



$$\lambda((N\uparrow)\circ)\uparrow$$

$$\begin{aligned}x &\rightarrow y \\ y &\rightarrow z \\ z &\rightarrow \alpha\end{aligned}$$

$$x \Leftrightarrow y$$

$$\alpha$$

$$\beta$$

$$\begin{matrix}x \\ \circ \\ z\end{matrix}$$

$$\lambda$$

RETURN OF THE RENAMINGS

RoR

All You Need Is `refl` — Almost

Return of the Renamings

Peter Thiemann Marius Weidner

Wadler70 @ Edinburgh

09 Sep 2026

universität freiburg

Substitution in PL metatheory: a solved problem?

On paper

$M[x := N]$

“capture-free substitution”

“Barendregt convention”

Substitution in PL metatheory: a solved problem?

On paper

$$M[x := N]$$

“capture-free substitution”
“Barendregt convention”

In a proof assistant

representations (vectors, functions, ...),
renamings,
composition lemmas,
...

Substitution in PL metatheory: a solved problem?

On paper

$M[x := N]$

“capture-free substitution”
“Barendregt convention”

In a proof assistant

representations (vectors, functions, ...),
renamings,
composition lemmas,
...

What if substitution lemmas were **definitional**?

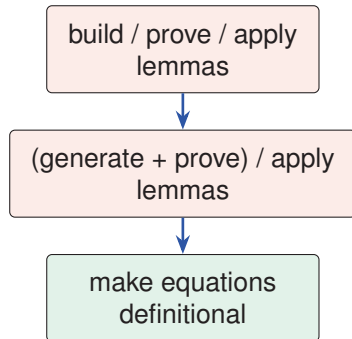
`proof = refl`

Automated Proof

- Before 2015: build and prove a substantial substitution library; apply lemmas manually
- Since 2015: generate library by Autosubst; apply lemmas via tactics
- We ask for something different:

Goal (Agda)

Intrinsically typed, inductive syntax, de Bruijn variables, substitution equations normalize to `refl`



What does `refl` mean?

Running example: $N \uparrow [M]_0 \equiv N$

Explicit propositional proof

Tactical
by `asimpl.`

Definitional
`refl`

```
begin
  N ↑ [ M ]0
≡ ⟨ ⟩
  (N [ wkR ]R) [ M .S ⟨ idR ⟩ ]S
≡ ⟨ compositionalityR S {e = N} {ρ1 = wkR} {σ2 = M .S ⟨ idR ⟩} ⟩
  N [ ⟨ wkR ⟩ ]S (M .S ⟨ idR ⟩ )S
≡ ⟨ cong (λ (σ : Sub Γ Γ) → N [ σ ]S) interact ⟩
  N [ ⟨ idR ⟩ ]S
≡ ⟨ coincidence {e = N} {ρ = idR} ⟩
  N [ idR ]R
≡ ⟨ identityr ⟩
  N
□
```

Benchmark equations [Wadler24]

Introduction

$$(N\uparrow)[M]_0 \equiv N$$

Double substitution

$$N[M]_1[L]_0 \equiv N[L\uparrow]_0[M]_0$$

Commuting substitution

$$N[M]_0[L]_0 \equiv N[L]_1[M[L]_0]_0$$

Discriminator EW

$$\eta\text{-weak: } (\lambda((N\uparrow)\bullet))\uparrow \equiv \lambda(((N\uparrow)\uparrow)\bullet)$$

Discriminators RoR

$$\eta\text{-id: } ((\text{id}^S \uparrow^S) \triangleright^S \bullet) \equiv \text{id}^S$$

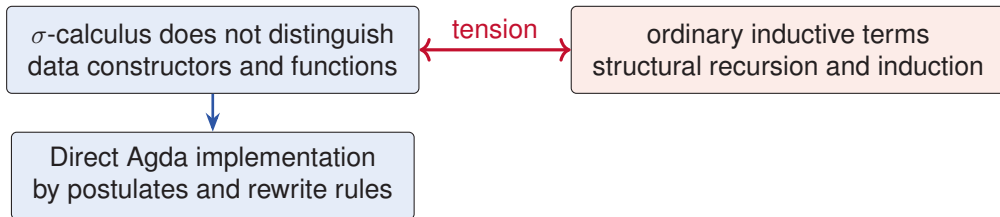
$$\eta\text{-law: } (((\text{id}^S \uparrow^S) \circ^S \sigma) \triangleright^S (\bullet [\sigma])) \equiv \sigma$$

σ -calculus: the equational gold standard

- Rewrite calculus on terms and substitutions
- Complete set of equations
- All benchmarks become definitional
- Confluent and terminating

M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. *Explicit Substitutions*. JFP 1 (1991), 375–416.

The dark side of the gold standard



How much of σ can we keep?

Explicit Weakening (EW) makes weakening syntax

Terms are inductive

$M, N ::=$	$\bullet$	newest variable
	$M\uparrow$	explicit weakening
	λM	abstraction
	$M N$	application

Substitutions are inductive

$\sigma ::=$	id	identity
	$\sigma\uparrow$	weakening
	$\sigma \triangleright M$	cons

Instantiation and composition are plain functions.

Philip Wadler. *Explicit Weakening* (2024).

EW: Three lemmas buy a lot

Proved and installed as rewrite rules

$$\begin{aligned}M[\sigma][\tau] &\equiv M[\sigma ; \tau] \\ \text{id} ; \sigma &\equiv \sigma \\ (\sigma ; \tau) ; v &\equiv \sigma ; (\tau ; v)\end{aligned}$$

introduction refl

double substitution refl

commuting substitution refl

Pros: Inductive syntax, intrinsically typed, much gets definitional.

EW: Three lemmas buy a lot

Proved and installed as rewrite rules

$$\begin{aligned}M[\sigma][\tau] &\equiv M[\sigma ; \tau] \\ \text{id} ; \sigma &\equiv \sigma \\ (\sigma ; \tau) ; v &\equiv \sigma ; (\tau ; v)\end{aligned}$$

introduction refl

double substitution refl

commuting substitution refl

Pros: Inductive syntax, intrinsically typed, much gets definitional.

Cons: Not confluent, some benchmarks break.

EW: A Roadblock

Desired equation (η -weak)

$$(\lambda((N\uparrow)\bullet))\uparrow \equiv \lambda(((N\uparrow)\uparrow)\bullet)$$

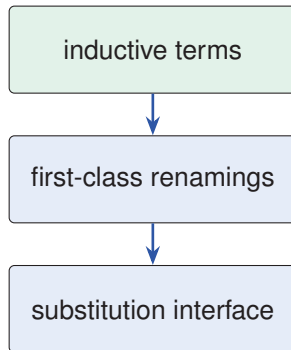
Both sides are normal forms in Explicit Weakening

$$\underbrace{(\lambda \cdots)}_{\text{outer constructor } \uparrow} \uparrow \quad \text{vs.} \quad \underbrace{\lambda(\cdots)}_{\text{outer constructor } \lambda}$$

Return of the Renamings (RoR)

Familiar ground

- Terms are inductive data
- Variables are de Bruijn indices
- Renamings return as maps
- Substitutions are maps
- Ren and Sub mostly opaque
- Rewriting installs σ -like theory with proved equations



RoR: Mapping from EW notation

Presentation notation

- $\bullet$ $:=$ var here
- λN $:=$ lam N
- LM $:=$ app $L M$
- $M\uparrow$ $:=$ $M[\text{wk}^R]^R$
- $M[\sigma]$ $:=$ $M[\sigma]^S$

RoR: Rewrite theory

σ -calculus equations (excerpt)

$$M[\rho]^R[\rho']^R \rightsquigarrow M[\rho ; \rho']^R$$

$$M[\rho]^R[\sigma]^S \rightsquigarrow M[\langle \rho \rangle ; \sigma]^S$$

$$M[\sigma]^S[\tau]^S \rightsquigarrow M[\sigma ; \tau]^S$$

Manufacturing definitional equality

- 1 opaque definitions of Ren and Sub
- 2 expose *only* equations as rewrite rules
- 3 check compatibility with reduction (local confluence)

On η -id and η -law

- not rewrite rules
- break confluence

RoR: proves what EW cannot

Return of the Renamings

$$((\lambda ((N \uparrow) \cdot \bullet)) \uparrow) \equiv (\lambda (((N \uparrow) \uparrow) \cdot \bullet))$$

refl

Explicit Weakening

$$(\lambda((N \uparrow) \bullet)) \uparrow \not\equiv \lambda(((N \uparrow) \uparrow) \bullet)$$

Outer constructors remain distinct.

impossible

Explicit Weakening embeds into RoR



Preservation

Translation commutes with instantiation and composition.

⇒ every EW-equality is also a RoR-equality.

$EW \subsetneq RoR$ strict by η -weak

RoR vs. σ -calculus; aka the meaning of “almost”

η -id

$$\text{var here} \cdot^S \text{wk}^S \equiv \text{id}^S$$

η -law

$$(\text{var here}[\sigma]^S) \cdot^S (\text{wk}^S ; \sigma) \equiv \sigma$$

- Both implied by full σ -calculus
- Neither is definitional in RoR
- Propositional proof “on top” of RoR when needed; not just `refl`

RoR scales

Formalizing System F

- Term substitution and type substitution
- Terms indexed by types: $Expr \Gamma T$
- ... which get substituted: $Expr \Gamma (T[id])$
- ... but $T \neq T[id]$,
- ... so we either insert transports or make $T \equiv T[id]$ definitional!
- ... transports can leak into lemmas
- ... better make them definitional

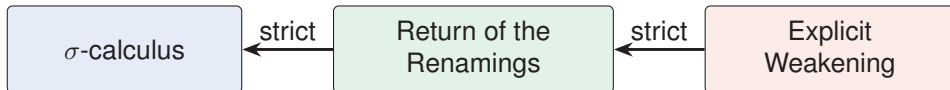
Proof of the pudding

Transports disappear in proofs

- canonicity for System F
- logical relations

Overview

	Inductive terms	Variables		Equational strength	Confluence
σ -calculus	×	unified calculus	term/substitution	full theory, including η	✓
RoR	✓	de Bruijn + renamings		extended σ -laws; no η	✓
EW	✓	• + explicit $_↑$ constructor		weaker again	×



All you need is `refl` — almost

- 1 σ -calculus: full equational theory
- 2 EW regains inductive syntax, but rules out useful equalities, confluence?¹
- 3 RoR keeps inductive syntax and standard variables while recovering a strictly stronger, confluent theory
demonstrated to work on significant developments

¹Non-confluent rewrite rules make definitional equality depend on the reduction path, undermining predictable type checking.

All You Need Is `ref1` — Almost

Peter Thiemann Marius Weidner

Backup: benchmark equations

Introduction

$$(N\uparrow)[M]_0 \equiv N$$

Double substitution

$$N[M]_1[L]_0 \equiv N[L\uparrow]_0[M]_0$$

Commuting substitution

$$N[M]_0[L]_0 \equiv N[L]_1[M[L]_0]_0$$

Through η -expansion

$$(\lambda(N\uparrow\bullet))[\sigma] \equiv \lambda((N[\sigma])\uparrow\bullet)$$

Through single substitution

$$(N[\text{id} \triangleright M])[\sigma] \equiv (N[\sigma\uparrow \triangleright \bullet])[\text{id} \triangleright (M[\sigma])]$$

Weakening through η

$$(\lambda(N\uparrow\bullet))\uparrow \equiv \lambda((N\uparrow)\uparrow\bullet)$$

η -id

$$(\text{id}\uparrow) \triangleright \bullet \equiv \text{id}$$

η -law

$$((\text{id}\uparrow) ; \sigma) \triangleright (\bullet[\sigma]) \equiv \sigma$$

Backup: typed pieces behind the notation

Explicit Weakening

- $\bullet : \Gamma \triangleright A \vdash A$
- $_ \uparrow : \Gamma \vdash A \rightarrow \Gamma \triangleright B \vdash A$
- $\text{id} : \Gamma \models \Gamma$
- $_ \uparrow : \Gamma \models \Delta \rightarrow \Gamma \triangleright A \models \Delta$
- $_ \triangleright _ : \Gamma \models \Delta \rightarrow \Gamma \vdash A \rightarrow \Gamma \models \Delta \triangleright A$

Return of the Renamings

- $\text{here} : \Gamma \triangleright A \ni A$
- $\text{there} : \Gamma \ni A \rightarrow \Gamma \triangleright B \ni A$
- $\text{var} : \Gamma \ni A \rightarrow \Gamma \vdash A$
- $\text{Ren } \Delta \Gamma : \Delta \ni A \rightarrow \Gamma \ni A$
- $\text{Sub } \Delta \Gamma : \Delta \ni A \rightarrow \Gamma \vdash A$

Both substitution notations read $M : \Delta \vdash A$ and $\sigma : \Delta \rightarrow \Gamma$, hence $M[\sigma] : \Gamma \vdash A$.