Bisimulation Up-To Techniques for Psi-Calculi

Johannes Åman Pohjola Joachim Parrow
Department of Information Technology, Uppsala University, Sweden
{johannes.aman-pohjola,joachim.parrow}@it.uu.se

Abstract

Psi-calculi is a parametric framework for process calculi similar to popular pi-calculus extensions such as the explicit fusion calculus, the applied pi-calculus and the spi calculus. Remarkably, machine-checked proofs of standard algebraic and congruence properties of bisimilarity apply to all calculi within the framework.

Bisimulation up-to techniques are methods for reducing the size of relations needed in bisimulation proofs. In this paper, we show how these bisimulation proof methods can be adapted to psi-calculi. We formalise all our definitions and theorems in Nominal Isabelle, and show examples where the use of up to-techniques yields drastically simplified proofs of known results. We also prove new structural laws about the replication operator.

Categories and Subject Descriptors F.3.1 [*Logics and Meanings of Programs*]: Specifying and Verifying and Reasoning about Programs; F.4.1 [*Logics and Meanings of Programs*]: Semantics of Programming Languages

General Terms Languages, Theory, Verification

Keywords Bisimulation up-to, process calculus, psi-calculi, Isabelle, Nominal Isabelle, nominal logic

1. Introduction

Bisimilarity is a common method to establish behavioural equivalence in labelled transition systems, and can be defined as follows. Let a transition labelled α from P to P' be written $P \xrightarrow{\alpha} P'$; a binary relation $\mathcal{R}$ is a *bisimulation* if the following diagram commutes:

$$\begin{array}{ccc} P & \mathcal{R} & Q \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle\alpha} \\ P' & \mathcal{R} & Q' \end{array}$$

i.e. whenever $P \mathcal{R} Q$ and $P \xrightarrow{\alpha} P'$ there exists Q' such that $Q \xrightarrow{\alpha} Q'$ and $P' \mathcal{R} Q'$, and symmetrically for the transitions of Q. To establish that two agents P and Q are bisimilar, written $P \sim Q$, we must find a bisimulation $\mathcal{R}$ that relates them. On one hand we want this $\mathcal{R}$ to be small in order to have few transitions to explore from the upper part. On the other hand we want $\mathcal{R}$ to be large in order to facilitate the proof of the lower part $P' \mathcal{R} Q'$. In informal proofs there is a tempting shortcut around this dilemma: we can always just elide the boring parts ("the other cases are similar"). In formal proof, however, such shortcuts requires rigorous justification. The so called *up-to techniques* introduced by Milner [16] permit us to use a small relation in the upper part (the source relation) and a large relation in the lower part (the target relation), leading to more efficient proofs. The contribution of the present paper is a systematic way to define sound up-to techniques for the psi-calculi framework, meaning that such techniques become easily available for a large number of advanced process calculi.

Milner's original idea is called "bisimulation up-to $\sim$" [16], meaning that the following diagram commutes:

$$\begin{array}{ccc} P & \mathcal{R} & Q \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle\alpha} \\ P' \sim & \mathcal{R} & \sim Q' \end{array}$$

and the result is that if a relation is a bisimulation up-to $\sim$ then it is included in $\sim$. As a consequence all known results about $\sim$ may be re-used when establishing the target relation. Sangiorgi [26] develops a general theory through a notion of *respectfulness* to increase the size of the target relation without increasing the size of the source relation in a bisimulation proof. Respectfulness is preserved by useful constructors such as union, composition and chaining. The main point is that the soundness of elaborate up-to techniques then follows immediately from the soundness of smaller building blocks. These results apply to the pi-calculus, and Hirschkoff implemented them as part of his Coq formalisation thereof [12]. There is also a Coq formalisation by Pous of up-to techniques for weak bisimulation that applies to calculi without binders [25]. The notion of respectfulness has since been refined, generalised and rebranded as *compatibility* by Pous [24].

Psi-calculi [5] is a family of applied process calculi that generalises the pi-calculus in three ways. First, instead of single names, the subjects and objects of input and output actions may be *terms* taken from an arbitrary set. 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 facts about the environment in which a process executes. The assertions of a process influence the evaluation of conditions and the connectivity between channel terms. The environment can change dynamically during process execution.

For every psi-calculus, the standard congruence properties and structural laws of both strong and weak bisimulation familiar from the pi-calculus hold. These results have been formalised by Bengtson [3, 4] in the interactive theorem prover Isabelle [19], yielding very high confidence in the correctness of these developments.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

Copyright is held by the owner/author(s). Publication rights licensed to ACM.

CPP'16, January 18–19, 2016, St. Petersburg, FL, USA
ACM. 978-1-4503-4127-1/16/01...$15.00
http://dx.doi.org/10.1145/2854065.2854080

Since its original publication, the psi-calculi family has been extended to include advanced language features such as higher-order data [20] and broadcast communication [6]. As extensions of psi-calculi grow in complexity, so do the bisimulation up-to techniques used to prove their meta-theory. In the Isabelle formalisation of psi-calculi by Bengtson, bisimulation up-to $\sim$ is the most complicated technique used. In broadcast psi-calculi, bisimulation up to the transitive closure of a relation is used to prove that binders commute. In higher-order psi-calculi, an up-to technique composed from the transitive closure, the union with bisimilarity, and the closure under binding sequences of a relation is used to show that higher-order bisimilarity is preserved by the parallel operator.

Of course each new such proof technique must be proven to be sound. Until now, these soundness proofs have been done on a case-by-case basis. Since these up-to techniques often have many similar elements some proof re-use would be desirable, but it is not clear how to combine old soundness results to derive new ones in general. This leads to duplicated effort in both the soundness proofs themselves, and in the bisimulation proofs in lieu of more advanced proof techniques.

We improve on this state of affairs by adopting the idea of compatibility mentioned above. The main difficulty in extending the results to the psi framework is the treatment of environmental assertions. Since the transition relation is parameterised on an assertion environment, so is bisimulation. Two processes with exactly the same transition behaviour need not be bisimilar, since their assertions may have different impact on the evaluation of parallel processes. In order to acquire a sound and compositional theory of compatible functions, this must be taken into account.

All definitions and proofs in this paper have been formalised and machine-checked in Nominal Isabelle [30]. The presentation here closely follows the formalisation, while attempting to abstract away from some of the more tedious details. Formal text in `this font` has been generated by Isabelle. The reader who is interested in further details is referred to the proof scripts, which are available online [1].

The structure of this paper is as follows. In Section 2 we will recapitulate necessary background on the syntax, semantics and bisimulation of psi-calculi. In Section 3 we develop bisimulation up-to techniques for psi-calculi. In Section 4 we apply this framework to simplify bisimulation proofs from the psi-calculi proof archives, and to prove a new result. In Section 5 we discuss alternative ways to build a framework for up-to techniques, and the trade-offs involved. In Section 6 we conclude and discuss related work.

2. Psi-Calculi

The following is a quick recapitulation of the psi-calculi framework as implemented in Isabelle/HOL-Nominal. For in-depth introductions with motivations and examples we refer the reader to [5] for psi-calculi, and to [3] for its Isabelle formalisation.

There is a countably infinite type of atomic *names* `name` ranged over by `a`, `b`, ..., `z`. Intuitively, names are the symbols that can be scoped and can be subject to substitution. A *nominal set* [10, 21] is a type 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 ∈ supp X` (pronounced "`a` is in the support of `X`"). A nominal set is *finitely supported* if all its elements have finite support. We write `a ♯ X`, pronounced "`a` is fresh for `X`", for `a ∉ supp X`, and if `A` is a set or list of names we write `A ♯* X` to mean `∀a∈A. a ♯ X`. In the following `as`, `bs`, ..., `zs` ranges over lists of names. The empty list is written `[]` and the concatenation of `xs` and `ys` is written `xs @ ys`. We use the same convention for lists of other nominal sets. A *name swap* is a pair of names; when applied to an element of a nominal set it replaces all occurences of any of the two names with the other. A sequence of name swaps is a *permutation*. Let `p` range over permutations. `p · X` denotes the application of the permutation `p` to the element `X`; its result is to sequentially apply all name swaps in `p` to `X`.

We say that a constant symbol `X` is *equivariant* if `p · X = X` for all `p`. A function symbol `f` is equivariant if `p · f X = f (p · X)` for all `p`, `X`. We say that a set `X` is equivariant, denoted `eqvt X`, if `∀x∈X. ∀p. p · x ∈ X`.

A *nominal datatype* is an inductively defined datatype that is a nominal set. Nominal Isabelle supports binding occurences of names in such inductive definitions, with the main benefit that nominal datatypes are automatically quotiented by alpha-conversion of bound names. For a simple example, the lambda-calculus can be defined as follows:

```
nominal_datatype lam =
    Var name
  | App lam lam
  | Abs "≪name≫ lam"
```

Here `≪name≫` denotes a binding occurence of a name. Nominal Isabelle will automatically derive: strong induction principles that can avoid most name clashes in induction proofs, akin to Barendregt's variable convention; lemmas characterising the equivariance, freshness and support of terms; and distinctness and injectivity lemmas. Injectivity is formulated up-to alpha so that we may easily derive e.g.

```
y ♯ T ⟹ Abs x T = Abs y ([(x, y)] · T)
```

A psi-calculus is defined by instantiating three nominal sets and four operators; technically these are represented as locale parameters [2].

Definition 2.1 (Psi-calculus parameters)**.** *A psi-calculus requires the three finitely supported nominal sets:*

`'t`	*the (data) terms, ranged over by* `M`, `N`
`'a`	*the assertions, ranged over by* Ψ
`'c`	*the conditions, ranged over by* φ

and the four equivariant operators:

$\leftrightarrow$: `'t ⇒ 't ⇒ 'c`	*Channel Equivalence*
$\otimes$: `'a ⇒ 'a ⇒ 'a`	*Composition*
$\mathbf{1}$: `'a`	*Unit*
$\vdash$: `'a ⇒ 'c ⇒ bool`	*Entailment*

and substitution functions `[as::=Ms]`*, substituting the terms* `Ms` *for names* `as`*, on each of* `'t`*,* `'a`*, and* `'c`*.*

The substitution functions can be chosen freely, but must satisfy certain natural laws regarding the treatment of names; see [5] for details. The binary functions above will be written in infix. Thus, `M ↔ N` is a condition, pronounced "`M` and `N` are channel equivalent". Intuitively, two terms are channel equivalent if they represent the same communication channel. We pronounce $\Psi \vdash \varphi$ as "Ψ entails φ". If Ψ and Ψ' are assertions then so is $\Psi \otimes \Psi'$, which intuitively represents the conjunction of the information in Ψ and Ψ'.

A term is something that can be sent and received; it may also be used as a communication channel whenever it is channel equivalent to some term. A condition is a test that agents may perform. An assertion is a fact about the environment that agents run in; it influences the evaluation of conditions through the entailment relation, and influences the communication topology through channel equivalence. The assertion environment is dynamic and may change as processes execute.

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 restrictions on the choice of parameters:

Definition 2.2 (Requisites on psi-calculi parameters)**.** *The parameters of a psi-calculus must be chosen so that they satisfy:*

1. *(Symmetry)* $\Psi \vdash M \leftrightarrow N \Longrightarrow \Psi \vdash N \leftrightarrow M$
2. *(Transitivity)* $[\![\Psi \vdash M \leftrightarrow N;\ \Psi \vdash N \leftrightarrow L]\!] \Longrightarrow \Psi \vdash M \leftrightarrow L$
3. *(Compositionality)* $\Psi \simeq \Psi' \Longrightarrow \Psi \otimes \Psi'' \simeq \Psi' \otimes \Psi''$
4. *(Identity)* $\Psi \otimes \mathbf{1} \simeq \Psi$
5. *(Associativity)* $(\Psi \otimes \Psi') \otimes \Psi'' \simeq \Psi \otimes \Psi' \otimes \Psi''$
6. *(Commutativity)* $\Psi \otimes \Psi' \simeq \Psi' \otimes \Psi$

These simple requisites suffice to guarantee that for all psi-calculi, the usual congruence and structural laws about bisimulation hold. Three properties that seem to be natural requisites turn out to be unnecessary for this purpose, and by not insisting on them psi-calculi gain significant expressive power. First, we do not assume reflexivity of channel equivalence. The main consequence of this that not every term must be a communication channel. However, since we require associativity and commutativity, $\Psi \vdash M \leftrightarrow K$ implies $\Psi \vdash M \leftrightarrow M$; e.g. every communication channel is equivalent with itself. It is also possible for a term to act as a communication channel in some environments, but not others. Moreover, notice that we do not require idempotence nor weakening of assertion composition, i.e. there may be conditions and assertions such that $\Psi \vdash \varphi$ holds, but $\Psi \otimes \Psi' \vdash \varphi$ or $\Psi \otimes \Psi \vdash \varphi$ do not. This means that we can accomodate e.g. logics to represent resource use.

The names of an assertion may be scoped by the ν-binders familiar from the pi-calculus; this can be used to make the information in an assertion private, e.g. if Ψ contains information about a private key k, then $(\nu k)\Psi$ restricts processes outside the scope of k from accessing this information. We refer to such scoped assertions as *frames*. A frame is an assertion together with a sequence of names that bind into it: it is of the form $\langle xs,\ \Psi\rangle$ where xs binds into the assertion Ψ. We use F, G to range over frames. We overload Ψ to also mean $\langle [],\ \Psi\rangle$, and $\otimes$ to composition on frames defined by

$$\langle A_F,\ \Psi_F\rangle \otimes \langle A_G,\ \Psi_G\rangle = \langle (A_F\ @\ A_G),\ \Psi_F \otimes \Psi_G\rangle$$

where $A_F \sharp^* A_G$, $A_F \sharp^* \Psi_G$ and $A_G \sharp^* \Psi_F$. We write $(\nu x)(\langle xs, \Psi\rangle)$ to mean $\langle (x \cdot xs),\ \Psi\rangle$.

We extend static equivalence to frames by letting $F \vdash \varphi$ iff

$$\exists A_F\ \Psi_F.\ F = \langle A_F,\ \Psi_F\rangle \wedge A_F \sharp^* \varphi \wedge \Psi_F \vdash \varphi$$

We also define $F \simeq G$ to mean that for all φ it holds that $F \vdash \varphi = G \vdash \varphi$. Intuitively a condition is entailed by a frame if it is entailed by the assertion and does not contain any names bound by the frame, and two frames are equivalent if they entail the same conditions.

Definition 2.3 (Psi-calculus agents)**.** *Given psi-calculus parameters as in Definition 2.1, the* agents *(`'t, 'a, 'c) psi`, ranged over by `P, Q, ...`, are defined inductively as a nominal datatype with constructors:*

$\mathbf{0}$	*Nil*
$\overline{M}\,N.P$	*Output*
$\underline{M}(\lambda xs)N.P$	*Input*
`P ‖ Q`	*Parallel*
$(\!\lvert\Psi\rvert\!)$	*Assertion*
`Cases Cs`	*Case*
$(\nu a)P$	*Restriction*
`!P`	*Replication*

Here `Cs` is of type `('c × ('t, 'a, 'c) psi) list`.

$(\nu a)P$ binds `a` in `P` and input $\underline{M}(\lambda xs)N.P$ binds `xs` 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 `P` is* assertion guarded*, denoted `guarded P`, if it contains no unguarded assertions. An agent is* well-formed *if in $\underline{M}(\lambda xs)N.P$ it holds that `xs` is a sequence without duplicates, that in a replication `!P` the agent `P` is assertion guarded, and that in `Cases [(φ0,P0),...,(φn,Pn)]`, the agents P_i are assertion guarded for all $i \leq n$.*

In pen-and-paper reasoning, we normally only consider well-formed agents. In the Isabelle formalisation we admit ill-formed agents in the sense that they inhabit the type `('t, 'a, 'c) psi`; this is not a problem since the operational semantics will be defined so that we cannot derive transitions from ill-formed agents.

The case statement is a form of non-deterministic guarded choice. Its semantics is that `Cases Cs` may act as any process `P` such that `(φ, P) ∈ Cs` and φ holds in the current assertion environment. Otherwise, the process constructs are standard.

In the literature, the parallel operator is often denoted | instead — here we use a different notation because ‖ results in less ambiguous Isabelle syntax.

The *frame `F P` of an agent `P`* is defined inductively as follows:

```
F (0) = 1
F M(λxs)N.P = 1
F M̄ N.P = 1
F (Cases Cs) = 1
F (P ‖ Q) = F P ⊗ F Q
F (⦇Ψ⦈) = Ψ
F ((νx)P) = (νx)F P
F (!P) = 1
```

Intuitively, its assertion component is the composition of all unguarded assertions of `P`, and its binders are the top-level binders of `P`. Note that we do not admit unguarded assertions in replication and case statements, so their frames are always $\mathbf{1}$. The motivation for this design choice is discussed in [14, pp. 60-61].

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

```
bn (M N) = []
bn (M̄ (νxs)N) = xs
bn (τ) = []
```

As in the pi-calculus, the output $\overline{M}\ (\nu xs)N$ represents an action sending `N` along `M` and opening the scopes of the names `xs`. Note that the names `xs` are not binding in $\overline{M}\ (\nu xs)N$; thus $\overline{M}\ (\nu[a])N$ and $\overline{M}\ (\nu[b])([(a,\ b)] \cdot N)$ are different actions. However, they are binding when they occur in a transition:

Definition 2.4 (Transitions)**.** *A* transition *is written $\Psi \rhd P \xrightarrow{\alpha} P'$, where `bn` α binds into both `P'` and the object of α. meaning that in the environment Ψ, `P` can do α to become P'. The transitions are defined in Figure 1.*

The semantics of Figure 1 closely follows the labelled semantics of the pi-calculus, with some additional complexity introduced to account for assertions occurring as agents.

For an example, the premise in the PAR rule intuitively requires that the transition from `P` can be derived in the frame of `Q`, because `Q` may contain assertions that influence the evaluation of conditions occurring in `P`. Analogously, the COM rule requires that the transitions from `P` and `Q` can be derived in each others' frames, and that the channels on which they send and receive are equivalent in the frame

$$\frac{\Psi \vdash M \leftrightarrow K \quad \texttt{distinct xs} \quad \texttt{set xs} \subseteq \texttt{supp N} \quad \texttt{|xs|} = \texttt{|Ts|}}{\Psi \rhd \underline{M}(\lambda xs)N.P \xrightarrow{\underline{K}\, N[xs::=Ts]} P[xs::=Ts]}\ \textsc{In}$$

$$\frac{\Psi \vdash M \leftrightarrow K}{\Psi \rhd \overline{M}\, N.P \xrightarrow{\overline{K} N} P}\ \textsc{Out}$$

$$\frac{\Psi \rhd P \xrightarrow{\alpha} P \quad (\varphi, P) \in Cs \quad \Psi \vdash \varphi \quad \texttt{guarded P}}{\Psi \rhd Cases\ Cs \xrightarrow{\alpha} P}\ \textsc{Case}$$

$$\frac{\Psi \otimes \Psi_Q \rhd P \xrightarrow{\alpha} P' \quad \texttt{bn}\ \alpha \sharp^* Q \quad \mathcal{F}\ Q = \langle A_Q, \Psi_Q \rangle \quad A_Q \sharp^* (\Psi, P, \alpha)}{\Psi \rhd P \parallel Q \xrightarrow{\alpha} P' \parallel Q}\ \textsc{Par}$$

$$\frac{\begin{array}{c}\Psi \otimes \Psi_Q \rhd P \xrightarrow{\underline{M}\, N} P' \quad \Psi \otimes \Psi_P \rhd Q \xrightarrow{\overline{K}\,(\nu xs)N} Q' \\ \Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K \\ xs \sharp^* P \quad \mathcal{F}\ P = \langle A_P, \Psi_P \rangle \quad \mathcal{F}\ Q = \langle A_Q, \Psi_Q \rangle \\ A_P \sharp^* (\Psi, P, Q, M, A_Q) \quad A_Q \sharp^* (\Psi, P, Q, K)\end{array}}{\Psi \rhd P \parallel Q \xrightarrow{\tau} (\nu xs)(P' \parallel Q')}\ \textsc{Com}$$

$$\frac{\Psi \rhd P \parallel !P \xrightarrow{\alpha} P \quad \texttt{guarded P}}{\Psi \rhd !P \xrightarrow{\alpha} P}\ \textsc{Rep}$$

$$\frac{\Psi \rhd P \xrightarrow{\alpha} P' \quad x \sharp \Psi \quad x \sharp \alpha}{\Psi \rhd (\nu x)P \xrightarrow{\alpha} (\nu x)P'}\ \textsc{Res}$$

$$\frac{\Psi \rhd P \xrightarrow{\overline{M}\,(\nu(xs\ @\ ys))N} P' \quad x \in \texttt{supp N} \quad x \sharp \Psi \quad x \sharp M \quad x \sharp xs \quad x \sharp ys}{\Psi \rhd (\nu x)P \xrightarrow{\overline{M}\,(\nu(xs\ @\ [x]\ @\ ys))N} P'}\ \textsc{Open}$$

Figure 1. Structured operational semantics. Symmetric versions of COM and PAR are elided.

of $P \parallel Q$. The somewhat intimidating freshness conditions serve to guarantee that Ψ_P and Ψ_Q are sufficiently fresh representatives of these frames. As in the pi-calculus, processes may tell each other secrets: a process may send messages containing restricted names to outside their scope. This extrudes the scope, so that it encapsulates also the receiver of the message; in the COM rule these are the names xs.

The OPEN rule may look intimidating, but on closer inspection it is essentially the same as in the pi-calculus. It says that if the agent P can send a message containing the name x on a channel that does not contain x, then so can $(\nu x)P$. This opens the scope of x, so that it may eventually be closed by the COM rule. In the conclusion, the name x may be inserted anywhere in the sequence $xs\ @\ ys$ of opened names. If we insist that x always goes at the front of the sequence, we lose an important algebraic law: $(\nu x)((\nu y)P)$ and $(\nu y)((\nu x)P)$ may exhibit transitions with different labels.

Let $\mathcal{R}, \mathcal{S}$ range over ternary relations on assertions and pairs of agents, i.e. an element of $\mathcal{R}$ is of kind (Ψ, P, Q).

Definition 2.5 (Simulation)**.** *We say that* P simulates Q in Ψ to $\mathcal{R}$, *written* $\Psi \rhd P \rightsquigarrow [\mathcal{R}]\ Q$, *if for all* α, Q' *such that* $\Psi \rhd Q \xrightarrow{\alpha} Q'$, $\texttt{bn}\ \alpha \sharp^* \Psi$ *and* $\texttt{bn}\ \alpha \sharp^* P$ *it holds that*

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

Definition 2.6 (Strong bisimulation)**.** *A relation* $\mathcal{R}$ *is a* strong bisimulation *if* $(\Psi, P, Q) \in \mathcal{R}$ *implies*

1. *Static equivalence:* $\mathcal{F}\ P \otimes \Psi \simeq \mathcal{F}\ Q \otimes \Psi$
2. *Symmetry:* $(\Psi, Q, P) \in \mathcal{R}$*; and*
3. *Extension of arbitrary assertion:* $\forall \Psi'.\ (\Psi \otimes \Psi', P, Q) \in \mathcal{R}$*; and*
4. *Simulation:* $\Psi \rhd P \rightsquigarrow [\mathcal{R}]\ Q$

Bisimilarity*, denoted* $\sim$*, is the largest bisimulation relation. We abbreviate* $(\Psi, P, Q) \in \sim$ *as* $\Psi \rhd P \sim Q$*, and write* $P \sim Q$ *for* $\mathbf{1} \rhd P \sim Q$.

We give the intuition behind the clauses in reverse order. The simulation clause is the same as in the pi-calculus, with freshness conditions in the definition of simulation to ensure that the choice of concrete representatives for the bound names of α does not impact the derivation of the matching transition in Q. Extension of arbitrary assertion represents changes in the environment caused by some external process; intuitively it means that bisimilarity must be preserved by all possible changes to the environment. Without this requisite, bisimilarity is not preserved by the parallel operator. The symmetry clause is simply a convenience that lets us avoid having two simulation clauses. Finally, static equivalence states that two bisimilar processes must have equivalent frames; intuitively this means that their impact on the behaviour of other processes must be the same.

We say that a relation is *statically equivalent* if it satisfies the static equivalence clause of Definition 2.6, and that a relation is *extensible* if it satisfies the extension of arbitrary assertion clause of Definition 2.6.

The following are some of the congruence properties and algebraic laws of $\sim$ proved in [5]:

Theorem 2.1 (Congruence properties of $\sim$)**.**

1. $\Psi \rhd P \sim Q \Longrightarrow \Psi \rhd \overline{M}\, N.P \sim \overline{M}\, N.Q$
2. $[\![\Psi \rhd P \sim Q;\ x \sharp \Psi]\!] \Longrightarrow \Psi \rhd (\nu x)P \sim (\nu x)Q$
3. $\Psi \rhd P \sim Q \Longrightarrow \Psi \rhd P \parallel R \sim Q \parallel R$
4. $[\![\Psi \rhd P \sim Q;\ \texttt{guarded P};\ \texttt{guarded Q}]\!] \Longrightarrow \Psi \rhd !P \sim !Q$

Bisimulation is not preserved by input, for the same reasons as in the pi-calculus. It is preserved by case, but we elide the exact formulation of this result since the details are unimportant.

Theorem 2.2 (Structural laws of $\sim$)**.**

1. $\Psi \rhd P \sim P$
2. $\Psi \rhd P \sim Q \Longrightarrow \Psi \rhd Q \sim P$
3. $[\![\Psi \rhd P \sim Q;\ \Psi \rhd Q \sim R]\!] \Longrightarrow \Psi \rhd P \sim R$
4. $\Psi \rhd P \parallel Q \sim Q \parallel P$
5. $\Psi \rhd P \parallel Q \parallel R \sim P \parallel (Q \parallel R)$
6. $\Psi \rhd P \parallel \mathbf{0} \sim P$
7. $\Psi \rhd (\nu x)((\nu y)P) \sim (\nu y)((\nu x)P)$
8. $x \sharp P \Longrightarrow \Psi \rhd (\nu x)(P \parallel Q) \sim P \parallel (\nu x)Q$
9. $\texttt{guarded}\ P \Longrightarrow \Psi \rhd !P \sim P \parallel !P$

As usual in name-passing calculi, we may obtain a congruence relation that is contained in bisimilarity, called simply *strong congruence*, by closing bisimilarity under all substitution sequences. The usual approach to proving that two processes P, Q are strongly

congruent is to exhibit a bisimulation relation such that for all substitutions σ it contains the pair $(P\sigma, Q\sigma)$; in such cases the up-to techniques introduced in this paper can be used to aid congruence proofs also. We will not use strong congruence in this paper: it is of scant interest from an up-to techniques perspective, since it is not defined coinductively.

3. Up-To techniques

In this section — or more precisely, in Section 3.2 — we develop bisimulation up-to techniques for psi-calculi. Before that, we begin by recapitulating the general theory of compatible functions on arbitrary complete lattices in Section 3.1.

3.1 Up-To Techniques and Complete Lattices

Pous [24] discovered that many of the basic ingredients of up-to techniques can be developed in a setting that abstracts away from bisimulation, and instead considers arbitrary coinductive objects, taking the view that a coinductive object is the greatest fixed point of a monotone function on a complete lattice. This is useful for our purposes since bisimulation for psi-calculi is different from standard LTS bisimulation, yet we may still reuse these foundations without making any particular adaptations for psi-calculi.

In this section, let f, g, b range over functions of type $'l \Rightarrow 'l$, where $'l$ is a complete lattice. We say that a function f is *b-compatible* if $f \circ b \leq b \circ f$. We write *mono f* to mean that f is monotonic, and *gfp f* for the greatest fixed point of f.

The following two lemmas are due to Pous; we contribute their Isabelle implementation. First, we show that compatibility yields sound up-to techniques, in the sense that in order to prove something about *gfp b*, it suffices that we can prove it about *gfp (b ∘ f)* for some b-compatible f.

Lemma 3.1 (Soundness of compatible functions)**.**

$[\![mono\ f;\ mono\ b;\ f \circ b \leq b \circ f]\!] \Longrightarrow gfp\ (b \circ f) \leq gfp\ b$

The main benefit of compatibility is that it has nice compositionality properties. Already in this very abstract setting we can show that compatibility is preserved by function composition and supremum. We can also show that the identity function, and the constant functions that always return some post-fixed point of b, are compatible.

Lemma 3.2 (Compatible functions)**.**

1. $[\![mono\ f;\ mono\ g;\ mono\ b;\ f \circ b \leq b \circ f;\ g \circ b \leq b \circ g]\!] \Longrightarrow f \circ g \circ b \leq b \circ (f \circ g)$
2. $[\![mono\ b;\ \bigwedge f.\ f \in F \Longrightarrow f \circ b \leq b \circ f]\!] \Longrightarrow \bigsqcup F \circ b \leq b \circ \bigsqcup F$
3. $mono\ b \Longrightarrow id \circ b \leq b \circ id$
4. $[\![mono\ b;\ c \leq b\ c]\!] \Longrightarrow (\lambda x.\ c) \circ b \leq b \circ (\lambda x.\ c)$

3.2 Up-To Techniques for Psi-Calculi

Recall that $\mathcal{R}, \mathcal{S}$ range over ternary relations on assertions and pairs of agents. Adopting the terminology of Sangiorgi [26], we will use f, g to range over endofunctions on such relations, which we will call *first-order functions*. $\mathcal{A}$ ranges over sets of first-order functions. Functions that have first-order functions as both arguments and return values will be called *constructors*.

We define *extended simulation*, which intuitively is a simulation step followed by an extension of the environment:

Definition 3.1 (Extended simulation)**.** *We say that* P simulates and extends Q in Ψ to $\mathcal{R}$, *written* $\Psi \rhd P \rightsquigarrow [\mathcal{R}]_{ext}\ Q$, *if for all* α, Q' *such that* $\Psi \rhd Q \xrightarrow{\alpha} Q'$, $bn\ \alpha \sharp^* \Psi$ *and* $bn\ \alpha \sharp^* P$ *it holds that*

$\exists P'.\ \Psi \rhd P \xrightarrow{\alpha} P' \wedge (\Psi, P', Q') \in \mathcal{R} \wedge (\forall \Psi'.\ (\Psi \otimes \Psi', P', Q') \in \mathcal{R})$

In the definition of extended simulation, we could do without the conjunct $(\Psi, P', Q') \in \mathcal{R}$ since up-to static equivalence, its effect can be obtained by extending with the unit assertion. We include it for convenience — it ensures that every extended simulation is also a simulation so that results about simulation can be easily lifted, and also serves to make the definition workable when not reasoning up-to static equivalence.

A foundation for bisimulation up-to techniques for psi-calculi is obtained immediately by instantiating the function b from Section 3.1 with a functional of bisimilarity, i.e. a function whose greatest fixed point is $\sim$. There are in fact several sensible candidates, leading to different trade-offs between what compatible functions are admitted and what proof obligations arise in bisimulation proofs, as we discuss in Section 5.

Our preference is for the functional $\mathcal{B}$:

Definition 3.2. *The bisimilarity functional* $\mathcal{B}$ *is defined in Isabelle as follows:*

```
definition 𝓑 :: "('a × ('t, 'a, 'c) psi × ('t, 'a, 'c)
psi) set ⇒ ('a × ('t, 'a, 'c) psi × ('t, 'a, 'c) psi)
set"
where "𝓑 𝓡 = {(Ψ,P,Q) | Ψ P Q.
  ℱ P ⊗ Ψ ≃ ℱ Q ⊗ Ψ ∧
  Ψ ▷ P ⇝[𝓡]ext Q ∧
  Ψ ▷ Q ⇝[𝓡⁻¹]ext P ∧
  (∀Ψ'. (Ψ ⊗ Ψ',  P, Q) ∈ 𝓡)}"
```

Intuitively, if $\mathcal{R} \subseteq \mathcal{B}\ \mathcal{S}$ then playing one round of the bisimulation game from a triple in $\mathcal{R}$ leads to a triple in $\mathcal{S}$; in particular we have that if $\mathcal{R} \subseteq \mathcal{B}\ \mathcal{R}$, then the symmetric closure of $\mathcal{R}$ is a bisimulation relation. There are two differences between $\mathcal{B}$ and the functional (implicitly) used in Definition 2.6: first, we use a second simulation clause for transitions from Q, rather than insist on symmetry. This allows us to use smaller candidate relations in proofs, and allows for up-to techniques that do not necessarily preserve symmetry. The second is that we use extended simulation instead of simulation; this slightly stronger requirement appears necessary to prove $\mathcal{B}$-compatibility of parallel contexts; we will explain the details in Section 3.3. Our experience is that the added difficulty of proving extended simulation as opposed to simulation is negligable in practice. From this point onward, we will use *compatible* to abbreviate *$\mathcal{B}$-compatible*.

First, some sanity checks on $\mathcal{B}$ are in order:

Lemma 3.3.

1. $mono\ \mathcal{B}$
2. $gfp\ \mathcal{B} = \sim$
3. $compatible\ \mathcal{B}$
4. $eqvt\ \mathcal{R} \Longrightarrow eqvt\ (\mathcal{B}\ \mathcal{R})$

We introduce a few useful functions and constructors. We will use the identity function id; the inverse function $\_^{-1}$; the constant functions $\mathcal{U}$ and $\mathcal{X}$ that map every relation to bisimilarity and the identity relation, respectively; the transitive closure $\_^*$ of a relation; and the closure $\mathcal{E}$ of a relation under static equivalence. The main reason compatibility is attractive is that compatible functions can be combined to form more complex functions which are themselves

compatible, using important constructors such as composition $\circ$, supremum $\bigsqcup$ and chaining $\frown$.

Definition 3.3 (Functions and constructors)**.**

1. $\mathcal{R}^{-1} = \{(\Psi,P,Q) \mid (\Psi,Q,P) \in \mathcal{R}\}$
2. $\mathcal{U}\ \mathcal{R} = (\lambda\ x.\ \sim)$
3. $\mathcal{P}\ \mathcal{R} = \{(\Psi,P,Q) \mid \exists p.\ (p\cdot\Psi,p\cdot P,p\cdot Q) \in \mathcal{R}\}$
4. $\mathcal{E}\ \mathcal{R} = \{(\Psi,P,Q) \mid \exists\ \Psi'.\ \Psi' \simeq \Psi \wedge (\Psi',P,Q) \in \mathcal{R}\}$
5. $\mathcal{X}\ \mathcal{R} = \{(\Psi,P,P) \mid True\}$

Definition 3.4 (Transitive closure)**.** *The* transitive closure $\mathcal{R}^\star$ *of a relation* $\mathcal{R}$ *is inductively defined by the rules*

$$\frac{(\Psi,\ P,\ Q) \in \mathcal{R}}{(\Psi,\ P,\ Q) \in \mathcal{R}^\star}$$

$$\frac{(\Psi,\ P,\ Q) \in \mathcal{R}^\star \qquad (\Psi,\ Q,\ R) \in \mathcal{R}}{(\Psi,\ P,\ R) \in \mathcal{R}^\star}$$

Definition 3.5 (Chaining)**.** *The* chaining $\frown$ *of two compatible functions is defined as*

$$(f \frown g)\ \mathcal{R} = \{(\Psi,\ P,\ Q) \mid \exists P'.\ (\Psi,\ P,\ P') \in f\ \mathcal{R} \wedge (\Psi,\ P',\ Q) \in g\ \mathcal{R}\}$$

Most of the above are straightforward adaptations of functions that occur frequently in the up-to techniques literature. $\mathcal{P}$ is the nominal logic analogue of Sangiorgi's up-to injective substitutions [26]; it has the very useful property that $eqvt\ (\mathcal{P}\ \mathcal{R})$ for all $\mathcal{R}$, meaning that it lets us reason up-to alpha in the lower parts of bisimulation diagrams even if we use a non-equivariant relation in the upper part. The main novelty is $\mathcal{E}$, which is very useful since bisimulation proofs in psi-calculi often rely on being able to rewrite the frame modulo static equivalence. Note that in a pi-calculus setting, $\mathcal{E}$ would collapse to the identity function.

All these functions can be used to build compatible functions, but not all possible combinations yield compatible functions. For an example, the chaining function does not preserve compatibility, but

$$compatible\ ((\mathcal{P} \circ f) \frown (\mathcal{P} \circ g))$$

holds if f and g are compatible and monotonic. When a large compatible function is constructed by composing many smaller functions, the function will quickly become cluttered with many such occurences of auxiliary functions: necessary to prove compatibility, but usually not needed in bisimulation proofs. These become unwieldy and annoying to work with. For an example, suppose that in a bisimulation proof we need to close under static equivalence on the left before chaining, but we do not need the closure under permutations. $\mathcal{E}$ is not compatible, but $\mathcal{E} \circ \mathcal{P}$ is. Applying this and the above fact about $\frown$ yields the compatible function $(\mathcal{P} \circ \mathcal{E} \circ \mathcal{P}) \frown (\mathcal{P} \circ id)$. With some manual effort we may simplify this to $(\mathcal{E} \circ \mathcal{P}) \frown \mathcal{P}$, but what we really wanted was $\mathcal{E} \frown id$. In order to avoid such tedium, we will be primarily interested in *quasi-compatibility* rather than compatibility:

Definition 3.6 (Quasi-compatibility)**.** *A first-order function* f *is* quasi-compatible*, written* $qcompatible\ f$*, if*

$$\exists g.\ compatible\ g \wedge mono\ g \wedge f \leq g$$

In other words, a function is quasi-compatible if it is included in a compatible monotonic function. Quasi-compatibility has the same nice compositionality properties as compatibility, but without the clutter:

Lemma 3.4. *$\mathcal{X}$, $\mathcal{E}$, $\mathcal{P}$, $\_^{-1}$, $\mathcal{U}$, id and $\_^\star$ are monotonic and quasi-compatible functions. If f, g are quasi-compatible and monotonic, then so is $f \circ g$ and $f \frown g$. If all $f \in F$ are quasi-compatible then so is $\bigsqcup F$.*

$$\frac{\begin{array}{c}(\Psi,\ P,\ Q) \in \mathcal{R} \\ qcompatible\ f \qquad \bigwedge \Psi\ P\ Q.\ \dfrac{(\Psi,\ P,\ Q) \in \mathcal{R}}{\mathcal{F}\ P \otimes \Psi \simeq \mathcal{F}\ Q \otimes \Psi} \\ \bigwedge \Psi\ \Psi'\ P\ Q.\ \dfrac{(\Psi,\ P,\ Q) \in \mathcal{R}}{(\Psi \otimes \Psi',\ P,\ Q) \in f\ \mathcal{R}} \\ \bigwedge \Psi\ P\ Q.\ \dfrac{(\Psi,\ P,\ Q) \in \mathcal{R}}{\Psi \rhd P \rightsquigarrow [f\ \mathcal{R}]_{ext}\ Q} \\ \bigwedge \Psi\ P\ Q.\ \dfrac{(\Psi,\ P,\ Q) \in \mathcal{R}}{\Psi \rhd Q \rightsquigarrow [(f\ \mathcal{R})^{-1}]_{ext}\ P}\end{array}}{\Psi \rhd P \sim Q}$$

Figure 2. Manually derived coinduction rule for bisimilarity up-to quasi-compatible functions.

Similar refined notions of compatibility can be found in the literature. Pous [24] applies up-to techniques to compatible functions themselves, yielding a notion of "compatibility up-to" that can be used to achieve similar aims. Hur et al. [13] have a construct called the largest compatible function, which is denoted † and defined as the join of all compatible functions. The connection with quasi-compatibility is that a function f is quasi-compatible iff $f \leq \dagger$. However, Hur et al. do not discuss applying † to justify the use of not-quite-compatible functions; rather † is applied to facilitate coinductive proofs in an incremental style, where no a priori commitment to e.g. a particular choice of compatible function is necessary.

The derived coinduction rule for proving bisimilarity up-to quasi-compatible functions is shown in Figure 2. All we need to do before we can use a particular up-to technique in a proof is to prove it quasi-compatible, and then instantiate this rule. Even better, quasi-compatibility proofs can usually be fully automated; registering the clauses of Lemma 3.4 as introduction rules with Isabelle's classical reasoner suffices in most practical cases. For an example, deriving Milner's original bisimulation up-to $\sim$ comes down to the following one-liner:

lemma *"qcompatible($\mathcal{U} \frown (id \frown \mathcal{U})$)"*
by *blast*

This compares favourably to the situation when reasoning on raw soundness: Bengtson uses a 37 line structured proof to derive the soundness of this technique. Further, obtaining soundness of the minor variant $\mathcal{U} \frown (\lambda\mathcal{R}.\ id\ \mathcal{R} \cup \sim) \frown \mathcal{U}$ is also fully automatic for us, while Bengtson uses a 36 line proof. This illustrates one of the major advantages of compatibility over soundness: its compositionality properties lets us avoid wasting effort on redundant proofs when deriving new up-to techniques.

An up-to technique that looks natural at a glance, but turns out to be unsound, is the closure of a relation under extension, defined as

$$\mathcal{T}\ \mathcal{R} = \{(\Psi \otimes \Psi',\ P,\ Q) \mid (\Psi,\ P,\ Q) \in \mathcal{R}\}$$

To see why, consider a psi-calculus with assertions Ψ and Ψ' such that $\Psi \vdash M \leftrightarrow L$, $\Psi \otimes \Psi' \vdash M \leftrightarrow M$, and $\Psi \otimes \Psi' \vdash L \leftrightarrow L$ but not $\Psi \otimes \Psi' \vdash M \leftrightarrow L$. Consider the relation $\mathcal{R} = \{(\Psi,\ \overline{M}\,N.\,\mathbf{0},\ \overline{L}\,N.\,\mathbf{0}),\ (\Psi,\ \mathbf{0},\ \mathbf{0})\}$. We have that $(\Psi,\ \overline{M}\,N.\,\mathbf{0},\ \overline{L}\,N.\,\mathbf{0}) \notin \sim$ since extending with Ψ' yields different transition behaviour. But since all triples in $\mathcal{R}$ have the same frames and outgoing transitions, we can easily show that $\mathcal{R} \subseteq \mathcal{B}\ (\mathcal{T}\ \mathcal{R})$; hence if $\mathcal{T}$ is sound there is a "proof" of $\Psi \rhd \overline{M}\,N.\,\mathbf{0} \sim \overline{L}\,N.\,\mathbf{0}$.

The ability to close a relation under contexts is a desirable proof technique. We here consider the closure under all operators save for

replication separately — the closure under arbitrary contexts can be obtained by taking the transitive closure of their union.

Definition 3.7 (Closure under contexts)**.**

1. $\mathcal{C}_{Res}\ \mathcal{R} = \{(\Psi, (\nu xs)P, (\nu xs)Q) \mid xs \sharp^* \Psi \wedge (\Psi, P, Q) \in \mathcal{R}\}$
2. $\mathcal{C}_{Par}\ \mathcal{R} = \{(\Psi, P \parallel R, Q \parallel R) \mid \exists A_R\ \Psi_R.\ \mathcal{F}\ R = \langle A_R, \Psi_R\rangle \wedge A_R \sharp^* \Psi \wedge A_R \sharp^* P \wedge A_R \sharp^* Q \wedge (\Psi \otimes \Psi_R, P, Q) \in \mathcal{R}\}$
3. $\mathcal{C}_{Out}\ \mathcal{R} = \{(\Psi, \overline{M}\,N.P, \overline{M}\,N.Q) \mid (\Psi, P, Q) \in \mathcal{R}\}$
4. $\mathcal{C}_{In}\ \mathcal{R} = \{(\Psi, \underline{M}(\lambda xs)N.P, \underline{M}(\lambda xs)N.Q) \mid \forall Ts.\ |xs| = |Ts| \longrightarrow (\Psi, P[xs::=Ts], Q[xs::=Ts]) \in \mathcal{R}\}$
5. $\mathcal{C}_{Case}\ \mathcal{R} = \{(\Psi, Cases\ ((\varphi, P) \cdot CsP), Cases\ ((\varphi, Q) \cdot CsP)) \mid (guarded\ P \wedge guarded\ Q \wedge (\Psi, P, Q) \in \mathcal{R}) \wedge (\forall \varphi\ P.\ (\varphi, P) \in set\ CsP \longrightarrow guarded\ P)\}$

Theorem 3.5. *All functions introduced in Definition 3.7 are monotonic and quasi-compatible.*

This is where quasi-compatibility leads to the most drastic simplifications: the witness to the quasi-compatiblility of $\mathcal{C}_{Par}$ is the somewhat unwieldy $\mathcal{C}_{Res} \circ \mathcal{C}_{Par} \circ \mathcal{E} \circ \mathcal{P}$.

A reader may wonder why $\mathcal{C}_{Par}$ is not defined using the following much simpler formulation:

$$\mathcal{C}_{Par'}\ \mathcal{R} = \{(\Psi, P \parallel R, Q \parallel R) \mid (\Psi, P, Q) \in \mathcal{R}\}$$

First, note that while the definition of $\mathcal{C}_{Par}$ is certainly more intimidating, it is usually more helpful than $\mathcal{C}_{Par'}$ in practice: it allows P and Q to be related by $\mathcal{R}$ in a more specific environment. More importantly, $\mathcal{C}_{Par'}$ is not quasi-compatible. To see why not, first consider the monotonic and quasi-compatible function

$$\uparrow \mathcal{R} = \{(\Psi \otimes \Psi', P, Q) \mid (\Psi, P \parallel (\!|\Psi'|\!), Q \parallel (\!|\Psi'|\!)) \in \mathcal{R}\}$$

that moves assertions occuring in the processes down to the frame. If $\mathcal{C}_{Par'}$ is quasi-compatible, then so is $\uparrow \circ \mathcal{C}_{Par'}$; however, this function is precisely the up-to extension function $\mathcal{T}$, which as we have seen is unsound. In the common special case where $\mathcal{R}$ is extensible, we can recover $\mathcal{C}_{Par'}$ since we then have $\mathcal{C}_{Par'}\ \mathcal{R} \subseteq \mathcal{C}_{Par}\ \mathcal{R}$.

We have only defined the closure under parallel contexts where the common context occurs to the right of the $\parallel$ operator. To obtain it to the left instead, we may use $\mathcal{U} \frown \mathcal{C}_{Par} \frown \mathcal{U}$ and the structural law $\Psi \rhd P \parallel Q \sim Q \parallel P$. Through a similar technique we may obtain the closure under case contexts where P and Q occur in positions other than the first.

The quantification over substitutions in $\mathcal{C}_{In}$ is necessary for soundness since bisimilarity is not preserved by input. However, note that we need only require membership in $\mathcal{R}$ for such substitutions of P and Q as may arise from the particular input prefix under consideration. This constitutes an improvement over [12, 26], where a quantification over all substitution sequences is used.

Finally we observe that bisimilarity is preserved by all quasi-compatible functions.

Theorem 3.6. *If* $qcompatible\ f$*, then* $f \sim \subseteq \sim$

As an immediate consequence, we obtain alternative proofs that bisimilarity is preserved by restriction, parallel, output, and case: we obsolete 400 lines of bisimulation proofs from Bengtson's formalisation by simply replacing them with invocations of Theorem 3.6. Its proof is 9 lines long.

An attentive reader may notice that we do not consider the closure of a relation under replication. For completeness it would be nice to also provide closure under replication contexts, but this omission is not very significant in practice — the most practically interesting context closures are those for restriction and parallel composition, since they are the only operators that occur on the right-hand side of the transition arrow in the rules of the operational semantics. We anticipate that a proof of its quasi-compatibility would be tedious but not really difficult — the proof strategy would be similar to Bengtson's proof that replication preserves bisimilarity [4]. We currently see no potential application for bisimulation up-to replication contexts beyond proving that replication preserves bisimilarity; in Section 4.1 we already have a proof of this result that is much simpler than a compatibility proof would be.

$$\frac{\bigwedge \mathcal{R}\ \mathcal{S}\ \Psi\ P\ Q.\ \dfrac{(\Psi, P, Q) \in f\ \mathcal{R} \qquad \mathcal{R} \subseteq \mathcal{B}\ \mathcal{S}}{\mathcal{F}\ P \otimes \Psi \simeq \mathcal{F}\ Q \otimes \Psi} \quad \bigwedge \mathcal{R}\ \mathcal{S}\ \Psi\ P\ Q.\ \dfrac{(\Psi, P, Q) \in f\ \mathcal{R} \qquad \mathcal{R} \subseteq \mathcal{B}\ \mathcal{S}}{\Psi \rhd P \rightsquigarrow [f\ \mathcal{S}]_{ext}\ Q} \quad \bigwedge \mathcal{R}\ \mathcal{S}\ \Psi\ P\ Q.\ \dfrac{(\Psi, P, Q) \in f\ \mathcal{R} \qquad \mathcal{R} \subseteq \mathcal{B}\ \mathcal{S}}{\Psi \rhd Q \rightsquigarrow [(f\ \mathcal{S})^{-1}]_{ext}\ P} \quad \bigwedge \mathcal{R}\ \mathcal{S}\ \Psi\ \Psi'\ P\ Q.\ \dfrac{(\Psi, P, Q) \in f\ \mathcal{R} \qquad \mathcal{R} \subseteq \mathcal{B}\ \mathcal{S}}{(\Psi \otimes \Psi', P, Q) \in f\ \mathcal{S}}}{compatible\ f}$$

Figure 3. Derived introduction rule for compatibility.

3.3 Anatomy of a Compatibility Proof

In this section we show the main ideas behind a non-trivial compatibility proof: namely the one for $\mathcal{C}_{Par}$. This is currently our most difficult compatibility proof, and will also illustrate why we believe our use of extended simulation is necessary.

As mentioned in Section 3, the witness to the quasi-compatibility of $\mathcal{C}_{Par}$ is $\mathcal{C}_{Res} \circ \mathcal{C}_{Par} \circ \mathcal{E} \circ \mathcal{P}$. We conduct compatibility proofs with the introduction rule shown in Figure 3, derived from the definitions of compatibility and $\mathcal{B}$.

The static equivalence clause of Figure 3 is discharged with tedious but straightforward arguments. For extension, we ignore the outermost $\mathcal{C}_{Res}$ in order to avoid bogging down the presentation. We have that $(\Psi \otimes \Psi_R, P, Q) \in \mathcal{E}\ (\mathcal{P}\ \mathcal{R})$, and wish to show that $(\Psi \otimes \Psi', P \parallel R, Q \parallel R) \in \mathcal{C}_{Par}\ (\mathcal{E}\ (\mathcal{P}\ \mathcal{S}))$. Since we do not have that $A_R \sharp^* \Psi'$, we construct a permutation p such that $p \cdot A_R$ is fresh for $\Psi', \Psi, P, Q, R, A_R, \Psi_R$. By $\mathcal{R} \subseteq \mathcal{B}\ \mathcal{S}$ and by compatibility of $\mathcal{E} \circ \mathcal{P}$, we may extend to

$$((\Psi \otimes \Psi_R) \otimes (p \cdot \Psi'), P, Q) \in \mathcal{E}\ (\mathcal{P}\ S)$$

Since $\mathcal{P}$ guarantees equivariance we get

$$((\Psi \otimes \Psi_R) \otimes (p \cdot \Psi'), P, Q) \in \mathcal{E}\ (\mathcal{P}\ S)$$

$\mathcal{E}$ lets us rewrite the frame modulo AC:

$$((\Psi \otimes (p \cdot \Psi')) \otimes \Psi_R, P, Q) \in \mathcal{E}\ (\mathcal{P}\ S)$$

Since A_R is fresh for $\Psi \otimes (p \cdot \Psi')$ we have

$$(\Psi \otimes (p \cdot \Psi'), P \parallel R, Q \parallel R) \in \mathcal{C}_{Par}\ (\mathcal{E}\ (\mathcal{P}\ S))$$

Finally, we use equivariance again to obtain

$$(\Psi \otimes \Psi', P \parallel R, Q \parallel R) \in \mathcal{C}_{Par}\ (\mathcal{E}\ (\mathcal{P}\ S))$$

For the simulation clauses, we elide $\mathcal{E}$ and $\mathcal{P}$, and ignore all binders occuring in frames and labels; the numerous intricacies that arise from the interaction between the various binding mechanisms

are essentially the same as those encountered when proving that bisimulation is preserved by parallel (see e.g. [3, pp. 360-371]).

Unfolding the definition of extended simulation, we know that $(\Psi \otimes \Psi_R, P, Q) \in \mathcal{R}$ and $\mathcal{R} \subseteq \mathcal{B}\ \mathcal{S}$ and $\Psi \rhd (\nu xs)(P \parallel R) \xrightarrow{\alpha} S'$, and in order to conclude we must find T' such that $\Psi \rhd (\nu xs)(Q \parallel R) \xrightarrow{\alpha} T'$ and $(\Psi \otimes \Psi', S', T') \in \mathcal{C}_{Res}\ (\mathcal{C}_{Par}\ \mathcal{S})$. The proof is by induction on xs. The most difficult part occurs in the base case, when the transition is inferred from a communication between P and R. We have $\Psi \otimes \Psi_R \rhd P \xrightarrow{\alpha} P'$ and $\Psi \otimes \Psi_P \rhd R \xrightarrow{\beta} R'$. From $(\Psi \otimes \Psi_R, P, Q) \in \mathcal{R}$ and $\mathcal{R} \subseteq \mathcal{B}\ \mathcal{S}$ we obtain $\Psi \otimes \Psi_R \rhd Q \xrightarrow{\alpha} Q'$ and $((\Psi \otimes \Psi_R) \otimes \Psi'', P', Q') \in \mathcal{S}$ for all Ψ'', and by applying certain technical lemmas from [3, pp. 360-371] we eventually obtain $\Psi \rhd Q \parallel R \xrightarrow{\tau} Q' \parallel R'$. The frame of R' is the frame of R augmented with whatever assertions became unguarded by the transition, so in other words $\Psi_{R'} \simeq \Psi_R \otimes \Psi'''$ for some Ψ'''. By instantiating Ψ'' we get $((\Psi \otimes \Psi_R) \otimes \Psi''' \otimes \Psi', P', Q') \in \mathcal{S}$, and rewriting the frame module AC we get $((\Psi \otimes \Psi') \otimes \Psi_{R'}, P', Q') \in \mathcal{S}$. It follows that $(\Psi \otimes \Psi', P' \parallel R', Q' \parallel R') \in \mathcal{C}_{Res}\ (\mathcal{C}_{Par}\ \mathcal{S})$ and we may conclude.

Therein lies the problem that we solve by introducing extended simulation: mimicking the behaviour of a process communicating with another process R requires taking two steps in the bisimulation game: one step perform the transition itself, and another to extend the environment with the new assertions added to the environment by the transition from R. However we only have that $\mathcal{R} \subseteq \mathcal{B}\ \mathcal{S}$, so after taking one step from $\mathcal{R}$ to $\mathcal{S}$ the bisimilarity functional is consumed and we are stuck with no way to extend. Using extended simulation, we get around this difficulty since we may then perform both the transition and any extension we may need with only one application of $\mathcal{B}$.

4. Examples

In this section, we apply the proof techniques introduced in Section 3 to obtain significantly shorter proofs of familiar results from the psi-calculi literature. We also prove a new structural law about replication.

4.1 Bisimilarity is Preserved by Replication

An Isabelle proof that bisimilarity is preserved by replication for psi-calculi is due to Bengtson [4] — formally, the result is that if $\Psi \rhd P \sim Q$, $\mathtt{guarded}\ P$ and $\mathtt{guarded}\ Q$ then $\Psi \rhd !P \sim !Q$. A detailed account can be found in [3, pp. 388-391].

Bengtson's proof uses bisimulation up-to $\sim$, with the candidate relation

$\{(\Psi, R \parallel !P, R \parallel !Q) \mid \Psi \rhd P \sim Q \wedge \mathtt{guarded}\ P \wedge \mathtt{guarded}\ Q\}$

This means that in the simulation part of the proof, we must consider all transitions from $R \parallel !P$, resulting in a transition inversion where four cases must be analysed (transitions from $R \parallel !P$ can be derived via PAR, COM or their symmetric counterparts). Essentially, this inversion re-proves a special case of the more general result that bisimilarity is preserved by parallel composition, and seems somewhat besides the point in a proof which is really about the replication operator, not the parallel operator.

We support this intuition by using the techniques introduced in Section 3 to give a simpler proof, which does not use a redundant parallel component in the candidate relation and hence does not feature a rule inversion at all. Our proof constitutes 63 lines of Isabelle code; Bengtson's constitutes 214 lines.

The idea is to pick the candidate relation

$\mathcal{R} = \{(\Psi, !P, !Q) \mid \mathtt{guarded}\ P \wedge \mathtt{guarded}\ Q \wedge \Psi \rhd P \sim Q\}$

and the compatible function

$f = \mathcal{U} \frown (\mathcal{C}_{Res} \circ \mathcal{C}_{Par}) \frown \mathcal{U}$

In the simulation part of the proof, we apply the following technical lemma by Bengtson[1]:

Lemma 4.1. *(Lemma 28.29 from [3]) If* $\Psi \rhd !P \xrightarrow{\alpha} P'$, *and* $\Psi \rhd P \sim Q$, *and* $\mathtt{bn}\ \alpha\ \sharp^*\ \Psi$, *and* $\mathtt{bn}\ \alpha\ \sharp^*\ P$, *and* $\mathtt{bn}\ \alpha\ \sharp^*\ Q$, *and* $\mathtt{bn}\ \alpha\ \sharp^*\ \mathtt{subject}\ \alpha$, *and* $\mathtt{guarded}\ Q$, *then there exists* Q', R, T *such that* $\Psi \rhd !Q \xrightarrow{\alpha} Q'$, $\Psi \rhd P' \sim R \parallel !P$, $\Psi \rhd Q' \sim T \parallel !Q$, $\Psi \rhd R \sim T$, $\mathtt{supp}\ R \subseteq \mathtt{supp}\ P'$ *and* $\mathtt{supp}\ T \subseteq \mathtt{supp}\ Q'$.

Intuitively, this lemma states that any derivative of $!P$ can be rewritten with $\sim$ to a certain normal form $R \parallel !P$ for some R. Moreover, if P and Q are bisimilar then Q has a derivative with a normal form $S \parallel !Q$ with R and S bisimilar.

After applying Lemma 4.1, all that remains is to show that $(\Psi, P', Q') \in f\ \mathcal{R}$, which is simply a matter of rewriting using the structural congruence laws.

4.2 Idempotence of Replication

We show that replication is idempotent, a structural congruence law that is new in the setting of psi-calculi, but familiar from the pi-calculus [18, 29].

Theorem 4.2. $\Psi \rhd !!P \sim !P$

Proof. We choose $\mathcal{R} = \{(\Psi, !!P, !P) \mid \mathtt{True}\}$ and $f = \mathcal{U} \frown (\mathcal{C}_{Res} \circ \mathcal{C}_{Par}) \frown \mathcal{U}$.

Static equivalence and extension are trivial.

The right-to-left direction of the simulation proof goes by applying Lemma 4.1 to obtain R such that $\Psi \rhd !P \xrightarrow{\alpha} P'$ and $P' \sim R \parallel !P$. By a simple derivation, $\Psi \rhd !!P \xrightarrow{\alpha} P' \parallel !!P$. From there, it remains to be shown that $(\Psi, P' \parallel !!P, P') \in f\ \mathcal{R}$, which follows by structural congruence.

In the left-to-right direction, an induction over the derivation of $\Psi \rhd !!P \xrightarrow{\alpha} P'$ is used. Each case is discharged by applying Lemma 4.1 and familiar algebraic laws. □

Corollary 4.3. $\Psi \rhd !P \sim !P \parallel !P$

Proof. $\Psi \rhd !P \parallel !P \sim !P \parallel !!P \sim !!P \sim !P$. □

The Isabelle proof of Theorem 4.2 is 310 lines long. Strikingly, there does not appear to have been any proof of the corresponding pi-calculus result in the literature until the publication of Sangiorgi and Rutten's book in 2011 [28]; it is left as an exercise to the reader in [18, 29]. Sangiorgi and Rutten's book uses a proof technique similar to ours.

For a comparison of different approaches to this kind of proof, we instead resort to Corollary 4.3, whose corresponding pi-calculus result has been proven twice by Milner [17, 18], and once by Sangiorgi and Walker [29].

Milner's first proof uses the candidate relation

$$\{(\nu\tilde{y})(!P \mid !P \mid Q),\ (\nu\tilde{y})(!P \mid Q) : \mathbf{true}\}.$$

While full details of the proof are not shown, the more complicated relation choice would certainly entail at least an induction over the

[1] Bengtson's proof of Lemma 4.1 is 134 lines long. It is used both in our proof and Bengtson's proof that bisimilarity is preserved by replication; but also in other proofs, so we include it in the line count of neither.

length of $\widetilde{y}$ and an inversion for each parallel operator before we even get to the replication operators.

Milner's second proof uses the relation $\{(!P \mid Q,\ !P \mid !P \mid Q) : \mathbf{true}\}$ and bisimulation up-to structural congruence. Sangiorgi and Walker's proof instead uses the candidate relation $\{(!P \mid !P,\ !P) : \mathbf{true}\}$ and a choice of f which closes the relation under contexts and structural congruence, a very similar approach to our proof of Theorem 4.2.

The shortest of these proofs is Milner's first proof, since it is presented in the least amount of detail. Interestingly, while Milner's second proof is presented at a similar level of abstraction to Sangiorgi and Walker's, the proofs themselves are about the same length despite the simpler relation used by the latter. We conclude that in a pen and paper setting, the choice of up-to techniques does not have as much impact as in a theorem prover setting. Pen and paper proofs can often be made easier by using informal arguments such as "the other cases are similar", "as the reader may care to check" et cetera in place of up-to techniques. In a theorem prover, where every single detail must be considered, up-to techniques can provide a much needed substitute for such shortcuts.

4.3 Encoding of Replication in Higher-Order Calculi

Higher-order psi-calculi extend psi-calculi by allowing terms to act as handles for agents. More formally, if Ψ entails the *clause* $M \Leftarrow P$, then the agent **run** M may act as P in Ψ, as governed by the rule

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

We recall from [20] the result that under certain assumptions on the calculus, replication can be encoded using the **run** construct. More precisely:

$$\Psi \rhd !P \sim (\nu a)(\mathbf{run}\ M \parallel (\!|C_\Psi(M \Leftarrow P \parallel \mathbf{run}\ M)|\!))$$

Here $C_\Psi(M \Leftarrow P \parallel \mathbf{run}\ M)$, pronounced the *characretistic assertion* of the clause $M \Leftarrow P \parallel \mathbf{run}\ M$, is an assertion that entails this particular clause but otherwise has no impact on the assertion environment. The name a is fresh in P but not in M. For a more detailed account we refer to [20].

While perhaps not a very surprising result, the Isabelle proof is actually one of the longest proofs in the psi-calculi literature — a theory file consisting of 8788 lines of code totalling 870 kB is dedicated entirely to it. Part of the reason why this proof is so long is the choice of candidate relation, which is the symmetric closure of the following:

$$\mathcal{R} = \{(\Psi,\ (\nu xs)(Q \parallel (\nu a)(\mathbf{run}\ M \parallel (\!|C_\Psi(M \Leftarrow P \parallel \mathbf{run}\ M)|\!))),\ (\nu xs)(Q \parallel !P))\}$$

In order to avoid bogging down the presentation, the definition of $\mathcal{R}$ given above is not fully formal since it omits several freshness conditions and requisites on the characteristic assertion. Readers interested in the gory details are referred to the Isabelle sources [1].

Focusing on the right-to-left direction of the simulation proofs, we must follow all transitions from $(\nu xs)(Q \parallel !P)$. The first step is an induction over the length of the binding sequence xs. In the base case, an inversion of the transition from $Q \parallel !P$ follows, yielding four cases which must be analysed: PAR, COM and their symmetric counterparts. In three of these cases, an induction over the derivation of the transition from $!P$ occurs, yielding five sub-cases: two each of PAR and COM, plus REP.

Fortunately, we can significantly improve upon the situation using bisimulation up-to techniques and the following much simpler relation:

$$\mathcal{S} = \{(\Psi,\ (\nu a)(\mathbf{run}\ M \parallel (\!|C_\Psi(M \Leftarrow P \parallel \mathbf{run}\ M)|\!)),\ !P)\}$$

The proof uses $f = \mathcal{U} \frown (\mathcal{C}_{Res} \circ \mathcal{C}_{Par}) \frown \mathcal{U}$, i.e. the same function as in previous examples. Of the steps necessary with $\mathcal{R}$ as the relation choice discussed above, only the innermost induction over the derivation from $!P$ must now be considered. We are also relieved of the obligation to close $\mathcal{S}$ under symmetry. Without otherwise changing anything, this significantly shortens the proof to 3263 lines and 248.6 kB, a size decrease of about 60% and 70% respectively.

5. Alternative Formulations

In this section, we explore alternative ways to derive up-to techniques for psi-calculi. Different choices of bisimilarity functional lead to different notions of compatibility. While in general we may choose any monotonic function f such that $gfp\ f = \sim$, not all choices are equally good; for an example, if we use $\mathcal{U}$ then $\mathcal{U}$-compatibility is precisely soundness and we lose most compositionality properties. Below we discuss two functionals that yield more interesting trade-offs when compared with $\mathcal{B}$.

A natural choice is to use the functional whose post-fixed points are the bisimulation relations of Definition 2.6, namely

definition $\mathcal{B}'$ :: "('a × ('t, 'a, 'c) psi × ('t, 'a, 'c) psi) set ⇒ ('a × ('t, 'a, 'c) psi × ('t, 'a, 'c) psi) set"
where "$\mathcal{B}'\ \mathcal{R}$ = {(Ψ,P,Q) | Ψ P Q.
$\mathcal{F}$ P ⊗ Ψ $\simeq_F$ $\mathcal{F}$ Q ⊗ Ψ ∧
Ψ ▷ P ⇝[$\mathcal{R}$] Q ∧
(Ψ,Q,P) ∈ $\mathcal{R}$ ∧
(∀Ψ'. (Ψ ⊗ Ψ', P, Q) ∈ $\mathcal{R}$)}"

We eschew this functional because neither the inverse function, the chaining function nor the symmetric closure function are $\mathcal{B}'$-compatible. Hence it seems too difficult to use anything but symmetric candidate relations with $\mathcal{B}'$, whereas $\mathcal{B}$ handles asymmetric relations without a hitch. It seems likely that the simulation clause of B' must be replaced with extended simulation in order to recover $\mathcal{C}_{Par}$.

The following functional was suggested by Pous:

definition $\mathcal{B}''$ :: "('a × ('t, 'a, 'c) psi × ('t, 'a, 'c) psi) set ⇒ ('a × ('t, 'a, 'c) psi × ('t, 'a, 'c) psi) set"
where "$\mathcal{B}''\ \mathcal{R}$ = {(Ψ,P,Q) | Ψ P Q.
(∀ Ψ'.
$\mathcal{F}$ P ⊗ Ψ ⊗ Ψ' $\simeq_F$ $\mathcal{F}$ Q ⊗ Ψ ⊗ Ψ' ∧
Ψ ⊗ Ψ' ▷ P ⇝[$\mathcal{R}$] Q ∧
Ψ ⊗ Ψ' ▷ Q ⇝[$\mathcal{R}^{-1}$] P)}"

This functional drops the extension clause in favour of a quantification over all environments in the simulation and static equivalence clauses. Its most appealing consequence is that the up-to extension function $\mathcal{T}$ is quasi-$\mathcal{B}''$-compatible. Using $\mathcal{T}$ we may always extend the environment after taking a transition, so we do not need to use extended simulation. We may also often use singleton candidate relations like $\{(\mathbf{1},\ P,\ Q)\}$ in cases where $\mathcal{B}$ and $\mathcal{B}'$ would require an infinite relation, viz. the closure of $\{(\mathbf{1},\ P,\ Q)\}$ under extension. However, note that this decrease in relation size does not correspond to smaller proof obligations in bisimulation proofs: because of the $\forall\ \Psi'$ we must always chase bisimulation diagrams and prove static equivalence in every possible environment, even when the candidate relation only uses $\mathbf{1}$.

A more radical alternative is to only consider candidate relations that are equivariant and extensible, and only admit first-order functions that preserve equivariance and extensibility. This avoids many of the technical complications of working with non-extensible and non-equivariant relations in compatibility proofs, at the cost of some flexibility. $\mathcal{T}$ and $\mathcal{P}$ collapse to the identity function. Since a bisimilarity functional for this setting would be unwieldy we present this alternative in terms of respectfulness [26] rather than compatibility:

Definition 5.1 (Progression)**.** *We say that* $\mathcal{R}$ progresses to $\mathcal{S}$*, denoted* $\mathcal{R} \longmapsto \mathcal{S}$ *, if* $(\Psi, P, Q) \in \mathcal{R}$ $(\Psi, P, Q) \in \mathcal{R}$ *implies* $\Psi \rhd P \rightsquigarrow[\mathcal{S}] Q$ *and* $\Psi \rhd Q \rightsquigarrow[\mathcal{S}^{-1}] P$

Definition 5.2 (Respectfulness)**.** *A function* $\mathtt{f}$ *is* respectful *if for all* $\mathcal{R}$ *it holds that*

1. *If* $\mathcal{R}$ *is statically equivalent, then so is* $\mathtt{f}\ \mathcal{R}$*.*
2. *If* $\mathcal{R}$ *is equivariant and extensible, then* $\mathtt{f}\ \mathcal{R}$ *is extensible.*
3. $\mathtt{f}$ *is equivariant.*
4. *If* $\mathcal{R} \subseteq \mathcal{S}$*, and* $\mathcal{R} \longmapsto \mathcal{S}$ *, and* $\mathcal{R}$ *is equivariant, and* $\mathcal{S}$ *is equivariant, statically equivalent and extensible, then* $\mathtt{f}\ \mathcal{R} \subseteq \mathtt{f}\ \mathcal{S}$ *and* $\mathtt{f}\ \mathcal{R} \longmapsto \mathtt{f}\ \mathcal{S}$

To see the connection between compatibility and respectfulness intuitively, progression is to respectfulness as the bisimilarity functional is to compatibility. We have developed a branch of our formalisation in terms of respectfulness, and all results presented in this paper about $\mathcal{B}$-compatible functions carry over to respectful functions. The practical difference in the difficulty of bisimulation proofs appears negligable; using respectfulness instead of $\mathcal{B}$-compatibility to prove the results in Section 4 yields proofs that are 70 rather than 63 lines long, 3324 rather than 3249 lines long, and 314 rather than 310 lines long. Respectfulness proofs are often easier than compatibility proofs: many respectful functions are smaller than their $\mathcal{B}$-compatible counterparts since we do not need $\mathcal{P}$. The proof archive becomes slightly smaller, since we do not need a theory of extended simulation.

This solution of only considering equivariant relations is similar to Hirschkoff's solution in [12, p. 161], where only "good" relations are considered. A "good" relation in his setting based on de Bruijn-indices is analogous to an equivariant relation in our nominal logic-based setting. Hirschkoff motivates this restriction with appeals to intuition, with remarks such as

> these transformations ... do not really have a strong significance, since they ... do some kind of 'administrative work' to keep the notation coherent. Therefore, if a pair (P, Q) of processes belongs to a given relation $\mathcal{R}$, there should be no reason for $[(p \cdot P, p \cdot Q)]$ not to be in $\mathcal{R}$.

Square brackets in the above quote demarcate a part where we have substituted de Bruijn-index specific language for the nominal logic analogue[2]. We understand and share Hirschkoff's reasons for considering only "good" relations, but our results indicate that nonetheless such simplifying assumptions are not strictly necessary if we formulate our results in terms of quasi-compatibility rather than compatibility.

We prefer $\mathcal{B}$-compatibility over respectfulness primarily because of the added generality. While we do not currently have any formal developments where we benefit from non-equivariant or non-extensible relations, we do not doubt that they exist. For an example where non-extensible relations help, suppose we want to use psi-calculi to model concurrent constraint programming. We assume a set of *constraints* ranged over by c, and an entailment relation $\vdash_c$ that is a binary relation between constraints. A constraint is *inconsistent* if it entails every other constraint. If our constraint system has *atomic tell*, i.e. a constraint can only be added to the environment if doing so does not lead to inconsistency, we are normally not interested in considering the behaviour of processes under inconsistent stores. If we model constraints as assertions, bisimulation in psi-calculi seems overly discriminating: it forces us to relate the behaviour of processes in inconsistent stores even though no such store is reachable.

One trick that lets us avoid considering inconsistent stores in bisimulation proofs, is to let the assertions be the constraints, and let the conditions be the constraints plus a special symbol $\bot$. Entailment is then defined as follows:

$$\begin{array}{ll} c \vdash c' & \text{if } \vdash_c c' \text{ and } c \text{ is consistent} \\ c \vdash \bot & \text{if } c \text{ is inconsistent} \end{array}$$

If $\bot$ is not in the range of the channel equivalence function, it follows that if the store is inconsistent, there are no transitions and everything is statically equivalent; thus $c \rhd P \sim Q$ for all inconsistent c and all P, Q.

Hence, if we work with $\mathcal{B}$-compatibility and close our compatible function under union with bisimilarity, we can use candidate relations that only contain consistent stores, since extensions that lead to inconsistency take us into $\sim$. By contrast, if we work with respectfulness then we must bog down our candidate relation with inconsistent stores.

6. Conclusion

In this paper, we have adapted the bisimulation proof method of Sangiorgi and Pous to the setting of psi-calculi in a way that accounts for assertion environments. We have illustrated the usefulness of this proof method by showing how it can make proofs of known results significantly better, i.e. shorter, easier to understand and less redundant; and used it to prove a new result. We have also mechanised all of our definitions and theorems in Nominal Isabelle.

The better proofs are worthwhile to obtain because they make the proofs more maintainable. Just like program code, proof scripts require maintenance: definitions change, or new versions of Isabelle break backwards-compatibility. For every such change, all proofs must be re-checked, and better proofs are naturally easier to check.

The developments of Section 3 consist of approximately 3750 lines of code. Using these developments, the proofs pertaining to the encoding of replication in higher-order psi-calculi were made approximately 5500 lines shorter. Of course we do not wish to imply a one-to-one correspondence between lines of code and maintainability; nonetheless, this result is a strong indication that our development efforts have already paid off.

For future work, we would like to apply these techniques to the extensions of psi-calculi which can be found in the literature. So far, these proofs have been carried out for the original formulation of psi-calculi and higher-order psi-calculi. While we are able to use the same proof scripts for higher-order psi-calculi after changing just a single line of code, different compatibility proofs for the up-to context techniques would be necessary for the extension with broadcasts.

Bisimulation up-to techniques for weak bisimulation represents another interesting area to explore. Soundness is more delicate for weak bisimulation than for the strong case; for an example, weak

[2] For an account of the formal relationship between de Bruijn-indices and nominal sets see [8].

bisimulation up-to weak bisimilarity is unsound. Solutions have been proposed by Sangiorgi and Milner [27] and Pous [23, 25] that could constitute a starting point for similar investigations in the field of psi-calculi.

In recent work Madiot et al. [15] suggest to recover up-to techniques for higher-order calculi by translating them to first-order transition systems (i.e., where the transition labels do not carry binders). This is a promising idea, but for our purposes it would only be beneficial if the difficulty of developing up-to techniques is less than the difficulty of (a) defining a first-order transition system, (b) proving full abstraction and (c) deriving bisimulation up-to context in the first-order transition system. It seems unlikely that this would hold in the general case of psi-calculi, so we prefer our more direct approach. Staying in a setting similar to the existing psi-calculi formalisation also has the advantage that we may leverage the cyclopean effort invested by Bengtson in developing infrastructure for reasoning about it in Isabelle.

Popescu and Gunter [22] define a coinductive proof system for bisimilarity that applies to transition systems in de Simone format [9], and prove it sound and complete in Isabelle. Hence, while our work is parametric on the term language and assertion logic, theirs is parametric on the process syntax. The choice of de Simone as the rule format precludes us from considering many standard features of modern process calculi, such as pi-calculus-style restriction operators. It seems difficult to generalise their result to more expressive settings: the soundness of the proof system relies on bisimulation up-to context being sound, and Sangiorgi [26] has shown that this fails to hold for many other rule formats such as tyft/tyxt, despite bisimulation being a congruence.

Other recent work by Chaudhuri et al. [7] is devoted to a formalisation of bisimulation up-to techniques for CCS and the pi-calculus with replication in the Abella theorem prover. The pi-calculus formalisation treats bound names with Abella's built-in ∇ quantifier for *generic judgements*, i.e. $\nabla x.P$ means that P holds for all x when nothing is assumed about x. A comparison between this specification style and Nominal Isabelle can be found in [11]; a comparative disadvantage of Nominal Isabelle is that a notion of substitution must be defined by the user, while it comes for free in Abella. Their main results are: the soundness of bisimulation up-to $\sim$ composed with bisimulation up-to context for CCS, and the soundness of bisimulation up-to $\sim$ for the pi-calculus. Bisimulation up-to context for the pi-calculus is deferred to future work, with a main hurdle being that "defining the notion of a process context in the π-calculus is tricky, because contexts are allowed to capture free names"[7, p. 164]. We do not know enough about Abella to understand the precise technical difficulties involved in that setting, but in Nominal Isabelle defining contexts is straightforward. The key insight is that names must be treated as non-binding when occuring in a context, but as binding once the hole has been filled. Here is how to set it up for psi-calculi in Nominal Isabelle; for convenience we consider only monadic contexts with parallel and restriction.

```
nominal_datatype ('t,'a,'c) psi_context =
    Hole
  | Par_contextL
      "('t::fs_name,'a::fs_name,'c::fs_name) psi_context"
      "('t,'a,'c) psi"
  | Par_contextR
      "('t,'a,'c) psi" "('t,'a,'c) psi_context"
  | Res_context
      name "('t,'a,'c) psi_context"

nominal_primrec fill_hole ::
 "('t::fs_name,'a::fs_name,'c::fs_name) psi_context ⇒
  ('t,'a,'c) psi ⇒ ('t,'a,'c) psi"
where
  "fill_hole Hole P = P"
| "fill_hole (Res_context x C) P = (⌊νx⌋)(fill_hole C P)"
| "fill_hole (Par_contextL C Q) P = (fill_hole C P) ∥ Q"
| "fill_hole (Par_contextR Q C) P = Q ∥ (fill_hole C P)"
  by(rule TrueI)+
```

Differences between settings aside, a main goal of Chaudhuri et al. is to keep the formalisation lightweight, in the sense that it does not start from foundations but rather builds on top of existing infrastructure. In many ways our work is lightweight in the same sense: we build on top of the nominal package, the existing psi-calculi formalisation and the lattice library. We arguably stray from this philosophy by eschewing the definitional command for coinductive sets, instead dealing explicitly with bisimilarity functionals and their fixed points. However, this allows us to go beyond Chaudhuri et al. in several ways. First, we consider compatibility whereas they only consider soundness; hence they lack a framework for combining up-to techniques. Second, we go beyond the pi-calculus and derive, once and for all, up-to techniques for all pi-calculus extensions that fall within the psi-calculi framework.

Acknowledgments

We are deeply grateful to Damien Pous for extensive technical discussions on the material in Sections 3 and 5. Johannes Borgström came up with the idea for the proof of Corollary 4.3. We are grateful to the reviewers for many constructive comments.

References

[1] J. Åman Pohjola. Formalisations of up-to techniques for psi-calculi. http://www.it.uu.se/research/group/mobility/theorem/upto.tgz, Oct. 2015. Isabelle formalisation of the definitions, theorems and proofs.

[2] C. Ballarin. Locales and locale expressions in Isabelle/Isar. In S. Berardi, M. Coppo, and F. Damiani, editors, *Types for Proofs and Programs, International Workshop, TYPES 2003, Torino, Italy, April 30 – May 4, 2003, Revised Selected Papers*, volume 3085 of *Lecture Notes in Computer Science*, pages 34–50. Springer-Verlag, 2003. URL http://dx.doi.org/10.1007/978-3-540-24849-1_3.

[3] J. Bengtson. *Formalising process calculi*. PhD thesis, Uppsala University, June 2010. URL http://urn.kb.se/resolve?urn=urn%3Anbn%3Ase%3Auu%3Adiva-122782.

[4] J. Bengtson and J. Parrow. Psi-calculi in Isabelle. In S. Berghofer, T. Nipkow, C. Urban, and M. Wenzel, editors, *Proc. of TPHOLs 2009*, volume 5674 of *Lecture Notes in Computer Science*, pages 99–114. Springer-Verlag, Aug. 2009. URL http://dx.doi.org/10.1007/978-3-642-03359-9_9.

[5] J. Bengtson, M. Johansson, J. Parrow, and B. Victor. Psi-calculi: A framework for mobile processes with nominal data and logic. *Logical Methods in Computer Science*, 7(1), 2011. URL http://dx.doi.org/10.2168/LMCS-7(1:11)2011.

[6] J. Borgström, S. Huang, M. Johansson, P. Raabjerg, B. Victor, J. Åman Pohjola, and J. Parrow. Broadcast psi-calculi with an application to wireless protocols. In G. Barthe, A. Pardo, and G. Schneider, editors, *Software Engineering and Formal Methods: SEFM 2011*, volume 7041 of *Lecture Notes in Computer Science*, pages 74–89. Springer-Verlag, Nov. 2011. URL http://dx.doi.org/10.1007/978-3-642-24690-6_7.

[7] K. Chaudhuri, M. Cimini, and D. Miller. A lightweight formalization of the metatheory of bisimulation-up-to. In X. Leroy and A. Tiu, editors, *Proceedings of the 2015 Conference on Certified Programs and Proofs, CPP 2015, Mumbai, India, January 15-17, 2015*, pages 157–166. ACM, 2015. ISBN 978-1-4503-3296-5. . URL http://doi.acm.org/10.1145/2676724.2693170.

[8] V. Ciancia and U. Montanari. A name abstraction functor for named sets. In *9th Workshop on Coalgebraic Methods in Computer Science*

(CMCS 2008), volume 203–5, pages 49–70, 2008. URL http://www.di.unipi.it/~ciancia/content/cm08.pdf.

[9] R. de Simone. Higher-level synchronising devices in meije-sccs. *Theoretical Computer Science*, 37:245 – 267, 1985. ISSN 0304-3975. . URL http://www.sciencedirect.com/science/article/pii/0304397585900933.

[10] M. J. Gabbay and A. M. Pitts. A new approach to abstract syntax with variable binding. *Formal Aspects of Computing*, 13:341–363, 2002. URL http://dx.doi.org/10.1007/s001650200016.

[11] A. Gacek, D. Miller, and G. Nadathur. Reasoning in abella about structural operational semantics specifications. *Electr. Notes Theor. Comput. Sci.*, 228:85–100, 2009. . URL http://dx.doi.org/10.1016/j.entcs.2008.12.118.

[12] D. Hirschkoff. A full formalisation of pi-calculus theory in the calculus of constructions. In *TPHOLs '97: Proceedings of the 10th International Conference on Theorem Proving in Higher Order Logics*, pages 153–169, London, UK, 1997. Springer-Verlag. ISBN 3-540-63379-0. URL http://dx.doi.org/10.1007/bfb0028392.

[13] C. Hur, G. Neis, D. Dreyer, and V. Vafeiadis. The power of parameterization in coinductive proof. In R. Giacobazzi and R. Cousot, editors, *The 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '13, Rome, Italy - January 23 - 25, 2013*, pages 193–206. ACM, 2013. ISBN 978-1-4503-1832-7. . URL http://doi.acm.org/10.1145/2429069.2429093.

[14] M. Johansson. *Psi-calculi: a framework for mobile process calculi: Cook your own correct process calculus - just add data and logic*. PhD thesis, Uppsala University, Division of Computer Systems, 2010. URL http://urn.kb.se/resolve?urn=urn%3Anbn%3Ase%3Auu%3Adiva-123139.

[15] J.-M. Madiot, D. Pous, and D. Sangiorgi. Bisimulations up-to: Beyond first-order transition systems. In P. Baldan and D. Gorla, editors, *CONCUR 2014*, volume 8704 of *Lecture Notes in Computer Science*, pages 93–108. Springer Berlin Heidelberg, 2014. ISBN 978-3-662-44583-9. . URL http://dx.doi.org/10.1007/978-3-662-44584-6_8.

[16] R. Milner. *Communication and Concurrency*. Prentice-Hall, Inc., 1989. ISBN 0131149849.

[17] R. Milner. The polyadic pi-calculus: a tutorial. Technical report, Logic and Algebra of Specification, 1991.

[18] R. Milner. *Communicating and mobile systems: the pi-calculus*. Cambridge University Press, New York, NY, USA, 1999. ISBN 0-521-65869-1.

[19] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL: a Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer-Verlag, 2002. ISBN 3-540-43376-7.

[20] J. Parrow, J. Borgström, P. Raabjerg, and J. Åman Pohjola. Higher-order psi-calculi. *Mathematical Structures in Computer Science*, FirstView: 1–37, 6 2013. ISSN 1469-8072. . URL http://journals.cambridge.org/article_S0960129513000170.

[21] A. M. Pitts. Nominal logic, a first order theory of names and binding. *Information and Computation*, 186:165–193, 2003. URL http://dx.doi.org/10.1016/s0890-5401(03)00138-x.

[22] A. Popescu and E. Gunter. Incremental pattern-based coinduction for process algebra and its isabelle formalization. In L. Ong, editor, *Foundations of Software Science and Computational Structures*, volume 6014 of *Lecture Notes in Computer Science*, pages 109–127. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-12031-2. . URL http://dx.doi.org/10.1007/978-3-642-12032-9_9.

[23] D. Pous. Up-to techniques for weak bisimulation. In *ICALP*, pages 730–741, 2005. URL http://dx.doi.org/10.1007/11523468_59.

[24] D. Pous. Complete lattices and up-to techniques. In Z. Shao, editor, *APLAS*, volume 4807 of *Lecture Notes in Computer Science*, pages 351–366. Springer, 2007. ISBN 978-3-540-76636-0. URL http://dx.doi.org/10.1007/978-3-540-76637-7_24.

[25] D. Pous. New up-to techniques for weak bisimulation. *Theor. Comput. Sci.*, 380(1-2):164–180, 2007. URL http://dx.doi.org/10.1016/j.tcs.2007.02.060.

[26] D. Sangiorgi. On the bisimulation proof method. *Mathematical Structures in Computer Science*, 8(5):447–479, Oct. 1998. ISSN 0960-1295. . URL http://dx.doi.org/10.1017/S0960129598002527.

[27] D. Sangiorgi and R. Milner. The problem of "weak bisimulation up to". In R. Cleaveland, editor, *CONCUR*, volume 630 of *Lecture Notes in Computer Science*, pages 32–46. Springer, 1992. ISBN 3-540-55822-5. URL http://dx.doi.org/10.1007/BFb0084781.

[28] D. Sangiorgi and J. Rutten, editors. *Advanced Topics in Bisimulation and Coinduction*. Cambridge University Press, New York, NY, USA, 1st edition, 2011. ISBN 1107004977, 9781107004979.

[29] D. Sangiorgi and D. Walker. *The Pi-Calculus - a theory of mobile processes*. Cambridge University Press, 2001. ISBN 978-0-521-78177-0.

[30] C. Urban and C. Tasson. Nominal techniques in Isabelle/HOL. In R. Nieuwenhuis, editor, *Proceedings of CADE 2005*, volume 3632 of *Lecture Notes in Computer Science*, pages 38–53. Springer-Verlag, 2005. ISBN 3-540-28005-7. URL http://dx.doi.org/10.1007/s10817-008-9097-2.