2}} \# \Psi, P, Q, R, N.$$

Since $\Psi_{R_{P2}} \simeq \Psi_{R_2} \otimes \Psi_2$ for some Ψ_2, we have

$$\Psi_{R_{P2}} \otimes \Psi \rhd_{B \cup \widetilde{b}_{R_2}} (P|R_{P1})\ \overline{\mathcal{S}}\ (Q|R_{Q1})$$

by Lemma 4.24. So we have

$$\Psi \rhd_B (P|R_{P1} \mid R_{P2})\ \overline{\mathcal{S}}\ (Q|R_{Q1} \mid R_{P2})$$

by Lemma 4.24 (1). Similarly,

$$\Psi \rhd_B (P|R_{Q1} \mid R_{P2})\ \overline{\mathcal{S}}\ (Q|R_{Q1} \mid R_{Q2}).$$

By the symmetry of $\dot{\sim}^{\text{HO}}$, we get

$$\Psi \otimes \Psi_R \rhd Q \dot{\sim}^{\text{HO}} P,$$

and by extension of arbitrary assertion (Definition 4.12.(3)), we get

$$\Psi \otimes \Psi_{R_{Q1}} \otimes \Psi_{R_{P2}} \rhd Q \dot{\sim}^{\text{HO}} P.$$

By the definition of $\mathcal{S}$, we get

$$\Psi \rhd (Q \mid R_{Q1} \mid R_{P2})\ \mathcal{S}\ (P \mid R_{Q1} \mid R_{P2}).$$

Since $B\#R_1, N$, we then have

$$\Psi \rhd_B (Q \mid R_{Q1} \mid R_{P2})\ \overline{\mathcal{S}}\ (P \mid R_{Q1} \mid R_{P2}).$$

By transitivity of $\overline{\mathcal{S}}$, we then get

$$\Psi \rhd_B (P \mid R_{P1} \mid R_{P2})\ \overline{\mathcal{S}}\ (Q \mid R_{Q1} \mid R_{Q2}).$$

Finally, using Lemma 4.24, we get

$$\Psi \rhd_B P \mid (\nu\widetilde{a})(R_{P1} \mid R_{P2})\ \overline{\mathcal{S}}\ Q \mid (\nu\widetilde{a})(R_{Q1} \mid R_{Q2}),$$

which completes this case. □

A variant of Lemma 4.28 treats the case where R makes a transition in the environment of P that can communicate with a transition of P in the environment of R. The processes R and Q can then perform matching transitions, leading to $\overline{\mathcal{S}}$-related derivatives.

Lemma 4.29 (subject switching lemma).

$$\begin{aligned}
&\Psi \otimes \Psi_R \rhd P \stackrel{\cdot}{\sim}^{\text{HO}} Q \wedge\\
&\mathcal{F}(P) = (\nu\widetilde{b}_P)\Psi_P \wedge\\
&\mathcal{F}(Q) = (\nu\widetilde{b}_Q)\Psi_Q \wedge\\
&\mathcal{F}(R) = (\nu\widetilde{b}_R)\Psi_R \wedge\\
&\Psi \otimes \Psi_R \rhd P \xrightarrow{\overline{M}(\nu\widetilde{a})N} P' \wedge\\
&\Psi \otimes \Psi_P \rhd R \xrightarrow{M\,N} R_P \wedge\\
&\Psi \otimes \Psi_P \otimes \Psi_R \vdash K \leftrightarrow M \wedge\\
&\widetilde{b}_P \# R, M, N, \Psi, P, Q \wedge\\
&\widetilde{b}_Q \# R, M, N, \Psi, P, Q \wedge\\
&\widetilde{b}_R \# K, N, \Psi, P, \widetilde{b}_P, \Psi_P, Q, \widetilde{b}_Q, \Psi_Q, R \wedge\\
&\widetilde{a} \# M, \Psi, P, Q, R, \widetilde{b}_P, \widetilde{b}_Q \wedge\\
&B \# P, Q, R, N, \widetilde{a}, \widetilde{b}_P, \widetilde{b}_Q, P'\\
&\quad\Longrightarrow \exists M', R_Q, Q'.\\
&\qquad \widetilde{b}_R, B \# M' \wedge\\
&\qquad \Psi \otimes \Psi_Q \otimes \Psi_R \vdash K \stackrel{\cdot}{\leftrightarrow} M' \wedge\\
&\qquad \Psi \otimes \Psi_Q \rhd R \xrightarrow{M'\,N} R_Q \wedge\\
&\qquad \Psi \otimes \Psi_R \rhd Q \xrightarrow{\overline{M}(\nu\widetilde{a})N} Q' \wedge\\
&\qquad \Psi \otimes \Psi_P \rhd_B (P' \mid R_P)\ \overline{\mathcal{S}}\ (Q' \mid R_Q).
\end{aligned}$$

Proof. The proof is by induction on the transition of R, and is similar to the proof of Lemma 4.28. □

The statement of Lemma 4.29 also needs to hold for output transitions of R, *mutatis mutandis*. We can then show the desired result.

Theorem 4.30. $\mathcal{S}$ is a HO-bisimulation up to $T \circ U \circ R$.

Proof sketch. We assume

$$\Psi \rhd P|R \; \mathcal{S} \; Q|R,$$

that is,

$$\Psi \otimes \mathcal{F}(R) \rhd P \; \dot{\sim}^{\text{HO}} \; Q.$$

Symmetry, extension with arbitrary assertion and static equivalence of conditions follow from the same properties of $\dot{\sim}^{\text{HO}}$. Static equivalence of clauses up to $T \circ U \circ R$ follows from the static equivalence of clauses of $\Psi \otimes \mathcal{F}(R) \rhd P \; \dot{\sim}^{\text{HO}} \; Q$.

We prove simulation up to $T \circ U \circ R$ by case analysis on the derivation of the transition of $P \mid R$, recalling that

$$\Psi \rhd P' \; (T(\mathcal{R})) \; Q'$$

if and only if

$$\Psi \rhd_{\varnothing} P' \; (T_{\text{a}}(\mathcal{R})) \; Q'.$$

We have:

— Par-L
This case follows from the bisimilarity of P and Q.

— Par-R
In this case,

$$\Psi \otimes \Psi_P \rhd R \xrightarrow{\alpha} R_P.$$

By Lemma 4.28, we have

$$\Psi \otimes \Psi_Q \rhd R \xrightarrow{\alpha} R_Q$$

with

$$\Psi \rhd_{\varnothing} P \mid R_P \; \overline{\mathcal{S}} \; Q \mid R_Q.$$

— Com-L
In this case,

$$\begin{gathered}
\Psi \otimes \Psi_R \rhd P \xrightarrow{\overline{K}\,(\nu \widetilde{a})N} P' \\
\Psi \otimes \Psi_P \rhd R \xrightarrow{M\,N} R_P \\
\Psi \otimes \Psi_P \otimes \Psi_R \vdash K \leftrightarrow M \\
\widetilde{b}_P \# K \\
\widetilde{b}_R \# M.
\end{gathered}$$

We may assume that $\widetilde{a} \# \Psi_R$.
By bisimilarity,

$$\Psi \otimes \Psi_R \rhd Q \xrightarrow{\overline{K}\,(\nu \widetilde{a})N} Q'$$

with

$$\Psi \otimes \Psi_R \rhd P' \; \dot{\sim}^{\text{HO}} \; Q'.$$

By Lemma 4.29, there are M', R_Q with $\widetilde{b}_R \# M'$ such that

$$\Psi \otimes \Psi_Q \otimes \Psi_R \vdash K \leftrightarrow M'$$

$$\Psi \otimes \Psi_Q \rhd R \xrightarrow{M'N} R_Q$$

$$\Psi \rhd_\varnothing (P' \mid R_P) \,\overline{\mathcal{S}}\, (Q' \mid R_Q).$$

Finally, by Lemma 4.24, we get

$$\Psi \rhd_\varnothing (\nu \widetilde{a})(P' \mid R_P) \,\overline{\mathcal{S}}\, (\nu \widetilde{a})(Q' \mid R_Q).$$

— Com-R

This case is the same as Com-L. □

It now follows that Theorem 4.16(1) holds.

Corollary 4.31. $P \dot{\sim}^{\text{HO}}_\Psi Q \Longrightarrow P \mid R \dot{\sim}^{\text{HO}}_\Psi Q \mid R$.

Proof. Assume that $\mathcal{F}(R) = (\nu \widetilde{b}_R)\Psi_R$ with $\widetilde{b}_R \# \Psi, P, Q$. By extension of arbitrary assertion, we get

$$P \dot{\sim}^{\text{HO}}_{\Psi \otimes \Psi_P} Q,$$

so

$$\Psi \rhd (P|R) \,\mathcal{S}\, (Q|R)$$

by the definition of $\mathcal{S}$. By Theorems 4.30 and 4.22, we then get $\mathcal{S} \subseteq \dot{\sim}^{\text{HO}}$, so $P \mid R \dot{\sim}^{\text{HO}}_\Psi Q \mid R$. □

4.5. *Formal proofs*

All theorems in this paper have been machine-checked with the interactive theorem prover Isabelle. The proof scripts (Åman-Pohjola and Raabjerg 2012) are adapted and extended from Bengtson's formalisation of psi-calculi (Bengtson 2010). These constitute 63,334 lines of Isabelle code; Bengtson's code is 37,417 lines. The bulk of the new code is related to Theorems 4.11 and 4.16, which have quite involved proofs that depart significantly from Bengtson's. It is interesting to observe how wildly the effort involved in conducting the proofs varies. We will briefly recount our experiences here.

With only minor modifications to Bengtson's proofs, we were able to re-prove all of the meta-theoretical results for psi-calculi (Theorems 4.2, 4.4, 4.5, and 4.6) in a matter of days. We believe that situations like these, where results need to be re-established under slightly different definitions, are among those where theorem provers truly shine.

By contrast, HO-bisimulation (Theorems 4.14, 4.16, and 4.13) is an example of where a small change to the definitions can give rise to man-months of work rather than days. This is because certain technical lemmas on which the old proofs depend are no longer valid in the context of HO-bisimulation. Hence, completely new proofs and proof ideas had to be developed. However, with HO-bisimulation in place, HO-congruence (Theorem 4.17) was mechanised in a matter of minutes.

The proofs related to canonical instances and the encoding of operators (*viz.* Theorems 3.6, 4.9, 4.10, and 4.11) also required man-months of work, but for different reasons.

Here, simple and intuitive proof ideas turned out to be cumbersome to mechanise. In the case of Theorem 3.6, the encoding of canonical instances is complicated and unintuitive, because of the necessity to side-step certain technical restrictions in the framework: for example, nominal datatype definitions cannot depend on locale parameters. Theorem 4.11 required almost 9,000 lines of proof script, even though the proof is conceptually simple. The main problem is the unwieldy candidate relation used for the proof, which includes many assumptions about the underlying psi-calculus. Moreover, it is closed under parallel composition and restriction, which significantly increases the size of the transition derivation trees we must follow and the amount of manual alpha-conversion we must perform, respectively. We believe that a much shorter proof can be obtained if a bisimulation up-to context technique is used instead, but we do not currently have a proof that such a technique is sound.

4.6. *Comparing higher-order equivalences*

Our definition of HO-bisimilarity is technically non-trivial, so we will provide some motivation for it in this section. Our primary concern was not to depart too much from the original bisimilarity since we have invested a substantial effort in an Isabelle proof repository and would like to re-use as much as possible. Therefore, our approach is to amend the original definition as little as possible in order to validate Theorem 4.13. Even so, there are a number of alternatives in the precise formulation of Clause (1b). The current definition requires in the conclusion that $\mathcal{R}(\mathbf{1}, P', Q')$, that is, that P' and Q' are again bisimilar in the assertion $\mathbf{1}$, which by Clause (3) is the same as requiring $\forall\Psi.\mathcal{R}(\Psi, P', Q')$. As a consequence, the following strengthening of Theorem 4.13 (note the assertions Ψ) is not true in general:

$$\Psi \rhd P \mathrel{\dot{\sim}}^{\text{HO}} Q \Rightarrow \Psi \rhd (\!|\Psi^{M \Leftarrow P}|\!) \mathrel{\dot{\sim}}^{\text{HO}} (\!|\Psi^{M \Leftarrow Q}|\!).$$

We have failed to define a version of higher-order bisimilarity where this holds. An obvious attempt is to adjust Clause (1b) to use $(\Psi, P', Q') \in \mathcal{R}$, that is, with Ψ in place of $\mathbf{1}$, but with this we fail to prove Theorem 4.16 (1), that is, that bisimilarity is preserved by parallel composition. The reason is that our proof strategy using the relation $\mathcal{S}$ in Section 4.4 relies on the fact that

$$\Psi \otimes \Psi' \rhd P \mathrel{\dot{\sim}} Q \quad \Rightarrow \quad \Psi \rhd (\!|\Psi'|\!) \mid P \mathrel{\dot{\sim}} (\!|\Psi'|\!) \mid Q.$$

This holds for ordinary bisimulation and for higher-order bisimulation, but will fail if Clause (1b) uses $(\Psi, P', Q') \in \mathcal{R}$. The counterexample is somewhat artificial and it remains to be seen whether we can formulate a subset of higher-order calculi where this property holds, or if there is a different proof strategy for Theorem 4.16.1 that does not require the property, and involving another candidate bisimulation relation.

Another possibility would be to include even more information in the assertion, as in $(\Psi \otimes \mathcal{F}(P), P', Q') \in \mathcal{R}$. In this case, we instead fail to establish that HO-bisimilarity is transitive; again we do not know if there is a counterexample. The problems are highly technical and are mainly concerned with how freshness conditions are propagated in the proofs.

The definition does not aspire to full abstraction with respect to observational criteria, and in this way it is very different from most existing work on higher-order calculi. It can immediately be seen that it is not complete in any sensible respect: the agents $\mathbf{0}$ and $(\!|\{M \Leftarrow \mathbf{0}\}|\!)$ should be indistinguishable from an observation viewpoint since neither has a transition and $M \Leftarrow \mathbf{0}$ does not give M any invocation possibilities, yet they fail bisimilarity on Clause (1b). On the other hand, it is straightforward to establish soundness for reasonable criteria. For example, say that a process P has the barb M if P has a transition with subject M, and that a congruence relation is barbed if related agents have the same barbs. Clause (4) in Definition 4.12 then directly gives us that HO-bisimilarity is barbed.

5. Conclusion

We have defined higher-order psi-calculi in a smooth extension from ordinary psi-calculi, meaning that we can re-use many of the mechanised proofs. Ordinary psi-calculi can be lifted in a systematic way to higher-order counterparts, yielding higher-order versions of the applied pi-calculus and the concurrent constraint pi-calculus.

We have also integrated the proofs with our existing proof repositories based on Isabelle/Nominal. In some cases, this process is surprisingly easy, but in others there are roadblocks related to the exact working of nominal datatypes with complicated constructors and locales. However, we regard this effort as worthwhile. For the main results, like Theorem 4.16, it is not efficient to embark on manual proofs, particularly as manual proofs in psi-calculi are notoriously error-prone because of their length, the number of cases to check and the numerous side conditions related to freshness of names.

There are several interesting avenues to explore. An obvious one is higher-order weak bisimulation and congruence. Here an immediate problem is that we can encode Sum, and therefore the usual example that weak bisimulation is not preserved by Sum may imply that it is not preserved by Parallel. For example, suppose we define weak higher-order bisimulation by adapting the weak bisimulation from Bengtson *et al.* (2010) in the same way as we do here for strong bisimulation. In other words, we require that a clause needs a weakly bisimilar clause. Now consider the agents

$$\begin{aligned} P &= (\!|\{M \Leftarrow \tau\,.\,a\,.\,\mathbf{0}\}|\!) \\ Q &= (\!|\{M \Leftarrow a\,.\,\mathbf{0}\}|\!) \\ R &= \mathbf{run}\ M \mid (\!|\{M \Leftarrow b\,.\,\mathbf{0}\}|\!). \end{aligned}$$

Here we have that P and Q are weakly higher-order bisimilar but $P|R$ and $Q|R$ are not. This indicates that a less straightforward definition will be necessary. An obvious attempt is to require that clauses are weakly congruent (rather than weakly bisimilar), and this requires that both weak congruence and weak bisimilarity are defined in one simultaneous co-inductive definition since each depends on the other.

The relationship between a calculus and its canonical higher-order counterpart should also be investigated. For example, bisimilarity on first-order processes is hopefully the same, and perhaps there is an interesting class of calculi where the canonical higher-order

calculus can be encoded. Finally, higher-order calculi should be combined with other extensions of the psi-calculi framework. We have successfully integrated higher-order calculi and ordinary bisimulation with the broadcast extension presented in Borgström *et al.* (2011). Here the total effort in the formalisation was roughly half a day, mainly to textually combine the proof files. This is a striking advantage of using formal proof repositories. We could also extend our recent work on sort systems (Borgström *et al.* 2013) to a higher-order setting.

In the invocation rule, the handle M must be exactly the same in the premise (where it occurs in $M \Leftarrow P$) and conclusion (where it occurs in **run** M). This means that we cannot directly describe the extraction of handles from complicated data structures. For example, consider one process defining two clauses $M_i \Leftarrow P_i$, and then sending the pair of the handles $\langle M_1, M_2 \rangle$. A receiving process might want to receive the pair and invoke its first element. Expressing this as $a(x)\,.\,\mathbf{run}\ \pi_1(x)$ will not work. After the communication of $\langle M_1, M_2 \rangle$, this becomes $\mathbf{run}\ \pi_1(\langle M_1, M_2 \rangle)$, but the environment contains $M_1 \Leftarrow P_1$ and not $\pi_1(\langle M_1, M_2 \rangle) \Leftarrow P_1$. What would be necessary here is a rewriting theory of projections, with axioms such as $\pi_1(\langle M_1, M_2 \rangle) \to M_1$, to be used in the entailment relation.

Most cases of simple extractions, such as projections, can be handled by pattern matching, as in the case $a(\lambda x, y)\langle x, y \rangle\,.\,\mathbf{run}\ x$. In more complicated structures, such as the representation of the encryption and decryption of handles, pattern matching will not be sufficient, and we must include information about the evaluation of handles in the assertions, where scoping can be used to make them local. This device is already present for communication subjects as the channel equivalence predicate. It remains to be seen if it is feasible to introduce a similar relation for handles.

Acknowledgements

We are very grateful to Magnus Johansson and Björn Victor for constructive and inspiring discussions.

References

Åman Pohjola, J. and Raabjerg, P. (2012) Isabelle proofs for higher-order psi-calculi. Proof scripts for higher-order psi-calculi. (Available at `http://www.it.uu.se/research/group/mobility/theorem/hopsi.tar.gz`.)

Bengtson, J. (2010) *Formalising process calculi*, Ph.D. thesis, Uppsala University.

Borgström, J., Gutkovas, R., Parrow, J., Victor, B. and Åman Pohjola, J. (2013) A Sorted Semantic Framework for High-Level Concurrency.

Borgström, J. *et al.* (2011) Broadcast psi-calculi with an application to wireless protocols. In: Barthe, G., Pardo, A. and Schneider, G. (eds.) Proceedings SEFM. *Springer-Verlag Lecture Notes in Computer Science* **7041** 74–89.

Bengtson, J., Johansson, M., Parrow, J. and Victor, B. (2009) Psi-calculi: Mobile processes, nominal data, and logic. In: *Proceedings of LICS 2009*, IEEE Computer Society 39–48. (Full version available at `http://user.it.uu.se/~joachim/psi-long.pdf`.)

Bengtson, J., Johansson, M., Parrow, J. and Victor, B. (2010) Weak equivalences in psi-calculi. In: *Proceedings of LICS 2010*, IEEE Computer Society 322–331.

Bengtson, J., Johansson, M., Parrow, J. and Victor, B. (2011) Psi-calculi: a framework for mobile processes with nominal data and logic. *Logical Methods in Computer Science* **7** (1) 2011.

Bengtson, J. and Parrow, J. (2009) Psi-calculi in Isabelle. In: Berghofer, S., Nipkow, T., Urban, C. and Wenzel, M. (eds.) Proceedings of TPHOLs 2009. *Springer-Verlag Lecture Notes in Computer Science* **5674** 99–114.

Demangeon, R., Hirschkoff, D. and Sangiorgi, D. (2009) Termination in higher-order concurrent calculi. In: Arbab, F. and Sirjani, M. (eds.) Proceedings FSEN. *Springer-Verlag Lecture Notes in Computer Science* **5961** 81–96.

Gabbay, M. and Pitts, A. (2001) A new approach to abstract syntax with variable binding. *Formal Aspects of Computing* **13** 341–363.

Johansson, M. (2010) *Psi-calculi: a framework for mobile process calculi*, Ph.D. thesis, Uppsala University.

Jeffrey, A. and Rathke, J. (2005) Contextual equivalence for higher-order pi-calculus revisited. *Logical Methods in Computer Science* **1** (1).

Johansson, M., Victor, B. and Parrow, J. (2010) A fully abstract symbolic semantics for psi-calculi. In: Proceedings of SOS 2009. *Electronic Proceedings in Theoretical Computer Science* **18** 17–31.

Lanese, I., Pérez, J. A., Sangiorgi, D. and Schmitt, A. (2008) On the expressiveness and decidability of higher-order process calculi. In: *Proceedings LICS*, IEEE Computer Society 145–155.

Lanese, I., Pérez, J. A., Sangiorgi, D. and Schmitt, A. (2010) On the expressiveness of polyadic and synchronous communication in higher-order process calculi. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F. and Spirakis, P. G. (eds.) Proceedings ICALP (2). *Springer-Verlag Lecture Notes in Computer Science* **6199** 442–453.

Pitts, A. M. (2003) Nominal logic, a first order theory of names and binding. *Information and Computation* **186** 165–193.

Sangiorgi, D. (1993) From pi-calculus to higher-order pi-calculus – and back. In: Gaudel, M.-C. and Jouannaud, J.-P. (eds.) Proceedings TAPSOFT. *Springer-Verlag Lecture Notes in Computer Science* **668** 151–166.

Sangiorgi, D. (1996) Bisimulation for higher-order process calculi. *Information and Computation* **131** (2) 141–178.

Sangiorgi, D. (1998) On the bisimulation proof method. *Mathematical Structures in Computer Science* **8** (5) 447–479. (An extended abstract also appeared in the Proceedings of MFCS '95, *Springer-Verlag Lecture Notes in Computer Science* **969** 479–488.)

Sangiorgi, D. (2001) Asynchronous process calculi: the first- and higher-order paradigms. *Theoretical Computer Science* **253** (2) 311–350.

Thomsen, B. (1989) A calculus of higher order communicating systems. In: *Proceedings POPL*, ACM Press 143–154.

Thomsen, B. (1993) Plain CHOCS: A second generation calculus for higher order processes. *Acta Informatica* **30** (1) 1–59.

Urban, C. (2008) Nominal techniques in Isabelle/HOL. *Journal of Automated Reasoning* **40** (4) 327–356.