HOCore in Coq

Petar Maksimovic, Alan Schmitt

HOCore in Coq

Petar Maksimović[1,2] & Alan Schmitt[1]

[1] Inria, France
[2] Mathematical Institute of the Serbian Academy of Sciences and Arts, Serbia

Abstract. We consider a recent publication on higher-order process calculi [12] and describe how its main results have been formalized in the Coq proof assistant. We highlight a number of important technical issues that we have uncovered in the original publication. We believe that these issues are not unique to the paper under consideration and require particular care to be avoided.

1 Introduction

Computer-aided verification has reached a point where it can be applied to state-of-the-art research domains, including compilers, programming languages, and mathematical theorems. Our goal is to contribute to this growing effort by formalizing a recent paper on process calculi.

The paper that we are examining is [12], by Lanese et al. It introduces HOCore—a minimal higher-order process calculus, which features input-prefixed processes, output processes, and parallel composition. Its syntax is as follows:

$$P ::= a(x).P \mid \overline{a}\langle P\rangle \mid P \parallel P \mid x \mid \mathbf{0}.$$

HOCore is minimal, in the sense that it features only the operators strictly necessary for higher-order communication. For one, there are no continuations following output messages. More importantly, there is no restriction operator, rendering all channels global and the dynamic creation of new channels impossible. The semantics of HOCore is presented in [12] in the form of a labeled transition system and it is shown that HOCore is a Turing-complete calculus. On the other hand, its observational equivalence is proven to be decidable. In fact, the main forms of strong bisimilarity considered in the literature (contextual equivalence, barbed congruence, higher-order bisimilarity, context bisimilarity, and normal bisimilarity [4,11,18,22]) all coincide in HOCore. Moreover, their synchronous and asynchronous versions coincide as well. Therefore, it is possible to decide that two processes are equivalent, yet it is impossible, in the general case, to decide whether or not they terminate. In addition, the authors give in [12] a sound and complete axiomatization of observational equivalence. Our

This work has been partially supported by the ANR project 2010-BLAN-0305 Pi-Coq, as well as by the Serbian Ministry of Education, Science and Technological Development, through projects III44006 and ON174026.

formalization in Coq [13] addresses all of the results of [12] concerning decidability of observational equivalence, coincidence of synchronous bisimilarities, as well as the soundness and completeness of the axiomatization.

Contributions. We present a formalization of the HOCore calculus and its behavioral theory in Coq. Throughout the development, we have strived to preserve a high degree of visual and technical correspondence between, on the one hand, the formulations of theorems and their proofs in the original paper and, on the other hand, their Coq counterparts. We introduce in Section 2 the syntactic and semantic concepts of HOCore: processes, freshness of variables and channels, structural congruence, and the labeled transition system. In particular, we focus on modeling bound variables using the canonical approach of Pollack et al. [16].

Sections 3 and 4 contain the main contributions of this paper. In Section 3, we present the formalization of various strong bisimilarities used in [12]: higher-order, context, normal, open normal, and IO-bisimilarity, together with the formal proofs that these five bisimilarities coincide and that they are decidable. In Section 4, we present the axiomatization of HOCore and the formal proof of its soundness and completeness. In both of these sections, we detail the errors, inaccuracies, and implicit assumptions made in the pen-and-paper proofs, and show how to correct them. These issues are not unique to the paper under consideration, and require particular care to be avoided.

Section 5 is reserved for the overview of related work. Finally, we summarize what we have learned in Section 6. The complete formalization, available online,[3] contains approximately 4 thousands of lines of code (kloc) of specification and 22 kloc of proofs. It was developed intermittently over a period of three years by the authors, Simon Boulier, and Martín Escardá. It relies on the TLC library—a general purpose Coq library equipped with many tactics for proof automation—developed by Arthur Charguéraud.[4] Due to lack of space, the details for some of the definitions and proofs can be found in the Appendix that is available online.

2 Formalizing HOCore

2.1 Syntax

HOCore is a minimal higher-order process calculus, based on the core of well-known calculi such as CHOCS [21], Plain CHOCS [23], and Higher-Order π-calculus [18,19,20]. The main syntactic categories of HOCore are channels (a, b, c), variables (x, y, z), and processes (P, Q, R). Channels and variables are atomic, whereas processes are defined inductively.

An input-prefixed process $a(x).P$ awaits to receive a process, e.g., Q, on the channel a; it then replaces all occurrences of the variable x in the process P with Q, akin to λ-abstraction. An output process[5] $\overline{a}\langle Q\rangle$ emits the process Q on the channel a. Parallel composition allows senders and receivers to interact.

[3] `http://www.irisa.fr/celtique/aschmitt/research/hocore/`

[4] `http://www.charguer aud.org/softs/tlc/`

[5] We also refer to an output process as an output message or, simply, a message.

$P,\ Q ::= \overline{a}\langle P\rangle$	output process
$\mid a(x).P$	input prefixed process
$\mid x$	process variable
$\mid P \parallel Q$	parallel composition
$\mid \mathbf{0}$	empty process

The only binding in HOCore occurs in input-prefixed processes. We denote the free variables of a process P by $\mathsf{fv}(P)$, and the bound ones by $\mathsf{bv}(P)$. In [12], processes are identified up to the renaming of bound variables. Processes that do not contain free variables are *closed* and those that do are *open*. If a variable x does not occur within a process P, we say that it is fresh with respect to that process, and denote this in the Coq development by $x\sharp P$. We denote lists with the symbol $\widetilde{\ }$: a list of variables is, for instance, denoted by $\widetilde{x}$. In addition, we define two notions of *size* for a process P: $s_{\parallel}(P)$, for which parallel compositions count; and $s(P)$, for which they do not. (see online Appendix)

Structural congruence. The structural congruence relation ($\equiv$) is the smallest congruence relation on processes for which parallel composition is associative, commutative, and has $\mathbf{0}$ as the neutral element.

Canonical representation of terms. We first address our treatment of bound variables and α-conversion. We have opted for the *locally named* approach of Pollack et al. [16]. There, the set of variables is divided into two distinct subsets: free (global) variables, denoted by X, Y, Z, and bound (local) variables, denoted by x, y, z. The main idea is to dispense with α-equivalence by calculating the value of a local variable canonically at the time of its binding. For this calculation, we rely on a *height function* f that takes a global variable X and a term P within which X will be bound, and computes the value of the corresponding local variable x. We then replace every occurrence of X with x in P.

To illustrate, consider the process $a(x).(x \parallel x)$. It is α-equivalent to $a(y).(y \parallel y)$, for any local variable y. We wish to choose a canonical representative for this entire family of processes. To that end, we start from the process $X \parallel X$, where X is free, calculate $x = f_X(X \parallel X)$, and take this x to be our canonical representative. In the following, we write $a[X]_f.P$ for $a(f_X(P)).[f_X(P)/X]P$.

The height function needs to meet several criteria so that it does not, for one, return a local variable already used in a binding position around the global variable to be replaced, resulting in a capture. We use the criteria presented in [16] that guarantee its well-definedness. Although it is not strictly necessary to choose a particular height function f satisfying these criteria, it is important to demonstrate that it exists, which we do. (see online Appendix)

Adjusting the syntax. The separation of variables into local and global ones is reflected in the syntax of processes

$$P,\ Q ::= a(x).P \ \mid\ \overline{a}\langle P\rangle \ \mid\ P \parallel Q \ \mid\ x \ \mid\ X \ \mid\ \mathbf{0}$$

which is encoded in Coq inductively as follows:

```
Inductive process : Set :=
 | Send    : chan    -> process -> process
 | Receive : chan    -> lvar    -> process -> process
 | Lvar    : lvar    -> process
 | Gvar    : var     -> process
 | Par     : process -> process -> process
 | Nil     : process.
```

Here, `chan`, `lvar`, and `var` are types representing channels, local variables, and global variables, respectively. We can now define substitution without any complications that would normally arise from the renaming of variables. (see online Appendix) We write $[Q/X]P$ for the substitution of the variable X with the process Q in P, and $[\widetilde{Q}/\widetilde{X}]P$ for the simultaneous substitution of distinct variables $\widetilde{X}$ with processes $\widetilde{Q}$ in P.

Well-formed processes. Given our height function f, we define *well-formed processes* as processes in which all local variables are bound[6] and computed using f. We refer to them as *wf-processes* and define the predicate `wf` that characterizes them inductively. Since all subsequent definitions, such as those of the labeled transition system, congruence, and bisimulations, consider wf-processes only, we define a dependent type corresponding to wf-processes, by pairing processes with the proof that they are well-formed.

```
Record wfprocess : Set := mkWfP {proc :> process; wf_cond : wf proc}.
```

A `wfprocess` is a record with one constructor, `mkWfP`. This record contains two fields: a process, named `proc`, and a proof that `proc` is well-formed, named `wf_cond`. We use Coq's coercion mechanism to treat a `wfprocess` as a `process` when needed. This approach allows us to reason about wf-processes directly, without additional hypotheses in lemmas or definitions. Each time a wf-process is constructed, however, we are required to provide a proof of its well-formedness. To this end, we introduce well-formed counterparts for all process constructors; e.g., we express parallel composition of wf-processes in the following way:

```
Definition wfp_Par (p q : wfprocess) :=
  mkWfP (Par p q) (WfPar p q (wf_cond p) (wf_cond q)).
```

where `WfPar` is used to construct well-formedness proofs for parallel processes:

[6] Here we have a slight overloading of terminology w.r.t. free and bound variables—while local variables are intended to model the bound variables of the object language, they can still appear free with respect to our adjusted syntax. For example, x is free in the process `Lvar x`, and we do not consider this process to be well-formed.

```
| WfPar  : forall (p q : process), (wf p) -> (wf q) -> (wf (Par p q))
```

Our initial development did not use dependent types, but relied on additional well-formedness hypotheses for most lemmas instead. We have discovered that, in practice, it is more convenient to bundle processes and well-formedness together. We can, for instance, use Coq's built-in notion of reflexive relations without having to worry about relating non wf-processes. Also, we can define transitions and congruence directly on wf-processes, thus streamlining the proofs by removing a substantial amount of code otherwise required to show that processes obtained and constructed during the proofs are indeed well-formed.

2.2 Semantics

The original semantics of HOCore is expressed via a labeled transition system (LTS) applied to a calculus with α-conversion. It features the following transition types: internal transitions $P \xrightarrow{\tau} P'$, input transitions $P \xrightarrow{a(x)} P'$, where P receives on the channel a a process that is to replace a local variable x in the continuation P', and output transitions $P \xrightarrow{\overline{a}\langle P'' \rangle} P'$, where P emits the process P'' on channel a and evolves to P'. This LTS is not satisfactory because input transitions mention the name of their bound variable. For one, the following transition has to be prevented to avoid name capture: $x \parallel a(x).x \xrightarrow{a(x)} x \parallel x$. To this end, side conditions are introduced and the entire semantics is defined "up-to α-conversion" to make sure terms do not get stuck because of the choice of a bound name. To simplify the formalization, we restate it using abstractions.

Abstractions and agents. An *agent* A is either a process P or an abstraction F. An *abstraction* can be viewed as a λ-abstraction inside a context.

$$A ::= P \mid F \qquad F ::= (x).P \mid F \parallel P \mid P \parallel F$$

We call these abstractions *localized*, since the binder (x) does not move (the usual approach to abstractions involves "lifting" the binder so that the abstraction remains of the form $(x).P$). Our approach, similar to the one in [24], allows us to avoid recalculation of bound variables. The application of an abstraction to a process, denoted by $F \bullet Q$, follows the inductive definition of abstractions.

$$((x).P) \bullet Q = [Q/x]P \quad (F \parallel P) \bullet Q = (F \bullet Q) \parallel P \quad (P \parallel F) \bullet Q = P \parallel (F \bullet Q)$$

Removal transitions. To simplify the definition of bisimulations that allow an occurrence of a free variable to be deleted from related processes, we also add transitions $X \xrightarrow{X} \mathbf{0}$, where a global variable dissolves into an empty process.

The Labeled Transition System. The LTS consists of the following seven rules.

Inp $a(x).P \xrightarrow{a} (x).P$ **Out** $\overline{a}\langle P\rangle \xrightarrow{\overline{a}\langle P\rangle} \mathbf{0}$ **Rem** $X \xrightarrow{X} \mathbf{0}$

Act1 If $P \xrightarrow{\alpha} A$, then $P \parallel Q \xrightarrow{\alpha} A \parallel Q$.
Act2 If $Q \xrightarrow{\alpha} A$, then $P \parallel Q \xrightarrow{\alpha} P \parallel A$.
Tau1 If $P \xrightarrow{\overline{a}\langle P''\rangle} P'$ and $Q \xrightarrow{a} F$, then $P \parallel Q \xrightarrow{\tau} P' \parallel (F \bullet P'')$.
Tau2 If $P \xrightarrow{a} F$ and $Q \xrightarrow{\overline{a}\langle P''\rangle} Q'$, then $P \parallel Q \xrightarrow{\tau} (F \bullet P'') \parallel Q'$.

In these rules, α denotes an arbitrary transition label. In Coq, we represent transitions inductively, providing a constructor for each rule. To illustrate, we present constructors corresponding to the rules Out, In, and Rem.

```
| TrOut : forall a p, {{wfp_Send a p -- LabOut a p ->> AP wfp_Nil}}
| TrIn  : forall a X p, {{wfp_Abs a X p -- LabIn a ->> AA (AbsPure X p)}}
| TrRem : forall X, {{wfp_Gvar X -- LabRem X ->> AP wfp_Nil}}
```

3 Coincidence and Decidability of Bisimilarities

We now present the formalization of the decidability and coincidence theorems of HOCore. We emphasize that, in order to arrive at those results, it was necessary to prove a number of auxiliary lemmas about structural congruence, substitution, multiple substitution, and transitions that were implicitly considered true in the original paper. Some of these lemmas were purely mechanical, while others required a substantial effort. (see online Appendix)

We begin by presenting the five forms of strong bisimilarity commonly used in higher-order process calculi and describe their formalization in Coq. Following [12], we first define a number of basic bisimulation clauses. We show here only those adjusted during the formalization. (see online Appendix)

Definition 1. *A relation $\mathcal{R}$ on HOCore processes is:*

- *a* variable *relation, if when $P\ \mathcal{R}\ Q$ and $P \xrightarrow{X} P'$, there exists a process Q', such that $Q \xrightarrow{X} Q'$ and $P'\ \mathcal{R}\ Q'$.*
- *an* input *relation, if when $P\ \mathcal{R}\ Q$ and $P \xrightarrow{a} F$, there exists an abstraction F', such that $Q \xrightarrow{a} F'$, and for all global variables X fresh in P and Q, it holds that $(F \bullet X)\ \mathcal{R}\ (F' \bullet X)$.*
- *an* input normal *relation, if when $P\ \mathcal{R}\ Q$ and $P \xrightarrow{a} F$, there exists an abstraction F', such that $Q \xrightarrow{a} F'$, and for all channels m fresh in P and Q it holds that $(F \bullet \overline{m}\langle \mathbf{0}\rangle)\ \mathcal{R}\ (F' \bullet \overline{m}\langle \mathbf{0}\rangle)$.*
- closed*, if when $P\ \mathcal{R}\ Q$ and $P \xrightarrow{a} F$, then there exist Q' and F', such that $Q \xrightarrow{a} F'$, and for all closed R, it holds that $(F \bullet R)\ \mathcal{R}\ (F' \bullet R)$.*

```
Definition var_relation (R : RelWfP) : Prop :=
  forall (p q : wfprocess), (R p q) ->
    forall (X : var) (p' : wfprocess), {{p -- LabRem X ->> (AP p')}} ->
      exists (q' : wfprocess), {{q -- LabRem X ->> (AP q')}} /\ (R p' q').
```

There are two main differences between these definitions and the ones presented in [12]. The first one, as previously mentioned, is the use of abstractions in place of variables in input transitions. The second difference is found in the definition of a *variable relation*; we use the transition **Rem**, while the one in [12] uses structural congruence. This change was motivated by the fact that we have chosen to define bisimulations semantically, in terms of processes capable of executing transitions. In that context, preserving the definition of a variable relation from [12] and involving structural congruence would be inconsistent. Moreover, working under the hypothesis that two processes are structurally congruent is very inconvenient, as their structure may be completely different.

Using these clauses, we define the five forms of strong bisimulation: higher-order ($\sim_{\mathtt{HO}}$), context ($\sim_{\mathtt{CON}}$), normal ($\sim_{\mathtt{NOR}}$), input-output ($\sim^{\circ}_{\mathtt{IO}}$), and open normal ($\sim^{\circ}_{\mathtt{NOR}}$) bisimulations. We present here only the ones that differ from those in [12].

Definition 2. *A relation $\mathcal{R}$ on HOCore processes is:*

- *an input-output* (IO) *bisimulation if it is symmetric, an input relation, an output relation, and a variable relation;*
- *an open normal bisimulation if it is symmetric, a τ-relation, an input relation, an output normal relation, and a variable relation.*

```
Definition IO_bisimulation (R : RelWfP) : Prop :=
 (Symmetric R) /\ (in_relation R) /\ (out_relation R) /\ (var_relation R).
```

For each of these bisimulations, we also consider a corresponding *bisimilarity*, i.e., the union of all corresponding bisimulations, and each of these bisimilarities is proven to be a bisimulation itself. Bisimilarity can naturally be viewed as a co-inductive notion, but its co-inductive aspects can also be expressed in a set-theoretic way, as follows:

```
Definition IObis (p q : wfprocess) : Prop :=
  exists R, (IO_bisimulation R) /\ (R p q).
```

To compare the set-theoretic and the native Coq co-inductive versions of bisimilarity, we have also defined the latter and proven it equivalent to the set-theoretic one by using a variation of Park's principle [7]. We use these two definitions interchangeably, depending on which is more suited to a given proof. For instance, the co-inductive definition is more convenient for statements in which the candidate relations are simple and follow the formulation of the statement, as there is no need to supply the relation during the proof process. In the cases where the candidate relations are more complicated, however, we find it more natural and closer to the paper proofs to use the set-theoretic definition.

We also define the extensions of $\sim_{\mathtt{HO}}$, $\sim_{\mathtt{CON}}$, and $\sim_{\mathtt{NOR}}$ to open processes by adding abstractions that bind all free variables, and denote these extensions by $\sim^{\star}_{\mathtt{HO}}$, $\sim^{\star}_{\mathtt{CON}}$, and $\sim^{\star}_{\mathtt{NOR}}$, respectively. (see online Appendix)

Relations and Bisimilarities "up-to". An important notion used in our development is that of two processes P and Q being in a relation $\mathcal{R}$ *up to* another relation $\mathcal{R}'$: assuming the processes are related by $\mathcal{R}$ before a transition, they are related by $\mathcal{R}'\mathcal{R}\mathcal{R}'$ after the transition. We establish the following connection between bisimilarities and bisimilarities "up-to", illustrated here only for $\sim^{\circ}_{\mathtt{NOR}}$-bisimilarity, but holding for all five.

Lemma 1. *Let $\mathcal{R}'$ be an equivalence relation that is also an ONOR-bisimulation. Then, $p \sim^{\circ}_{\mathtt{NOR}} q$ if and only if $p \sim^{\circ}_{\mathtt{NOR}} q$ up to $\mathcal{R}'$.*

We mainly use this technique to reason up to structural congruence. We prove $\equiv$ is an equivalence relation, and that it is an $\sim^{\circ}_{\mathtt{NOR}}$-bisimulation. By Lemma 1, we may reason up to $\equiv$ to show that processes are $\sim^{\circ}_{\mathtt{NOR}}$-bisimilar. This allows us to consider small bisimulation candidates and use $\equiv$ after a transition to get back in the candidate.

Existential vs. Universal Quantification of Freshness. Definitions of input, input normal, and output normal relations all feature universal quantification on a fresh channel and/or variable. Our formalization has revealed that this condition leads to major problems in proving a number of statements, such as transitivity of $\sim^{\circ}_{\mathtt{NOR}}$-bisimilarity, Lemma 1, and Lemma 3; these problems were not addressed in [12]. (see online Appendix) To correct this, we defined existentially-quantified $\sim^{\circ}_{\mathtt{IO}}$-, $\sim_{\mathtt{NOR}}$-, and $\sim^{\circ}_{\mathtt{NOR}}$-bisimilarities and proved they coincide with their universally-quantified counterparts. We could consider closing candidate relations under variable freshness instead, but this would complicate the formal proofs substantially. Much more importantly, existentially-quantified IO-bisimulation cannot be avoided in the formalization of the decidability procedure.

Decidability of IO-bisimilarity. The simplicity of IO-bisimilarity lets us show directly that not only it is a congruence, but that it is also decidable.

Lemma 2. *The following properties concerning $\sim^{\circ}_{\mathtt{IO}}$ hold:*

1. *$\sim^{\circ}_{\mathtt{IO}}$ is a congruence: if $P \sim^{\circ}_{\mathtt{IO}} P'$, then: (1) $a[X]_f.P \sim^{\circ}_{\mathtt{IO}} a[X]_f.P'$; (2) for all Q, $P \parallel Q \sim^{\circ}_{\mathtt{IO}} P' \parallel Q$; and (3) $\overline{a}\langle P\rangle \sim^{\circ}_{\mathtt{IO}} \overline{a}\langle P'\rangle$.*
2. *IO-bisimilarity and existential IO-bisimilarity coincide.*
3. *IO-bisimilarity and IO-bisimilarity up to $\equiv$ coincide.*
4. *$\sim^{\circ}_{\mathtt{IO}}$ is a τ-bisimulation.*
5. *$\sim^{\circ}_{\mathtt{IO}}$ is decidable.*

To show decidability, we specify a brute force algorithm that tests every possible transition for the two processes, up to a given depth. As IO-bisimilarity testing always results in smaller processes (there is no τ clause in this bisimilarity), we can show that this algorithm decides IO-bisimilarity, if the depth considered is large enough. This allows us to conclude:

```
Theorem IObis_constructive_decidable : forall p q, {p ≈ q} + {~(p ≈ q)}.
```

We should note here that the brute force algorithm would not be applicable to the definition of IO-bisimilarity as stated in Definition 2, because we would need to consider universal quantification on the fresh variable X in the case of an input transition. However, since we have proven the equivalence of universally- and existentially-quantified IO-bisimilarities, it is sufficient to check bisimilarity for only one arbitrary fresh X. More importantly, we strongly emphasize that, although the TLC library is based on classical logic, the decidability procedure that we have formalized is fully constructive.

Coincidence of Bisimilarities. First, we address one of the two most technically challenging lemmas of this part of the development (Lemma 4.15 of [12]).

Lemma 3. *If* $(m[X]_f.P') \parallel P \sim^{\circ}_{\mathtt{NOR}} (m[X]_f.Q') \parallel Q$ *for some fresh* m*, then* $P \sim^{\circ}_{\mathtt{NOR}} Q$ *and* $P' \sim^{\circ}_{\mathtt{NOR}} Q'$*.*

Proof. In order to show the first part of the lemma, we show that the relation

$$\mathcal{S} = \bigcup_{j=1}^{\infty} \left\{ (P,\, Q) \;:\; \left(\prod_{k \in 1..j} m_k[X_k]_f.P_k \right) \parallel P \sim^{\circ}_{\mathtt{NOR}} \left(\prod_{k \in 1..j} m_k[X_k]_f.Q_k \right) \parallel Q \right\}$$

is an existential $\sim^{\circ}_{\mathtt{NOR}}$-bisimulation, where $\{P_i\}_1^{\infty}$, $\{Q_i\}_1^{\infty}$ are arbitrary processes, $\{m_i\}_1^{\infty}$ are channels fresh with respect to P, Q, $\{P_i\}_1^{\infty}$, and $\{Q_i\}_1^{\infty}$, and each variable X_i is fresh with respect to P_i and Q_i. The proof proceeds as in [12], by examining possible transitions of P and showing these transitions are matched by Q. To this end, we study processes of the form $(\prod_{k \in 1..j} m_k[X_k]_f.P_k) \parallel P$ and prove several lemmas about their properties in relation to transitions and structural congruence. The original proof silently relies on the use of existentially-quantified $\sim^{\circ}_{\mathtt{NOR}}$-bisimilarity and up-to structural congruence proof techniques for this bisimilarity (up-to techniques are proven, but only for IO-bisimilarity). These assumed results were not trivial to formally establish.

For the second part, we determined that it is unnecessary to formalize the informal procedure to consume $\sim^{\circ}_{\mathtt{NOR}}$-bisimilar processes presented in [12]. Instead, it is sufficient to proceed by induction on $(s(P) + o(P))$, where $o(P)$ is the number of outputs in P, and use the already proven first part, reducing the formal proof to a technical exercise. We believe that the formalization of this consumption procedure would have been very difficult and would require a substantial amount of time and effort. This illustrates the insights one can gain through formal proving when it comes to proof understanding and simplification. □

We are now ready to state and prove the coincidence of bisimilarities.

Theorem 1. *The five strong bisimilarities coincide in HOCore.*

Proof. We prove the following implications, the direct corollary of which is our goal: (1) $\sim^{\circ}_{\mathtt{IO}}$ implies $\sim^{\star}_{\mathtt{HO}}$; (2) $\sim^{\star}_{\mathtt{HO}}$ implies $\sim^{\star}_{\mathtt{CON}}$; (3) $\sim^{\star}_{\mathtt{CON}}$ implies $\sim^{\star}_{\mathtt{NOR}}$; (4) $\sim^{\star}_{\mathtt{NOR}}$ implies $\sim^{\circ}_{\mathtt{NOR}}$; (5) $\sim^{\circ}_{\mathtt{NOR}}$ implies $\sim^{\circ}_{\mathtt{IO}}$.

The only major auxiliary claim that needs to be proven here is that an open normal bisimulation also satisfies the output relation clause; this is an immediate consequence of Lemma 3. □

To conclude this section, we focus on the error discovered in the proof of the right-to-left direction of Lemma 4.14 of [12], stating that higher order bisimilarity implies IO-bisimilarity. Its proof was the most challenging to formalize in this part of the development and this formalization has led us to several important insights. First, we need the following auxiliary lemma.

Lemma 4. *$P \sim^{\star}_{\mathtt{HO}} Q$ if and only if for all closed $\widetilde{R}$, $[\widetilde{R}/\widetilde{X}]P \sim_{\mathtt{HO}} [\widetilde{R}/\widetilde{X}]Q$, where $\widetilde{X} = \mathsf{fv}(P) \cup \mathsf{fv}(Q)$. (Lemma 4.13 of [12]).*

While the proof of this claim takes only several lines on paper, its Coq version amounts to more than three hundred lines of code, requiring a number of auxiliary lemmas that appear evident in a pen-and-paper context. These lemmas mostly deal with with permutations of elements inside a list and the treatment of channels in the "opening" of $\sim_{\mathtt{HO}}$. We can now proceed to our desired claim.

Lemma 5. *$\sim^{\star}_{\mathtt{HO}}$ implies $\sim^{\circ}_{\mathtt{IO}}$.*

Discussion. It suffices to prove that the relation $\mathcal{R} = \{(P, Q) \mid [\widetilde{M}/\widetilde{X}]P \sim^{\star}_{\mathtt{HO}} [\widetilde{M}/\widetilde{X}]Q\}$ is an IO-bisimulation, where $\widetilde{X}$ and $\widetilde{M}$ are, respectively, lists of variables and messages carrying $\mathbf{0}$ on fresh channels. We show $\mathcal{R}$ is an input, an output, and a variable relation . The proofs of all three cases in [12] share the same structure that relies on the application of both directions of Lemma 4.

First, the unguarded variables[7] are substituted with processes of the form $\overline{m}\langle\mathbf{0}\rangle$. Then, the remaining free variables are substituted with arbitrary closed processes $\widetilde{R}$, using Lemma 4 left-to-right to show that the result is still in $\mathcal{R}$. Next, the processes do a transition, and it is proposed to apply Lemma 4 right-to-left to conclude. This last step is not justified, as we now explain. The main idea is to show that the bisimulation diagrams are closed under multiple substitution, i.e., that for all processes P and Q, all global variables X, and all closed processes $\widetilde{R}$, given $[\widetilde{R}/\widetilde{X}]P \sim_{\mathtt{HO}} [\widetilde{R}/\widetilde{X}]Q$ and $[\widetilde{R}/\widetilde{X}]P \xrightarrow{\alpha} [\widetilde{R}/\widetilde{X}]P'$, there exists a Q' such that $[\widetilde{R}/\widetilde{X}]Q \xrightarrow{\alpha} [\widetilde{R}/\widetilde{X}]Q'$ and $[\widetilde{R}/\widetilde{X}]P' \sim_{\mathtt{HO}} [\widetilde{R}/\widetilde{X}]Q'$. Then, the idea is to apply Lemma 4 right-to-left to obtain $P' \sim^{\star}_{\mathtt{HO}} Q'$. This is accomplished by first showing that there exists an S, such that $[\widetilde{R}/\widetilde{X}]Q \xrightarrow{\alpha} S$ and $[\widetilde{R}/\widetilde{X}]P' \sim_{\mathtt{HO}} S$, and then showing that there exists a Q', such that $S = [\widetilde{R}/\widetilde{X}]Q'$. The crucial mistake is that Q' here depends on $\widetilde{R}$, unless proven otherwise. Thus, the requirement for a unique Q' with a universal quantification on $\widetilde{R}$ of Lemma 4 right-to-left is not satisfied, and the lemma cannot be applied.

[7] Variables that appear in an execution context, i.e., those not "guarded" by an input.

Our initial attempt at this proof followed this approach, which failed when Coq refused to let us generalize $\widetilde{R}$ since Q' depended on it. We thus proceeded differently, by first substituting every free variable with processes of the form $\overline{m}\langle\mathbf{0}\rangle$, using Lemma 4 left-to-right to show that we remain in the relation. We then applied several auxiliary lemmas to finish the proof. While the error in this proof was corrected relatively easily, its very existence in the final version of [12] is indicative of the need for a more formal treatment of proofs in this domain, a need further substantiated by the findings presented in the following Section.

4 Axiomatization of HOCore

In this section, we present a formal proof of the soundness and completeness of the axiomatization of HOCore and reflect on the errors discovered in the pen-and-paper proofs. First, we briefly outline the treatment of axiomatization in [12]. The authors begin by formulating a cancellation lemma for $\sim^{\circ}_{\mathtt{IO}}$:

Lemma 6. *For all processes P, Q, R, if $P \parallel R \sim^{\circ}_{\mathtt{IO}} Q \parallel R$, then also $P \sim^{\circ}_{\mathtt{IO}} Q$.*

Next, they introduce the notion of a prime process and prime decomposition.

Definition 3. *A process P is prime if $P \not\sim^{\circ}_{\mathtt{IO}} \mathbf{0}$ and $P \sim^{\circ}_{\mathtt{IO}} P_1 \parallel P_2$ implies $P_1 \sim^{\circ}_{\mathtt{IO}} \mathbf{0}$ or $P_2 \sim^{\circ}_{\mathtt{IO}} \mathbf{0}$. When $P \sim^{\circ}_{\mathtt{IO}} \prod_{i=1}^{n} P_i$, with each P_i prime, we say that $\prod_{i=1}^{n} P_i$ is a prime decomposition of P.*

The authors then prove that every process admits a prime decomposition unique up to bisimilarity and permutation of indices. Next, extended structural congruence $(\equiv_E,)$ is defined by adding to $\equiv$ the following distribution law:

$$a(x).\left(P \parallel \prod_{1}^{k-1} a(x).P\right) \equiv_E \prod_{1}^{k} a(x).P.$$

Next, a reduction relation $\rightsquigarrow$ and normal forms are defined as follows:

Definition 4. *We write $P \rightsquigarrow Q$ if there exist processes P' and Q', such that $P \equiv P'$, $Q' \equiv Q$, and Q' is obtained from P' by rewriting a subterm of P' using the distribution law left-to-right. A process P is in* normal form *if it cannot be further simplified in the system $\equiv_E$ using $\rightsquigarrow$.*

Finally, the following three claims conclude the axiomatization section of [12]:

Lemma 7. *If $a(x).P \sim^{\circ}_{\mathtt{IO}} Q \parallel Q'$, with $Q, Q' \not\sim^{\circ}_{\mathtt{IO}} \mathbf{0}$, then $a(x).P \sim^{\circ}_{\mathtt{IO}} \prod_1^k a(x).R$, with $k > 1$ and $a(x).R$ in normal form.*

Lemma 8. *For all P, Q in normal form, if $P \sim^{\circ}_{\mathtt{IO}} Q$, then $P \equiv Q$.*

Theorem 2. *$\equiv_E$ is a sound and complete axiomatization of $\sim^{\circ}_{\mathtt{IO}}$ in HOCore.*

Our formalization of this section has led to the discovery of the following imprecisions and errors.

Cancellation Lemma. For the proof of Lemma 6 in [12], the authors attempt to re-use the proof from [14], that proceeds by induction on the sum of sizes of P, Q, and R. This, however, is not possible, as that proof was designed for a calculus with operators structurally different than the ones used in HOCore. Instead, we need to use the candidate relation $\mathcal{R}$, parameterized by a natural number n, such that $P\ \mathcal{R}\ Q$ at level n if and only if there exists a process R such that $s(P) + s(Q) + s(R) \leq n$ and $P \parallel R \sim^{\circ}_{\mathtt{IO}} Q \parallel R$, and prove that this candidate relation is an IO-bisimulation by total induction on n.

"Deep" prime decomposition. In the proof of Lemma 7, the authors in [12] claim that every process is bisimilar to a parallel composition of a collection of prime processes in normal form. They prove this by taking the normal form of a process, performing a prime decomposition on this normal form, and concluding that all of the constitutents in this prime decomposition are both prime and in normal form. This, however, is not necessarily true, as there is no proof that taking the prime decomposition of a process preserves the fact that it is in normal form. In particular, the definition of primality considers only the top-level structure of the process: we can prove that every output message $\overline{a}\langle P\rangle$ is prime for all a and P, but we cannot obtain any information on the primality of P from this. On the other hand, normal forms are defined "in depth". Because of these discrepancies, the proof in [12] cannot be concluded. To remedy this, we need to define a "deep" version of primality that recursively requires of all sub-processes of P to be prime as well, together with the notion of "deep" prime decomposition, using which we can prove Lemma 7 properly.

Normal forms and the proof of Lemma 8. The proof of Lemma 8 in [12] exhibits several errors. It is performed by induction on $s(P)$, simultanously proving $\mathcal{A}$ and $\mathcal{C}$, where $\mathcal{A}$ is an auxiliary lemma stating that if P is an input-prefixed process in normal form, then it is prime, and $\mathcal{C}$ is the main claim. However, the induction must actually be performed on $\mathcal{A} \wedge (\mathcal{A} \to \mathcal{C})$, since $\mathcal{C}$ cannot be proven otherwise (the third case of item 2 on page 27 of [12] fails). Much more importantly, the seemingly trivial case when $P = a(x).P' \parallel \mathbf{0}$ is not covered by the proof, and it turned out that it cannot be covered at all with the proof structure as presented in [12]. As we were unable to immediately discover an alternate proof method, we decided to adjust the notion of normal form by disallowing empty processes in parallel composition. This meant that normal forms could no longer be defined using $\rightsquigarrow$, but had to be defined directly instead. Interestingly, we can say that normal forms are now "more normal" w.r.t. those in [12], as they are unique only up to commutativity and associativity of parallel composition. With these corrections, we were able to conclude the proof of Lemma 8.

5 Related Work

We first discuss alternative approaches to the treatment of binding and α-conversion, starting from *Nominal Isabelle* [25,10,27], where nominal datatypes are used to represent α-equivalence classes. This extension of Isabelle has been successfully applied in a number of formalizations of various calculi with binders

[15,2,26], and taking advantage of it would probably reduce the size of our development. However, since one of our main results is the formalization of the decidability procedure for IO-bisimulation, and since stating decidability in Isabelle is not a trivial task, we have ultimately opted to use Coq.

The locally nameless approach [1,5] is the one most similar to the one we are using. There, variables are also split into two categories—local and global—but local variables are calculated by using de Bruijn indices instead of a height function. We find this approach to be both equally viable and equally demanding—freshness and well-formedness would both have to be defined, albeit differently, the proofs would retain their level of complexity, and the amount of code required would not be reduced substantially.

In the Higher-Order-Abstract-Syntax (HOAS), binders of the meta-language are used to encode the binding structure of the object language. In some settings, HOAS can streamline the development and provide an elegant solution for dealing with α-conversion, but in the case of process calculi it also brings certain drawbacks. As stated in [9], there are difficulties with HOAS in dealing with meta-theoretic issues concerning names, especially when it comes to the opening of processes, to the extent that certain meta-theoretic properties that involve substitution and freshness of names inside proofs and processes cannot be proven inside the framework and must be postulated instead.

In light of everything previously stated in this section and since HOCore does not have a restriction operator, we have decided to use the canonical locally named approach [16] for variables bound by the input operator. We have not yet considered more general approaches for name binding, such as [17]. In fact, our development may easily be adapted to other models for binders, as long as binders remain canonical (α-convertibility is equality).

We now turn to works related to the formalization of process calculi and their properties. To the best of the authors' knowledge, there have been no formalizations of the higher-order π-calculus so far. There are, however, many formalizations of the π-calculus in various proof assistants, including Coq, such as [8] (where the author uses de Bruijn indices) and [9] (where the authors use HOAS). The work closest to ours is the recent formalization in Isabelle of higher-order Psi-calculi [15], an extension of [2] to the higher-order setting. We have developed our own formalization because we wanted to stay as close to the paper proofs as possible, in particular when it came to the handling of higher-order and the definitions of the many bisimulations involved. Although we feel that translating HOCore and its bisimulations into a higher-order psi-calculus constitutes a very interesting problem, it is beyond the scope of this paper.

Preliminary versions of this paper have appeared in [3,6]. The former contained the initial results of the effort, which included the locally nameless technique for dealing with binders, the correction of the semantics, and the results on decidability. The latter added to this the new and more solid treatment of well-formed processes as records and the proof of the coincidence of bisimilarities. This version rounds this formalization effort up with results on the soundness

and completeness of the axiomatization, as well as the discovery and correction of a number of errors and imprecisions made in [12].

6 Lessons Learned and Conclusions

This formalization of HOCore in Coq has given us a deeper understanding of both the calculus itself and the level of precision required for proving properties of bisimulations in a higher-order setting. Moreover, it has led to the discovery of several major flaws in the proofs of [12]. First, there was the improper generalization of a hypothesis in the proof of Lemma 5. This flaw was due to a lapse in the tracking of the context inside which a property is derived, and it is in precisely such tracking that proof assistants excel. Next, the proof of Lemma 6 was incorrect, as it relied on a previously existing technique not applicable to higher-order process calculi. Finally, two notions crucial for the axiomatization of HOCore—prime decomposition and normal forms—were not defined correctly and did not take into account, respectively, the need for structural recursion in primality and the subtle impact of empty processes in parallel composition. These errors have all been corrected during the formalization and did not ultimately affect the correctness of the main results of [12], but we feel that it is their very presence in a peer-reviewed, state-of-the-art paper that strongly underlines the need for a more precise formal treatment of proofs in this domain.

We have also identified two missing results required by some lemmas. The first concerns existentially-quantified bisimulations: certain properties of bisimulations cannot be proven if the variables or channels featured in freshness conditions of bisimulation clauses are universally quantified. We solved this by introducing bisimulations with existential quantification and demonstrating that they are equivalent to the universally quantified ones. We have also discovered that existentially-quantified bisimulations are necessary for the formulation of the decidability procedure. The second missing result concerns transitivity of bisimulations, which is, in turn, required to prove that open normal bisimilarity and existential open normal bisimilarity coincide. This claim was not trivial to formalize; it required the use of up-to techniques, which were also left untreated for some bisimilarities in [12].

In the end, we were able to prove the decidability of IO-bisimilarity, the main coincidence theorem, and their supporting lemmas, as well as the results on the soundness and completeness of the axiomatization. We realized that some of the lemmas of the paper are actually not needed, such as Lemma 5. Additionally, we significantly simplified some proofs. For instance, the proof of Lemma 3 (Lemma 4.15 in [12]) does not require the consumption procedure described in [12]; induction on the combined measure $m(P)$ is sufficient and more elegant.

As a side effect of our precise treatment of bound variables, we have detected and corrected an error in the original semantics of HOCore, streamlining it in the process. Labels for input transitions no longer contain the variable to be substituted, but evolve to a localized abstraction instead. We have also introduced

the deletion of a variable as a transition in the LTS, removing a special case in the definition of the bisimulations and avoiding the use of structural congruence.

In fact, many of the choices that were made during the formalization were guided by the intent to avoid structural congruence. Although it is very convenient to be able to freely change the structure of a process, doing so interferes considerably with local reasoning. There were cases, however, where we could not avoid it, in particular when proving properties of normal bisimulations. There, we have used "up-to structural congruence" techniques to keep the bisimulation candidates small enough. We have discovered that the discord between the rigid syntax of a formal process and the intuition that it models a "soup" where processes can move around freely and interact with each other made the formalizations of some of the proofs more difficult than anticipated.

We have also proved that the co-inductive and set-theoretic definitions of IO-bisimilarity are equivalent, and used them interchangeably, depending on the complexity of the candidate relation at hand.

As for further work, our immediate aim is to formalize the results presented in Section 5 of [12], pertaining to the correctness and completeness of IO-bisimilarity w.r.t. barbed congruence. At this point, we have the correctness proof, which, given Theorem 1, means that all of the five forms of bisimilarity in HOCore are correct w.r.t. barbed congruence. Afterwards, we plan to extend the formalization of HOCore with name restriction, before tackling more complex features such as passivation.

In conclusion, we hope that the experience described in this paper will serve as a motivational step towards a more systematic use of proof assistants in the domain of process calculi.

References

1. B. Aydemir, A. Charguéraud, B. C. Pierce, R. Pollack, and S. Weirich. Engineering formal metatheory. In *ACM SIGPLAN–SIGACT Symposium on Principles of Programming Languages (POPL)*, pages 3–15. ACM, Jan. 2008.
2. J. Bengtson and J. Parrow. Psi-calculi in Isabelle. In *Proceedings of the 22nd International Conference on Theorem Proving in Higher Order Logics*, TPHOLs '09, pages 99–114, Berlin, Heidelberg, Aug. 2009. Springer-Verlag.
3. S. Boulier and A. Schmitt. Formalisation de HOCore en Coq. In *Actes des 23èmes Journées Francophones des Langages Applicatifs*, Jan. 2012.
4. Z. Cao. More on bisimulations for higher order *pi*-calculus. In *Proc. of FoSSaCS'06*, volume 3921 of *LNCS*, pages 63–78. Springer, 2006.
5. A. Charguéraud. The locally nameless representation. *Journal of Automated Reasoning*, pages 1–46, 2011. 10.1007/s10817-011-9225-2.
6. M. Escarrá, P. Maksimović, and A. Schmitt. HOCore in Coq. In *Actes des 26èmes Journées Francophones des Langages Applicatifs*, Jan. 2015.
7. E. Gimenez. *A Tutorial on Recursive Types in Coq*, 1998. Tech. Rep. No. 0221.
8. D. Hirschkoff. A full formalisation of pi-calculus theory in the calculus of constructions. In *Proceedings of the 10th International Conference on Theorem Proving in Higher Order Logics*, volume 1275, pages 153–169. Springer, Aug. 1997.

9. F. Honsell, M. Miculan, and I. Scagnetto. pi-calculus in (co)inductive-type theory. *Theoretical Computer Science*, 253(2):239–285, Feb. 2000.
10. B. Huffman and C. Urban. A new foundation for Nominal Isabelle. In M. Kaufmann and L. C. Paulson, editors, *Interactive Theorem Proving, First International Conference, ITP 2010, Edinburgh, UK, July 2010.*, pages 35–50. Springer, 2010.
11. A. Jeffrey and J. Rathke. Contextual equivalence for higher-order pi-calculus revisited. *Log. Meth. Comput. Sci.*, 1(1):1–22, 2005.
12. I. Lanese, J. A. Pérez, D. Sangiorgi, and A. Schmitt. On the expressiveness and decidability of higher-order process calculi. *Information and Computation*, 209(2):198–226, Feb. 2011.
13. The Coq development team. *Coq reference manual*, 2014. version. 8.4.
14. R. Milner and F. Moller. Unique decomposition of processes. *Theor. Comput. Sci.*, 107(2):357–363, 1993.
15. J. Parrow, J. Borgström, P. Raabjerg, and J. Åman Pohjola. Higher-order psi-calculi. *Mathematical Structures in Computer Science*, FirstView:1–37, 3 2014.
16. R. Pollack, M. Sato, and W. Ricciotti. A canonical locally named representation of binding. *J. Autom. Reasoning*, pages 1–23, May 2011. 10.1007/s10817-011-9229-y.
17. N. Pouillard and F. Pottier. A fresh look at programming with names and binders. In *Proceedings of the fifteenth ACM SIGPLAN International Conference on Functional Programming (ICFP 2010)*, pages 217–228, Sept. 2010.
18. D. Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms.* PhD thesis, Univ. of Edinburgh, Dept. of Comp. Sci., 1992.
19. D. Sangiorgi. Bisimulation for higher-order process calculi. *Information and Computation*, 131(2):141–178, dec 1996.
20. D. Sangiorgi. π-calculus, internal mobility and agent-passing calculi. *Theor. Comput. Sci.*, 167(2):235–274, 1996.
21. B. Thomsen. A calculus of higher order communicating systems. In *Proc. of POPL'89*, pages 143–154. ACM Press, 1989.
22. B. Thomsen. *Calculi for Higher Order Communicating Systems.* PhD thesis, Imperial College, 1990.
23. B. Thomsen. Plain CHOCS: A second generation calculus for higher order processes. *Acta Inf.*, 30(1):1–59, 1993.
24. A. Tiu and D. Miller. Proof search specifications of bisimulation and modal logics for the pi-calculus. *ACM Transactions on Computational Logic (TOCL)*, 11:13:1–13:35, January 2010.
25. C. Urban. Nominal techniques in Isabelle/HOL. *Journal of Automated Reasoning*, 40(4):327–356, 2008.
26. C. Urban, J. Cheney, and S. Berghofer. Mechanizing the metatheory of LF. *ACM Trans. Comput. Log.*, 12(2):15, 2011.
27. C. Urban and C. Kaliszyk. General bindings and alpha-equivalence in nominal Isabelle. In *Programming Languages and Systems - 20th European Symposium on Programming, ESOP 2011, Saarbrücken, Germany*, pages 480–500. Springer, 2011.