Towards Normalization by Evaluation for the Calculus of Constructions

Andreas Abel

Department of Computer Science
Ludwig-Maximilians-University Munich

Functional and Logic Programming Symposium
FLOPS 2010, Sendai, Japan
19-21 April 2010

Normalization by Evaluation for Type Theory

- Efficient algorithm for deciding definitional equality.
  - Leroy/Gregoire (ICFP 2002): Compile Coq to byte code.
  - Boespflug (PADL 2010): Compile (via Haskell) to machine code.
- Extend type theory by η in a systematic way.
- "Semantic" normalization: beneficial for meta theory.
- This work:
  1. Basis: Untyped Normalization by Evaluation.
  2. Extended by η-expansion for Π-types.
  3. Model to show termination and completeness of normalization.
- Future work: Show soundness and type preservation.

The Calculus of Constructions

- Full PTS (pure type system) over sorts $s ::= * \mid \Box$.

$$\frac{}{\Gamma \vdash * : \Box} \qquad \frac{\Gamma, x : U \vdash T : s}{\Gamma \vdash \Pi\, U\, (\lambda x T) : s}$$

$$\frac{\Gamma, x : U \vdash t : T}{\Gamma \vdash \lambda x : U.\, t : T} \qquad \frac{\Gamma \vdash t : \Pi\, U\, T \qquad \Gamma \vdash u : U}{\Gamma \vdash t\, u : T\, u}$$

$$\frac{\Gamma \vdash t : T \qquad \text{“}T = U\text{”}}{\Gamma \vdash t : U}$$

- Write $(x : U) \to T$ for $\Pi\, U\, \lambda x T$.

Features of the CoC

- Dependency: $P : A \to *$ for $A : *$.
- Impredicativity: $\exists\, A\, P := (X : *) \to ((x : A) \to P\, x \to X) \to X$.
- Scale to large eliminations: $T : \mathsf{Nat} \to *$.

$$\begin{array}{lcl} T\,0 & = & \mathsf{Bool} \\ T\,(n+1) & = & \mathsf{Bool} \to T\,n \end{array}$$

- Scale to η.

Eta laws

- Function type.

$$\frac{\Gamma \vdash t : U \to T}{\Gamma \vdash t = \lambda x{:}U.\, t\, x : U \to T}\; x \notin \mathsf{FV}(t)$$

- Unit type.

$$\frac{\Gamma \vdash t : 1}{\Gamma \vdash t = () : 1}$$

- Identity type: proof irrelevance.

$$\frac{\Gamma \vdash p, q : \mathsf{Id}_T\, t\, t'}{\Gamma \vdash p = q : \mathsf{Id}_T\, t\, t'} \qquad \frac{\Gamma \vdash p : \mathsf{Id}_T\, t\, t}{\Gamma \vdash p = \mathsf{refl} : \mathsf{Id}_T\, t\, t}$$

Type conversion rule

1. Untyped conversion.

$$\frac{\Gamma \vdash t : T \qquad \Gamma \vdash T' : s}{\Gamma \vdash t : T'} \; T =_{\beta(\eta)} T'$$

   - Requires confluence of $\beta(\eta)$-reduction.
   - Does not extend to η for the unit type.

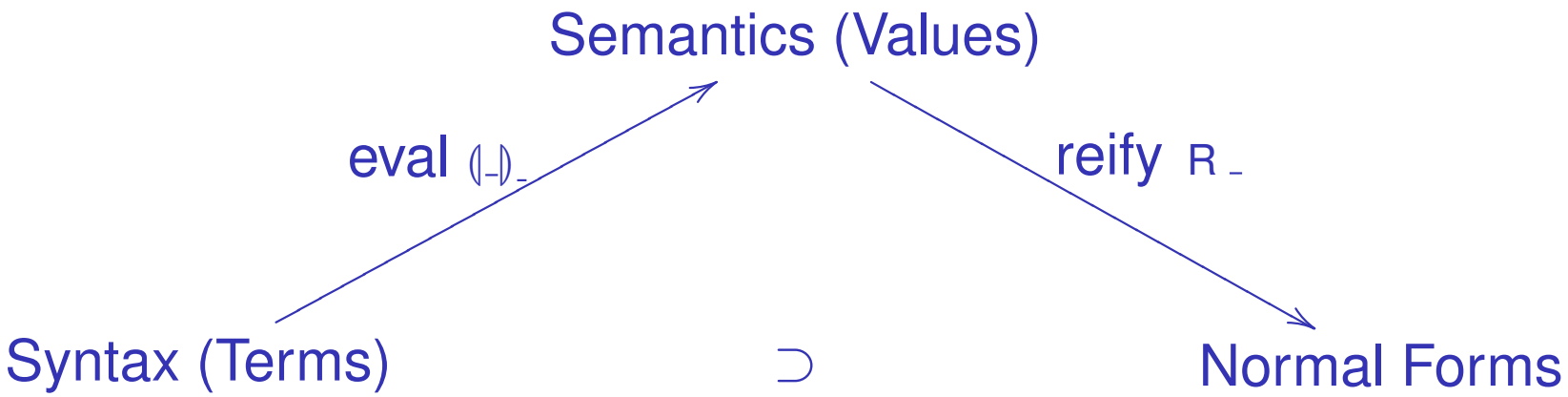
2. Typed conversion, *judgemental equality*.

$$\frac{\Gamma \vdash t : T \qquad \Gamma \vdash T = T' : s}{\Gamma \vdash t : T'}$$

   - Supports all η-laws.
   - Simplifies model construction.
   - Difficult: Injectivity of function types.

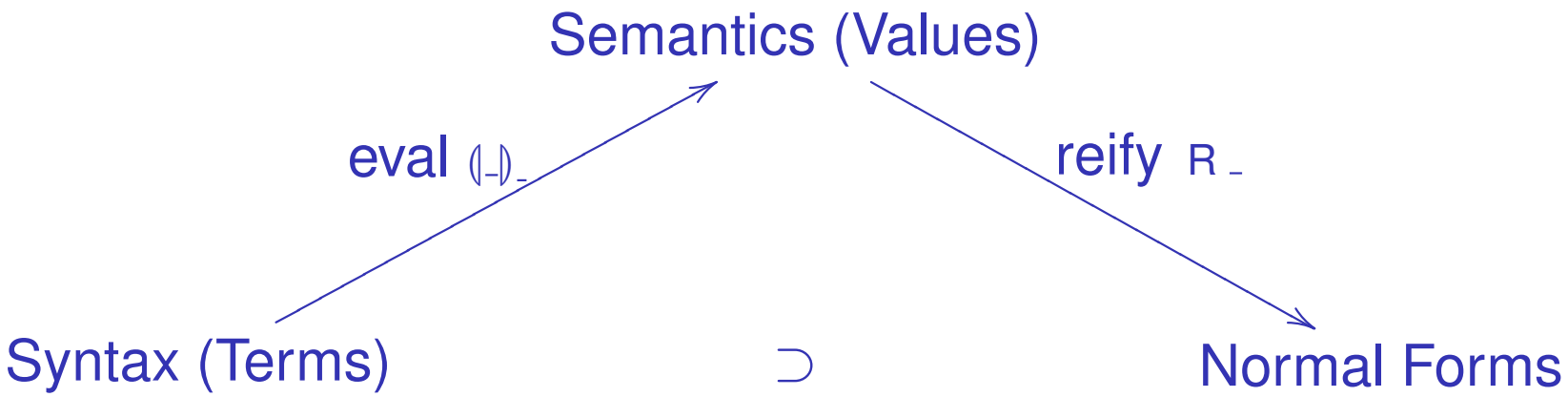
$$\Gamma \vdash \Pi\, U\, T = \Pi\, U'\, T' : s \quad \Longrightarrow \quad \Gamma \vdash U = U' : s' \quad \text{and} \quad \Gamma \vdash T = T' : U \to s$$

What is Normalization By Evaluation?



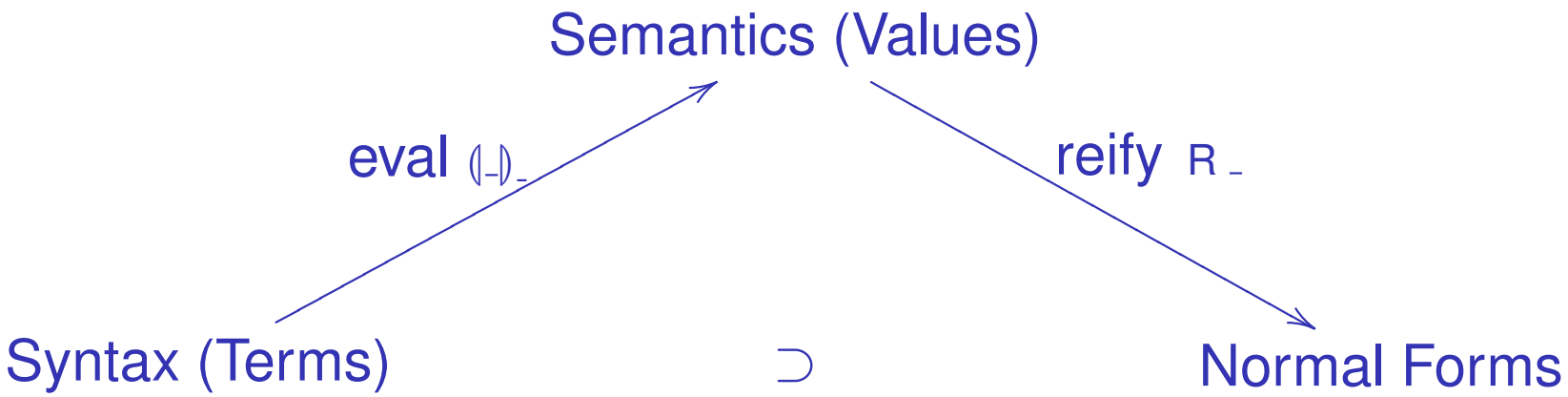
- You have: an interpreter ((|_|)_).
- You buy: a reifyer (R _).
- You get for free: a *full normalizer*!
- Famous instance: Leroy/Gregoire's compiled reduction.

What is Normalization By Evaluation?



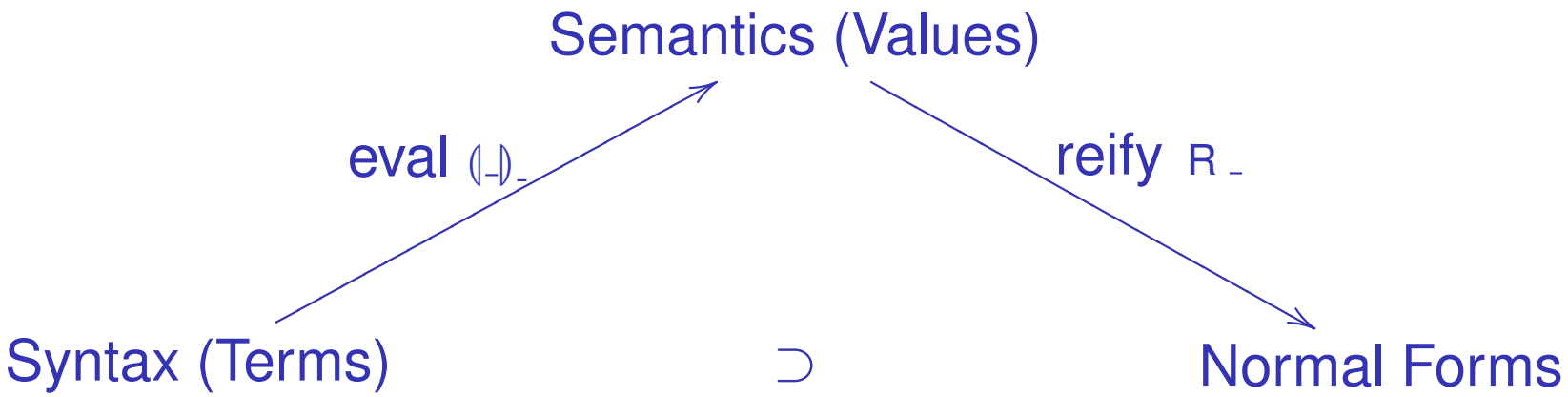
- You have: an interpreter ((|_|)_).
- You buy: a reifyer (R _).
- You get for free: a *full normalizer*!
- Famous instance: Leroy/Gregoire's compiled reduction.

What is Normalization By Evaluation?



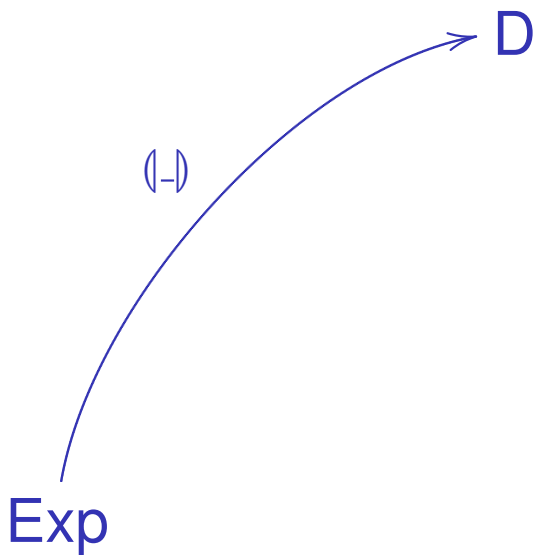
- You have: an interpreter ((|_|)_).
- You buy: a reifyer (R _).
- You get for free: a *full normalizer*!
- Famous instance: Leroy/Gregoire's compiled reduction.

What is Normalization By Evaluation?



- You have: an interpreter ((|_|)_).
- You buy: a reifyer (R _).
- You get for free: a *full normalizer*!
- Famous instance: Leroy/Gregoire's compiled reduction.

Interpretation



Interpretation in Applicative Structure

- Set of values D.
- Application operation $\_ \cdot \_ : \mathsf{D} \times \mathsf{D} \to \mathsf{D}$.
- Interpretation $(\!|t|\!)_\eta \in \mathsf{D}$ for term t and environment η:

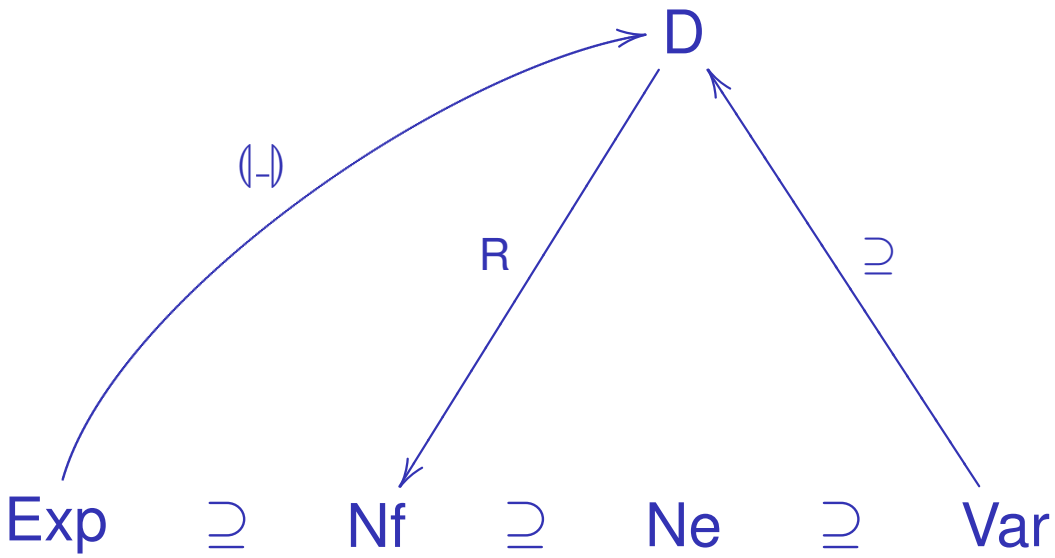
$$\begin{array}{rcl} (\!|x|\!)_\eta & = & \eta(x) \\ (\!|r\ s|\!)_\eta & = & (\!|r|\!)_\eta \cdot (\!|s|\!)_\eta \\ (\!|\lambda x t|\!)_\eta \cdot d & = & (\!|t|\!)_{\eta[x \mapsto d]} \end{array}$$

- Example: Scott domain $\mathsf{D} = \mathsf{Abs}\,[\mathsf{D} \to \mathsf{D}]$.

$$\begin{array}{rcl} (\!|\lambda x t|\!)_\eta & = & \mathsf{Abs}\,(d \mapsto (\!|t|\!)_{\eta[x \mapsto d]}) \\ (\mathsf{Abs}\,f) \cdot d & = & f(d) \end{array}$$

Untyped Normalization-by-Evaluation

Var x_k
Ne $u ::= x_k\ \vec{v}$
Nf $v ::= u \mid \lambda x_k v$



Untyped NbE

- Extend D by neutral values $x_k\,\vec{d}$.
- Application:

$$\begin{aligned}(\!|\lambda x t|\!)_\eta \cdot d &= (\!|t|\!)_{\eta[x\mapsto d]}\\ (x_k\,\vec{d}) \cdot d &= x_k\,(\vec{d}, d)\end{aligned}$$

- Reification (avoiding freshness problems!):

$$\begin{aligned}\mathrm{R}_n\,(\!|\lambda x t|\!)_\eta &= \lambda x_n.\, R_{n+1}((\!|\lambda x t|\!)_\eta \cdot x_n)\\ \mathrm{R}_n\,(x_k\,\vec{d}) &= x_k\,R_n(\vec{d})\end{aligned}$$

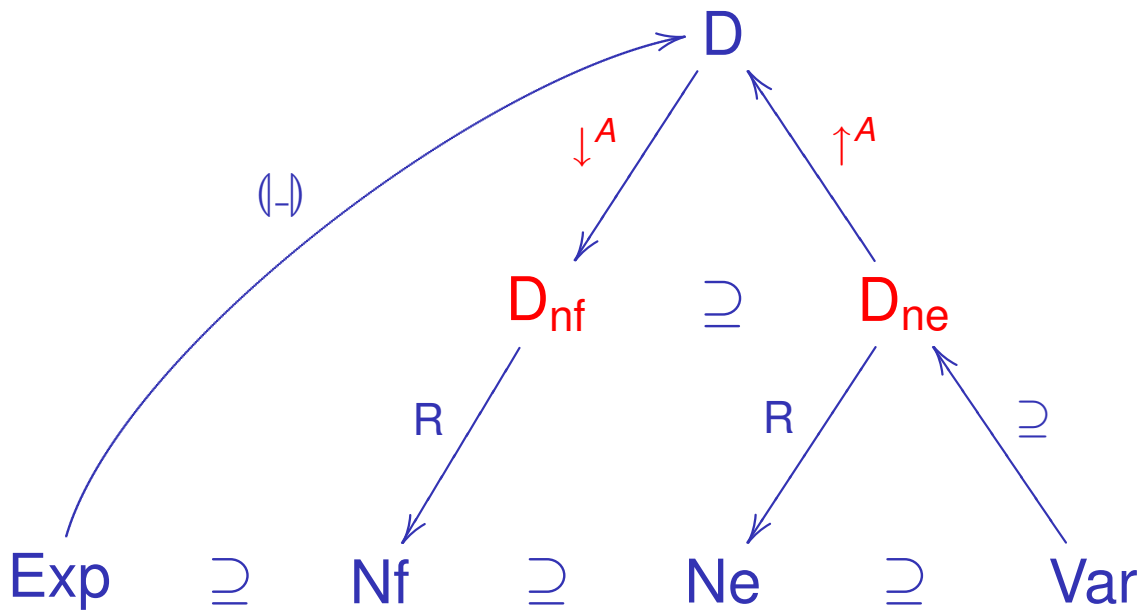
- Normal form: $\mathrm{R}_1\,(\!|t|\!)$.

Example Normalization

Let $I = \lambda y\, y$ identity.

$$
\begin{array}{lll}
 & \mathrm{R}_1(\!|\lambda x.\, I\, x\, I|\!) & = \lambda x_1.\, \mathrm{R}_2(x_1 \cdot (\!|I|\!)) \\
= & \lambda x_1.\, \mathrm{R}_2((\!|\lambda x.\, I\, x\, I|\!) \cdot x_1) & = \lambda x_1.\, x_1\, (\mathrm{R}_2(\!|I|\!)) \\
= & \lambda x_1.\, \mathrm{R}_2((\!|I\, x\, I|\!)_{x \mapsto x_1}) & = \lambda x_1.\, x_1\, (\lambda x_2.\, \mathrm{R}_3((\!|\lambda y y|\!) \cdot x_2)) \\
= & \lambda x_1.\, \mathrm{R}_2((\!|\lambda y y|\!) \cdot (\!|x|\!)_{x \mapsto x_1} \cdot (\!|I|\!)) & = \lambda x_1.\, x_1\, (\lambda x_2.\, \mathrm{R}_3(\!|y|\!)_{y \mapsto x_2}) \\
= & \lambda x_1.\, \mathrm{R}_2((\!|y|\!)_{y \mapsto (\!|x|\!)_{x \mapsto x_1}} \cdot (\!|I|\!)) & = \lambda x_1.\, x_1\, (\lambda x_2.\, \mathrm{R}_3\, x_2) \\
= & \lambda x_1.\, \mathrm{R}_2((\!|x|\!)_{x \mapsto x_1} \cdot (\!|I|\!)) & = \lambda x_1.\, x_1\, (\lambda x_2.\, x_2)
\end{array}
$$

η-Expansion in the Semantics



η-Expansion for Simple Types

- $\uparrow^A$: Lazy η-expansion of neutrals at type A.
- $\downarrow^A$: Marker for η-expansion during reification.

Simple types

$$
\begin{array}{lcl}
(\uparrow^{A\to B} u)\cdot a & = & \uparrow^B (u\,(\downarrow^A a)) \\
\mathrm{R}_n(\downarrow^{A\to B} f) & = & \lambda x_n.\ \mathrm{R}_{n+1}\downarrow^B (f\cdot\uparrow^A x_n) \\
\mathrm{R}_n\downarrow^o\uparrow^o(x_k\,\vec{d}) & = & x_k\ \mathrm{R}_n\,\vec{d}
\end{array}
$$

η-Expansion for Dependent Types

- $\uparrow^A$: Lazy η-expansion of neutrals at type A.
- $\downarrow^A$: Marker for η-expansion during reification.

Dependent types

$$
\begin{array}{lcl}
(\uparrow^{\Pi\, A\, F}\, u) \cdot a & = & \uparrow^{F \cdot a}(u\,(\downarrow^A a)) \\
\mathrm{R}_n(\downarrow^{\Pi\, A\, F}\, f) & = & \lambda x_n.\ \mathrm{R}_{n+1}\downarrow^{(F \cdot \uparrow^A x_n)}(f \cdot \uparrow^A x_n) \\
\mathrm{R}_n \downarrow^o \uparrow^o (x_k\, \vec{d}) & = & x_k\ \mathrm{R}_n\, \vec{d}
\end{array}
$$

Normalization algorithm

- $\mathsf{nbe}^T\, t = \mathsf{R}_1 \downarrow^{(\!|T|\!)} (\!|t|\!)$
- Soundness: If $\vdash t : T$ then $\vdash t = \mathsf{nbe}^T\, t : T$.
- Completeness: If $\vdash t = t' : T$ then $\mathsf{nbe}^T\, t =_\alpha \mathsf{nbe}^T\, t'$.
- Show completeness using a PER model:
  1. If $\vdash t = t' : T$ then $((\!|t|\!), (\!|t'|\!)) \in [\![T]\!]$.
  2. If $\vdash T : *$ then $(\!|T|\!) \Vdash [\![T]\!]$.
  3. If $(a, a') \in \mathcal{A}$ and $A \Vdash \mathcal{A}$ then $\mathsf{R}_n \downarrow^A a =_\alpha \mathsf{R}_n \downarrow^A a'$.

Interpretation of Types

- A type T is interpreted by a pair $(A, \mathcal{A})$.
  1. $A \in \mathsf{D}$
  2. $\mathcal{A} \subseteq \mathsf{D} \times \mathsf{D}$ is a partial equivalence
  3. $A \Vdash \mathcal{A}$, meaning:
     1. $(\uparrow^A u, \uparrow^A u') \in \mathcal{A}$ for all u, u' with $\forall n.\ \mathsf{R}_n\, u =_\alpha \mathsf{R}_n\, u' \in \mathsf{Ne}$
     2. if $(d, d') \in \mathcal{A}$ then $\forall n.\ \mathsf{R}_n \downarrow^A d =_\alpha \mathsf{R}_n \downarrow^A d' \in \mathsf{Nf}$ for all n.
- Equality of types $(A, \mathcal{A}) = (A', \mathcal{A}')$ holds if
  1. $\mathcal{A} = \mathcal{A}'$
  2. $\downarrow^* A = \downarrow^* A' \in \mathsf{Nf}$
  3. $\uparrow^A = \uparrow^{A'}$ and $\downarrow^A = \downarrow^{A'}$

Semantic Π

- Definition:

$$\Pi\,\mathcal{A}\,\mathcal{F} = \{(f, f') \mid \forall (a, a') \in \mathcal{A},\ (f \cdot a, f' \cdot a') \in \mathcal{F}(a)\}$$

- Realizability:

$$\begin{array}{lrcll} \text{if} & A & \Vdash & \mathcal{A} & \\ \text{and} & F \cdot a & \Vdash & \mathcal{F}(a) & \text{for all } a \in \mathcal{A} \\ \text{then} & \Pi\,A\,F & \Vdash & \Pi\,\mathcal{A}\,\mathcal{F} & \end{array}$$

Conclusions

- Principled $\beta\eta$-normalization for CoC.
- Avoiding freshness issues during reification.
- Partial formalization in Coq.
- Can be extended to Calculus of Inductive Constructions!?
- Next: show soundness and Π-injectivity.