A Formalization of the Reversible Concurrent Calculus $CCSK^P$ in Beluga

Gabriele Cecilia

Augusta University

20 June 2025

ICE 2025

Table of Contents

Reversible Concurrent Calculi

Concurrent Calculi:

- Abstract models for concurrent systems
- Examples: CCS, π-calculus

Reversible Concurrent Calculi

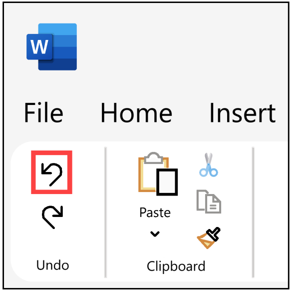
Concurrent Calculi:

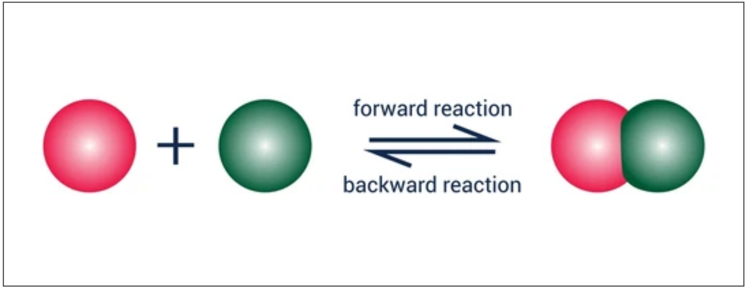
- Abstract models for concurrent systems
- Examples: CCS, π-calculus

Reversible Concurrent Calculi:

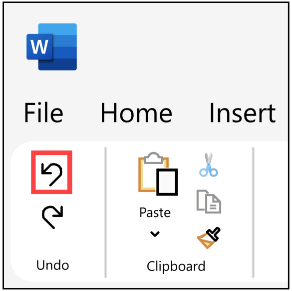
- Abstract models for concurrent systems *in which every action can be undone*
- Examples: CCSK, RCCS, CCSK^{P}

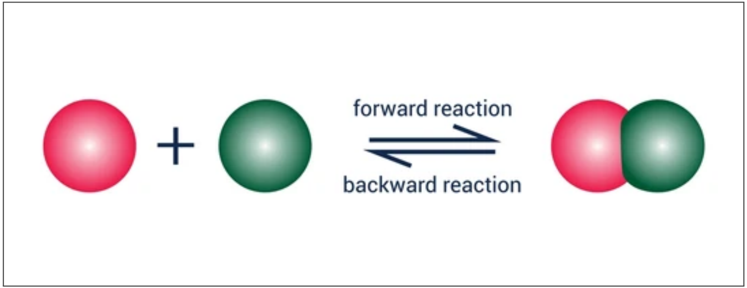
Reversibility





Reversibility





- Accurate representation of reversible systems
- Computational efficiency: chips, debuggers, quantum computing, ...

Formalization

How one line of code caused a $60 million loss

60,000 people lost full phone service, half of AT&T's network was down, and 500 airline flights were delayed

NOV 13, 2023

On January 15th, 1990, AT&T's New Jersey operations center detected a widespread system malfunction, shown by a plethora of red warnings on their network display.

Despite attempts to rectify the situation, the network remained compromised for 9 hours, leading to a 50% failure rate in call connections.

AT&T lost over **$60 million** as a result with **over 60,000 of Americans left with fully disconnected phones.**

How a single line of code brought down a half-billion euro rocket launch

It's Tuesday, June 4th, 1996, and the European Space Agency is set to launch its new Ariane 5 rocket for the first time. This is the culmination of a decade of design, testing and a budget spending billions of euros.

Formalization

Mechanized Metatheory for the Masses: The POPLMARK Challenge

Brian E. Aydemir[1], Aaron Bohannon[1], Matthew Fairbairn[2], J. Nathan Foster[1],
Benjamin C. Pierce[1], Peter Sewell[2], Dimitrios Vytiniotis[1],
Geoffrey Washburn[1], Stephanie Weirich[1], and Steve Zdancewic[1]

[1] Department of Computer and Information Science,
University of Pennsylvania
[2] Computer Laboratory, University of Cambridge

Abstract. How close are we to a world where every paper on programming languages is accompanied by an electronic appendix with machine-checked proofs?

Existing Concurrent Calculi Formalizations

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 indexes
Bengtson 09	Formalizing Process Calculi	Nominal techniques
Miller et al. 99	Foundational Aspects of Syntax	HOAS
Honsell et al. 01	π-Calculus in (Co)Inductive Type Theory	HOAS

Existing Concurrent Calculi Formalizations

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 indexes
Bengtson 09	Formalizing Process Calculi	Nominal techniques
Miller et al. 99	Foundational Aspects of Syntax	HOAS
Honsell et al. 01	π-Calculus in (Co)Inductive Type Theory	HOAS

... but no reversible concurrent calculi formalizations.

Table of Contents

CCS with Keys and Proof labels (CCSK^{P})

CCSK: CCS with Keys

- A reversible extension of CCS
- Phillips & Ulidowski, 2007
- Processes and transitions enriched with communication keys

CCS with Keys and Proof labels (CCSK^{P})

CCSK: CCS with Keys

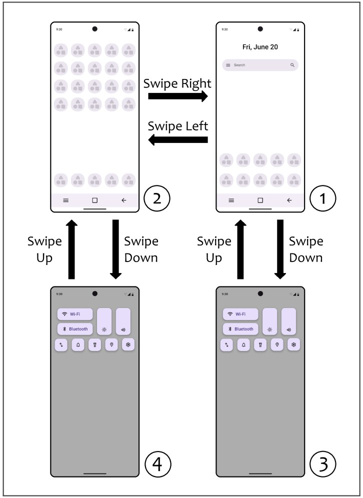
- A reversible extension of CCS
- Phillips & Ulidowski, 2007
- Processes and transitions enriched with communication keys

CCSK^{P}: CCS with Keys and Proof labels

- A proved transition system for CCSK
- Aubert, 2024
- Semantics enriched with proof labels and causality relations

Example: Smartphone

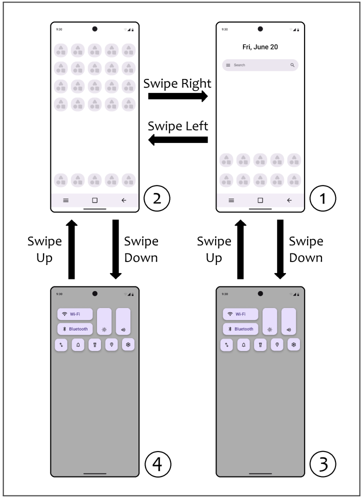
Example: Smartphone



Representation in CCS:

$$P_1 \stackrel{\text{def}}{=} \text{left}.P_2 + \text{down}.P_3$$
$$P_2 \stackrel{\text{def}}{=} \text{right}.P_1 + \text{down}.P_4$$
$$P_3 \stackrel{\text{def}}{=} \text{up}.P_1$$
$$P_4 \stackrel{\text{def}}{=} \text{up}.P_2$$

Example: Smartphone



Representation in CCSK$^{\mathsf{P}}$:

$$X = \mathsf{left}.\mathsf{down} + \mathsf{down}$$

$$X \mathrel{\substack{\longmapsto \\ \leftsquigarrow}} \mathsf{left}[k].\mathsf{down} + \mathsf{down}$$

Syntax

- Infinite set of names N: $a, b, \ldots$
- Complementary names $\overline{\text{N}}$: $\overline{a}, \overline{b}, \ldots$
- Symbol for interactions τ
- Infinite set of keys: $k, m, \ldots$

(The first three items are grouped by a brace:) Labels, ranged over by $\alpha, \beta, \ldots$

Syntax

- Infinite set of names N: $a, b, \ldots$
- Complementary names $\overline{\mathrm{N}}$: $\overline{a}, \overline{b}, \ldots$
- Symbol for interactions τ
- Infinite set of keys: $k, m, \ldots$

(The first three items: } Labels, ranged over by $\alpha, \beta, \ldots$)

Definition

The set of processes is defined by the following syntax:

$$X, Y \ ::= \ \mathbf{0} \ | \ \alpha.X \ | \ \alpha[k].X \ | \ X + Y \ | \ (X \mid Y) \ | \ X \setminus a$$

Notation:

keys(X): set of keys occurring in X

std(X): predicate true when X has no keys (X is said to be *standard*)

Semantics

Given by a Labelled Transition System (LTS).

Transitions are labelled by *proof labels*.

Examples of transitions:

- $l.d + d \xmapsto{+_L l[k]} l[k].d \, + \, d$
- $l.d + d \xmapsto{+_R d[m]} l.d \, + \, d[m]$
- $a \mid b[m] \overset{|_R b[m]}{\leadsto} a \mid b$
- $a \mid b[m] \xmapsto{|_L a[k]} a[k] \mid b[m]$

Semantics

Definition

The set of proof labels is defined by the following syntax:

$$\theta \ ::= v\alpha[k] \quad | \quad v\langle |_{\mathrm{L}} v_1 \lambda[k],\ |_{\mathrm{R}} v_2 \overline{\lambda}[k] \rangle$$

where λ ranges over $\mathrm{N} \cup \overline{\mathrm{N}}$ and v, v_1 and v_2 range over strings of symbols $\{|_{\mathrm{L}}, |_{\mathrm{R}}, +_{\mathrm{L}}, +_{\mathrm{R}}\}$.

Semantics

Definition

Semantics is given by a combined LTS, made of the union of forward ($\longmapsto$) and backward ($\rightsquigarrow$) transition rules such as the following:

$$\frac{}{\alpha.X \xrightarrow{\alpha[k]} \alpha[k].X}\ \text{pref} \quad \text{std}(X) \qquad\qquad \frac{}{\alpha[k].X \overset{\alpha[k]}{\rightsquigarrow} \alpha.X}\ \underline{\text{pref}} \quad \text{std}(X)$$

$$\frac{X \xrightarrow{\theta} X'}{\alpha[k].X \xrightarrow{\theta} \alpha[k].X'}\ \text{kpref} \quad \not{k}(\theta) \neq k \qquad\qquad \frac{X' \overset{\theta}{\rightsquigarrow} X}{\alpha[k].X' \overset{\theta}{\rightsquigarrow} \alpha[k].X}\ \underline{\text{kpref}} \quad \not{k}(\theta) \neq k$$

<u>Notation:</u>

Given a combined transition t: $X \overset{\theta}{\leadsto} X'$, X and X' are said the *source* and *target* of t.
Two transitions t_1 and t_2 are *connected* if there exists a *path* (i.e., a sequence of transitions) between the source of t_1 and the target of t_2.

Causality Relations

We can define three binary relations on proof labels characterizing causality of transitions:

- Dependence ($\times$)
- Independence (ι)
- Connectivity ($\curlyvee$)

Introduced by Aubert et al. in *“Independence and Causality in the Reversible Concurrent Setting”* (2025). ($\star$)

Connectivity Relation

Action

$$\frac{}{\alpha[k] \curlyvee \theta}\ \mathrm{A}^1 \qquad \frac{}{\theta \curlyvee \alpha[k]}\ \mathrm{A}^2$$

Choice

$$\frac{\theta_1 \curlyvee \theta_2}{+_{\mathrm{d}}\theta_1 \curlyvee +_{\mathrm{d}}\theta_2}\ \mathrm{C}^1_{\mathrm{d}} \qquad \frac{}{+_{\mathrm{d}}\theta_1 \curlyvee +_{\bar{\mathrm{d}}}\theta_2}\ \mathrm{C}^2_{\mathrm{d}}$$

Parallel

$$\frac{\theta_1 \curlyvee \theta_2}{|_{\mathrm{d}}\theta_1 \curlyvee |_{\mathrm{d}}\theta_2}\ \mathrm{P}^1_{\mathrm{d}} \qquad \frac{}{|_{\mathrm{d}}\theta_1 \curlyvee |_{\bar{\mathrm{d}}}\theta_2}\ \mathrm{P}^2_{\mathrm{d}}$$

Synchronization

$$\frac{\theta \curlyvee \theta_{\mathrm{d}}}{|_{\mathrm{d}}\theta \curlyvee \langle |_{\mathrm{L}}\theta_{\mathrm{L}}, |_{\mathrm{R}}\theta_{\mathrm{R}}\rangle}\ \mathrm{S}^1_{\mathrm{d}} \qquad \frac{\theta_{\mathrm{d}} \curlyvee \theta}{\langle |_{\mathrm{L}}\theta_{\mathrm{L}}, |_{\mathrm{R}}\theta_{\mathrm{R}}\rangle \curlyvee |_{\mathrm{d}}\theta}\ \mathrm{S}^2_{\mathrm{d}}$$

$$\frac{\theta_1 \curlyvee \theta'_1 \qquad \theta_2 \curlyvee \theta'_2}{\langle |_{\mathrm{L}}\theta_1, |_{\mathrm{R}}\theta_2\rangle \curlyvee \langle |_{\mathrm{L}}\theta'_1, |_{\mathrm{R}}\theta'_2\rangle}\ \mathrm{S}^3$$

Dependence Relation

Action

$$\frac{}{\alpha[k] \times \theta}\ \mathrm{A}^1 \qquad \frac{}{\theta \times \alpha[k]}\ \mathrm{A}^2$$

Choice

$$\frac{\theta \times \theta'}{+_{\mathrm{d}}\theta \times +_{\mathrm{d}}\theta'}\ \mathrm{C}^1_{\mathrm{d}} \qquad \frac{}{+_{\mathrm{d}}\theta \times +_{\bar{\mathrm{d}}}\theta'}\ \mathrm{C}^2_{\mathrm{d}}$$

Parallel

$$\frac{\theta \times \theta'}{|_{\mathrm{d}}\theta \times |_{\mathrm{d}}\theta'}\ \mathrm{P}^1_{\mathrm{d}} \qquad \frac{\not{k}(\theta) = \not{k}(\theta')}{|_{\mathrm{d}}\theta \times |_{\bar{\mathrm{d}}}\theta'}\ \mathrm{P}^2_{\mathrm{d}}$$

Synchronization

$$\frac{\theta \times \theta_{\mathrm{d}}}{|_{\mathrm{d}}\theta \times \langle |_{\mathrm{L}}\theta_{\mathrm{L}}, |_{\mathrm{R}}\theta_{\mathrm{R}}\rangle}\ \mathrm{S}^1_{\mathrm{d}} \qquad \frac{\theta_{\mathrm{d}} \times \theta}{\langle |_{\mathrm{L}}\theta_{\mathrm{L}}, |_{\mathrm{R}}\theta_{\mathrm{R}}\rangle \times |_{\mathrm{d}}\theta}\ \mathrm{S}^2_{\mathrm{d}}$$

$$\frac{\theta_i \times \theta'_i \qquad \theta_j \curlyvee \theta'_j \qquad i, j \in \{1,2\}, i \neq j}{\langle |_{\mathrm{L}}\theta_1, |_{\mathrm{R}}\theta_2\rangle \times \langle |_{\mathrm{L}}\theta'_1, |_{\mathrm{R}}\theta'_2\rangle}\ \mathrm{S}^3$$

Independence Relation

Action

(empty)

Choice

$$\frac{\theta \;\iota\; \theta'}{+_{\mathrm{d}}\theta \;\iota\; +_{\mathrm{d}}\theta'}\ \mathrm{C}^1_{\mathrm{d}}$$

Parallel

$$\frac{\theta \;\iota\; \theta'}{|_{\mathrm{d}}\theta \;\iota\; |_{\mathrm{d}}\theta'}\ \mathrm{P}^1_{\mathrm{d}} \qquad \frac{\not{k}(\theta) \neq \not{k}(\theta')}{|_{\mathrm{d}}\theta \;\iota\; |_{\bar{\mathrm{d}}}\theta'}\ \mathrm{P}^2_{\mathrm{d}}$$

Synchronization

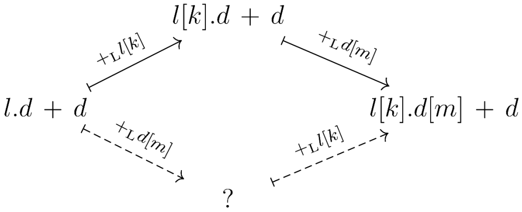
$$\frac{\theta \;\iota\; \theta_{\mathrm{d}}}{|_{\mathrm{d}}\theta \;\iota\; \langle |_{\mathrm{L}}\theta_{\mathrm{L}}, |_{\mathrm{R}}\theta_{\mathrm{R}}\rangle}\ \mathrm{S}^1_{\mathrm{d}} \qquad \frac{\theta_{\mathrm{d}} \;\iota\; \theta}{\langle |_{\mathrm{L}}\theta_{\mathrm{L}}, |_{\mathrm{R}}\theta_{\mathrm{R}}\rangle \;\iota\; |_{\mathrm{d}}\theta}\ \mathrm{S}^2_{\mathrm{d}}$$

$$\frac{\theta_1 \;\iota\; \theta'_1 \qquad \theta_2 \;\iota\; \theta'_2}{\langle |_{\mathrm{L}}\theta_1, |_{\mathrm{R}}\theta_2\rangle \;\iota\; \langle |_{\mathrm{L}}\theta'_1, |_{\mathrm{R}}\theta'_2\rangle}\ \mathrm{S}^3$$

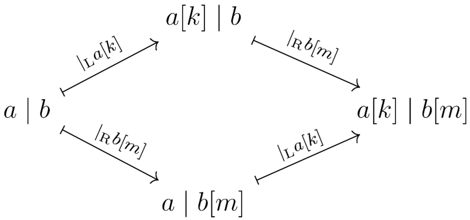
Causality Relations

Examples:

- $+_{\mathrm{L}} l[k] \times +_{\mathrm{L}} d[m]$:



- $|_{\mathrm{L}} a[k] \; \iota \; |_{\mathrm{R}} b[m]$:



Properties of Causality Relations

Main results proven in Section 3-4 of $(\star)$:

Theorem 1 (Characterization of connectivity of proof labels)

(i) If $t_1 : X_1 \xmapsto{\theta_1} X_1'$ and $t_2 : X_2 \xmapsto{\theta_2} X_2'$ are connected, then $\theta_1 \curlyvee \theta_2$.
(ii) If $\theta_1 \curlyvee \theta_2$, then there exist $t_1 : X_1 \xmapsto{\theta_1} X_1'$ and $t_2 : X_2 \xmapsto{\theta_2} X_2'$ such that t_1 and t_2 are connected.

Theorem 2 (Complementarity of dependence and independence)

(i) If $\theta_1 \iota \theta_2$ then $\theta_1 \curlyvee \theta_2$.
(ii) If $\theta_1 \times \theta_2$ then $\theta_1 \curlyvee \theta_2$.
(iii) If $\theta_1 \curlyvee \theta_2$ then either $\theta_1 \iota \theta_2$ or $\theta_1 \times \theta_2$, but not both.

Properties of Causality Relations

- *Soundness* of the causality relations
- Dependence and independence are usually *defined by complementarity*.

  Separate axioms $\rightarrow$ easier to deal with them

Table of Contents

Beluga

Developed at the Complogic group at McGill University, Canada

$\longrightarrow$ Two-level system (LF level, computation level)

$\longrightarrow$ Encoding of object-level binding constructs through Higher-Order Abstract Syntax (HOAS)

$\longrightarrow$ Terms are paired with the *contexts* that give them meaning

$\longrightarrow$ *Curry-Howard isomorphism*: proofs as recursive functional programs, propositions as types

Higher-Order Abstract Syntax (HOAS)

Bound variables of the object language as arguments of meta-language functions

$\rightarrow$ α-renaming and capture-avoiding substitutions managed by the meta-language

$\rightarrow$ Focus on the development of the target system, no technical details of names handling

Useful for encoding systems with a complex binders infrastructure (e.g., π-calculus)

Encoding of Syntax

Names, Keys, Labels and Processes

```
LF names: type =;
LF keys: type =
  | z: keys
  | s: keys → keys;
LF labels: type =
  | inp: names → labels
  | out: names → labels
  | tau: labels;
```

```
LF proc: type =
  | null: proc                                     % 0
  | pref: labels → proc → proc                     % A.X
  | kpref: labels → keys → proc → proc             % A[k].X
  | sum: proc → proc → proc                        % X+Y
  | par: proc → proc → proc                        % X|Y
  | nu: (names → proc) → proc;                     % X\a
```

Examples:

- $l.d + d$ $\quad\rightarrow\quad$ `sum (pref l (pref d zero)) (pref d zero)`
- $a \mid b[m]$ $\quad\rightarrow\quad$ `par (pref a zero) (kpref b m zero)`
- $(\overline{a}.b) \setminus b$ $\quad\rightarrow\quad$ `nu \b.(pref (out a) (pref (inp b) zero))`

Encoding of Syntax

Terms are paired with contexts, containing assumptions. Contexts are classified through schemas

We need a context of the form "`x:names`":

Context declaration

```
schema ctx = names;
```

Consequence: we can define *contextual* processes `[g ⊢ X]`, where `g` contains the free variables occurring in the open process `X`.

Encoding of Semantics

Proof Labels

```
LF pr_lab: type =
  | pr_base: labels → keys → pr_lab           % α[k]
  | pr_suml: pr_lab → pr_lab                  % +Lθ
  | pr_sumr: pr_lab → pr_lab                  % +Rθ
  | pr_parl: pr_lab → pr_lab                  % |Lθ
  | pr_parr: pr_lab → pr_lab                  % |Rθ
  | pr_sync: pr_lab → pr_lab → pr_lab;        % ⟨|Lθ1, |Rθ2⟩
```

Examples:

- $+_{\mathrm{L}} l[k] \quad \rightarrow$ `pr_suml (pr_base l k)`
- $|_{\mathrm{R}} b[m] \quad \rightarrow$ `pr_parr (pr_base b m)`

Encoding of Semantics

Forward and Backward Transitions (and auxiliary predicates)

```
LF key: pr_lab → keys → type = ...        LF std: proc → type = ...

LF fstep: proc → pr_lab → proc → type =
  | fs_pref: std X → fstep (pref A X) (pr_base A K) (kpref A K X)
  ...

LF bstep: proc → pr_lab → proc → type =
  | bs_pref: std X → bstep (kpref A K X) (pr_base A K) (pref A X)
  ...
```

Idea: $X \xmapsto{\theta} Y$ holds $\Leftrightarrow$ `(fstep X T Y)` is inhabited

Analogously, causality relations are encoded by three type families `conn`, `dep` and `indep`.

Writing proofs in Beluga

Proofs in Beluga:

- Total (recursive) functions
- Proof term written by the user, without tactics: interactivity through computation holes
- Lack of syntactic sugar for existentials and conjunctions
  $\rightarrow$ additional type families to encode proof statements

Proof of Theorems 1 and 2: key insights and findings

- Basic properties of keys, proof labels and transitions to be encoded
  $\rightarrow$ 15 additional lemmas

  Examples: decidability of equality of keys, standard processes have no keys, loop lemma (each transition can be reversed), ...

- **Theorem 2 (complementarity):** direct and uneventful encoding

Proof of Theorems 1 and 2: key insights and findings

Theorem 1 (connectivity):

- Some auxiliary lemmas are not required
- Some statements have been slightly refined
- Lengthy proofs, due to many nested pattern matchings
- Low-level details to fill out $\rightarrow$ 20 additional lemmas
- Non-constructive subcase $\rightarrow$ new proof strategy (+ 500 lines of code)

Example of proof in Beluga

Lemma: for all keys K, K does not occur in a standard process

```
rec no_key_in_std: (g:ctx) {K:[⊢ keys]} [g ⊢ std X] → [g ⊢ notin K[] X] =
  / total d (no_key_in_std _ _ _ d) /
mlam K ⇒ fn d ⇒ case d of
  | [g ⊢ std_null] ⇒ [g ⊢ not_null]
  | [g ⊢ std_pref D] ⇒ let [g ⊢ N] = no_key_in_std [⊢ K] [g ⊢ D] in
    [g ⊢ not_pref N]
  | [g ⊢ std_sum D1 D2] ⇒ let [g ⊢ N1] = no_key_in_std [⊢ K] [g ⊢ D1] in
    let [g ⊢ N2] = no_key_in_std [⊢ K] [g ⊢ D2] in [g ⊢ not_sum N1 N2]
  | [g ⊢ std_par D1 D2] ⇒ let [g ⊢ N1] = no_key_in_std [⊢ K] [g ⊢ D1] in
    let [g ⊢ N2] = no_key_in_std [⊢ K] [g ⊢ D2] in [g ⊢ not_par N1 N2]
  | [g ⊢ std_nu \a.D] ⇒
    let [g,a:names ⊢ N] = no_key_in_std [⊢ K] [g,a:names ⊢ D] in
    [g ⊢ not_nu \a.N];
```

Table of Contents

Evaluation

Technical overview:

- Size of the encoding: $\sim$2000 lines of code
- Informal theorems and lemmas covered: 13
- Technical and auxiliary lemmas: 36

Evaluation

Benefits of using Beluga:

- HOAS $\rightarrow$ no handling of bound names
- Proof term matches the informal proof

Drawbacks of using Beluga:

- Lack of syntactic sugar for existentials and conjunctions
- Limited support for custom notation
- No abstraction mechanism for repeated proof patterns

Overall, other proof assistants might be a better fit for this system.

Conclusions

<u>**Results:**</u>

- First formalization of a reversible concurrent calculus
- Verified the correctness of Sections 3-4 of ($\star$): causality relations are sound
- Filled out details and provided a new proof strategy

Future Work

- Journal paper version of ($\star$), with more results formalized
- Covering subsystems of CCSK^{P} (CCS, CCSK)
- Formalizing other (results about) reversible concurrent calculi, such as: *"An Axiomatic Theory for Reversible Computation"* by Lanese et al.

Thank you for listening!
Any questions?

Scan to access the formalization repository: