



Algorithmic Conversion with Surjective Pairing: A Syntactic and Untyped Approach

University of Pennsylvania
Yiyun Liu, Stephanie Weirich



Algorithmic Conversion with Surjective Pairing: A Syntactic and Untyped Approach

University of Pennsylvania
Yiyun Liu, Stephanie Weirich



Yiyun Liu

Ph. D. Candidate at Penn

liuyiyun@seas.upenn.edu

- Designer of DCOI: *Dependent Calculus of Indistinguishability*
- POPL 24 – DCOI the language
- POPL 25 – DCOI the logic
- POPL 26 – this work
- Look for his forthcoming dissertation *Dependency tracking and Dependent types*



Algorithmic Conversion with Surjective Pairing: A Syntactic and Untyped Approach

University of Pennsylvania
Yiyun Liu, Stephanie Weirich

Conversion

- Conversion is the heart of dependent type theory

$$\frac{\Gamma \vdash a : A \quad A = B \quad \Gamma \vdash B : \mathcal{U}_i}{\Gamma \vdash a : B}$$

- Type theory must *define* what it means for types (and programs) to be **equivalent**
- More types/terms are equivalent \Rightarrow more programs type check

Conversion with surjective pairing

- More types/terms are equivalent \Rightarrow more programs type check
- Congruence relation: reflexive, symmetric, transitive, congruent
- Conversion **always** includes β -rules

$$\frac{}{(\lambda x. a) b = a[b/x]}$$

$$\frac{}{\pi_1 (a, b) = a}$$

$$\frac{}{\pi_2 (a, b) = b}$$

- Conversion **sometimes** includes η -rules

$$\frac{x \notin \mathbf{freevar}(b)}{(\lambda x. b x) = b} \quad \frac{}{(\pi_1 a, \pi_2 a) = a}$$

Surjective pairing



Algorithmic Conversion with Surjective Pairing: A Syntactic and Untyped Approach

University of Pennsylvania
Yiyun Liu, Stephanie Weirich

How to decide when types are equal?

- We don't just want to define what terms type check, we also want to **implement** a type checker
- Algorithm for β -equality: **reduce and compare!**

$$\frac{a \rightsquigarrow_{\beta}^* c \quad b \rightsquigarrow_{\beta}^* c}{a \Downarrow b}$$

- Family of algorithms: β -reductions can occur in any subterm, in any order, and not necessarily to a normal form

Is a reduce-and-compare algorithm correct?

- **Reduce-and-compare**

$$\frac{a \rightsquigarrow_{\beta}^* c \quad b \rightsquigarrow_{\beta}^* c}{a \Downarrow b}$$

- To prove that $a \Downarrow b$ decides $a = b$, we need to show
 - **Termination:** $a \Downarrow b$ terminates for well-typed terms (via a logical relation)
 - **Soundness:** If $a \Downarrow b$, then $a = b$ (immediate by definition)
 - **Completeness:** If $a = b$, then $a \Downarrow b$ (most cases straightforward, transitivity requires confluence)

What about η ?

- Let's extend **reduce-and-compare** with η -reductions

$$\frac{x \notin \text{freevar}(a)}{(\lambda x. a x) \rightsquigarrow_{\eta} a} \qquad \frac{}{(\pi_1 a, \pi_2 a) \rightsquigarrow_{\eta} a}$$

- Need to show that \Downarrow , with η -rules added, satisfies termination, soundness, and completeness
- **Bad news:** Reduction with surjective pairing is not confluent!
Can't show that $a \Downarrow b$ is transitive (needed for completeness)



Algorithmic Conversion with Surjective Pairing: A Syntactic and Untyped Approach

University of Pennsylvania
Yiyun Liu, Stephanie Weirich

Untyped vs. Typed equality

- Definition of equivalence may be an *untyped* or a *typed* relation

$$\frac{\Gamma \vdash a : A \quad A = B \quad \Gamma \vdash B : \mathcal{U}_i}{\Gamma \vdash a : B}$$

$$\frac{\Gamma \vdash a : A \quad \Gamma \vdash A = B : \mathcal{U}_i}{\Gamma \vdash a : B}$$

- **Untyped** relation is unaware of types
- **Typed** relation only relates terms with the same type
- This is controversial

Untyped vs. Typed equality

- Algorithms for deciding untyped equality based on η -reduction

$$\frac{x \notin \text{freevar}(a)}{(\lambda x. a x) \rightsquigarrow_{\eta} a}$$

- Limited to *functions only* due to **lack of confluence**
- Untyped* equality is extensible to **irrelevant arguments**, where equal terms may not have equal types (cf. DCOI)

- Algorithms for deciding typed equality typically based on η -expansion as η -reduction may **not preserve types**
- Typed* equality is extensible to η -rules for products and unit

$$\frac{\Gamma \vdash a : \Sigma x : A. B}{\Gamma \vdash a = (\pi_1 a, \pi_2 a) : \Sigma x : A. B}$$

$$\frac{}{\Gamma \vdash a = \mathbf{unit} : \mathbf{Unit}}$$

Our Contribution

We show the **correctness** of reduce-and-compare for both **untyped*** and **typed** equivalence with η -rules for functions and products

$$\frac{x \notin \mathbf{freevar}(b)}{(\lambda x. b x) = b}$$
$$\frac{}{(\pi_1 a, \pi_2 a) = a}$$

$$\frac{\Gamma \vdash b : \Pi x : A. B}{\Gamma \vdash (\lambda x. b x) = b : \Pi x : A. B}$$
$$\frac{\Gamma \vdash a : \Sigma x : A. B}{\Gamma \vdash a = (\pi_1 a, \pi_2 a) : \Sigma x : A. B}$$

*Our specification of **untyped** equality is nonstandard.

Our correctness proof (overview)

- **Termination:** $a \Downarrow b$ terminates for well-typed terms (logical relation)
- **Completeness:** If $a = b$, then $a \Downarrow b$ *Challenge: lack of confluence*
 1. Find a “good” set of terms where reduction is confluent.
 2. Show that all well-typed terms are in this set.
- **Soundness (untyped):** If $a \Downarrow b$ then $a = b$
*Restrict transitivity rule to terms in the “good” set
- **Soundness (typed):** If $a \Downarrow b$ (and $\Gamma \vdash a, b : A$) then $\Gamma \vdash a = b : A$
Challenge: η -reduction is not typed preserving!
 1. Find a **different** algorithm that is easy to prove sound for typed conversion
 2. Show that this algorithm is complete with respect to any reduce-and-compare algorithm

Completeness via SN set

Completeness: If $a = b$, then $a \Downarrow b$ *Challenge: lack of confluence*

1. Prove confluence for terms in set **SN**, defined by Van Raamsdonk and Severi (1995)
2. Prove that all well-typed terms are in **SN** using a logical relation (Joachimski and Matthes 2003, Abel et al. 2019)

SN: An inductive characterization of strongly β -normalizing terms

- Does not include **stuck** terms, eliminating ill-formed cases of the confluence proof
- Surprise! SN characterizes $\beta\eta$ -normalizing terms (JM 2003)

Soundness via Coquand's algorithm

- **Soundness (typed):** If $a \Downarrow b$ and $\Gamma \vdash a, b : A$ then $\Gamma \vdash a = b : A$
Challenge: η -reduction is not typed preserving!
- Coquand's algorithmic conversion ($a \leftrightarrow b$): an untyped algorithm that **η -expands** using the shape of terms
- *Proof:*
 - $a \Downarrow b$ implies $a \leftrightarrow b$
(by strong normalization and confluence)
 - $\Gamma \vdash a, b : A$ and $a \leftrightarrow b$ implies $\Gamma \vdash a = b : A$
(by preservation and injectivity of type constructors)



Algorithmic Conversion with Surjective Pairing: A **Syntactic** and Untyped Approach

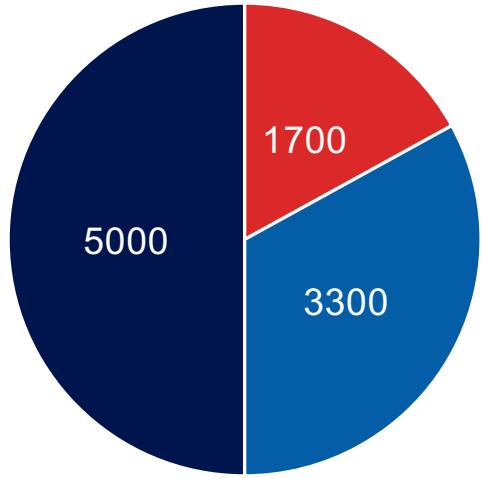
University of Pennsylvania
Yiyun Liu, Stephanie Weirich

Rocq Mechanization: mostly syntactic



- Syntactic approach: reasoning about a system in terms of its judgements instead of its models
- Works for a rich dependent type theory
 - Functions, pairs, natural numbers with an induction principle
 - An infinite predicative universe hierarchy, w/ contravariant subtyping
- Can extract an executable, certified conversion algorithm
- Minimal requirements from proof assistant (libraries & logical strength)
 - Key tool: autosubst-ocaml

Rocq Mechanization: mostly syntactic



- Logical Relation
- Confluence
- Type system properties

Factored proof: logical relation for normalization separate from type-agnostic syntactic results

- Semantic proof via logical predicate (~1700 LoC)
- Type-system agnostic confluence proof (~3300 LoC)
- Properties about the type system (preservation, substitution, etc.) & algorithmic conversion (~5000 LoC)
- Does not include code generated by autosubst-ocaml

Conclusion and Future Work

- We can include surjective pairing in dependent type theories that use **untyped** or **typed** equality
- We can implement conversion using a reduction-based algorithms and mechanically prove it correct to either system
- Future work
 - Update definitions to be more like Rocq/Agda/Lean
 - Extend proofs to DCOI and irrelevant arguments (untyped equality)
 - Employ more automation in syntactic proofs about reduction



Algorithmic Conversion with Surjective Pairing: A Syntactic and Untyped Approach

University of Pennsylvania
Yiyun Liu, Stephanie Weirich