A completeness proof for bisimulation in the pi-calculus using Isabelle

Jesper Bengtson and Joachim Parrow

Department of Information Technology
Uppsala University
Uppsala, Sweden

Abstract

We use the interactive theorem prover Isabelle to prove that the algebraic axiomatization of bisimulation equivalence in the pi-calculus is sound and complete. This is the first proof of its kind to be wholly machine checked. Although the result has been known for some time the proof had parts which needed careful attention to detail to become completely formal. It is not that the result was ever in doubt; rather, our contribution lies in the methodology to prove completeness and get absolute certainty that the proof is correct, while at the same time following the intuitive lines of reasoning of the original proof. Completeness of axiomatizations is relevant for many variants of the calculus, so our method has applications beyond this single result. We build on our previous effort of implementing a framework for the pi-calculus in Isabelle using the nominal data type package, and strengthen our claim that this framework is well suited to represent the theory of the pi-calculus, especially in the smooth treatment of bound names.

Keywords: pi-calculus, Theorem Provers, Isabelle, Bisimulation, Axiomatization

1 Introduction

We derive a soundness and completeness proof for the axiomatization of strong late bisimilarity in the pi-calculus [8]. The proof is machine checked in the interactive theorem prover Isabelle [9], using the nominal data type package [1]. It represents a continuation of our work [3] on formalizing the pi-calculus in Isabelle. The completeness proof is the first formally verified proof of its kind. The original manual counterpart is little more than a sketch, and our mechanically verified proof follows it quite closely in structure. Therefore we regard our contribution as threefold: it clarifies exactly what is needed for the full formalization of this particular proof, it opens the possibility to check similar proofs more easily, and it supports our claim that the Isabelle formulation in the nominal data type package is very well suited for deriving this kind of result.

The pi-calculus, like most similar calculi, is equipped with a structural operation semantics (SOS) that defines the meaning of agents in terms of the communication actions they can perform. This semantics is used to define bisimulation equivalence as, loosely put, the largest equivalence such that whatever an agent can do, any equivalent agent can mimic and remain equivalent. There are several variants of this and we study one of the most basic ones, namely late strong bisimilarity.

But there is also a completely different way to characterize an equivalence: postulate a set of algebraic laws and say that two agents are equivalent precisely when they can be proved so from the laws and standard equational reasoning. This way does not use the SOS semantics at all.

We here use algebraic laws from [8] and formalize the now well known result that they precisely capture late strong bisimilarity. The proof follows the standard structure and is partitioned into a soundness part (saying that each law is sound, i.e. that in all instances the right hand side is bisimilar to the left hand side) and a completeness part (saying that if two agents are bisimilar then there exists a proof of equivalence from the algebraic laws). As is typical in these situations the soundness proof is straightforward but tedious to write out in detail, while the completeness proof requires a bit more ingenuity. A common way, which we shall follow, is to define a subset of the agents to be so called head normal forms. These have their immediate communication abilities as the outmost syntactic operators, so in a sense the initial behaviour is apparent from the syntactic structure. We then conclude the result by first showing that each agent has a provably equivalent head normal form, and that two bisimilar head normal forms are provably equivalent.

The original manual proof of this particular result dates back to the very first presentation of the pi-calculus [8] and its sketch occupies about one page (on pages 67 and 68) in the journal. The result was not controversial since it is one of many similar results on complete axiomatizations in process algebras, the first being by Hennessy and Milner on CCS [5]. Variants of the proof have been used in variants of the calculus, but never written down in full detail. It contains several sweeping statements which require quite some attention to detail to verify formally. An example is the claim that if two head normal forms have provably equivalent summands then the agents are provably equivalent using laws for commutativity, associativity and idempotence of the sum operator (formalized as Lemma 4.10 in Section 4 below).

In conclusion we regard this as one of the most extensive results from a mechanized theorem prover concerning the meta theory of process algebra. We go beyond other significant efforts on the pi-calculus [11,7,4,6] which do not treat completeness at all. There are today several other mechanized proofs from meta mathematics, for example on Gödel's incompleteness result [12] and on typing in system F in the PoplMark Challenge [2].

2 The pi-calculus and its complete axiomatization

2.1 Syntax

We assume the reader to be familiar with the basics of the pi-calculus. In this paper we use a standard variant of the monadic pi-calculus without recursion and replication. The syntax of the calculus is defined in Table 1. We let P, Q etc. range over agents and $a, b, \ldots, z$ over names.

In the input Prefix $a(x) . P$ is said to *bind* x in P, and occurrences of x in P

Agents $P ::=$	$\mathbf{0}$	Nil
	$\overline{a}x \,.\, P$	Output Prefix
	$a(x) \,.\, P$	Input Prefix
	$\tau \,.\, P$	Silent Prefix
	$P + P$	Sum
	$P \mid P$	Parallel
	$[x = y]P$	Match
	$[x \neq y]P$	Mismatch
	$(\nu x)P$	Restriction

Table 1
The syntax of the π-calculus.

are then called *bound*. In contrast the output Prefix $\overline{a}x \,.\, P$ does not bind x. These Prefixes are said to have *subject* a and *object* x, where the object is called *free* in the output Prefix and *bound* in the input Prefix. The silent Prefix τ has neither subject nor object. The Restriction operator $(\nu x)P$ also binds x in P. The *free names* $\texttt{fn}(P)$ are those with a not bound occurrence, and we say that is x is *fresh for* P to mean that $x \notin \texttt{fn}(P)$, or just that x *fresh* to mean it is fresh for all agents in the context of the discussion. Similarly the *bound names* $\texttt{bn}(P)$ are those with a bound occurrence.

A *substitution* is a function from names to names. We write $\{x/y\}$ for the substitution that maps y to x and is identity for all other names. The agent $P\{x/y\}$ is P where all free names x are replaced by y, with alpha-conversion wherever needed to avoid captures.

A sum of several agents $P_1 + \cdots + P_n$ is written $\sum_{i=1}^{n} P_i$, or just $\sum_j P_j$ when n is unimportant or obvious, and we here allow the case $n = 0$ when the sum means $\mathbf{0}$.

2.2 Structural Operational Semantics

The *actions* ranged over by α consist of four classes:

(i) The internal action τ.

(ii) The (free) output actions of kind $\overline{a}x$.

(iii) The input actions of kind $a(x)$.

(iv) The bound output actions $\overline{a}\nu x$.

We write $x \notin \alpha$ to mean that x does not occur in α. The structural operational semantics is given in Table 2. As can be seen it is a late version, without structural congruence.

prefix $$\frac{}{\alpha\,.\,P \xrightarrow{\alpha} P}$$

sum $$\frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$$

match $$\frac{P \xrightarrow{\alpha} P'}{[x = x]P \xrightarrow{\alpha} P'}$$

mismatch $$\frac{P \xrightarrow{\alpha} P',\ x \neq y}{[x \neq y]P \xrightarrow{\alpha} P'}$$

par $$\frac{P \xrightarrow{\alpha} P',\ \mathtt{bn}(\alpha) \cap \mathtt{fn}(Q) = \emptyset}{P|Q \xrightarrow{\alpha} P'|Q}$$

com $$\frac{P \xrightarrow{a(x)} P',\ Q \xrightarrow{\overline{a}u} Q'}{P|Q \xrightarrow{\tau} P'\{u/x\}|Q'}$$

res $$\frac{P \xrightarrow{\alpha} P',\ x \notin \alpha}{(\nu x)P \xrightarrow{\alpha} (\nu x)P'}$$

open $$\frac{P \xrightarrow{\overline{a}x} P',\ a \neq x}{(\nu x)P \xrightarrow{\overline{a}\nu x} P'}$$

close $$\frac{P \xrightarrow{a(x)} P',\ \ Q \xrightarrow{\overline{a}\nu u} Q'}{P|Q \xrightarrow{\tau} (\nu u)(P'\{u/x\}|Q')}$$

Table 2
The operational semantics. The symmetric versions of **sum, par, com** and **close** are elided. Agents are identified up to alpha equivalence, i.e. choice of bound names.

2.3 Bisimulation

We recapitulate the definition of strong late bisimulation.

Definition 2.1 A *bisimulation* is a symmetric binary relation $\mathcal{R}$ on agents satisfying the following: $P\mathcal{R}Q$ and $P \xrightarrow{\alpha} P'$ where $\mathtt{bn}(\alpha)$ is fresh implies that

(i) If $\alpha = a(x)$ then $\exists Q' :\ \ Q \xrightarrow{a(x)} Q' \ \wedge\ \forall u :\ P'\{u/x\}\mathcal{R}Q'\{u/x\}$

(ii) If α is not an input then $\exists Q' :\ Q \xrightarrow{\alpha} Q' \ \wedge P'\mathcal{R}Q'$

P and Q are *bisimilar*, written $P \sim Q$, if they are related by a bisimulation.

2.4 Axiomatization

The axioms for strong late bisimilarity are given in Table 3 and 4. We also implicitly use the laws of equational reasoning, i.e., that equality between agents is reflexive, symmetric and transitive. Note that substitutivity (that an agent can replace an equal agent in any expression) is *not* implied, since bisimilarity is not a congruence. Instead the **congr** laws define the substitutive properties.

The first laws **str1-5** are normally part of a structural congruence which also contains more laws. We here use only the structural laws that are actually needed for the completeness proof. In particular we will not need any structural laws for parallell since all instances of those can be derived from the expansion law. Incidentally, law **str5** can be replaced by the two simpler laws $(\nu x)\mathbf{0} = \mathbf{0}$ and $(\nu x)(\nu x)P = (\nu x)P$.

We say that two agents P and Q are *provably equivalent*, written $P \equiv Q$, if their equality can be established by these axioms and equational reasoning. The main result is the theorem that relates provable equivalence with bisimilarity:

Theorem 2.2 $P \sim Q$ *iff* $P \equiv Q$

This theorem has been known since the late 1980's when the pi-calculus was first conceived. Its validity has never been in doubt, even though all proofs presented until now have used hand waving at several points. To give the reader an overview of the proof, and also to appreciate the level of detail this kind of proof usually is presented with, we here cite parts of the proof from [10]. Soundness, i.e. the implication from right to left, merits no more than "It is easily seen that all laws are sound, so if $P = Q$ is provable then it must hold that $P \sim Q$." (The original proof in [8] is just a little more detailed; it goes on to postulate bisimulation relations for all algebraic laws without actually proving that these relations are bisimulations). Concerning the completeness part, on page 529 we read, for the subcalculus without the parallel operator:

Let the *depth* of an agent be the maximal nesting of its Prefixes.

Proposition 7. Using the axioms every agent P is provably equal to a *head normal form* (hnf) of kind $\sum_i \alpha_i \,.\, P_i$ of no greater depth.

The proof is by induction over the structure of the agent and all cases are easy. If P is a Match or Mismatch then **m1**–**mm2** applies; if it is a Restriction then **r3** is used to distribute it onto the summands and **r1**–**r2** to push it through the Prefixes or form part of a bound output Prefix.

Proposition 8 If $P \sim Q$ then $P = Q$ is provable from the axioms.

The proof is by induction on the depths of P and Q. By Proposition 7 we can assume P and Q are head normal forms. The base case $P = Q = \mathbf{0}$ is trivial. For the inductive step we prove that for each summand in P there is a provably equal summand in Q and vice versa. For example, take a summand $a(x) \,.\, P'$ in P. Assume by alpha-conversion that all top-level input actions have the same

str1	$P + \mathbf{0} = P$
str2	$P + Q = Q + P$
str3	$P + (Q + R) = (P + Q) + R$
str4	$(\nu x)(\nu y)P = (\nu y)(\nu x)P$
str5	$x \notin \texttt{fn}(P) \to (\nu x)P = P$
congr1	If $P = Q$ then $\overline{a}u\,.\,P = \overline{a}u\,.\,Q$
	$\tau\,.\,P = \tau\,.\,Q$
	$P + R = Q + R$
	$P \vert R = Q \vert R$
	$R \vert P = R \vert Q$
	$(\nu x)P = (\nu x)Q$
congr2	If $P\{y/x\} = Q\{y/x\}$ for all $y \in \texttt{fn}(P, Q, x)$ then $a(x)\,.\,P = a(x)\,.\,Q$

idem	$P + P$	$=$	P	
m1	$[x = x]P$	$=$	P	
m2	$[x = y]P$	$=$	$\mathbf{0}$	if $x \neq y$
mm1	$[x \neq x]P$	$=$	$\mathbf{0}$	
mm2	$[x \neq y]P$	$=$	P	if $x \neq y$
r1	$(\nu x)\alpha\,.\,P$	$=$	$\alpha\,.\,(\nu x)P$	if $x \notin \alpha$
r2	$(\nu x)\alpha\,.\,P$	$=$	$\mathbf{0}$	if x is the subject of α
r3	$(\nu x)(P + Q)$	$=$	$(\nu x)P + (\nu x)Q$	

Table 3
Axioms for strong late bisimilarity, except for Parallel.

bound object x. Then from $P \sim Q$ and $P \xrightarrow{a(x)} P'$ we get $Q \xrightarrow{a(x)} Q'$ such that $P'\{u/x\} \sim Q'\{u/x\}$ for all u. By induction they are also provably equal for all u. So $a(x)\,.\,Q'$ is a summand of Q and from **congr2** we get that it is provably equal to $a(x)\,.\,P'$. The other cases are similar and simpler. So, each summand of P is provably equivalent to a summand of Q and therefore, by the laws **idem** (and **str**), P is provably equivalent to Q.

The original proof in [8] uses the same idea and is just a little bit more detailed (writing out "the other cases" rather then saying they are "similar and simpler").

exp
Let $P = \sum_i \alpha_i \,.\, P_i$ and $Q = \sum_j \beta_j \,.\, Q_j$ where $\mathtt{bn}(\alpha_i) \cap \mathtt{fn}(Q) = \emptyset$ and $\mathtt{bn}(\beta_j) \cap \mathtt{fn}(P) = \emptyset$ for all i, j. Then

$$P|Q \quad = \quad \sum_i \alpha_i \,.\, (P_i|Q) \; + \; \sum_j \beta_j \,.\, (P|Q_j) \; + \sum_{\alpha_i \; comp \; \beta_j} \tau \,.\, R_{ij}$$

where the relation $\alpha_i comp \beta_j$ and R_{ij} are defined through the follwing four cases:

(i) $\alpha_i = a(x)$ and $\beta_j = \overline{a}u$ in which case $R_{ij} = P_i\{u/x\}|Q_j$,

(ii) $\alpha_i = a(x)$ and $\beta_j = (\nu u)\overline{a}u$ in which case $R_{ij} = (\nu u)(P_i\{u/x\}|Q_j)$,

(iii) The converse of 1,

(iv) The converse of 2.

Table 4
The traditional expansion law for strong bisimilarity. Here α_i and β_j range over the prefix forms (Input, Output and Silent) and also over the combination of bound output prefix $(\nu u)\overline{a}u$.

Both get the definition of depth wrong. In the quote above, "maximal nesting" is not correct if the agent is a parallel composition, and in [8] the definition only applies to head normal forms and is later used for all agents.

3 The implementation in Isabelle: Soundness

In our earlier work [3] we have implemented the semantics and various bisimulation definitions in Isabelle, and we refer the reader to that paper for details on the representation of the SOS semantics, associated induction rules, and the co-inductive definition of bisimulation. Our results include formal proofs of substitutivity properties such as $P \sim R \to P|Q \sim R|Q$, and many algebraic laws.

Our implementation uses the nominal datatype package and treats bound names efficiently, so that we almost never have to worry about alpha variants of agents or actions.

The only exceptions are when bound names are explicitly mentioned and arguments about their identities are part of the proofs. An example is in the algebraic law $(\nu x)(\nu y)P = (\nu y)(\nu x)P$, where the proof will split in one case for $x = y$ and another case for $x \neq y$. In all other situations the implementation will guarantee that bound names are always suitably fresh, thus automatically formalizing the assumption that manual proofs often make.

We will begin by showing soundness and completeness for the sub-calculus that only contains Nil, Prefix, Sum, Match and Mismatch. We will then expand our proof to encompass also Restriction and Parallel.

To prove soundness of the axiomatization, we must prove that all provably equivalent pairs of processes are also bisimilar. In [3] we proved that strong bisimulation is preserved by all operators except input-prefix and that all structurally congruent terms are also bisimilar. This meant that the proofs for **str** and **congr1** from Table 3 are already complete. The soundness proofs for the rest of the axioms are trivial, with the exception of **congr2**.

Lemma 3.1 *If $P\{y/x\} \sim Q\{y/x\}$ for all $y \in fn(P, Q, x)$ then $a(x).P \sim a(x).Q$*

Proof By definition of bisimulation and case analysis for the cases where x and y clash with the introduced name u. $\square$

We can then prove our theorem:

Theorem 3.2 *If $P \equiv Q$ then $P \sim Q$*

4 Completeness

At the core of the completeness proofs are *head normal forms*, or *hnf* for short. A term is in head normal form if it is a sum of prefixed processes. In Isabelle, we use the following function to determine whether or not a term is in *hnf*.

Definition 4.1 Definition of *hnf*.

$$\begin{aligned}
&\texttt{hnf}(\mathbf{0}) &&= \texttt{True}\\
&\texttt{hnf}(\alpha.P) &&= \texttt{True}\\
&\texttt{hnf}(P+Q) &&= \texttt{hnf}(P) \wedge \texttt{hnf}(Q) \wedge P \neq \mathbf{0} \wedge Q \neq \mathbf{0}\\
&\texttt{hnf}\ \_ &&= \texttt{False}
\end{aligned}$$

We will reason about hnfs in terms of their *summands*. The summand of a term is the set of its prefixed subterms that are composed by the +-operator.

Definition 4.2 Definition of *summands*

$$\begin{aligned}
&\texttt{summands}(\alpha.P) &&= \{\alpha.P\}\\
&\texttt{summands}(P+Q) &&= \texttt{summands}(P) \cup \texttt{summands}(Q)\\
&\texttt{summands}\ \_ &&= \{\}
\end{aligned}$$

In the intuitive version of the completeness proof (Proposition 8 cited above), there is an appeal to the law **idem** to remove duplicate instances of provably equivalent summands. In order to reason formally about this we would like to let the summands be sets of equivalence classes (corresponding to $\equiv$) of terms. In Isabelle it turns out to be more efficient to implement this by the notion of a *unique hnf*, or *uhnf* where only one representative of each equivalence class is kept in the set.

Definition 4.3 Definition of *uhnf*
$\texttt{uhnf}(P) = \texttt{hnf}(P) \wedge \forall P'\ P'' \in \texttt{summands}(P).\ P' \neq P'' \longrightarrow \neg(P' \equiv P'')$

Our main proof will work by induction over the depth of the terms that are being proved equal. Intuitively, the depth of a term is an upper bound of the number of transitions it can make. We define the following function:

Definition 4.4 Depth of a term

$$\begin{aligned}
\texttt{depth}(\mathbf{0}) &= 0\\
\texttt{depth}(\alpha.P) &= 1 + \texttt{depth}(P)\\
\texttt{depth}([a=b]P) &= \texttt{depth}([a \neq b]P) = \texttt{depth}((\nu x)P) = \texttt{depth}(P)\\
\texttt{depth}(P+Q) &= \texttt{max}(\texttt{depth}(P),\ \texttt{depth}(\texttt{Q}))\\
\texttt{depth}(P \mid Q) &= \texttt{depth}(P) + \texttt{depth}(\texttt{Q})
\end{aligned}$$

For the rest of the completeness proof, we will mostly work with *uhnf*s. We therefore need to be able to show that every term can be rewritten as a provably equal *uhnf*. We must also make sure that the generated uhnfs have no larger depth than the original terms as this would otherwise break our induction on the depth in the main proof. In the following we present the sequence of lemmas needed for Isabelle to complete the proof, together with indications of the involved proof strategies.

First, we will need the following auxiliary lemmas:

Lemma 4.5 *If $Q \in \texttt{summands}(P)$ and $Q \equiv Q'$ then $P + Q' \equiv P$.*

Proof By induction on the structure of P. □

Lemma 4.6 *If $\texttt{uhnf}(P)$ and $\texttt{uhnf}(Q)$ then there exists an R such that $\texttt{uhnf}(R)$, $P + Q \equiv R$ and $\texttt{depth}(R) \leq \texttt{depth}(P+Q)$.*

Proof By induction on the structure of P. In the base case, Lemma 4.5 is used to filter out the terms which are provably equal to some term in the $\texttt{summands}(Q)$.
Base case: P is of the form $\alpha.P'$. If $Q = \mathbf{0}$ then set R to P, otherwise if there is a $Q' \in \texttt{summands}(Q)$ s.t. $Q' \equiv P$ then $Q + P \equiv Q$ by Lemma 4.5 so set R to Q, otherwise set R to $P + Q$.
Inductive step: Since $\texttt{uhnf}(P)$, only the case where P is of the form $P' + P''$ applies:
We have that $\texttt{uhnf}(P' + P'')$ hence $\texttt{uhnf}(P')$ and $\texttt{uhnf}(P'')$. From IH twice we obtain a Q' where $\texttt{uhnf}(Q')$, $P'' + Q \equiv Q'$ and $\texttt{depth}(Q') \leq \texttt{depth}(P'' + Q)$. □

With these lemmas in place we can prove:

Lemma 4.7 *For every P, there exists a Q s.t. $\texttt{uhnf}(Q)$, $P \equiv Q$ and $\texttt{depth}(Q) \leq \texttt{depth}(P)$.*

Proof By induction on the structure of P. In the case where P is of the form $P' + P''$, Lemma 4.6 is used. □

In order to prove the correspondence between bisimilarity and the axioms, we need to form a link to the transition system. We find such a link in the summands.

Lemma 4.8 *If $\texttt{hnf}(P)$ then $P \xrightarrow{\alpha} P'$ iff $\alpha.P' \in \texttt{summands}(P)$.*

Proof By induction on the structure of P. □

The intuitive way to express equality using summands is that if there is a bijection f between the summands of two agents such that $f(P) \equiv P$ for all P, then the agents are provably equal. Clearly the way to prove this must be through an induction over the summands, and there we shall need the following auxiliary lemma:

Lemma 4.9 *If* $\texttt{uhnf}(Q)$ *and* $P \in \texttt{summands}(Q)$ *then there exists a* Q' *s.t.* $P + Q' \equiv Q$ *and* $\texttt{summands}(Q') = \texttt{summands}(Q) - \{P\} \wedge \texttt{uhnf}(Q')$.

Proof By induction on the structure of P. **str2** and **str3** are used to pull P to the top level of Q. The intuition is that since the $+$-operator is commutative and associative, one can always pull any summand to the front of the term. Since the term is on *uhnf*, we know that no other provably equal sub terms exist and hence the summands of the remainder will be the summands of the original term with the pulled term removed. □

We can now prove the following lemma:

Lemma 4.10 *If* $\texttt{uhnf}(P)$ *and* $\texttt{uhnf}(Q)$ *and there exists a bijection* $f : \texttt{summands}(P) \rightarrow \texttt{summands}(Q)$ *s.t.* $P \equiv f(P)$ *then* $P \equiv Q$.

Proof By induction on $\texttt{summands}(P)$.
Base case: $\texttt{summands}(P) = \{\}$
Since f is a bijection, $\texttt{summands}(Q) = \{\}$ and since $\texttt{uhnf}(P)$ and $\texttt{uhnf}(Q)$, $P = \mathbf{0}$ and $Q = \mathbf{0}$. Hence $P \equiv Q$ by reflexivity of $\equiv$.
Inductive step: $\texttt{summands}(P) = \{P'\} \cup S$
From Lemma 4.9 we obtain a P'' where $P' + P'' \equiv P$, $\texttt{summands}(P'') = \texttt{summands}(P) - \{P'\}$ and $\texttt{uhnf}(P'')$. Since $\texttt{uhnf}(P'')$, $S = \texttt{summands}(P)$. Since $f(P') \in \texttt{summands}(Q)$ we obtain a Q'' where $f(P') + Q'' \equiv Q$, $\texttt{summands}(Q'') = \texttt{summands}(Q) - \{f(P')\}$ and $\texttt{uhnf}(Q'')$. Since f is a bijection and $\texttt{uhnf}(P'')$ and $\texttt{uhnf}(Q'')$ there exists a bijective function $f' : \texttt{summands}(P') \rightarrow \texttt{summands}(Q'')$, then by IH $P'' = Q''$. Hence $P \equiv P' + P'' \equiv f(P'') + Q'' \equiv Q$. □

In order to do induction we need to know that processes have greater depth than their derivatives.

Lemma 4.11 *If* $\texttt{uhnf}(P)$ *and* $P \xrightarrow{\alpha} P'$ *then* $\texttt{depth}(P') < \texttt{depth}(P)$.

Proof By induction on the proof of $P \xrightarrow{\alpha} P'$. □

We can now prove completeness.

Theorem 4.12 *If* $P \sim Q$ *then* $P \equiv Q$

Proof From Lemma 4.7 we can first convert P and Q to provably equal $\texttt{uhnf}$s. The proof is then done by induction over $\texttt{depth}(P) + \texttt{depth}(Q)$. The idea is to exhibit a bijection f satisfying the premises of Lemma 4.10.
Base case: $\texttt{depth}(P) + \texttt{depth}(Q) = 0$
The only case where this can hold is when $P = \mathbf{0}$ and $Q = \mathbf{0}$. Hence $P \equiv Q$ by reflexivity of $\equiv$.
Inductive step: $\texttt{depth}(P) + \texttt{depth}(Q) \leq n$ (where n is a natural number).
Pick an arbitrary $\alpha.P'$ from $\texttt{summands}(P)$. We need to find $f(\alpha.P')$, i.e. a Q' s.t. $\alpha.P' \equiv \alpha.Q'$.
If α is a free action, by Lemma 4.8 we have $P \xrightarrow{\alpha} P'$. Since $P \sim Q$, we know

that there is a Q' s.t. $Q \xrightarrow{\alpha} Q'$ and $P' \sim Q'$. From Lemma 4.7 we get a P'' where $\texttt{uhnf}(P'')$, $P' \equiv P''$ and $\texttt{depth}(P'') \leq \texttt{depth}(P')$ and a Q'' where $\texttt{uhnf}(Q'')$, $Q' \equiv Q''$ and $\texttt{depth}(Q'') \leq \texttt{depth}(Q')$. By transitivity and symmetry of $\sim$ and Theorem 3.2 we get that $P'' \sim Q''$ and thus by the fact that $\texttt{depth}(P'')+\texttt{depth}(Q'') < n$ (Lemma 4.11) and IH we get that $P'' \equiv Q''$ and by transitivity and symetry of $\equiv$, $P' \equiv Q'$ and hence $\alpha.P' \equiv \alpha.Q'$ by **congr1**.
If α is an input action of the form $a(x)$ we prove that $P'\{y/x\} \equiv Q'\{y/x\}$ for all $y \in \texttt{fn}(P,\ Q,\ x)$. The strategy is the same as for the free actions and we get that $P'\{u/x\} \sim Q'\{u/x\}$ for all u, more specifically $u = y$, hence $P'\{y/x\} \equiv Q'\{y/x\}$ and hence $a(x).P' \equiv a(x).Q'$ by **congr2**. □

5 Restriction

We add restriction to the calculus and the three laws **r1–r3**. These laws are first proved sound in the expected way; the most tedious is **r1** which requires a case analysis on α.

For our completeness proof we need to expand our definitions of *summands* and *hnf*. When restriction is added to the calculus, we have the possibility of bound outputs. A process which generates a bound output is of form $(\nu x)\bar{a}x.P$ where $x \neq a$. We extend our definition of *hnf* with the following case:

$$\texttt{hnf}((\nu x)P) = \exists a\ P'.\ a \neq x \wedge P = \bar{a}x.P'$$

and our definition of *summands* with:

$$\begin{aligned}\texttt{summands}((\nu x)P) = {} & \texttt{if } (\exists a\ P'.\ a \neq x \wedge P = \bar{a}x.P') \texttt{ then} \\ & \quad \{(\nu x)P\} \\ & \texttt{else} \\ & \quad \{\}\end{aligned}$$

We also have to expand our proofs that state that for every term there exists a provably equal *uhnf*. As with the +-operator we need an auxiliary lemma.

Lemma 5.1 *If* $\texttt{uhnf}(P)$ *then there exists a* Q *such that* $\texttt{uhnf}(Q)$, $(\nu x)P \equiv Q$ *and* $\texttt{depth}(Q) \leq \texttt{depth}((\nu x)Q)$.

Proof By induction on the structure of P. In the base case a case analysis is done to check if the restricted name collides with any names in the prefix. □

The extension to Lemma 4.7 now becomes quite simple, but in the restriction case, a second induction over P is required in order to prove how restriction behaves when paired with each of the other operators.

We will also need an expanded version of Lemma 4.8 to create our link between summands and transitions.

Lemma 5.2 *If* $\texttt{uhnf}(P)$ *and* $a \neq x$ *then* $P \xrightarrow{a\nu x} P'$ *iff* $(\nu x)\bar{a}x.P' \in \texttt{summands}(P)$.

Proof By induction on the structure of P. □

Theorem 4.12 has to be extended to take care of bound outputs. The summands will have the form $(\nu x)\bar{a}x.P$ and we find derivatives P' and Q' s.t. $P' \sim Q'$, hence $\bar{a}x.P' \equiv \bar{a}x.Q'$ and $(\nu x)\bar{a}b.P' \equiv (\nu x)\bar{a}b.Q'$ by **congr1**.

6 Parallel

We complete our proof by adding the parallel operator. To do this we must encode the expansion law in Isabelle. Unfortunately, the law as presented here and elsewhere is not completely formally correct. The reason is that it makes use of a function Σ which takes a set of agents as a parameter and returns the sum of them. However, such a function cannot be defined in the usual inductive way (as $\Sigma(\{P\} \cup S) = P + \Sigma S$) since this does not define Σ uniquely (elements can be pulled from a set in different order, resulting in different order of the summands). The reason the expansion law nevertheless is considered valid is that it implicitly operates on equivalence classes of agents up to $\equiv$ and here the sum operator is idempotent, associative and commutative. As there is no easy way to incorporate such functions in Isabelle we need to be a little more creative. We define a relation $\mathcal{S}$ on `agent` $\times$ `agent set` with the intuition that if F is a set of agents then $\mathcal{S}(P, F)$ if P can be obtained as a sum of the agents in F.

Definition 6.1 The relation $\mathcal{S}$.

$$(\mathbf{0},\ \{\}) \in \mathcal{S}$$

$$(P,\ \{P\}) \in \mathcal{S}$$

$$Q \in F \wedge (P,\ F - \{Q\}) \in \mathcal{S} \Longrightarrow (P + Q,\ F) \in \mathcal{S}$$

We must now create the set of all terms generated by the expansion law (Table 4).

Definition 6.2 The function `expand` which generates a set of all terms created by the expansion law. Symmetric versions have been elided. Here $\alpha.P$ ranges also over object restrictions of prefixes of kind $(\nu x)\bar{a}x.P$.

$$\texttt{expand}(P, Q) =$$

$$\{\alpha.(P' \mid Q)\ :\ \alpha.P' \in \texttt{summands}(P) \wedge \texttt{bn}(\alpha) \cap \texttt{fn}(Q) = \emptyset\} \cup$$

$$\{\tau.(P'\{y/x\} \mid Q')\ :\ a(x).P' \in \texttt{summands}(P)\ \wedge \bar{a}b.Q' \in \texttt{summands}(Q)\} \cup$$

$$\{\tau.(\nu y)(P'\{y/x\} \mid Q')\ :\ a(x).P' \in \texttt{summands}(P) \wedge y \notin \texttt{fn}(P) \wedge (\nu y)\bar{a}y.Q' \in \texttt{summands}(Q)\}$$

We can now add the expansion law to our axioms in the following way:

Definition 6.3 The expansion law.
$$\texttt{hnf}(P) \wedge \texttt{hnf}(Q) \wedge (R,\ \texttt{expand}(P, Q)) \in \mathcal{S} \Longrightarrow P \mid Q \equiv R$$

6.1 Soundness

To prove soundness of Def. 6.3 we will need the following auxilliary lemmas.

Lemma 6.4 *If $(P,\ F) \in \mathcal{S}$, $Q \in F$ and $Q \xrightarrow{\alpha} Q'$ then $P \xrightarrow{\alpha} Q'$.*

Proof By rule induction on the construction of $\mathcal{S}$. □

Lemma 6.5 *If $(R,\ F) \in \mathcal{S}$ and $R \xrightarrow{\alpha} R'$ then there is a P in F s.t. $P \xrightarrow{\alpha} R'$.*

Proof By rule induction on the construction of $\mathcal{S}$. In the inductive step a case analysis is made on whether or not the inserted term can do the desired transition. If so, that term is picked, otherwise the term is obtained through the induction hypothesis. □

Lemma 6.6 *If $\texttt{hnf}(P)$, $\texttt{hnf}(Q)$ and $(R,\ \texttt{expand}(P,Q)) \in \mathcal{S}$ then $P \mid Q \xrightarrow{\alpha} P'$ iff $R \xrightarrow{\alpha} P'$.*

Proof In this proof Lemma 4.8 is used to translate summands to transitions and vice versa.

$\Longrightarrow$ By case analysis of $P \mid Q \xrightarrow{\alpha} P'$. Each case matches one construction in the expansion law and lemma 6.4 is used to prove that R can make the desired transition.

$\Longleftarrow$ Lemma 6.5 is used to find the summand in R in $\texttt{expand}(P,Q)$ which can make the transition. A case analysis of R is made and each case is matched to its corresponding rule in the operational semantics.

□

We can now prove soundness.

Lemma 6.7 $\texttt{hnf}(P) \wedge \texttt{hnf}(Q) \wedge (R,\ \texttt{expand}(P,Q)) \in \mathcal{S} \Longrightarrow P \mid Q \sim R$.

Proof Follows trivially from the definition of $\sim$ and Lemma 6.6. □

6.2 Completeness

All we need to do to show completeness is to prove that every expanded term has a provably equal `uhnf` of no lesser depth. We will need the following auxiliary lemmas.

Lemma 6.8 *If $(P,\ F) \in \mathcal{S}$ and for all Q in F, $\texttt{uhnf}(Q)$ then there exists a P' s.t. $\texttt{uhnf}(P')$, $P \equiv P'$ and $\texttt{depth}(P') \leq \texttt{depth}(P)$.*

Proof By rule induction over the construction of $\mathcal{S}$. □

Lemma 6.9 *If $\texttt{uhnf}(P)$ and $\texttt{uhnf}(Q)$ then there exists an R s.t. $(R,\ \texttt{expand}(P,Q)) \in \mathcal{S}$ and $\texttt{depth}(R) \leq \texttt{depth}(P \mid Q)$.*

Proof By case analysis of the construction of $\texttt{expand}(P,Q)$. □

We finally need to modify Lemma 4.7. In the parallel case, we use the parallel congruence laws from **congr1** and Lemma 6.9 to find an expanded R of no greater depth than $\texttt{depth}(P \mid Q)$. Lemma 6.8 can then be used to convert R to a provably equal `uhnf`.

In the restriction case one also has to take care of the situation when a restriction is done on a parallel composition. This is done by first applying the expansion law to the term and then Lemma 5.1.

7 Conclusion

We have presented a machine checked proof of the complete axiomatization of strong late bisimilarity in the pi-calculus. It is difficult to estimate the time this has required since it was done in parallel with the development of the infrastructure in Isabelle. As an example, the effort corresponding to Section 6 (adding the parallel operator) turned out to be roughly one day, most of which was spent proving soundness. Our experience is that the Isabelle system with its nominal datatype package and our implementation of the pi-calculus [3] works well for this task, in that the treatment of bound names is facilitated. This is not really apparent from the presentation in this paper since the gain is in many low-level details. Ultimately such judgements must be subjective, and we refer to our previous paper for a comparison with other approaches. But our present result strengthens our claim: similar completeness proofs have been around for more than 20 years, and no such proof has been formalized inside a theorem prover before now.

The Isabelle code can be obtained from our homepage
`http://www.it.uu.se/research/group/mobility/theorem`

There are several obvious continuations of this work. Strong late bisimilarity is but one of a large family of equivalences. There is the early variety and the weak varieties where the completeness proofs are more complex and rely on saturation of terms with summands, and there are the corresponding congruences where the hnfs are different since matching and mismatching cannot be reduced. The saturation technique could be adapted also to the proof in the present paper, with the idea to establish first that if $P \sim Q$ then $Q \equiv Q + P$ by adding each of the summands of P to Q; this would obviate the need for uhnfs but it is unclear if it would give a shorter or more striking proof.

References

[1] `http://isabelle.in.tum.de/nominal/`

[2] `http://fling-l.seas.upenn.edu/~plclub/cgi-bin/poplmark`

[3] Jesper Bengtson and Joachim Parrow. Formalising the pi-calculus using nominal logic. In *Proceedings of FOSSACS 2007*, volume 4423 of *Lecture Notes in Computer Science*, pages 63–77. Springer Verlag, 2007.

[4] M. J. Gabbay. The π-calculus in FM. In Fairouz Kamareddine, editor, *Thirty-five years of Automath*. Kluwer, 2003.

[5] Matthew Hennessy and Robin Milner. Algebraic laws for nondeterminism and concurrency. *J. ACM*, 32(1):137–161, 1985.

[6] Daniel 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.

[7] Furio Honsell, Marino Miculan, and Ivan Scagnetto. π-calculus in (co)inductive type theory. *Theoretical Computer Science*, 253(2):239–285, 2001.

[8] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, I/II. *Inf. Comput.*, 100(1):1–77, 1992.

[9] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL: a proof assistant for higher-order logic*. Springer-Verlag, 2002.

[10] Joachim Parrow. An introduction to the pi-calculus. In Jan Bergstra et al, editor, *Handbook of Process Algebra*, pages 479–543. Elsevier, 2001.

[11] Christine Röckl and Daniel Hirschkoff. A fully adequate shallow embedding of the π-calculus in Isabelle/HOL with mechanized syntax analysis. *J. Funct. Program.*, 13(2):415–451, 2003.

[12] Natarajan Shankar. *Metamathematics, Machines and Gödel's Proof.* Cambridge University Press, 1994.