Priorities Without Priorities: Representing Preemption in Psi-Calculi

Johannes Åman Pohjola
Uppsala University, Sweden
johannes.aman-pohjola@it.uu.se

Joachim Parrow
Uppsala University, Sweden
joachim.parrow@it.uu.se

Psi-calculi is a parametric framework for extensions of the pi-calculus with data terms and arbitrary logics. In this framework there is no direct way to represent action priorities, where an action can execute only if all other enabled actions have lower priority. We here demonstrate that the psi-calculi parameters can be chosen such that the effect of action priorities can be encoded.

To accomplish this we define an extension of psi-calculi with action priorities, and show that for every calculus in the extended framework there is a corresponding ordinary psi-calculus, without priorities, and a translation between them that satisfies strong operational correspondence. This is a significantly stronger result than for most encodings between process calculi in the literature.

We also formally prove in Nominal Isabelle that the standard congruence and structural laws about strong bisimulation hold in psi-calculi extended with priorities.

1 Introduction

Priorities in process calculi allow certain actions to take precedence over others. This is useful when modelling systems because it admits more fine-grained control over the model's behaviour. Phenomena that exhibit prioritised behaviour include eg. interrupts in operating systems, and exception handling in programming languages. In this paper we demonstrate how priorities can be represented in the psi-calculi framework, by encoding them into the logical theory that determines how actions are generated by process syntax.

Psi-calculi [3] is a family of applied process calculi that generalises the pi-calculus in three ways. First, the subjects (designating the communication channels) and objects (designating the communicated data) of input and output actions may be *terms* taken from an arbitrary set, and not just single names. Second, equality tests on names are replaced by tests of predicates called *conditions*, taken from an arbitrary logic. Finally, the process syntax is extended with *assertions*, which can be seen as introducing new facts about the environment in which a process executes. The unguarded assertions of a process influence the evaluation of conditions and the connectivity between channel terms, and can change as the process executes.

In this paper, we show that the psi-framework is sufficiently expressive to represent action priorities derived from a priority order on the communication channels. We are interested in priorities for two reasons. First, previous work on priorities indicate that they are highly expressive: Jeffrey defines a process calculus with time and priority where timed processes can be encoded in the untimed fragment of the calculus [15]; Jensen shows that CCS augmented with priority choice can encode broadcast communication [16]; and Versari et al. shows that CCS with priority and only the prefix and parallel operators can solve both leader election (unlike the pi-calculus), and the "last man standing"-problem (unlike the broadcast pi-calculus) [23]. Second, we are not aware of another process calculus (without priorities) where adding priorities has been shown to yield no increased expressiveness. The prevailing

J. Borgström, S. Crafa (Eds.): Combined Workshop on Expressiveness in Concurrency and Structural Operational Semantics (EXPRESS/SOS 2014)
EPTCS 160, 2014, pp. 2–15, doi:10.4204/EPTCS.160.2

methods to introduce priorities in process algebras are through semantic rules with negative premises or new auxiliary relations to express the absence of higher-priority actions; we shall need none of those.

We accomplish our result as follows. First we define an extension of the psi-calculi framework with explicit channel priorities, where the priority level of a channel can change dynamically during process execution, as defined by an auxiliary relation representing absence of actions. We formally prove, using the interactive theorem prover Isabelle [18], that in this setting strong bisimilarity satisfies the usual algebraic laws and congruence properties familiar from the pi-calculus. We proceed to show that for every psi-calculus with priorities, separate choice and prefix-guarded replication, it can be encoded in a standard psi-calculus without priorities. This encoding satisfies particularly strong quality criteria, namely strong operational correspondence, meaning that the translation does not introduce any protocol in the target language. The main idea is that we use a non-monotonic logic for the assertions, where the appearance of enabled high-priority channels can temporarily prevent lower priority channels from resulting in actions.

The rest of the paper is structured as follows. In Section 2 we briefly recapitulate the essentials of psi-calculi, and in Section 3 we define the extension with explicit channel priorities. Section 4 contains an encoding into standard psi-calculi. In Section 5 we establish strong operational correspondence and briefly discuss other criteria for encodings, among them full abstraction, and Section 6 contains conclusions with future work.

Full proofs of all theorems presented in this paper are available online at `http://www.it.uu.se/research/group/mobility/prio-proofs.pdf`.

2 Psi-calculi

The following is a quick recapitulation of the psi-calculi framework. For an in-depth introduction with motivations and examples we refer the reader to [3].

We assume a countably infinite set of atomic *names* $\mathcal{N}$ ranged over by $a, b, \ldots, z$. Intuitively, names are the symbols that can be scoped and be subject to substitution. A *nominal set* [21, 12] is a set equipped with a formal notion of what it means to swap names in an element; this leads to a notion of when a name a occurs in an element X, written $a \in \mathrm{n}(X)$ (pronounced "a is in the support of X"). We write $a\#X$, pronounced "a is fresh for X", for $a \notin \mathrm{n}(X)$, and if A is a set of names we write $A\#X$ to mean $\forall a \in A \,.\, a\#X$. In the following $\tilde{a}$ is a finite sequence of names. The empty sequence is written ε and the concatenation of $\tilde{a}$ and $\tilde{b}$ is written $\tilde{a}\tilde{b}$. We say that a function symbol is *equivariant* if all name swappings distribute over it.

A *nominal datatype* is a nominal set together with a set of functions on it. In particular we shall consider substitution functions that substitute elements for names. If X is an element of a datatype, $\tilde{a}$ is a sequence of names without duplicates and $\tilde{Y}$ is an equally long sequence of elements of possibly another datatype, the *substitution* $X[\tilde{a} := \tilde{Y}]$ is an element of the same datatype as X. The substitution function can be chosen freely, but must satisfy certain natural laws regarding the treatment of names; it must be equivariant, the names $\widetilde{a}$ in $X[\widetilde{a} := \widetilde{T}]$ must be alpha-convertible as if they were binding in X. See [3] for details.

A psi-calculus is defined by instantiating three nominal data types and four equivariant operators; formally it is a tuple $(\mathbf{T}, \mathbf{A}, \mathbf{C}, \vdash, \otimes, \leftrightarrows, \mathbf{1})$ as follows.

Definition 1 (Psi-calculus parameters)**.** *A psi-calculus requires the three (not necessarily disjoint) nominal data types: the (data) terms* $\mathbf{T}$*, ranged over by* M, N*, the conditions* $\mathbf{C}$*, ranged over by* φ*, the*

assertions $\mathbf{A}$*, ranged over by* Ψ*, and the four operators:*

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

The binary functions above will be written in infix. Thus, $M \leftrightarrows N$ is a condition, pronounced "M and N are channel equivalent". We write $\Psi \vdash \varphi$, pronounced "Ψ entails φ", for $(\Psi,\varphi)\in\vdash$, and if Ψ and Ψ' are assertions then so is $\Psi\otimes\Psi'$, which intuitively represents the conjunction of the information in Ψ and Ψ'.

We say that two assertions are *statically equivalent*, written $\Psi \simeq \Psi'$ if they entail the same conditions, i.e. for all φ we have that $\Psi\vdash\varphi$ iff $\Psi'\vdash\varphi$. We impose certain requisites on the sets and operators: channel equivalence must be symmetric and transitive, $\otimes$ must be compositional with regard to $\simeq$, and the assertions with $(\otimes,\mathbf{1})$ form an abelian monoid modulo $\simeq$. Finally, substitution $M[\tilde{a}:=\tilde{T}]$ on terms must be such that if the names $\tilde{a}$ are in the support of M, the support of $\tilde{T}$ must be in the support of $M[\tilde{a}:=\tilde{T}]$.

A *frame* is an assertion together with a sequence of names that bind into it: it is of the form $(\nu\tilde{b})\Psi$ where $\tilde{b}$ binds into the assertion Ψ. We use F,G to range over frames. We overload Ψ to also mean $(\nu\varepsilon)\Psi$ and $\otimes$ to 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 write $\Psi\otimes F$ to mean $(\nu\varepsilon)\Psi\otimes F$, and $(\nu c)((\nu\tilde{b})\Psi)$ for $(\nu c\tilde{b})\Psi$.

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 it holds that $F\vdash\varphi$ iff $G\vdash\varphi$.

Definition 2 (Psi-calculus agents)**.** *Given a psi-calculus* $\mathscr{P}$ *with parameters as in Definition 1, the* agents $\mathbf{P}(\mathscr{P})$*, 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
$(\!\vert\Psi\vert\!)$	Assertion

Restriction $(\nu a)P$ *binds* a *in* P *and input* $\underline{M}(\lambda\tilde{x})N.P$ *binds* $\tilde{x}$ *in both* N *and* P*. An occurrence of a subterm in an agent is* guarded *if it is a proper subterm of an input or output term. An agent is* assertion guarded *if it contains no unguarded assertions. An agent is* well-formed *if in* $\underline{M}(\lambda\tilde{x})N.P$ *it holds that* $\tilde{x}\subseteq \mathrm{n}(N)$ *is a sequence without duplicates, that in a replication* $!P$ *the agent* P *is assertion guarded, and that in* $\mathbf{case}\ \varphi_1:P_1\,[]\cdots[]\,\varphi_n:P_n$ *the agents* P_i *are assertion guarded.*

The agent $\mathbf{case}\ \varphi_1:P_1\,[]\cdots[]\,\varphi_n:P_n$ is sometimes abbreviated as $\mathbf{case}\ \tilde{\varphi}:\tilde{P}$. We sometimes write $\underline{M}(x).P$ for $\underline{M}(\lambda x)x.P$. From this point on, we only consider well-formed agents.

The *frame* $\mathscr{F}(P)$ *of an agent* P is defined inductively as follows:

$$\mathscr{F}(\underline{M}(\lambda\tilde{x})N.P)=\mathscr{F}(\overline{M}N.P)=\mathscr{F}(\mathbf{0})=\mathscr{F}(\mathbf{case}\ \tilde{\varphi}:\tilde{P})=\mathscr{F}(!P)=\mathbf{1}\qquad \mathscr{F}((\!\vert\Psi\vert\!))=(\nu\varepsilon)\Psi$$

$$\mathscr{F}(P\mid Q)=\mathscr{F}(P)\otimes\mathscr{F}(Q)\qquad \mathscr{F}((\nu b)P)=(\nu b)\mathscr{F}(P)$$

The *actions* ranged over by α,β are of the following three kinds:
Output $\overline{M}(\nu\tilde{a})N$*, input* $\underline{M}\,N$, and *silent* τ. Here we refer to M as the *subject* and N as the *object*. We

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

$$\textsc{Case}\ \frac{\Psi \rhd P_i \xrightarrow{\alpha} P' \qquad \Psi \vdash \varphi_i}{\Psi \rhd \mathbf{case}\ \tilde{\varphi} : \tilde{P} \xrightarrow{\alpha} P'} \qquad \textsc{Par}\ \frac{\Psi_Q \otimes \Psi \rhd P \xrightarrow{\alpha} P'}{\Psi \rhd P \mid Q \xrightarrow{\alpha} P' \mid Q}\ \mathrm{bn}(\alpha)\#Q$$

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

$$\textsc{Rep}\ \frac{\Psi \rhd P \mid !P \xrightarrow{\alpha} P'}{\Psi \rhd !P \xrightarrow{\alpha} P'} \qquad \textsc{Scope}\ \frac{\Psi \rhd P \xrightarrow{\alpha} P'}{\Psi \rhd (\nu b)P \xrightarrow{\alpha} (\nu b)P'}\ b\#\alpha,\Psi$$

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

Table 1: Structured operational semantics. Symmetric versions of COM and PAR are elided. In the rule COM we assume that $\mathscr{F}(P) = (\nu\tilde{b}_P)\Psi_P$ and $\mathscr{F}(Q) = (\nu\tilde{b}_Q)\Psi_Q$ where $\tilde{b}_P$ is fresh for all of $\Psi, \tilde{b}_Q, Q, M$ and P, and that $\tilde{b}_Q$ is similarly fresh. In the rule PAR we assume that $\mathscr{F}(Q) = (\nu\tilde{b}_Q)\Psi_Q$ where $\tilde{b}_Q$ is fresh for Ψ, P and α. In OPEN the expression $\tilde{a} \cup \{b\}$ means the sequence $\tilde{a}$ with b inserted anywhere.

define $\mathrm{bn}(\overline{M}(\nu\tilde{a})N) = \tilde{a}$, and $\mathrm{bn}(\alpha) = \emptyset$ if α is an input or τ. 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 $\tilde{a}$.

Definition 3 (Transitions)**.** *A* transition *is written* $\Psi \rhd P \xrightarrow{\alpha} P'$*, meaning that in the environment* Ψ*,* P *can do* α *to become* P'*. The transitions are defined inductively in Table 1.We abbreviate* $\mathbf{1} \rhd P \xrightarrow{\alpha} P'$ *as* $P \xrightarrow{\alpha} P'$*.*

We identify alpha-equivalent agents, frames and transitions. In a transition the names in $\mathrm{bn}(\alpha)$ bind into both the action object and the derivative, therefore $\mathrm{bn}(\alpha)$ is in the support of α but not in the support of the transition.

Definition 4 (Strong bisimulation)**.** *A* strong bisimulation $\mathscr{R}$ *is a ternary relation on assertions and pairs of agents such that* $\mathscr{R}(\Psi,P,Q)$ *implies*

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

We define $\Psi \rhd P \mathrel{\dot\sim} Q$ *to mean that there exists a bisimulation* $\mathscr{R}$ *such that* $\mathscr{R}(\Psi, P, Q)$*, and write* $P \mathrel{\dot\sim} Q$*, pronounced P and Q are* (strongly) bisimilar*, for* $\mathbf{1} \rhd P \mathrel{\dot\sim} Q$*.*

Definition 5 (Strong congruence)**.** *We define* $P \sim_\Psi Q$ *to mean that for all substitution sequences* σ*,* $\Psi \rhd P\sigma \mathrel{\dot\sim} Q\sigma$ *holds. We write* $P \sim Q$*, pronounced P is* (strongly) congruent *to Q, to mean* $P \sim_{\mathbf{1}} Q$*.*

We have shown [3] that strong bisimilarity preserves all operators except input, and that strong congruence is a congruence and satisfies the expected algebraic laws for structural congruence.

3 Extension: Psi-calculi with priorities

The most common approaches to implementing priorities in process calculi are (1) to add a priority operator Θ such that $\Theta(P)$ may only take the highest-priority actions of P as defined by some ordering on actions [1], and (2) to alway enforce priorities, rather than only at special operators [9, 10]. In order to avoid introducing a new operator, we follow the second approach.

We associate a priority level to actions that may depend on the assertion environment, and hence change dynamically as a process evolves. The intuition is that we write $\Psi \vdash M : p$ to mean that the priority level of communication on the channel M in the environment Ψ is p, where lower values of p indicate *higher* priority. Priorities are subject to some natural constraints: they must be equivariant, and in a given assertion, channel equivalent terms must have the same unique priority level.

Definition 6 (Psi-calculi with priorities)**.** *A* psi-calculus with priorities*, ranged over by* $\mathscr{P}, \mathscr{Q}$*, is a tuple* $(\mathbf{T}, \mathbf{A}, \mathbf{C}, \vdash, \otimes, \leftrightarrows, \mathbf{1}, :)$ *such that*

1. $(\mathbf{T}, \mathbf{A}, \mathbf{C}, \vdash, \otimes, \leftrightarrows, \mathbf{1})$ *is a psi-calculus, and*
2. $:$ *of type* $\mathbf{T} \times \mathbb{N} \Rightarrow \mathbf{C}$ *is an equivariant operator written in infix, i.e., we write* $M : p$ *for* $:(M, p)$*, such that for all* Ψ, M, N*, if* $\Psi \vdash M \leftrightarrows N$ *then there is a unique* $p \in \mathbb{N}$ *such that* $\Psi \vdash M : p$ *and* $\Psi \vdash N : p$*.*

The semantics of psi-calculi with priorities is as the semantics of psi-calculi, but with two changes. The first is that τ actions are replaced with $\tau : p$ actions, where p is the priority level of the transition. The second is that the rules are augmented with side conditions that prevent a process from taking low priority actions. This has a natural formulation in terms of negative premises [5], but in order to make implementation in Isabelle easier we instead define the semantics in two layers, following [9, 10, 22].

The bottom layer is denoted with the transition arrow $\longrightarrow_{\mathtt{b}}$ and is used to determine which transitions would be available, disregarding priorities. The semantics of $\longrightarrow_{\mathtt{b}}$ is exactly as in Table 1 with the sole extension that the COM rule generates an action of kind $\tau : p$, where p is derived from the priority of the channel. We then define a predicate $\mathtt{H}(\alpha, \Psi, P)$, which intuitively means that no τ transition whose priority is higher than that of α can be derived from P in Ψ. Finally we define $\longrightarrow_{\mathtt{p}}$ to represent transitions respecting priorities, where the CASE, PAR, and COM rules get side conditions using $\mathtt{H}$.

Definition 7.

$$\mathtt{H}(\alpha, \Psi, P) \triangleq \neg\exists n\, P'.(\Psi \rhd P \xrightarrow{\tau:n}_{\mathtt{b}} P' \wedge n < \mathtt{PRIO}(\Psi \otimes \mathscr{F}(P), \alpha))$$

where $\mathtt{PRIO}(F, \alpha)$ *is defined to be* p *if either* $\alpha = \tau : p$ *or* $F \vdash \mathrm{subj}(\alpha) : p$*, and* $\longrightarrow_{\mathtt{b}}$ *is defined in Definition 8 below.*

Definition 8 (Transitions with priorities)**.** *The transitions of psi-calculi with priorities are defined inductively by the same rules as in Table 1, but with all occurrences of* $\longrightarrow$ *replaced with* $\longrightarrow_{\mathtt{p}}$*, and the* CASE*,* COM *and* PAR *rules replaced by the following:*

$$\textsc{Case}\ \frac{\Psi \rhd P_i \xrightarrow{\alpha}_{\mathrm{p}} P' \qquad \Psi \vdash \varphi_i}{\Psi \rhd \mathbf{case}\ \widetilde{\varphi} : \widetilde{P} \xrightarrow{\alpha}_{\mathrm{p}} P'}\ \mathtt{H}(\alpha, \Psi, \mathbf{case}\ \widetilde{\varphi} : \widetilde{P})$$

$$\textsc{Par}\ \frac{\Psi_Q \otimes \Psi \rhd P \xrightarrow{\alpha}_{\mathrm{p}} P'}{\Psi \rhd P \mid Q \xrightarrow{\alpha}_{\mathrm{p}} P' \mid Q}\ \begin{array}{l}\mathtt{H}(\alpha, \Psi, P \mid Q)\\ \mathrm{bn}(\alpha)\#Q\end{array}$$

$$\textsc{Com}\ \frac{\begin{array}{cc}\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K & \Psi \otimes \Psi_P \otimes \Psi_Q \vdash M : p\\ \Psi_Q \otimes \Psi \rhd P \xrightarrow{\overline{M}(\nu \widetilde{a})N}_{\mathrm{p}} P' & \Psi_P \otimes \Psi \rhd Q \xrightarrow{\underline{K}N}_{\mathrm{p}} Q'\end{array}}{\Psi \rhd P \mid Q \xrightarrow{\tau:p}_{\mathrm{p}} (\nu \widetilde{a})(P' \mid Q')}\ \begin{array}{l}\mathtt{H}(\tau : p, \Psi, P \mid Q)\\ \widetilde{a}\#Q\end{array}$$

The transition relation $\longrightarrow_{\mathtt{b}}$ *is defined by the same rules as* $\longrightarrow_{\mathrm{p}}$*, but with all side conditions involving* H *omitted.*

Strong bisimulation and strong congruence on psi-calculi with priorities can be obtained from Definitions 4-5 by replacing all occurrences of $\longrightarrow$ with $\longrightarrow_{\mathrm{p}}$. The meta-theory pertaining to strong bisimulation from the original psi-calculi carries over to psi-calculi with priorities, and formal proofs in Isabelle have been carried out:

Theorem 1. *Strong congruence* $\sim$ *on psi-calculi with priorities is a congruence, and satisfies*

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

As an example, Versari's $\pi@$ [22] is an extension of the pi-calculus with priorities. Input and output prefixes in $\pi@$ are of form $\mu : k(y)$ and $\overline{\mu} : k\langle z \rangle$, where μ is the subject, k is the priority level and y and z are the objects. The semantics is the standard reduction semantics of the pi-calculus, augmented with side conditions stating that no higher-priority reduction is possible, similar to our use of the H predicate.

$\pi@$ can be recovered in our framework as follows. For simplicity we consider only monadic synchronisation. Let the terms be the union of $\mathscr{N}$ (corresponding to objects in $\pi@$) and $\{a : n | a \in \mathscr{N}, n \in \mathbb{N}\}$ (corresponding to subjects annotated with their priority level), let the conditions be the booleans and the assertions be $\{\mathbf{1}\}$. Define channel equivalence and : so that $a : n$ is equivalent to itself and has priority n.

As an immediate consequence, we equip $\pi@$ with a labelled semantics and a theory of strong bisimulation; no labelled semantics or bisimulation theory has been previously developed for $\pi@$.

Note that in our representation of $\pi@$, it is possible to write agents where the term $a : n$ occurs in object position. We can rule out such ill-formed agents by using the sort system for psi-calculi described in [6], the details of which are beyond the scope of the present paper.

For a slightly more involved example, we consider dynamic priorities. We define a psi-calculus with priorities based on the pi-calculus, with the addition that channels may have one of two priority levels: 0 (high) and 1 (low). Rather than annotating prefixes with a priority level, we let channels have high priority by default, and let our assertions be the set of channels whose priority have been flipped to low priority. If a channel is asserted to be flipped twice, the assertions cancel each other and the channel is flipped back to high priority. Thus we may flip the priority of a channel a dynamically by asserting $\{a\}$. Similarly, asserting $\{a,b\}$ flips the priorities of both a and b. Composition of assertions is exclusive or, e.g. $\{a\} \otimes \{a,b\} = \{b\}$. To illustrate how this calculus can be used, suppose we want to enforce a fairness scheme such that synchronisations on two channels x and y are guaranteed to interleave. This can be achieved by swapping the priorities of x and y after every such synchronisation, as in the following derivation sequence, where for all $z \in \{x,y\}$ we let $P_z = (\!|\{z\}|\!) \mid !\overline{x}.(\!|\{x,y\}|\!) \mid !\overline{y}.(\!|\{x,y\}|\!)$.

$$\begin{array}{lcll} \mathbf{1} & \rhd & P_y \mid x.x.x \mid y & \xrightarrow{\tau:0}_{\mathtt{p}} \quad P_x \mid x.x \mid y \\ & & & \xrightarrow{\tau:0}_{\mathtt{p}} \quad P_y \mid x.x \\ & & & \xrightarrow{\tau:0}_{\mathtt{p}} \quad P_x \mid x \\ & & & \xrightarrow{\tau:1}_{\mathtt{p}} \quad P_y \end{array}$$

Note that the above τ sequence is the only possible τ sequence — as long as both x and y are available they are guaranteed to be consumed alternatingly.

Formally, we define this psi-calculus by letting $\mathbf{T} = \mathscr{N}$, $\mathbf{C} = \{x = y \mid x,y \in \mathbf{T}\} \cup \{M : n \mid M \in \mathbf{T} \wedge n \in \mathbb{N}\}$ and by letting $\mathbf{A}$ be the finite sets of names. Moreover, let $\mathbf{1}$ be the empty set and $A \otimes B = (A \cup B) - (A \cap B)$. Entailment is defined so that $\Psi \vdash x = y$ iff $x = y$, $\Psi \vdash x : 1$ iff $x \in \Psi$, and $\Psi \vdash x : 0$ iff $x \notin \Psi$. Finally, we let channel equivalence be syntactic equality on names.

The definition of composition as the pairwise exclusive or on the elements of its arguments achieves the priority flip in a manner that is associative, commutative and compositional. This is a useful general technique for constructing psi-calculi where facts can be retracted.

4 Encoding priorities

In this section we present a translation from psi-calculi with priorities to the original psi-calculi. The main idea is that we augment the assertions with information about prefixes, and ensure that the frame of a process records precisely its enabled prefixes. The $\mathtt{H}$ predicate is thus obtained from the entailment relation.

The main technical complication with this idea is that when P takes a transition to P', some of the top-level prefixes of P may be absent in P'. The frame of P' will always be the frame of P composed with assertions that are guarded in P and unguarded in P'; in other words $\mathscr{F}(P') \simeq (\nu \widetilde{b_{P'}})(\Psi_P \otimes \Psi)$. It follows that composing with this Ψ must in effect retract the prefixes no longer available in P' from Ψ_P. For this purpose we use a non-monotonic logic, where assertions contain multisets with negative occurrence [4].

4.1 Preliminaries: integer-indexed multisets

Intuitively, an integer-indexed multiset is like a regular multiset, except that the number of occurrences of an element may be negative. We use *finite integer-indexed multisets with a maximum element* (henceforth abbreviated *FIMM*), ranged over by E. Let $\mathbb{Z}^\infty$ denote $\mathbb{Z} \cup \{\infty\}$. Formally, the FIMMs over a set S is the

set of functions $E : S \Rightarrow \mathbb{Z}^\infty$ such that for all but finitely many elements $s \in S$, $E(s) = 0$. We define some of the usual operations on sets as follows:

$$x \in E \triangleq E(x) > 0 \qquad \emptyset \triangleq \lambda x.0 \qquad E \cup E' \triangleq \lambda x.(E(x) + E'(x))$$

The maximal element ∞ will be used to represent prefixes under a replication operator (these are permanently enabled and cannot ever be retracted). We will write $\{(z_0)x_0, \ldots, (z_n)x_n\}$ for the multiset E such that $E(x_i) = z_i$ if $0 \leq i \leq n$, and $E(x_i) = 0$ otherwise. We will sometimes write x_i to mean $(1)x_i$ and $-x_i$ to mean $(-1)x_i$.

4.2 Preliminaries: Requisites and guarding elements

From this point in the paper, we restrict attention to psi-calculi with separate choice and prefix-guarded replication. In other words, case statements have the form **case** $\widetilde{\varphi} : \widetilde{\alpha.P}$, where either every α_i is an input, or every α_i is an output. Moreover, replications are of the form $!\alpha.P$. These restrictions significantly simplify our definitions and proofs. In the conclusion we briefly discuss what would be involved to lift them.

We also require that substitution has no effect on terms where the names being substituted do not occur, i.e. that if $\widetilde{x}\#M$ then $M[\widetilde{x} := \widetilde{T}] = M$. This natural requirement on substitution is found in the original publication on psi-calculi [2], but is often omitted since it is not needed for the standard structural and congruence properties of bisimulation.

Further, for convenience we will assume that the psi-calculus under consideration has a condition $\top$ that is always true in every context, i.e. it is such that $\forall\Psi.\Psi \vdash \top$, $\forall\sigma.\top\sigma = \top$ and $\mathsf{n}(\top) = \emptyset$. If such a condition is absent, it can simply be added.

A *guarding element* is simply a prefix guarded by a condition. Enriching the assertions with FIMMs of guarding elements will provide all the information necessary to encode H in the entailment relation.

Definition 9 (guarding elements)**.** *The set of* guarding elements *of a psi-calculus* $\mathscr{P} = (\mathbf{T}, \mathbf{A}, \mathbf{C}, \vdash, \otimes, \leftrightarrow, \mathbf{1})$ *is denoted* $\mathbf{F}(\mathscr{P})$ *and defined as*

$$\mathbf{F}(\mathscr{P}) = \mathbf{C} \times (\{\overline{M}N : M, N \in \mathbf{T}\} \cup \{\underline{M}(\lambda\widetilde{x})X : M, N \in \mathbf{T}\})$$

We consider guarding elements as implicitly quotiented by alpha-equivalence, where the names $\widetilde{x}$ *in the input prefix* $\underline{M}(\lambda\widetilde{x})X$ *bind into* N*. We will sometimes write* α *to mean* $(\top, \alpha)$*.*

4.3 The encoding

Assume a psi-calculus with priorities $\mathscr{P} = (\mathbf{T}, \mathbf{A}, \mathbf{C}, \vdash, \otimes, \leftrightarrow, \mathbf{1}, :)$. We shall encode it in the psi-calculus $\mathscr{Q} = (\mathbf{T}, \mathbf{A}', \mathbf{C}', \vdash', \otimes', \leftrightarrow', (\mathbf{1}, \emptyset))$, whose parameters are defined as follows:

$$\begin{array}{rcl}
\mathbf{A}' & = & \mathbf{A} \times (\mathbf{F}(\mathscr{P}) \Rightarrow \mathbb{Z}^\infty) \\
\mathbf{C}' & = & \mathbf{C} \uplus (\mathbb{Z}^\infty \times \mathbf{F}(\mathscr{P})) \uplus \{M \leftrightarrow' N : M, N \in \mathbf{T}\} \\
(\Psi, E) \otimes' (\Psi', E') & = & (\Psi \otimes \Psi', E \cup E') \\
\\
(\Psi, E) \vdash' \varphi & = & \Psi \vdash \varphi \qquad \text{if } \varphi \in \mathbf{C} \\
(\Psi, E) \vdash' (z)(\varphi, \alpha) & = & E(\varphi, \alpha) = z \\
(\Psi, E) \vdash' M \leftrightarrow' N & = & \Psi \vdash M \leftrightarrow N \wedge \neg\exists M'\, N'\, n\, m\, X\, K\, \widetilde{x}\, \widetilde{L}\, \varphi\, \varphi'.\Psi \vdash M' \leftrightarrow N' \\
 & & \wedge \Psi \vdash M : m \wedge \Psi \vdash M' : n \wedge n < m \wedge (\varphi, \underline{M'}(\lambda\widetilde{x})X) \in E \\
 & & \wedge (\varphi', \overline{N'}K) \in E \wedge K = X[\widetilde{x} := \widetilde{L}] \wedge \Psi \vdash \varphi \wedge \Psi \vdash \varphi'
\end{array}$$

Assertions in $\mathbf{A}'$ augment the original assertions with FIMMs of guarding elements, representing the top-level prefixes of a process. The conditions are augmented with multiplicity tests on elements of the FIMMs (only needed for technical reasons concerning the compositionality of $\otimes'$), as well as channel equivalence statements. Composition and entailment of multiplicity tests and conditions in $\mathbf{C}$ are straightforward. The definition of entailment of channel equivalence statements intuitively means that two channels M,N are equivalent in (Ψ,E) if (1) they are equivalent in Ψ, and (2) E does not contain prefixes that can communicate with each other with a priority higher than that of M,N. This is the mechanism by which we prevent lower-priority actions in the translations: those actions that would be ruled out by $\mathtt{H}$ in $\mathscr{P}$ are ruled out in $\mathscr{Q}$ by not being channel equivalent to anything.

In order to avoid bogging down the notation with brackets, we introduce some syntactic sugar for assertions in $\mathbf{A}'$. We will sometimes write Ψ for $(\Psi,\emptyset)$ and E for $(\mathbf{1},E)$. Further, we will sometimes write single-element multisets without the curly brackets, ie. $(z)x$ for $\{(z)x\}$. For an example, combined with the previously introduced syntactic sugar for multisets and guarding elements, we may write $(\mathbf{1},\{(1)(\top,\alpha)\})$ as simply α, and $(\mathbf{1},\{(-1)(\top,\alpha)\})$ as $-\alpha$.

Lemma 2. *$\mathscr{Q}$ is a psi-calculus, meaning that it satisfies the requisites outlined in Section 2.*

The translation of agents from $\mathscr{P}$ to $\mathscr{Q}$ is defined by the function $[\![\_]\!] : \mathbf{P}(\mathscr{P}) \Rightarrow \mathbf{P}(\mathscr{Q})$. The main idea is that in parallel to every prefix, we add the prefix as an assertion (recall that $(\!|\Psi|\!)$ denotes the assertion Ψ occurring as a process), so that it can be used when deciding channel equivalences. The continuation after the prefix contains the same prefix negatively, and since $\{\alpha\}\cup\{-\alpha\}=\emptyset$ the effect is to retract the prefix from the frame once it has been used, and thus ensures that the frame of an agent $[\![P]\!]$ contains an up-to-date copy of the top-level prefixes of P. Since replicated prefixes are permanently enabled, a replicated prefix is asserted with infinite multiplicity to ensure that it is never retracted. For **case** statements, we make sure to retract the guarding elements associated with the other branches after a particular branch has been chosen.

$$\begin{aligned}
[\![\mathbf{0}]\!] &= \mathbf{0}\\
[\![(\!|\Psi|\!)]\!] &= (\!|(\Psi,\emptyset)|\!)\\
[\![P \mid Q]\!] &= [\![P]\!] \mid [\![Q]\!]\\
[\![(\nu x)P]\!] &= (\nu x)[\![P]\!]\\
[\![\alpha.P]\!] &= (\!|\alpha|\!) \mid \alpha.([\![P]\!] \mid (\!|-\alpha|\!))\\
[\![!\,\alpha.P]\!] &= (\!|(\infty)\alpha|\!) \mid !\,\alpha.([\![P]\!] \mid (\!|-\alpha|\!))\\
[\![\mathbf{case}\ \widetilde{\varphi} : \widetilde{\alpha.P}]\!] &= (\!|(\widetilde{\varphi},\widetilde{\alpha})|\!) \mid \mathbf{case}\ \widetilde{\varphi} : \widetilde{\alpha}.(\widetilde{[\![P]\!]} \mid (\!|(-1)(\widetilde{\varphi},\widetilde{\alpha})|\!))
\end{aligned}$$

Recall that we require that substitution has no effect on terms where the names being substituted do not occur. To see why, consider the encoding of the input prefix $\alpha = \underline{M}(\lambda\widetilde{x})N$, where $\widetilde{x}$ is chosen to be fresh in M. If the encoding takes a transition $[\![\alpha]\!] \xrightarrow{\underline{M}N[\widetilde{x}:=\widetilde{L}]} (\!|\alpha|\!) \mid (\!|-\alpha[\widetilde{x}:=\widetilde{L}]|\!)$, we need that $\alpha[\widetilde{x}:=\widetilde{L}] = \alpha$ to achieve a retraction of α. This follows from our requirement since $\widetilde{x}$ does not occur freely in α.

5 Quality of the encoding

In this section, we show that the encoding presented in Section 4.3 satisfies strong operational correspondence, and briefly discuss two other quality criteria: Gorla's framework [13] and full abstraction.

Let $\equiv$, pronounced *structural congruence*, be the smallest congruence on processes that satisfies the commutative monoid laws with respect to $(\mid,\mathbf{0})$ and the rules $!P \equiv P \mid !P$ and $\mathbf{0} \equiv (\!|\mathbf{1}|\!)$ and $(\!|\Psi|\!) \mid (\!|\Psi'|\!) \equiv (\!|\Psi\otimes\Psi'|\!)$.

The main result of this paper is a one-to-one transition correspondence between agents in $\mathscr{P}$ and their encodings in $\mathscr{Q}$:

Theorem 3 (Strong operational correspondence)**.**

1. *If* $\Psi \rhd P \xrightarrow{\alpha}_{\mathrm{p}} P'$ *and* $\mathrm{bn}(\alpha)\#P$ *and* $\alpha \neq \tau : p$*, then there exists* P'' *such that* $(\Psi,\emptyset) \rhd [\![P]\!] \xrightarrow{\alpha} P''$ *and* $[\![P']\!] \equiv P''$.
2. *If* $\Psi \rhd P \xrightarrow{\tau:p}_{\mathrm{p}} P'$*, then there exists* P'' *such that* $(\Psi,\emptyset) \rhd [\![P]\!] \xrightarrow{\tau} P''$ *and* $[\![P']\!] \equiv P''$.
3. *If* $(\Psi,\emptyset) \rhd [\![P]\!] \xrightarrow{\alpha} P'$ *and* $\mathrm{bn}(\alpha)\#P$ *and* $\alpha \neq \tau$*, then there exists* P'' *such that* $[\![P'']\!] \equiv P'$ *and* $\Psi \rhd P \xrightarrow{\alpha}_{\mathrm{p}} P''$.
4. *If* $(\Psi,\emptyset) \rhd [\![P]\!] \xrightarrow{\tau} P'$*, then there exists* p *and* P'' *such that* $[\![P'']\!] \equiv P'$ *and* $\Psi \rhd P \xrightarrow{\tau:p}_{\mathrm{p}} P''$.

Note that a simplification of the encoding with $[\![!\,\alpha.P]\!] = (\!|(\infty)\alpha|\!) \mid !\,\alpha.[\![P]\!]$ would render the above theorem false, since we would then lose the property that $[\![!\,\alpha.P]\!] \equiv [\![\alpha.P \mid !\,\alpha.P]\!]$, and transitions may unfold replications.

Gorla [13] proposes a unified approach to encodability results, wherein a translation function is considered an encoding if it satisfies the five properties *compositionality*, *name invariance*, *operational correspondence*, *divergence reflection*, and *success sensitiveness*.

Because our encoding satisfies strong operational correspondence, the three last criteria follow immediately. Name invariance is immediate since our encoding is equivariant, and compositionality holds with the caveat that we must consider replicated prefixes $!\alpha.P$ as an operator in itself, rather than considering the replication and the prefix as separate operators, and likewise for **case**-guarded prefixes.

Full abstraction means that two agents are equivalent iff their translations are equivalent. The encoding presented in Section 4.3 is not fully abstract with respect to strong bisimilarity. This is because we require bisimilar agents to be statically equivalent, but the translation function introduces assertions such that the translation of bisimilar agents may not be statically equivalent. For a simple example, consider the agents $P = \alpha.\mathbf{0}$ and $Q = \alpha.P$, where α is an output prefix. Clearly $P \mid P \dot{\sim}_{\mathrm{p}} Q$ holds, but for $[\![P]\!] = (\!|\alpha|\!) \mid \alpha.(\mathbf{0} \mid (\!|-\alpha|\!))$ and $[\![Q]\!] = (\!|\alpha|\!) \mid \alpha.([\![P]\!] \mid (\!|-\alpha|\!))$, we have $\mathscr{F}([\![P \mid P]\!]) \vdash' (2)\alpha$ but $\mathscr{F}([\![Q]\!]) \nvdash' (2)\alpha$ and hence $[\![P \mid P]\!] \not\dot{\sim} [\![Q]\!]$.

At first glance, this difference between $[\![P \mid P]\!]$ and $[\![Q]\!]$ seems to be an unimportant technicality: the conditions $(2)\alpha$ and $(1)\alpha$ are not intended to be used as guards in **case**-statements. Their only use is in the evaluation of channel equivalences, but $\mathscr{F}([\![P \mid P]\!])$ and $\mathscr{F}([\![Q]\!])$ entail the same channel equivalences since the set of prefixes available coincides. To motivate that they must be considered different, consider the distinguishing context $R = (\!|-\alpha|\!) \mid (\!|\beta|\!) \mid \gamma.0$, where β is an input that can synchronise with α, and γ has lower priority than α; we have that $R \mid [\![Q]\!]$ can take an action on γ, but $R \mid [\![P \mid P]\!]$ cannot. This highlights an interesting difference between $\mathscr{P}$ and $\mathscr{Q}$: in $\mathscr{P}$, a prefix describes both an interaction possibility and a constraint on other (lower-priority) interactions; in $\mathscr{Q}$, the interaction possibility and interaction constraint are two separate syntactical elements. This means that in $\mathscr{Q}$ we may write $(\!|\alpha|\!)$, which is a process with no transitions that blocks lower-priority transitions as though it had an α-transition; conversely $\alpha.P$ has a non-blocking α-transition that may be blocked by higher-priority transitions.

Note that in the counterexample to full abstraction presented above, the context R is not in the range of $[\![\cdot]\!]$. Thus our encoding may well satisfy weak full abstraction [19], meaning that full abstraction holds if we restrict attention to contexts in the range of $[\![\cdot]\!]$. An investigation of this is deferred to future work.

A related question is whether a fully abstract encoding of $\mathscr{P}$ into some psi-calculus exists. The following theorem, inspired by recent work by Gorla and Nestmann [14] and Parrow [20] , shows that

because of the generality of the psi-calculi framework a trivial fully abstract "encoding" with strong bisimilarity as the target equivalence always exists, regardless of the source language and source equivalence under consideration.

Let **S** be a set ranged over by s, and $\sim$ be an equivalence on **S**. Then there is a psi-calculus $\mathscr{S}$ with no terms, with elements of **S** as assertions and conditions, where entailment is $\sim$. Define the encoding $[\![\_]\!]_{\mathbf{S}} : \mathbf{S} \Rightarrow \mathbf{P}(\mathscr{S})$ by $[\![s]\!]_{\mathbf{S}} \triangleq (\!|s|\!)$.

Theorem 4. *$s \sim s'$ iff $[\![s]\!]_{\mathbf{S}} \dot{\sim} [\![s']\!]_{\mathbf{S}}$*

This "encoding" simply embeds both the source language and source equivalence into a target language with no transition behaviour at all. We conclude that a meaningful approach to full abstraction would have to impose additional criteria. For an example, if we consider Gorla's criteria presented earlier, this "encoding" satisfies name invariance and divergence reflection, but fails to satisfy compositionality, operational correspondence and success sensitiveness.

6 Conclusion

In this paper, we have defined an extension of the psi-calculi framework with dynamic action priorities, and translated it to the original framework. This illustrates the high expressiveness of the assertion mechanism in psi-calculi: usually, it is necessary to introduce negative premises or define a multi-layered transition system in order to obtain action priorities in a given calculus; for psi-calculi, what is already there suffices.

The extension with explicit priorities is interesting in its own right despite the encoding. Expressiveness is not usefulness. Modelling a system with priorities in terms of the translation would be more cumbersome than representing priorities directly. Also, strong bisimulation in the extension is useful for proving equivalences that fail to hold in the encoding.

The most closely related development to psi-calculi with priorities is the *attributed pi-calculus with priorities*, written $\pi(\mathscr{L})$ [17]. It is designed as a generalisation of $\pi@$ [22] and the stochastic pi-calculus. Input and output prefixes take the form $e_1[e'_1]?\widetilde{x}$ and $e_2[e'_2]!\widetilde{y}$, where e_1 and e_2 are subjects, $\widetilde{x}$ and $\widetilde{y}$ are objects and e'_1 and e'_2 are interaction constraints, which may be instantiated to priorities or stochastic rates. e ranges over expressions in an *attribute language*, which is a kind of call-by-value λ-calculus equipped with a big-step reduction relation. The idea in the case of priorities is that if the expressions e_1 and e_2 reduce to the same channel name, and $\widetilde{e}$ reduces to some values $\widetilde{v}$, and the application $e'_1 e'_2$ reduces to the priority level r, then $e_1[e'_1]?\widetilde{x}.P \mid e_2[e'_2]!\widetilde{e}.Q$ reduces to $P[\widetilde{x} := \widetilde{v}] \mid Q$, unless another pair of prefixes can similarly communicate on a higher priority level. The focus is on developing type systems to prevent mismatches, on showing how the calculus can be applied to model phenomena in systems biology, and on the development and implementation of a stochastic simulation algorithm.

While $\pi(\mathscr{L})$ and our approach both generalise $\pi@$, the way the priorities are set up have several interesting differences that suggest incomparable expressive power in general. Priority levels in $\pi(\mathscr{L})$ are taken from an arbitrary partial order, whereas our priorities are natural numbers. Thus in $\pi(\mathscr{L})$ we may have systems where actions have mutually incomparable priority levels, unlike psi-calculi with priorities. The reason we use natural numbers is that the proof of Theorem 1 uses induction and successor arithmetic on the priority level; for future work we would like to investigate alternative proof strategies that would permit a generalised notion of priorities. In psi-calculi, priority levels are associated to communication channels, whereas in $\pi(\mathscr{L})$ they are associated with a particular pair of prefixes. The priority level of a particular pair in $\pi(\mathscr{L})$ is however static and cannot be influenced by the environment in any way, whereas in our approach priorities are dynamic and may change arbitrarily as the assertion

environment evolves. While psi-calculi has no explicit notion of computation on data such as that given by the attribute language, the substitution function can be chosen so that it performs explicit computation on data, or implicit computation can be performed during the evaluation of entailments. For a detailed discussion of how to express computation on data in psi-calculi we refer to [6].

The translation assumes separate choice and prefix-guarded replication. An interesting question is if these assumptions can be relaxed. Allowing mixed choice is possible, but a different definition of guarding elements must be made, that records which prefixes occur in different branches of the same **case**-statement. With the current definition, $[\![\mathbf{case}\ \top : \overline{M}\ [\!]\ \top : \underline{M}]\!]$ has the same guarding elements as $[\![\overline{M} \mid \underline{M}]\!]$, meaning that the former erroneously blocks other transitions as if a communication on M could be derived. Allowing unguarded choice and replication would be more difficult, but we conjecture that it is possible at the expense of compositionality. The solution would involve extending the guarding elements to contain whole syntax trees, including binders. We then lose compositionality since if e.g. $[\![\mathbf{case}\ \top : P\ [\!]\ \top : Q]\!]$ takes a transition from Q, the derivative must contain an assertion that retracts all interaction possibilities offered by P. Hence the translation of Q depends on P, violating compositionality.

Another way to introduce priorities in process calculi is with a *priority choice* operator $P+\rangle Q$, as is done for CCS in [8]. It is like the standard choice operator, with two exceptions. First, P and Q may for technical reasons not contain unguarded output prefixes. Second, transitions from P take precedence over Q. More precisely, its semantics is defined so that it may always act as P, but may act as Q only if no synchronisation on the prefixes of P is possible in the current environment. This operator could be encoded in psi-calculi using techniques similar to those presented in this paper. The main idea is to augment the assertions with information about output prefixes as in Section 4.3, and to translate priority choice as $[\![P+\rangle Q]\!] = \mathbf{case}\ \top : [\![P]\!]\ [\!]\ \varphi_P : [\![Q]\!]$, where φ_P is a condition that holds if no output prefixes matching the inputs of P are enabled in the current environment. A more detailed investigation of this idea is deferred to future work.

We would also like to investigate if a result by Jensen [16], that broadcast communication can be encoded in CCS with priority choice up-to weak bisimulation, can be adapted to our setting. If broadcast psi-calculi [7] can be encoded in psi-calculi with priorities, then by transitivity so can the original psi-calculi. This would contrast with the situation in the pi-calculus, where broadcast communication cannot be encoded [11].

Since both the original psi-calculi and their extension with priorities have been formalised in Nominal Isabelle, we aim to formalise the correspondence results in this paper, in order to be more certain of their correctness. As a first step, it would be necessary to develop a formalisation of FIMMs in Isabelle, and integrate it with the nominal logic package.

References

[1] Jos C. M. Baeten, Jan A. Bergstra & Jan Willem Klop (1986): *Syntax and Defining Equations for an Interrupt Mechanism in Process Algebra*. *Fundamenta Informaticae* IX(2), pp. 127–168.

[2] Jesper Bengtson, Magnus Johansson, Joachim Parrow & Björn Victor (2009): *Psi-calculi: Mobile processes, nominal data, and logic*. In: *Proceedings of LICS 2009*, IEEE Computer Society, pp. 39–48, doi:10.1109/lics.2009.20.

[3] Jesper Bengtson, Magnus Johansson, Joachim Parrow & Björn Victor (2011): *Psi-calculi: A framework for mobile processes with nominal data and logic*. *Logical Methods in Computer Science* 7(1), doi:10.2168/lmcs-7(1:11)2011.

[4] Wayne D. Blizard (1990): *Negative membership*. *Notre Dame Journal of Formal Logic* 31(3), pp. 346–368, doi:10.1305/ndjfl/1093635499.

[5] Roland N. Bol & Jan Friso Groote (1996): *The Meaning of Negative Premises in Transition System Specifications*. *J. ACM* 43(5), pp. 863–914, doi:10.1145/234752.234756.

[6] Johannes Borgström, Ramūnas Gutkovas, Joachim Parrow, Björn Victor & Johannes Åman Pohjola (2014): *A Sorted Semantic Framework for Applied Process Calculi (Extended Abstract)*. In: *Trustworthy Global Computing*, Springer Science + Business Media, pp. 103–118, doi:10.1007/978-3-319-05119-2_7.

[7] Johannes Borgström, Shuqin Huang, Magnus Johansson, Palle Raabjerg, Björn Victor, Johannes Åman Pohjola & Joachim Parrow (2011): *Broadcast Psi-calculi with an Application to Wireless Protocols*. In: *Software Engineering and Formal Methods: SEFM 2011, Lecture Notes in Computer Science* 7041, Springer-Verlag, pp. 74–89, doi:10.1007/s10270-013-0375-z.

[8] Juanito Camilleri & Glynn Winskel (1991): *CCS with priority choice*. In: *Proceedings Sixth Annual IEEE Symposium on Logic in Computer Science*, IEEE Comput. Sco. Press, pp. 246–255, doi:10.1109/lics.1991.151649.

[9] Rance Cleaveland & Matthew Hennessy (1988): *Priorities in Process Algebras*. In: *LICS*, IEEE Computer Society, pp. 193–202, doi:10.1109/lics.1988.5118.

[10] Rance Cleaveland, Gerald Lüttgen & V. Natarajan (2001): *Priority in Process Algebra*. In Jan A. Bergstra, Alban Ponse & Scott A. Smolka, editors: *Handbook of Process Algebra*, Elsevier Science Publishers, pp. 711–765, doi:10.1016/B978-044482830-9/50030-8.

[11] Cristian Ene & Traian Muntean (1999): *Expressiveness of point-to-point versus broadcast communications*. In: *Proceedings of FCT'99, Lecture Notes in Computer Science* 1684, Springer-Verlag, pp. 258–268, doi:10.1007/3-540-48321-7_21.

[12] Murdoch J. Gabbay & Andrew M. Pitts (2002): *A New Approach to Abstract Syntax with Variable Binding*. *Formal Aspects of Computing* 13, pp. 341–363, doi:10.1007/s001650200016.

[13] Daniele Gorla (2008): *Towards a Unified Approach to Encodability and Separation Results for Process Calculi*. In: *CONCUR, Lecture Notes in Computer Science* 5201, Springer, pp. 492–507, doi:10.1007/978-3-540-85361-9_38.

[14] Daniele Gorla & Uwe Nestmann (2014): *Full Abstraction for Expressiveness: History, Myths and Facts*. *Mathematical Structures in Computer Science*. To appear.

[15] Alan Jeffrey (1991): *Translating Timed Process Algebra into Prioritized Process Algebra*. In: *FTRTFT, Lecture Notes in Computer Science* 571, Springer, pp. 493–506, doi:10.1007/3-540-55092-5_27.

[16] Claus Torp Jensen (1994): *Interpreting Broadcast Communication in CCS with Priority Choice*. In: *Proceedings of the 6th Nordic Workshop on Programming Theory*, 203–5, pp. 49–70.

[17] Mathias John, Cédric Lhoussaine, Joachim Niehren & Adelinde Uhrmacher (2010): *The Attributed Pi-Calculus with Priorities*. *Transactions on Computational Systems Biology XII* 5945/2010, pp. 13–76, doi:10.1007/978-3-642-11712-1_2.

[18] Tobias Nipkow, Lawrence C. Paulson & Markus Wenzel (2002): *Isabelle/HOL: a Proof Assistant for Higher-Order Logic*. *Lecture Notes in Computer Science* 2283, Springer-Verlag, doi:10.1007/3-540-45949-9_7.

[19] Joachim Parrow (2008): *Expressiveness of Process Algebras*. *Electronic Notes in Theoretical Computer Science* 209, pp. 173–186, doi:10.1016/j.entcs.2008.04.011.

[20] Joachim Parrow (2014): *General Conditions for Full Abstraction*. *Mathematical Structures in Computer Science*. To appear.

[21] Andrew M. Pitts (2003): *Nominal Logic, A First Order Theory of Names and Binding*. *Information and Computation* 186, pp. 165–193, doi:10.1016/s0890-5401(03)00138-x.

[22] Cristian Versari (2007): *A Core Calculus for a Comparative Analysis of Bio-inspired Calculi*. In Rocco De Nicola, editor: *ESOP*, *Lecture Notes in Computer Science* 4421, Springer, pp. 411–425, doi:10.1007/978-3-540-71316-6_28.

[23] Cristian Versari, Nadia Busi & Roberto Gorrieri (2007): *On the Expressive Power of Global and Local Priority in Process Calculi*. In: *CONCUR*, *Lecture Notes in Computer Science* 4703, Springer, pp. 241–255, doi:10.1007/978-3-540-74407-8_17.