Normalization by Evaluation for Martin-Löf Type Theory

Daniel Gratzer

October 1, 2018

Goal

Produce a function $\mathsf{nf}(\Gamma, t, A) : \mathbf{Ctx} \times \mathbf{Term} \times \mathbf{Type} \rightharpoonup \mathbf{Term}$ so that the following 3 conditions hold:

1. $\Gamma \vdash t_1 \equiv t_2 : A \implies \mathsf{nf}(\Gamma, t_1, A) = \mathsf{nf}(\Gamma, t_2, A)$
2. If $\Gamma \vdash t : A$ then $\Gamma \vdash t \equiv \mathsf{nf}(\Gamma, t, A) : A$
3. If $\Gamma \vdash t : A$ then $\mathsf{nf}(\Gamma, t, A)$ is a *normal form*
   – more on this shortly.

Why Bother?

Why bother to do this when it's so much easier to not do things?

1. Lars told me to prove normalization for a type theory

Why Bother?

Why bother to do this when it's so much easier to not do things?

1. Lars told me to prove normalization for a type theory
2. Termination, canonicity, consistency are corollaries
3. Decidability of type-checking
   This because of the *conversion rule*:

$$\frac{\Gamma \vdash A \equiv B \qquad \Gamma \vdash t : A}{\Gamma \vdash t : B}$$

4. Adequacy in logical frameworks depends on normalization
5. Completeness of focused proof strategies is equivalent
6. Coherence theorems are normalization theorems in disguise

Why Normalization by Evaluation (NbE)?

Techniques for proving normalization abound, why NbE?

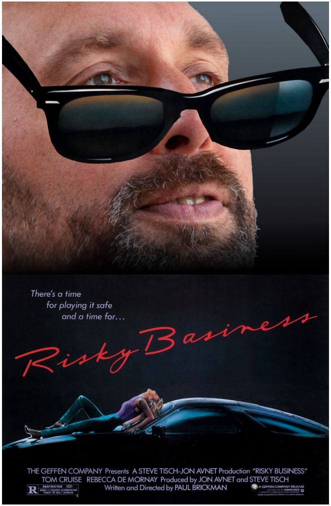
1. Scales to support many languages
   - full dependent types
   - proof-irrelevant types
   - impredicative quantification
   - sized types
   - (conjectured) fitch-style guarded dependent type theory
   - (conjectured) cubical type theory.
2. Amenable to formalization in a (stronger) type theory
3. Practical for implementation*
4. Principled semantic interpretation

What Semantic Interpretation?

It’s too much to discuss today, Jon & Bas have a paper though.

What Semantic Interpretation?

It's too much to discuss today, Jon & Bas have a paper though.



Why Not X Instead?[1]

The most common alternatives to NbE are based on rewriting:

- Define some relation $\rightarrow$ (*steps to*) between terms
- a term is normal when it cannot be reduced further with $\rightarrow$.
- Use logical relations/reducibility candidates to show that $\rightarrow$ terminates for well-typed terms.

[1] for X $\neq$ NbE

Why Not X Instead?[1]

The most common alternatives to NbE are based on rewriting:

- Define some relation $\to$ (*steps to*) between terms
- a term is normal when it cannot be reduced further with $\to$.
- Use logical relations/reducibility candidates to show that $\to$ terminates for well-typed terms.

Not all equalities make sense as reduction rules!

[1] for $X \neq$ NbE

Why Not X Instead?[1]

The most common alternatives to NbE are based on rewriting:

- Define some relation $\to$ (*steps to*) between terms
- a term is normal when it cannot be reduced further with $\to$.
- Use logical relations/reducibility candidates to show that $\to$ terminates for well-typed terms.

Not all equalities make sense as reduction rules!
These proofs are extremely brittle!

[1] for X $\neq$ NbE

Why Not X Instead?[1]

The most common alternatives to NbE are based on rewriting:

- Define some relation $\to$ (*steps to*) between terms
- a term is normal when it cannot be reduced further with $\to$.
- Use logical relations/reducibility candidates to show that $\to$ terminates for well-typed terms.

Not all equalities make sense as reduction rules!
These proofs are extremely brittle!
Entangles questions of reduction strategy!

[1] for X $\neq$ NbE

A Language

We need to specify the language that we're going to normalize.

The Main Judgments

Our type theory is divided into various judgments:

$\Gamma \vdash$	Γ is a valid context
$\Gamma \vdash T$	In context Γ, T is a type
$\Gamma \vdash t : T$	In context Γ, t has type T

The Main Judgments

Our type theory is divided into various judgments:

$\Gamma \vdash$	Γ is a valid context
$\Gamma \vdash T$	In context Γ, T is a type
$\Gamma \vdash t : T$	In context Γ, t has type T

Corresponding equality judgments: $\Gamma \vdash t_1 \equiv t_2 : T$.

Explicit Substitutions

We use *explicit substitutions*, $\Gamma \vdash \sigma : \Delta$, in our type theory:

$$\frac{\Gamma \vdash}{\Gamma \vdash \cdot : () \qquad \Gamma \vdash 1 : \Gamma} \qquad\qquad \frac{\Gamma \vdash T}{\Gamma.T \vdash \uparrow^1 : \Gamma}$$

$$\frac{\Gamma \vdash \sigma_1 : \Delta \qquad \Delta \vdash \sigma_2 : \Xi}{\Gamma \vdash \sigma_2 \circ \sigma_1 : \Xi}$$

$$\frac{\Gamma \vdash \sigma : \Delta \qquad \Delta \vdash T \qquad \Gamma \vdash t : T\{\sigma\}}{\Gamma \vdash \sigma.t : \Delta.T}$$

Crucial rule:

$$\frac{\Gamma \vdash t : T \qquad \Delta \vdash \sigma : \Gamma}{\Delta \vdash t\{\sigma\} : T\{\sigma\}}$$

A Language

The rules for types and contexts:

$$\frac{}{() \vdash} \qquad \frac{\Gamma \vdash \qquad \Gamma \vdash A}{\Gamma.A \vdash}$$

$$\frac{\Gamma \vdash A \qquad \Gamma.A \vdash B}{\Gamma \vdash A \to B} \qquad \frac{\Gamma \vdash}{\Gamma \vdash \mathsf{Unit}} \qquad \frac{\Gamma \vdash}{\Gamma \vdash \mathcal{U}} \qquad \frac{\Gamma \vdash A : \mathcal{U}}{\Gamma \vdash A}$$

A Language

The rules for terms:

$$\frac{\Gamma \vdash}{\Gamma \vdash \mathsf{Unit} : \mathcal{U} \qquad \Gamma \vdash \mathsf{tt} : \mathsf{Unit}} \qquad \frac{\Gamma \vdash A : \mathcal{U} \qquad \Gamma.A \vdash B : \mathcal{U}}{\Gamma \vdash A \to B : \mathcal{U}}$$

$$\frac{\Gamma \vdash A \qquad \Gamma.A \vdash t : B}{\Gamma \vdash \lambda t : A \to B} \qquad \frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash t(u) : B\{1.u\}}$$

$$\frac{\Gamma_1.T.\Gamma_2 \vdash \qquad |\Gamma_2| = k}{\Gamma_1.T.\Gamma_2 \vdash \mathbf{x}_k : T\{\uparrow^{k+1}\}}$$

The Wrinkle

We need the *conversion rule* for any sort of type theory.

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash A \equiv B}{\Gamma \vdash t : B}$$

Dependence means term equality matters for type equality.

$$\frac{\Gamma \vdash A \equiv B : \mathcal{U}}{\Gamma \vdash A \equiv B}$$

The Wrinkle – The Main Equality Rules

$$\frac{\Gamma \vdash u : A \qquad \Gamma.A \vdash t : B}{\Gamma \vdash (\lambda t)(u) \equiv t\{1.u\} : B\{1.u\}}$$

$$\frac{\Gamma \vdash t : A \to B}{\Gamma \vdash \lambda(t\{\uparrow^1\}(\mathbf{x}_0)) \equiv t : A \to B}$$

$$\frac{\Gamma \vdash t : \mathsf{Unit}}{\Gamma \vdash t \equiv \mathsf{tt} : \mathsf{Unit}}$$

Neutral and Normal Forms

Let us isolate special terms which will be *canonical* for $\equiv$.

1. Neutral terms: variables or normals stuck on variables.
2. Normal forms: terms in β-normal and η-long forms.

$$\frac{\Gamma \vdash \mathbf{x}_n : A}{\Gamma \vdash^{\mathsf{neu}} \mathbf{x}_n : A} \qquad \frac{\Gamma \vdash^{\mathsf{neu}} e : A \to B \qquad \Gamma \vdash^{\mathsf{nf}} v : A}{\Gamma \vdash^{\mathsf{neu}} e(v) : B\{1.v\}}$$

Neutral and Normal Forms

Let us isolate special terms which will be *canonical* for $\equiv$.

1. Neutral terms: variables or normals stuck on variables.
2. Normal forms: terms in β-normal and η-long forms.

$$\frac{\Gamma \vdash \mathbf{x}_n : A}{\Gamma \vdash^{\mathsf{neu}} \mathbf{x}_n : A} \qquad \frac{\Gamma \vdash^{\mathsf{neu}} e : A \to B \qquad \Gamma \vdash^{\mathsf{nf}} v : A}{\Gamma \vdash^{\mathsf{neu}} e(v) : B\{1.v\}}$$

$$\frac{\Gamma \vdash}{\Gamma \vdash^{\mathsf{nf}} \mathsf{tt} : \mathsf{Unit} \qquad \Gamma \vdash^{\mathsf{nf}} \mathsf{Unit} : \mathcal{U}} \qquad \frac{\Gamma \vdash A \qquad \Gamma.A \vdash^{\mathsf{nf}} t : B}{\Gamma \vdash^{\mathsf{nf}} \lambda t : A \to B}$$

$$\frac{\Gamma \vdash^{\mathsf{nf}} A : \mathcal{U} \qquad \Gamma.A \vdash^{\mathsf{nf}} B : \mathcal{U}}{\Gamma \vdash^{\mathsf{nf}} A \to B : \mathcal{U}} \qquad \frac{\Gamma \vdash^{\mathsf{neu}} e : \mathcal{U}}{\Gamma \vdash^{\mathsf{nf}} e : \mathcal{U}}$$

Normalization by Evaluation

Now we have a goal, construct $\Gamma \vdash^{\mathsf{nf}} \mathsf{nf}(\Gamma, t, A) : A$ given $\Gamma \vdash t : A$.

Normalization by Evaluation – Historical Context

Original idea:

normalize programs using the ambient semantic universe.

Latent in Martin-Löf's original proofs of the decidability of typing.

Normalization by Evaluation – Historical Context

Next found in implementation of Minlog:

$$\texttt{eval} : (\texttt{Term t}) \to \texttt{t}$$
$$\texttt{quote} : \texttt{t} \to (\texttt{Term t})$$

$$\texttt{normalize} = \texttt{quote . eval}$$

Done in Scheme for the simply-typed lambda calculus at first, adapted to other settings.

Normalization by Evaluation – Historical Context

To adapt to a proof people opted for domains instead of a PL

$$D \cong (D \to D) \oplus (\mathbb{N} \cup \mathbb{V})_\perp$$

Then define the following:

$$\mathsf{eval} : \mathbf{Term} \to D \qquad \mathsf{quote} : D \rightharpoonup \mathbf{Term}$$

Normalization by Evaluation – Historical Context

These historical approaches are imperfect:

- Intrinsic typing proved intractable for impredicativity or dependent types.
- Using domains adds unnecessary complexity and is far removed from implementations.
- The direct “reflect to the metatheory” approach does not scale to extrensic typing.

Normalization by Evaluation – Historical Context

These historical approaches are imperfect:

- Intrinsic typing proved intractable for impredicativity or dependent types.
- Using domains adds unnecessary complexity and is far removed from implementations.
- The direct “reflect to the metatheory” approach does not scale to extrensic typing.

Many presentations now use a different semantic model: syntax.

A Syntactic Semantic Domain

Construct a syntax in which all expressions are canonical.

Divided between neutrals, normals, values, closures.

A Syntactic Semantic Domain – Neutrals

Neutral elements represent computations which are stuck on some variable.

$$e ::= \mathbf{x}_\ell \mid \mathsf{app}(e, \downarrow^A v)$$

N.B. The argument to $\mathsf{app}(e, -)$ must be fully evaluated and annotated.

A Syntactic Semantic Domain – Closures

What happens when we go under a binder?

A Syntactic Semantic Domain – Closures

What happens when we go under a binder?

We choose to suspend evaluation and record the current state with a closure.

$$f ::= t\{\rho\}$$

ρ is the *environment* we're interpreting t. This removes the need for domains, is called *defunctionalization*.

A Syntactic Semantic Domain – Values

It's difficult to isolate η-long forms for dependent type theory.
We settle for isolating β-normal forms for now.

$$v, A ::= \lambda.\ f \mid \mathsf{tt} \mid \mathsf{Unit} \mid \mathsf{Uni} \mid \Pi\, A.\ F$$

A Syntactic Semantic Domain – Values

It's difficult to isolate η-long forms for dependent type theory.
We settle for isolating β-normal forms for now.

$$v, A ::= \lambda.\ f \mid \mathsf{tt} \mid \mathsf{Unit} \mid \mathsf{Uni} \mid \Pi\, A.\ F \mid {\uparrow^{A}} e$$

Need to include neutrals *with* type information to allow η-expansions later.

A Syntactic Semantic Domain

$$
\begin{array}{lcl}
v, A & ::= & \lambda.\ f \mid \mathsf{tt} \mid \mathsf{Unit} \mid \mathsf{Uni} \mid \Pi\, A_1.\ F \mid {\uparrow^{A}}\ e \\
f, F & ::= & t\{\rho\} \\
e & ::= & \mathbf{x}_{\ell} \mid \mathsf{app}(e, v) \\
n & ::= & {\downarrow^{A}}\ v \\
\rho & ::= & \cdot \mid \rho.v
\end{array}
$$

Paying the Piper – Typing Information

The usage of $\downarrow^{A} v$ and $\uparrow^{A} e$ seems very arbitrary. Why do we need typing information?

- We need type information to know whether η-expansion is necessary now that we have neutrals of all types.
  In the domain-theoretic or intrinsic formulation this was baked in as we disallowed such neutrals.

Paying the Piper – Typing Information

The usage of $\downarrow^A v$ and $\uparrow^A e$ seems very arbitrary. Why do we need typing information?

- We need type information to know whether η-expansion is necessary now that we have neutrals of all types.
  In the domain-theoretic or intrinsic formulation this was baked in as we disallowed such neutrals.
- Coquand proposed adding $\downarrow^A v$ to mark a value that should be η-expanded at type A during quotation.
- Quotation proceeds by casing on this type.

The Algorithm

Now that we have defined our sorts of terms, we can define the algorithm.

1. Evaluate a term to a value in some environment

$$\rho \models t \Downarrow v$$

2. Quote a normal form back to a term in a context of length c.

$$c \Vdash n \Uparrow t$$

3. Inject/reflect a term context into an environment.

$$\uparrow\Gamma \rightsquigarrow \rho$$

The Algorithm

$$\begin{aligned}\mathsf{nf}(\Gamma, t, T) = t' \iff& \\ &\uparrow\Gamma \rightsquigarrow \rho \wedge \\ &(\rho \models t \Downarrow v) \wedge (\rho \models T \Downarrow A) \wedge \\ &|\Gamma| \Vdash {\downarrow^{A}} v \Uparrow t'\end{aligned}$$

The relational presentation is ideal for a constructive setting.

The Algorithm – Defining Evaluation

The evaluation judgment is defined by inspection on t.

$$\frac{}{\rho.v \models \mathbf{x}_0 \Downarrow v} \qquad \frac{}{\rho \models \mathsf{tt} \Downarrow \mathsf{tt}} \qquad \frac{}{\rho \models \mathsf{Unit} \Downarrow \mathsf{Unit}}$$

$$\frac{}{\rho \models \mathcal{U} \Downarrow \mathsf{Uni}} \qquad \frac{}{\rho \models \lambda t \Downarrow \lambda.\ t\{\rho\}} \qquad \frac{\rho \models T_1 \Downarrow A}{\rho \models T_1 \to T_2 \Downarrow \Pi\, A.\ T_2\{\rho\}}$$

The Algorithm – Defining Evaluation

The evaluation judgment is defined by inspection on t.

$$\frac{}{\rho.v \models \mathbf{x}_0 \Downarrow v} \qquad \frac{}{\rho \models \mathsf{tt} \Downarrow \mathsf{tt}} \qquad \frac{}{\rho \models \mathsf{Unit} \Downarrow \mathsf{Unit}}$$

$$\frac{}{\rho \models \mathcal{U} \Downarrow \mathsf{Uni}} \qquad \frac{}{\rho \models \lambda t \Downarrow \lambda.\ t\{\rho\}} \qquad \frac{\rho \models T_1 \Downarrow A}{\rho \models T_1 \to T_2 \Downarrow \Pi\, A.\ T_2\{\rho\}}$$

What about the only construct in our language that computes?

The Algorithm – Defining Evaluation

Application uses an auxiliary relation: $v_1 \mathbin{@} v_2 \rightsquigarrow v$.

$$\frac{\rho.a \models t \Downarrow v}{\lambda.\ t\{\rho\} \mathbin{@} a \rightsquigarrow v} \qquad \frac{\rho.a \models T \Downarrow B}{\uparrow^{\Pi\, A.\ T\{\rho\}} e \mathbin{@} a \rightsquigarrow \uparrow^{B} \mathsf{app}(e, \downarrow^{A} a)}$$

$$\frac{\rho \models t \Downarrow v_1 \qquad \rho \models u \Downarrow v_2 \qquad v_1 \mathbin{@} v_2 \rightsquigarrow v}{\rho \models t(u) \Downarrow v}$$

Rule of thumb:

each eliminator gets an auxiliary judgment to either perform β-reduction or construct a new neutral.

The Algorithm – Defining Evaluation

We use a judgment so that syntactic substitutions produce new semantic environments.

$$\frac{}{\rho \models 1 \Downarrow \rho} \qquad \frac{}{\rho.v \models \uparrow^1 \Downarrow \rho} \qquad \frac{\rho_1 \models \sigma_1 \Downarrow \rho_2 \qquad \rho_2 \models \sigma_2 \Downarrow \rho_3}{\rho_1 \models \sigma_2 \circ \sigma_1 \Downarrow \rho_3}$$

$$\frac{\rho_1 \models \sigma \Downarrow \rho_2 \qquad \rho_2 \models t \Downarrow v}{\rho_1 \models \sigma.t \Downarrow \rho_2.v}$$

The Algorithm – Defining Evaluation

We use a judgment so that syntactic substitutions produce new semantic environments.

$$\frac{}{\rho \models 1 \Downarrow \rho} \qquad \frac{}{\rho.v \models \uparrow^1 \Downarrow \rho} \qquad \frac{\rho_1 \models \sigma_1 \Downarrow \rho_2 \qquad \rho_2 \models \sigma_2 \Downarrow \rho_3}{\rho_1 \models \sigma_2 \circ \sigma_1 \Downarrow \rho_3}$$

$$\frac{\rho_1 \models \sigma \Downarrow \rho_2 \qquad \rho_2 \models t \Downarrow v}{\rho_1 \models \sigma.t \Downarrow \rho_2.v}$$

Using this, we can interpret $t\{\sigma\}$:

$$\frac{\rho \models \sigma \Downarrow \rho' \qquad \rho' \models t \Downarrow v}{\rho \models t\{\sigma\} \Downarrow v}$$

The Algorithm – Defining Quotation

In order to define $c \Vdash n \Uparrow t$ we need to define two other forms of quotation:

- $c \Vdash v \Uparrow T$ – quotation of semantic types.
- $c \Vdash e \Uparrow t$ – quotation of neutrals.

The Algorithm – Defining Quotation

Quotation for normals proceeds by casing on the type.

$$\frac{v \mathbin{@} \uparrow^{A} \mathbf{x}_c \rightsquigarrow b \qquad \rho.\mathbf{x}_c \models T \Downarrow B \qquad c+1 \Vdash \downarrow^{B} b \Uparrow t}{c \Vdash \downarrow^{\Pi\, A.\ T\{\rho\}} v \Uparrow \lambda t}$$

$$\frac{}{c \Vdash \downarrow^{\mathsf{Unit}} v \Uparrow \mathsf{tt}}$$

$$\frac{}{c \Vdash \downarrow^{\mathsf{Uni}} \mathsf{Unit} \Uparrow \mathsf{Unit}}$$

$$\frac{c \Vdash \downarrow^{\mathsf{Uni}} A \Uparrow T_1 \qquad \rho.\mathbf{x}_c \models T \Downarrow B \qquad c+1 \Vdash \downarrow^{\mathsf{Uni}} B \Uparrow T_2}{c \Vdash \downarrow^{\mathsf{Uni}} \Pi\, A.\ T\{\rho\} \Uparrow T_1 \to T_2}$$

$$\frac{c \Vdash e \Uparrow t}{c \Vdash \downarrow^{-} \uparrow^{-} e \Uparrow t}$$

The Algorithm – Defining Quotation

Quotation for neutrals proceeds by casing on the neutral itself.

$$\frac{}{c \Vdash \mathbf{x}_\ell \Uparrow \mathbf{x}_0\{\uparrow^{c-(\ell+1)}\}} \qquad \frac{c \Vdash e \Uparrow t_1 \qquad c \Vdash n \Uparrow t_2}{c \Vdash \mathsf{app}(e, n) \Uparrow t_1(t_2)}$$

The Algorithm – Defining Quotation

Quotation for neutrals proceeds by casing on the neutral itself.

$$\frac{}{c \Vdash \mathbf{x}_\ell \Uparrow \mathbf{x}_0\{\uparrow^{c-(\ell+1)}\}} \qquad \frac{c \Vdash e \Uparrow t_1 \qquad c \Vdash n \Uparrow t_2}{c \Vdash \mathsf{app}(e, n) \Uparrow t_1(t_2)}$$

Quotation for types likewise proceed by casing on the type.

$$\frac{}{c \Vdash \mathsf{Unit} \Uparrow \mathsf{Unit}} \qquad \frac{}{c \Vdash \mathsf{Uni} \Uparrow \mathcal{U}}$$

$$\frac{c \Vdash A \Uparrow T_1 \qquad \rho.\mathbf{x}_c \models T \Downarrow B \qquad c+1 \Vdash B \Uparrow T_2}{c \Vdash \Pi\, A.\ T\{\rho\} \Uparrow T_1 \to T_2}$$

$$\frac{c \Vdash e \Uparrow t}{c \Vdash \uparrow^{-} e \Uparrow t}$$

Final Step

1. ~~Evaluate a term to a value in some environment~~
2. ~~Quote a normal form back to a term in a context of length c.~~
3. Inject/reflect a term context into an environment.

Final Step

1. ~~Evaluate a term to a value in some environment~~
2. ~~Quote a normal form back to a term in a context of length c.~~
3. Inject/reflect a term context into an environment.

$$\frac{}{\uparrow () \rightsquigarrow \cdot} \qquad \frac{\uparrow\Gamma \rightsquigarrow \rho \qquad \rho \models T \Downarrow A}{\uparrow\Gamma.T \rightsquigarrow \rho.\uparrow^{A} \mathbf{x}_{|\Gamma|}}$$

Why is This Correct?

Now we have to prove some stuff.

1. $\Gamma \vdash t_1 \equiv t_2 : A \implies \mathsf{nf}(\Gamma, t_1, A) = \mathsf{nf}(\Gamma, t_2, A)$
2. If $\Gamma \vdash t : A$ then $\Gamma \vdash t \equiv \mathsf{nf}(\Gamma, t, A) : A$
3. If $\Gamma \vdash t : A$ then $\mathsf{nf}(\Gamma, t, A)$ is a normal form

Why is This Correct?

Now we have to prove some stuff.

1. $\Gamma \vdash t_1 \equiv t_2 : A \implies \mathsf{nf}(\Gamma, t_1, A) = \mathsf{nf}(\Gamma, t_2, A)$
2. If $\Gamma \vdash t : A$ then $\Gamma \vdash t \equiv \mathsf{nf}(\Gamma, t, A) : A$
3. ~~If $\Gamma \vdash t : A$ then $\mathsf{nf}(\Gamma, t, A)$ is a normal form~~
   Can now prove this by induction!

Completeness

$$\Gamma \vdash t_1 \equiv t_2 : A \implies \mathsf{nf}(\Gamma, t_1, A) = \mathsf{nf}(\Gamma, t_2, A)$$

Proof intuition: build a PER model!

- Each type A is associated with a PER of values: $[\![A]\!] = R$.
- Each PER satisfies the *neutral-normal yoga*

Completeness – Neutral-normal yoga

Fix two distinguished PERs:

$$\mathcal{N}f = \{(n_1, n_2) \mid \forall m.\ \exists t.\ m \Vdash n_1 \Uparrow t \wedge m \Vdash n_2 \Uparrow t\}$$
$$\mathcal{N}e = \{(e_1, e_2)\ \mid \forall m.\ \exists t.\ m \Vdash e_1 \Uparrow t \wedge m \Vdash e_2 \Uparrow t\}$$

For each $R = [\![A]\!]$ we require that R is sandwiched between these two PERs.

$$\{(\uparrow^A e_1, \uparrow^A e_2) \mid (e_1, e_2) \in \mathcal{N}e\}$$
$$\subseteq R \subseteq$$
$$\{(v_1, v_2) \mid (\downarrow^A v_1, \downarrow^A v_2) \in \mathcal{N}f\}$$

Completeness – The fundamental lemma

We can define a notion of related environments $\rho_1 = \rho_2 \in [\![\Gamma]\!]$.

1. If $\Gamma \vdash t_1 \equiv t_2 : T$ then for all $\rho_1 = \rho_2 \in [\![\Gamma]\!]$ the following holds.
   - $\rho_1 \models t_1 \Downarrow v_1$
   - $\rho_2 \models t_2 \Downarrow v_2$
   - $\rho_1 \models T \Downarrow A$
   - $[\![A]\!] = R$
   - $(v_1, v_2) \in R$
2. If $\Gamma \vdash T_1 \equiv T_2$ then for all $\rho_1 = \rho_2 \in [\![\Gamma]\!]$ the following holds.
   - $\rho_1 \models T_1 \Downarrow A_1$
   - $\rho_2 \models T_2 \Downarrow A_2$
   - $[\![A_1]\!] = [\![A_2]\!] = R$
   - $\forall m.\ \exists T.\ m \Vdash A_1 \Uparrow T \wedge m \Vdash A_2 \Uparrow T$

Completeness – explicit substitutions

Without explicit substitutions, the fundamental lemma is doomed: no β rules will hold!

Completeness – explicit substitutions

Without explicit substitutions, the fundamental lemma is doomed: no β rules will hold!
Let us suppose that $\rho \models u \Downarrow v_a$:

$$
\begin{aligned}
\rho \models (\lambda t)(u) \Downarrow v &\iff \\
(\lambda.\ t\{\rho\}) @ v_a \rightsquigarrow v &\iff \\
\rho.v_a \models t \Downarrow v &\iff \\
(\rho \models 1.u \Downarrow \rho.v_a) \wedge (\rho.v_a \models t \Downarrow v) &\iff \\
\rho \models t\{1.u\} \Downarrow v &
\end{aligned}
$$

With implicit substitutions this last step fails!

Completeness – explicit substitutions

Without explicit substitutions, the fundamental lemma is doomed: no β rules will hold!
Let us suppose that $\rho \models u \Downarrow v_a$:

$$\begin{aligned}
\rho \models (\lambda t)(u) \Downarrow v &\iff \\
(\lambda.\ t\{\rho\}) \mathbin{@} v_a \rightsquigarrow v &\iff \\
\rho.v_a \models t \Downarrow v &\iff \\
(\rho \models 1.u \Downarrow \rho.v_a) \wedge (\rho.v_a \models t \Downarrow v) &\iff \\
\rho \models t\{1.u\} \Downarrow v &
\end{aligned}$$

With implicit substitutions this last step fails!
I learned this Saturday afternoon. Whoops.

Completeness

the fundamental lemma + neutral-normal yoga = completeness

Soundness

To prove if $\Gamma \vdash t : A$ then $\Gamma \vdash t \equiv \mathsf{nf}(\Gamma, t, A) : A$ we construct a logical relation!

Soundness – the logical relation

We define some relation $\Gamma \models t : T \circledR v \in A$.

Soundness – the logical relation

We define some relation $\Gamma \models t : T \circledR v \in A$.

$$\Gamma \models t : T \circledR v \in A \implies \exists t'.\ \big(|\Gamma| \Vdash \downarrow^{A} v \Uparrow t'\big) \wedge \big(\Gamma \vdash t \equiv t' : T\big)$$

Soundness – the fundamental lemma

We can extend the logical relation to substitutions: $\Gamma \models \sigma : \Gamma \circledR \rho$.

- If $\Gamma \vdash t : T$
- for any σ and ρ such that $\Delta \models \sigma : \Gamma \circledR \rho$
- for any v and A such that $\rho \models t \Downarrow v$ and $\rho \models T \Downarrow A$

Soundness – the fundamental lemma

We can extend the logical relation to substitutions: $\Gamma \models \sigma : \Gamma \circledR \rho$.

- If $\Gamma \vdash t : T$
- for any σ and ρ such that $\Delta \models \sigma : \Gamma \circledR \rho$
- for any v and A such that $\rho \models t \Downarrow v$ and $\rho \models T \Downarrow A$
- $\Delta \models t\{\sigma\} : T\{\sigma\} \circledR v \in A$

Soundness – the fundamental lemma

We can extend the logical relation to substitutions: $\Gamma \models \sigma : \Gamma \circledR \rho$.

- If $\Gamma \vdash t : T$
- for any σ and ρ such that $\Delta \models \sigma : \Gamma \circledR \rho$
- for any v and A such that $\rho \models t \Downarrow v$ and $\rho \models T \Downarrow A$
- $\Delta \models t\{\sigma\} : T\{\sigma\} \circledR v \in A$

If this holds then $\Gamma \vdash t : T$ implies $\Gamma \vdash t \equiv \mathsf{nf}(\Gamma, t, T) : T$

Dependent Types Complicates Things

- Defining the PER model for completeness requires either induction-recursion or Allen-style spines.
- The logical-relation is well-founded only with respect to an ordering on semantic types.
- All type constructions must be done relationally to account for universes.
  e.g., $[\![A]\!]$ must be $[\![A = B]\!]$

Dependent Types Complicates Things

- Defining the PER model for completeness requires either induction-recursion or Allen-style spines.
- The logical-relation is well-founded only with respect to an ordering on semantic types.
- All type constructions must be done relationally to account for universes.
  e.g., $[\![A]\!]$ must be $[\![A = B]\!]$

Happy to discuss these issues offline.

Dependent Types Complicates Things

- Defining the PER model for completeness requires either induction-recursion or Allen-style spines.
- The logical-relation is well-founded only with respect to an ordering on semantic types.
- All type constructions must be done relationally to account for universes.
  e.g., $[\![A]\!]$ must be $[\![A = B]\!]$

Happy to discuss these issues offline.
Thanks.