Barbed Similarity for the π-Calculus in Beluga:

A Case Study in Coinductive Reasoning

Lea Trogni · Università degli Studi di Milano, Dipartimento di Matematica

Joint work with
Gabriele Cecilia · Augusta University, School of Computer and Cyber Sciences
Alberto Momigliano · Università degli Studi di Milano, Dipartimento di Informatica

Roadmap

Part 1

Introduction

Part 2

π-Calculus

Syntax, semantics & structural congruence

Part 3

Behavioural Equivalences

Barbed similarity & precongruence

Part 4

Context Lemma

Proof strategy & key steps

Part 5

Conclusions

Results & future work

PART 1

Introduction

Concurrent Calculi and Their Formalization

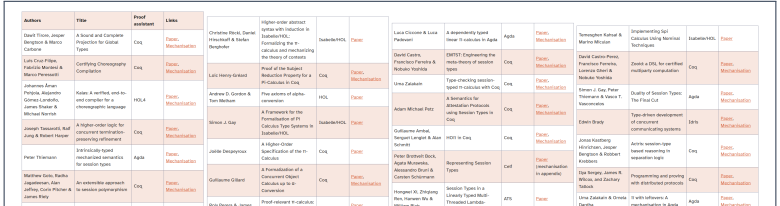
Concurrent Calculi: abstract models of concurrent systems

They have been studied and formalized extensively over the past decades:

Author, Year	Publication	Technique
Nesi 94	*A Formalization of the Process Algebra CCS in HOL*	Named syntax
Melham 94	*A Mechanized Theory of the π-Calculus in HOL*	Named syntax
Hirschkoff 97	*A Full Formalisation of π-Calculus Theory in the Calculus of Constructions*	De Bruijn indices
Miller et al. 99	*Foundational Aspects of Syntax*	HOAS
Honsell et al. 01	*π-Calculus in (Co)Inductive Type Theory*	HOAS
Bengtson et al. 09	*Formalizing the π-Calculus using Nominal Logic*	Nominal logic

Concurrent Calculi and Their Formalization

Today, we have dozens of formalizations:



Concurrent Calculi Formalization Benchmark (2024)

Three benchmark problems to assess the state of the art and improve the mechanizations of concurrent calculi

Each problem addresses one common challenge occurring when formalizing them:

- Linearity & behavioural type systems
- Name passing & scope extrusion
- **Coinduction & infinite behaviour**

Third challenge: formalize a **context lemma** for barbed congruence in the π-calculus

The Context Lemma, in a Nutshell

Contextual equivalences and preorders are obtained by requiring a behavioural relation to hold in every context.
the equivalence is required to hold for **all** contexts $\Rightarrow$ often difficult to handle

A context lemma replaces quantification over arbitrary contexts with a simpler criterion

CCFB3 — To obtain barbed congruence in the π-calculus, it is sufficient to prove closure under:

- parallel composition
- substitutions

The Proof Assistant Beluga

We formalized the context lemma in **Beluga**

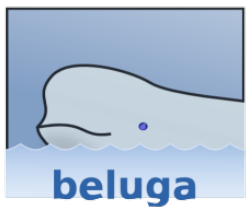
⇒ Developed by Brigitte Pientka at McGill University

⇒ Proofs are (co)recursive programs (Curry-Howard isomorphism)

⇒ Inductive proofs are total, coinductive ones are productive

⇒ Two-level system:

1. Representation level: (open) terms in Contextual LF
2. Computation level: contextual objects, (co)inductive predicates, theorems



Why Beluga?

Main features used in this work:

- HOAS (Higher-Order Abstract Syntax) to represent binders
- Built-in simultaneous substitutions
- Support for (co)inductive proofs via (co)pattern matching

Moreover, it complements existing Beluga solutions [1,2] of the first two problems

[1]: D. Zachon, C. Sano, A. Momigliano & B. Pientka (2025): *Split Decisions: Explicit Contexts for Substructural Languages*

[2]: G. Cecilia & A. Momigliano (2024): *A Beluga Formalization of the Harmony Lemma in the π-Calculus*

PART 2

π-Calculus

Syntax, semantics & structural congruence

Syntax

Processes: $P \coloneqq 0 \mid x(z).P \mid \bar{x}y.P \mid (P \mid P') \mid \nu z\, P \mid !P$

Key aspects:

- Name-passing communications
- Replication, yielding infinite behaviour

Syntax

Processes: $P \coloneqq 0 \mid x(z).P \mid \bar{x}y.P \mid (P \mid P') \mid \nu z\, P \mid !P$

Key aspects:

- Name-passing communications
- Replication, yielding infinite behaviour

Inputs and restrictions bind the name z in $P \Rightarrow$ formalized using HOAS:

```
LF proc: type =
  | p_zero: proc
  | p_in:  names → (names → proc) → proc
  ...
;
```

Semantics

Operational behaviour is described by the LTS for late semantics.

$$\underline{\text{Actions:}} \qquad \alpha ::= \tau \mid \bar{x}y \mid x(z) \mid \bar{x}(z)$$

Transition rules for replication:

$$\text{S-REP}\ \frac{P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P' \mid !P} \qquad \text{S-REP-COM}\ \frac{P \xrightarrow{\bar{x}y} P' \qquad P \xrightarrow{x(z)} P''}{!P \xrightarrow{\tau} (P' \mid P''\{y/z\}) \mid !P} \qquad \text{S-REP-CLOSE}\ \frac{P \xrightarrow{\bar{x}(z)} P' \qquad P \xrightarrow{x(z)} P''}{!P \xrightarrow{\tau} ((\nu z)(P' \mid P'')) \mid !P}$$

Semantics

> The formalization of the semantics follows previous Beluga developments [2] based on early work by Honsell et al. and Miller et al.

Some actions bind names $\Rightarrow$ two types `f_act`, `b_act`

LTS rules encoded via two mutually defined judgment families

```
LF fstep: proc → f_act → proc → type
LF bstep: proc → b_act → (names → proc) → type
```

- Side conditions involving free and bound names are not required.

Contexts & HOAS

A **context** is a process with a hole (for example, $C = x(z).[\,] \mid Q$).
Congruences are defined as relations that are **contextually closed**.

> Contexts may capture free names $\Rightarrow$ they cannot be defined directly as a datatype in a HOAS encoding.

Key idea (following Lancelot, Accattoli and Vemclefs [3])

Instead of using a context datatype, define contextual closure via the predicate

$$(P, P', Q, Q') \quad \text{if and only if} \quad P' = C[P],\ Q' = C[Q] \quad \text{for some context } C$$

[3]: A. Lancelot, B. Accattoli & M. Vemclefs (2025): *Barendregt's Theory of the λ-Calculus, Refreshed and Formalized*

Encoding of Contexts & Structural Congruence

Some free variables occurring in P and Q can become bound in $C[P]$ and $C[Q]$

- Their Beluga contexts are different and need to be made explicit
- Their relation cannot be formalized as an LF predicate; instead we define an *inductive* one that relates four contextual objects

```
inductive PctxM: (g: ctx) (g': ctx) [g' ⊢ proc] → [g ⊢ proc]
  → [g' ⊢ proc] → [g ⊢ proc] → ctype
```

We provide two encodings of structural congruence:

`inductive CongM` based on contextual closure

`LF cong` based on compatibility

PART 3

Behavioural Equivalences

Barbed similarity & precongruence

Observables & Barbs

Behavioural equivalences are based on **observable behaviour**.

Barbs

The ability to immediately perform an input or output on a specific channel:

$$P \downarrow_x \qquad P \downarrow_{\bar{x}}$$

Internal actions

Whenever $P \mathcal{R} Q$, every internal action of P must be matched by Q.

Remark. Since symmetry is never used in the proof of the Context Lemma, we work with **preorders** rather than equivalences.

Barbed Similarity: Definition & Encoding

A relation $\mathcal{R}$ is a **strong barbed simulation** if $P \mathcal{R} Q$ implies:

1. $P \downarrow_x \Rightarrow Q \downarrow_x$
2. $P \downarrow_{\bar{x}} \Rightarrow Q \downarrow_{\bar{x}}$
3. $P \xrightarrow{\tau} P' \Rightarrow \exists Q' .\ Q \xrightarrow{\tau} Q' \wedge P' \mathcal{R} Q'$

Strong barbed similarity, denoted by $\dot{\leq}_B$, is the union of all strong barbed simulations.

> Strong barbed similarity can be defined **coinductively**, since it is the largest strong barbed simulation.

Barbed Similarity: Definition & Encoding

In Beluga, a coinductive *n*-ary relation is defined via its **observations**, with the following syntax:

```
| (<Obs_name> : <Rel_name> <arg_1> ... <arg_n>) :: <observation>
```

Barbed Similarity: Definition & Encoding

In Beluga, a coinductive *n*-ary relation is defined via its **observations**, with the following syntax:

```
| (<Obs_name> : <Rel_name> <arg_1> ... <arg_n>) :: <observation>
```

Strong barbed similarity is mutually defined with an auxiliary inductive type, expressing the existential quantifier in its definition:

```
coinductive BarbSim: (g:ctx) [g ⊢ proc] → [g ⊢ proc] → ctype =
  ...
  | (BarbSim_tau: BarbSim [g ⊢ P] [g ⊢ Q])
    :: [g ⊢ fstep P f_tau P'] → Ex_sim_barb [g ⊢ P] [g ⊢ Q] [g ⊢ P']

and inductive Ex_sim_barb: ... =
  | Ex_sim_barb_def: [g ⊢ fstep Q f_tau Q'] → BarbSim [g ⊢ P'] [g ⊢ Q']
    → Ex_sim_barb [g ⊢ P] [g ⊢ Q] [g ⊢ P']
```

Barbed Precongruence

Barbed similarity is not closed under all contexts. Thus, we define the induced precongruence.

Two processes P and Q are **barbed precongruent**, denoted by $P \leq_B Q$, iff for every context C:

$$C[P] \dot{\leq}_B C[Q].$$

Using the quaternary relation `PctxM`:

```
inductive BarbPre: (g':ctx) [g' ⊢ proc] → [g' ⊢ proc] → ctype =
  | BC: {P:[g' ⊢ proc]} {Q:[g' ⊢ proc]}
        ((g:ctx) {CP:[g ⊢ proc]} {CQ:[g ⊢ proc]}
          pctxM [g' ⊢ P] [g ⊢ CP] [g' ⊢ Q] [g ⊢ CQ]
              → BarbSim [g ⊢ CP] [g ⊢ CQ])
                  → BarbPre [g' ⊢ P] [g' ⊢ Q]
```

PART 4

Context Lemma

Proof strategy & key steps

Statement of the Context Lemma

Context Lemma (sufficient condition).
If $P\sigma \mid R \dot{\leq}_B Q\sigma \mid R$ for every substitution σ and process R, then $C[P] \dot{\leq}_B C[Q]$ for every context C.

The proof combines:

- Inductive and coinductive reasoning;
- Strengthening lemmas for substitutions;
- The inclusion of structural congruence in barbed similarity;
- Up-to techniques.

Encoding Substitutions

The hypothesis on the pair (P, Q)

$$\forall R, \sigma : \quad P\sigma \mid R \dot{\leq}_B Q\sigma \mid R$$

is encoded as `BarbPre'` using Beluga's **built-in simultaneous substitutions**:

```
inductive BarbPre': (g:ctx) [g ⊢ proc] → [g ⊢ proc] → ctype =
    | BC' : ((h:ctx) {$S : $[h ⊢ g]} {R : [h ⊢ proc]}
      → BarbSim [h ⊢ P[$S] p_par R] [h ⊢ Q[$S] p_par R])
      → BarbPre' [g ⊢ P] [g ⊢ Q]
```

- Substitutions are contextual objects, whose names start with `$`
- The **type** of a substitution is determined by its codomain and domain contexts
- Substitutions can be **weakened** and **composed**

Proof Roadmap

Properties of $\dot{\leq}_B$

- Contains all simulations up to $\dot{\leq}_B$
- Is a **preorder**
- Preserved by **prefixes** and **restriction**
- Contains structural congruence: $\equiv \subseteq \dot{\leq}_B$

Properties of $\leq_B$

- Is a **barbed simulation**
- Is a **preorder**
- Is a **precongruence**
- Is the **largest precongruence** $\subseteq \dot{\leq}_B$

Proof Strategy: Key Steps

Goal: `BarbPre'` $\subseteq$ `BarbPre`

Steps:

1. `BarbPre'` is included in barbed similarity.
   Take the identity substitution $\sigma = \mathrm{id}$, then $P \equiv P\sigma \mid 0 \dot{\leq}_B Q\sigma \mid 0 \equiv Q$.
2. `BarbPre'` is a precongruence. By induction on the structure of the given context:
   - **Prefixes:** defining auxiliary relations and showing that they are simulations.
   - **Parallel/restriction:** using $\equiv \subseteq \dot{\leq}_B$.
   - **Replication:** defining an auxiliary relation and showing that it is a simulation up to $\dot{\leq}_B$.
3. Since `BarbPre` is the **largest precongruence** included in barbed similarity, we conclude.

Sample Proof: Compatibility with Restriction

```
rec res_barb_sim : (g:ctx) BarbSim [g, x : names ⊢ P] [g, x : names ⊢ Q]
  → BarbSim [g ⊢ p_res (\x. P)] [g ⊢ p_res (\x. Q)] =
  ...
  | b .BarbSim_tau s ⇒
      let [_ ⊢ fs_res \x. S] = s in
      let Ex_sim_barb_def [_,x:names ⊢ S'] b' =
           b .BarbSim_tau [_,x:names ⊢ S] in
      let s' = [_ ⊢ fs_res \x. S'] in
      let b'' = res_barb_sim b' in
      Ex_sim_barb_def s' b''
```

PART 5

Conclusions

Results & future work

CCFB3: Lessons Learned

Recall the LTS rules for replication:

$$\textsc{S-Rep}\ \frac{P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P' \mid !P} \qquad \textsc{S-Rep-Com}\ \frac{P \xrightarrow{\bar{x}y} P' \qquad P \xrightarrow{x(z)} P''}{!P \xrightarrow{\tau} (P' \mid P''\{y/z\}) \mid !P} \qquad \textsc{S-Rep-Close}\ \frac{P \xrightarrow{\bar{x}(z)} P' \qquad P \xrightarrow{x(z)} P''}{!P \xrightarrow{\tau} ((\nu z)(P' \mid P'')) \mid !P}$$

- CCFB3 only lists the first LTS replication rule $\Rightarrow$ structural congruence not included in barbed similarity

CCFB3: Lessons Learned

Moreover:

- CCFB3 is a valid case study for evaluating the mechanization of coinductive proofs
- Structural congruence: more central than initially anticipated
- Up-to techniques: less central than initially anticipated (only used in one proof)

Beluga: Lessons Learned

- HOAS $\Rightarrow$ seamless treatment of names and binders
- Indexed codata $\Rightarrow$ seamless treatment of coinductive reasoning via observations
- Close correspondence between formal and informal proofs

Beluga: Lessons Learned

- HOAS $\Rightarrow$ seamless treatment of names and binders
- Indexed codata $\Rightarrow$ seamless treatment of coinductive reasoning via observations
- Close correspondence between formal and informal proofs
- Beluga does not currently check productivity automatically $\Rightarrow$ guardedness of the corecursive definitions was therefore verified manually.
- One auxiliary function typechecks, but Beluga's totality checker cannot certify it because of substitution-unification problems $\Rightarrow$ termination and coverage were checked manually.
- Beluga's two-level architecture $\Rightarrow$ no higher-order predicates to reason about bisimulations or up-to relations more generally

Future Work

- Extend this result to weak barbed similarity
- Add *sum* and *match* to the syntax
- Encode other (bi)similarities and formalize the proofs of the inclusions between the induced congruences
- Use other semantics to induce barbed similarity and compare the proofs of the context lemma

Conclusions

Main work

- Formalized one implication of the context lemma for barbed similarity in the π-calculus
- Approximately 1500 lines of code, divided in 23 definitions—three of which are coinductive (some 10% of codebase), and 53 theorems

Additional results

- Developed the formalization of process contexts-as-relations in Beluga
- Constructed and formalized a counterexample showing that the benchmark replication rules do not validate replication unfolding.
- Proved equivalence of early, late and reduction semantics in barbed similarity definition
- Gained some insight about the current limitations of Beluga

Thank you!

Any questions?

Lea Trogni · lea.trogni@studenti.unimi.it
Gabriele Cecilia · gcecilia@augusta.edu
Alberto Momigliano · alberto.momigliano@unimi.it

Code: github.com/LeaTrogni/formalizing-barbed-similarity-for-the-pi-calculus-in-beluga