Normalization by Evaluation for Martin-Löf Type Theory

Daniel Gratzer

October 1, 2018
Goal

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

1. \( \Gamma \vdash t_1 \equiv t_2 : A \implies \text{nf}(\Gamma, t_1, A) = \text{nf}(\Gamma, t_2, A) \)
2. If \( \Gamma \vdash t : A \) then \( \Gamma \vdash t \equiv \text{nf}(\Gamma, t, A) : A \)
3. If \( \Gamma \vdash t : A \) then \( \text{nf}(\Gamma, t, A) \) 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:
   \[
   \Gamma \vdash A \equiv B \quad \Gamma \vdash t : A
   \]
   \[
   \Gamma \vdash t : B
   \]
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.
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 \( X \neq \text{NbE} \)
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.

Not all equalities make sense as reduction rules!

\[^1\text{for } X \neq \text{NbE} \]
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.

Not all equalities make sense as reduction rules!
These proofs are extremely brittle!
Why Not X Instead?\footnote{for $X \neq \text{NbE}$}

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.

\begin{quote}
Not all equalities make sense as reduction rules!
These proofs are extremely brittle!
Entangles questions of reduction strategy!
\end{quote}
A Language

We need to specify the language that we’re going to normalize.
Our type theory is divided into various judgments:

\[ \Gamma \vdash \Gamma \text{ is a valid context} \]
\[ \Gamma \vdash T \quad \text{In context } \Gamma, \ T \text{ is a type} \]
\[ \Gamma \vdash t : T \quad \text{In context } \Gamma, \ t \text{ has type } T \]
The Main Judgments

Our type theory is divided into various judgments:

\[ \Gamma \vdash \Gamma \text{ is a valid context} \]
\[ \Gamma \vdash T \quad \text{In context } \Gamma, T \text{ is a type} \]
\[ \Gamma \vdash t : T \quad \text{In context } \Gamma, t \text{ has type } T \]

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:

$\Gamma \vdash\cdot : ()$  
$\Gamma \vdash 1 : \Gamma$  
$\Gamma.T \vdash \uparrow^1 : \Gamma$

$\Gamma \vdash \sigma_2 \circ \sigma_1 : \Xi$

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

$\Gamma \vdash \sigma.t : \Delta.T$

Crucial rule:

$\Gamma \vdash t : T$  
$\Delta \vdash \sigma : \Gamma$  
$\Delta \vdash t\{\sigma\} : T\{\sigma\}$
A Language

The rules for types and contexts:

\[
\begin{align*}
() & \vdash \\
\Gamma \vdash A & \quad \Gamma \vdash A \\
\Gamma \vdash \Gamma.A & \vdash \\
\Gamma \vdash A & \quad \Gamma.A \vdash B \\
\Gamma \vdash A \rightarrow B & \\
\Gamma \vdash \text{Unit} & \quad \Gamma \vdash \mathcal{U} \\
\Gamma \vdash A : \mathcal{U} & \quad \Gamma \vdash A
\end{align*}
\]
A Language

The rules for terms:

\[
\Gamma \vdash \text{Unit} : \mathcal{U} \quad \Gamma \vdash \text{tt} : \text{Unit} \\
\Gamma \vdash A : \mathcal{U} \quad \Gamma.A \vdash B : \mathcal{U} \\
\Gamma \vdash A \quad \Gamma.A \vdash t : B \\
\Gamma \vdash \lambda t : A \rightarrow B \\
\Gamma \vdash t : A \rightarrow B \quad \Gamma \vdash u : A \\
\Gamma \vdash t(u) : B\{1.u\} \\
\Gamma_1.T.\Gamma_2 \vdash |\Gamma_2| = k \\
\Gamma_1.T.\Gamma_2 \vdash x_k : T\{^{k+1}\} 
\]
We need the *conversion rule* for any sort of type theory.

\[
\frac{\Gamma \vdash t : A \quad \Gamma \vdash A \equiv B}{\Gamma \vdash t : B}
\]

Dependence means term equality matters for type equality.

\[
\frac{\Gamma \vdash A \equiv B : U}{\Gamma \vdash A \equiv B}
\]
The Wrinkle – The Main Equality Rules

\[
\frac{\Gamma \vdash u : A \quad \Gamma.A \vdash t : B}{\Gamma \vdash (\lambda t)(u) \equiv t\{1.u\} : B\{1.u\}}
\]

\[
\frac{\Gamma \vdash t : A \rightarrow B}{\Gamma \vdash \lambda(t\{↑1\}(x_0)) \equiv t : A \rightarrow B}
\]

\[
\frac{\Gamma \vdash t : \text{Unit}}{\Gamma \vdash t \equiv \text{tt} : \text{Unit}}
\]
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 $\beta$-normal and $\eta$-long forms.

\[
\begin{align*}
\Gamma \vdash x_n : A & \quad \Gamma \vdash^{\text{neu}} e : A \rightarrow B & \quad \Gamma \vdash^{\text{nf}} v : A \\
\Gamma \vdash^{\text{neu}} x_n : A & \quad \Gamma \vdash^{\text{neu}} e(v) : B \{1.v\}
\end{align*}
\]
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 $\beta$-normal and $\eta$-long forms.

\[
\begin{align*}
\Gamma \vdash & x_n : A \\
\Gamma & \vdash^{\text{neu}} x_n : A \\
\Gamma \vdash & e : A \rightarrow B \\
\Gamma & \vdash^{\text{neu}} e(v) : B\{1.v\} \\
\Gamma & \vdash \text{tt} : \text{Unit} \\
\Gamma & \vdash^{\text{nf}} \text{Unit} : \mathcal{U} \\
\Gamma & \vdash^{\text{nf}} A : \mathcal{U} \\
\Gamma.A & \vdash^{\text{nf}} B : \mathcal{U} \\
\Gamma & \vdash^{\text{nf}} A \rightarrow B : \mathcal{U} \\
\Gamma & \vdash^{\text{nf}} e : \mathcal{U} \\
\Gamma & \vdash^{\text{nf}} e : \mathcal{U}
\end{align*}
\]
Now we have a goal, construct $\Gamma \vdash_{\text{nf}} \text{nf}(\Gamma, t, A) : A$ 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:

\[
\begin{align*}
\text{eval} &: \text{(Term t)} \rightarrow t \\
\text{quote} &: t \rightarrow \text{(Term t)}
\end{align*}
\]

\[
\text{normalize} = \text{quote} \cdot \text{eval}
\]

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

\[ \text{eval : Term} \to D \quad \text{quote : } D \to \text{Term} \]
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 extrinsic 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 extrinsic 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.
Neutral elements represent computations which are stuck on some variable.

\[ e ::= x^\ell \mid \text{app}(e, \downarrow^A v) \]

N.B. The argument to \text{app}(e, \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_) must be fully evaluated and annotated.
What happens when we go under a binder?

\[ f ::= t \{ \rho \} \]

\( \rho \) is the environment we're interpreting \( t \). This removes the need for domains, is called defunctionalization.
What happens when we go under a binder?

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

\[ f ::= t\{\rho} \]

\(\rho\) 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 $\eta$-long forms for dependent type theory. We settle for isolating $\beta$-normal forms for now.

\[ v, A ::= \lambda. f \mid \text{tt} \mid \text{Unit} \mid \text{Uni} \mid \Pi A. F \]
It’s difficult to isolate $\eta$-long forms for dependent type theory. We settle for isolating $\beta$-normal forms for now.

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

Need to include neutrals with type information to allow $\eta$-expansions later.
A Syntactic Semantic Domain

\[ \begin{align*}
\nu, A & ::= \lambda f \mid \text{tt} \mid \text{Unit} \mid \text{Uni} \mid \Pi A_1. F \mid \uparrow^A e \\
\nu, A & ::= t\{\rho}\ \\
\nu & ::= x_\ell \mid \text{app}(e, v) \\
\nu & ::= \downarrow^A v \\
\rho & ::= \cdot \mid \rho.v
\end{align*} \]
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 $\eta$-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.*
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 $\eta$-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 $\eta$-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

\[ nf(\Gamma, t, T) = t' \iff \]
\[ \uparrow\Gamma \sim\rho \land \]
\[ (\rho \models t \downarrow v) \land (\rho \models T \downarrow A) \land \]
\[ |\Gamma| \vdash A v \uparrow t' \]

The relational presentation is ideal for a constructive setting.
The evaluation judgment is defined by inspection on $t$.

\[
\begin{align*}
\rho \cdot v & \models x_0 \downarrow v \\
\rho & \models tt \downarrow tt \\
\rho & \models \text{Unit} \downarrow \text{Unit} \\
\rho & \models U \downarrow \text{Uni} \\
\rho & \models \lambda t \downarrow \lambda. \ t\{\rho\} \\
\rho & \models T_1 \rightarrow T_2 \downarrow \Pi A. \ T_2\{\rho\}
\end{align*}
\]
The Algorithm – Defining Evaluation

The evaluation judgment is defined by inspection on $t$.

$$\rho.v \models x_0 \downarrow v$$
$$\rho \models tt \downarrow tt$$
$$\rho \models \text{Unit} \downarrow \text{Unit}$$

$$\rho \models U \downarrow \text{Uni}$$
$$\rho \models \lambda t \downarrow \lambda. t\{\rho\}$$
$$\rho \models T_1 \downarrow A$$
$$\rho \models T_1 \rightarrow T_2 \downarrow \Pi A. T_2\{\rho\}$$

What about the only construct in our language that computes?
The Algorithm – Defining Evaluation

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

$$
\begin{align*}
\rho.a & \models t \downarrow v \\
\lambda. \ t\{\rho\} \ @ \ a & \rightsquigarrow v
\end{align*}
$$

$$
\begin{align*}
\rho.a & \models T \downarrow B \\
\Pi A. \ T\{\rho\} \ e \ @ \ a & \rightsquigarrow B \ \text{app}(e, \downarrow^A \ a)
\end{align*}
$$

$$
\begin{align*}
\rho & \models t \downarrow v_1 \\
\rho & \models u \downarrow v_2 \\
v_1 \ @ \ v_2 & \rightsquigarrow v
\end{align*}
$$

$$
\begin{align*}
\rho & \models t(u) \downarrow v
\end{align*}
$$

Rule of thumb:

each eliminator gets an auxiliary judgment to either perform $\beta$-reduction or construct a new neutral.
We use a judgment so that syntactic substitutions produce new semantic environments.

\[
\begin{align*}
\rho & \models 1 \downarrow \rho \\
\rho.v & \models \uparrow^1 \downarrow \rho \\
\rho_1 \models \sigma_1 \downarrow \rho_2 & \quad \rho_2 \models \sigma_2 \downarrow \rho_3 \\
\rho_1 \models \sigma \downarrow \rho_2 & \quad \rho_2 \models t \downarrow v \\
\rho_1 \models \sigma.t \downarrow \rho_2.v
\end{align*}
\]
The Algorithm – Defining Evaluation

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

\[
\begin{align*}
\rho & \vdash 1 \downarrow \rho \\
\rho.v & \vdash \uparrow^1 \downarrow \rho \\
\end{align*}
\]

\[
\begin{align*}
\rho_1 & \vdash \sigma_1 \downarrow \rho_2 & \rho_2 & \vdash \sigma_2 \downarrow \rho_3 \\
\rho_1 & \vdash \sigma_2 \circ \sigma_1 \downarrow \rho_3 \\
\end{align*}
\]

\[
\begin{align*}
\rho_1 & \vdash \sigma \downarrow \rho_2 & \rho_2 & \vdash t \downarrow v \\
\rho_1 & \vdash \sigma.t \downarrow \rho_2.v \\
\end{align*}
\]

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

\[
\begin{align*}
\rho & \vdash \sigma \downarrow \rho' & \rho' & \vdash t \downarrow v \\
\rho & \vdash t\{\sigma\} \downarrow v \\
\end{align*}
\]
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.

\[
v @ \uparrow^A x_c \leadsto b \quad \rho.x_c \models T \Downarrow B \quad c + 1 \models \downarrow^B b \uparrow t
\]

\[
c \models \downarrow \Pi A. \ T\{\rho\} \ v \uparrow \lambda t
\]

\[
c \models \downarrow^{\text{Unit}} v \uparrow \text{tt}
\]

\[
c \models \downarrow^{\text{Unit}} \text{Unit} \uparrow \text{Unit}
\]

\[
c \models \downarrow^{\text{Unit}} A \uparrow T_1 \quad \rho.x_c \models T \Downarrow B \quad c + 1 \models \downarrow^{\text{Unit}} B \uparrow T_2
\]

\[
c \models \downarrow^{\text{Unit}} \Pi A. \ T\{\rho\} \uparrow T_1 \rightarrow T_2
\]

\[
c \models e \uparrow t
\]

\[
c \models \downarrow^- \uparrow^- e \uparrow t
\]
The Algorithm – Defining Quotation

Quotation for neutrals proceeds by casing on the neutral itself.

\[
\frac{c \vdash x \uparrow x_0 \{ \uparrow^{c-(\ell+1)} \}}{c \vdash x_\ell \uparrow x_0 \{ \uparrow^{c-(\ell+1)} \}}
\]

\[
\frac{c \vdash e \uparrow t_1 \hspace{1cm} c \vdash n \uparrow t_2}{c \vdash \text{app}(e, n) \uparrow t_1(t_2)}
\]
The Algorithm – Defining Quotation

Quotation for **neutrals** proceeds by casing on the neutral itself.

\[ c \vdash x_\ell \uparrow x_0 \{ \uparrow^{c-(\ell+1)} \} \quad \text{if} \quad c \vdash e \uparrow t_1 \quad \text{and} \quad c \vdash n \uparrow t_2 \]

\[ c \vdash \text{app}(e, n) \uparrow t_1(t_2) \]

Quotation for **types** likewise proceed by casing on the type.

\[ c \vdash \text{Unit} \uparrow \text{Unit} \]

\[ c \vdash \text{Uni} \uparrow \mathcal{U} \]

\[ c \vdash A \uparrow T_1 \quad \rho.x_c \models T \downarrow B \quad c + 1 \vdash B \uparrow T_2 \]

\[ c \vdash \Pi A. \ T\{\rho\} \uparrow T_1 \rightarrow T_2 \]

\[ c \vdash e \uparrow t \]

\[ c \vdash \uparrow^* e \uparrow t \]
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.

\[
\begin{align*}
\uparrow() & \rightsquigarrow \cdot \\
\uparrow\Gamma \rightsquigarrow \rho & \quad \rho \models T \downarrow A
\end{align*}
\]

\[
\begin{align*}
\uparrow\Gamma . T & \rightsquigarrow \rho . \uparrow^A x |\Gamma|
\end{align*}
\]
Now we have to prove some stuff.

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

Now we have to prove some stuff.

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

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

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:

\[
\mathcal{N} f = \{ (n_1, n_2) \mid \forall m. \exists t. m \models n_1 \uparrow t \land m \not\models n_2 \uparrow t \} \\
\mathcal{N} e = \{ (e_1, e_2) \mid \forall m. \exists t. m \models e_1 \uparrow t \land m \not\models e_2 \uparrow t \} 
\]

For each \( R = \llbracket A \rrbracket \) 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.
   - $\rho_1 \models T_1 \downarrow A_1$
   - $\rho_2 \models T_2 \downarrow A_2$
   - $[A_1] = [A_2] = R$
   - $\forall m. \exists T. \ m \models A_1 \uparrow T \land m \models A_2 \uparrow T$
Without explicit substitutions, the fundamental lemma is doomed: no $\beta$ rules will hold!
Completeness – explicit substitutions

Without explicit substitutions, the fundamental lemma is doomed: no \( \beta \) rules will hold!
Let us suppose that \( \rho \vdash u \downarrow v_a \):

\[
\rho \vdash (\lambda t)(u) \downarrow v \iff \\
(\lambda. t\{\rho\}) @ v_a \rightsquigarrow v \iff \\
\rho.v_a \vdash t \downarrow v \iff \\
(\rho \vdash 1.u \downarrow \rho.v_a) \land (\rho.v_a \vdash t \downarrow v) \iff \\
\rho \vdash t\{1.u\} \downarrow v
\]

With implicit substitutions this last step fails!
Completeness – explicit substitutions

Without explicit substitutions, the fundamental lemma is doomed: no $\beta$ rules will hold!

Let us suppose that $\rho \vdash u \Downarrow v_a$:

$$\rho \vdash (\lambda t)(u) \Downarrow v \iff (\lambda. t\{\rho\}) \circ v_a \rightsquigarrow v \iff \rho.v_a \vdash t \Downarrow v \iff (\rho \vdash 1.u \Downarrow \rho.v_a) \land (\rho.v_a \vdash t \Downarrow v) \iff \rho \vdash t\{1.u\} \Downarrow v$$

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 $\Gamma \vdash t \equiv \text{nf}(\Gamma, t, A) : A$ we construct a logical relation!
Soundness – the logical relation

We define some relation $\Gamma \models t : T \circledast v \in A$. 
Soundness – the logical relation

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

$\Gamma \models t : T \supseteq v \in A \implies \exists t'. (|\Gamma| \vdash A v \uparrow t') \land (\Gamma \vdash t \equiv t' : T)$
We can extend the logical relation to substitutions: \( \Gamma \models \sigma : \Gamma \circ \rho \).

- If \( \Gamma \vdash t : T \)
- for any \( \sigma \) and \( \rho \) such that \( \Delta \models \sigma : \Gamma \circ \rho \)
- 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: $\Gamma \models \sigma : \Gamma \odot \rho$.

- If $\Gamma \vdash t : T$
- for any $\sigma$ and $\rho$ such that $\Delta \models \sigma : \Gamma \odot \rho$
- for any $v$ and $A$ such that $\rho \models t \downarrow v$ and $\rho \models T \downarrow A$
- $\Delta \models t\{\sigma\} : T\{\sigma\} \odot v \in A$
Soundness – the fundamental lemma

We can extend the logical relation to substitutions: \( \Gamma \models \sigma : \Gamma \otimes \rho \).

- If \( \Gamma \vdash t : T \)
- for any \( \sigma \) and \( \rho \) such that \( \Delta \models \sigma : \Gamma \otimes \rho \)
- for any \( v \) and \( A \) such that \( \rho \models t \downarrow v \) and \( \rho \models T \downarrow A \)
- \( \Delta \models t\{\sigma\} : T\{\sigma\} \otimes v \in A \)

If this holds then \( \Gamma \vdash t : T \) implies \( \Gamma \vdash t \equiv \text{nf}(\Gamma, t, T) : T \)
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., $\llbracket A \rrbracket$ must be $\llbracket A = B \rrbracket$
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., $\lbrack A \rbrack$ must be $\lbrack A = B \rbrack$

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.