A Formalisation of a Dependently Typed Language as an Inductive-Recursive Family

Nils Anders Danielsson

Chalmers

January 31, 2007

Introduction

- Abstract syntax data type for dependently typed language.
- No raw terms.
- Full normalisation (NBE).
- Equality checker.
- Type checker.
- Structurally recursive.

Meta language
Object language
Contexts
Types
Terms
Variables
Substitutions
Equality
Why ...?
Implicit substitutions
Normalisation
Normal forms
Values
Environments
Evaluation
reify + reflect
Normalisation
Equality
Type checker

Meta language

- AgdaLight (Ulf Norell).
- Inductive-recursive families, implicit arguments.
- "Epigram with Haskell-like syntax."

Meta language
Object language
Contexts
Types
Terms
Variables
Substitutions
Equality
Why ...?
Implicit substitutions
Normalisation
Normal forms
Values
Environments
Evaluation
reify + reflect
Normalisation
Equality
Type checker

Object language

- Variant of Martin-Löf's logical framework.
- Explicit substitutions.
- de Bruijn indices.
- Almost all definitions are mutually recursive.

Meta language
Object language
Contexts
Types
Terms
Variables
Substitutions
Equality
Why ...?
Implicit substitutions
Normalisation
Normal forms
Values
Environments
Evaluation
reify + reflect
Normalisation
Equality
Type checker

Contexts

data *Ctxt* : *Set* **where**
  $\varepsilon \;\; : \mathit{Ctxt}$
  $(\triangleright) : (\Gamma : \mathit{Ctxt}) \to \mathit{Ty}\ \Gamma \to \mathit{Ctxt}$

$$\frac{}{\varepsilon\ \mathit{context}} \qquad \frac{\Gamma\ \mathit{context} \qquad \Gamma \vdash \tau\ \mathit{type}}{\Gamma \triangleright \tau\ \mathit{context}}$$

Meta language
Object language
Contexts
Types
Terms
Variables
Substitutions
Equality
Why ...?
Implicit substitutions
Normalisation
Normal forms
Values
Environments
Evaluation
reify + reflect
Normalisation
Equality
Type checker

Types

data *Ty* : *Ctxt* → *Set* **where**
  $\star : \mathit{Ty}\ \Gamma$
  $\mathit{El} : \Gamma \vdash \star \to \mathit{Ty}\ \Gamma$
  $\Pi : (\tau : \mathit{Ty}\ \Gamma) \to \mathit{Ty}\ (\Gamma \triangleright \tau) \to \mathit{Ty}\ \Gamma$

$$\frac{}{\Gamma \vdash \star\ \mathit{type}} \qquad \frac{\Gamma \vdash t : \star}{\Gamma \vdash \mathit{El}\ t\ \mathit{type}}$$

$$\frac{\Gamma \vdash \tau_1\ \mathit{type} \qquad \Gamma \triangleright \tau_1 \vdash \tau_2\ \mathit{type}}{\Gamma \vdash \Pi\ \tau_1\ \tau_2\ \mathit{type}}$$

Meta language
Object language
Contexts
Types
Terms
Variables
Substitutions
Equality
Why ...?
Implicit substitutions
Normalisation
Normal forms
Values
Environments
Evaluation
reify + reflect
Normalisation
Equality
Type checker

Types

$$\begin{array}{l}
\textbf{data}\ \mathit{Ty} : \mathit{Ctxt} \to \mathit{Set}\ \textbf{where} \\
\quad \star : \mathit{Ty}\ \Gamma \\
\quad \mathit{El} : \Gamma \vdash \star \to \mathit{Ty}\ \Gamma \\
\quad \Pi : (\tau : \mathit{Ty}\ \Gamma) \to \mathit{Ty}\ (\Gamma \rhd \tau) \to \mathit{Ty}\ \Gamma
\end{array}$$

$$\begin{array}{l}
(/) : \mathit{Ty}\ \Gamma \to \Gamma \Rightarrow \Delta \to \mathit{Ty}\ \Delta \\
\star \quad\quad\ / \rho = \star \\
\mathit{El}\ t \quad / \rho = \mathit{El}\ (t /_{\vdash} \rho) \\
\Pi\ \tau_1\ \tau_2 / \rho = \Pi\ (\tau_1 / \rho)\ (\tau_2\ /\ \rho \uparrow \tau_1)
\end{array}$$

Terms

$$\frac{\Gamma \vdash v : \tau}{\Gamma \vdash \mathit{var}\ v : \tau} \qquad \frac{\Gamma \rhd \tau_1 \vdash t : \tau_2}{\Gamma \vdash \lambda\ t : \Pi\ \tau_1\ \tau_2}$$

$$\frac{\Gamma \vdash t_1 : \Pi\ \tau_1\ \tau_2 \qquad \Gamma \vdash t_2 : \tau_1}{\Gamma \vdash t_1 @ t_2 : \tau_2\ /\ \mathit{sub}\ t_2}$$

$$\frac{\Gamma \vdash t : \tau_1 \qquad \mathit{eq} : \tau_1 =_{\star} \tau_2}{\Gamma \vdash t \mathbin{::^{\equiv}_{\vdash}} \mathit{eq} : \tau_2}$$

$$\frac{\Gamma \vdash t : \tau \qquad \rho : \Gamma \Rightarrow \Delta}{\Delta \vdash t /_{\vdash} \rho : \tau\ /\ \rho}$$

Terms

$$\begin{array}{l}
\textbf{data}\ (\vdash) : (\Gamma : \mathit{Ctxt}) \to \mathit{Ty}\ \Gamma \to \mathit{Set}\ \textbf{where} \\
\quad \mathit{var} \quad : \Gamma \ni \tau \qquad\qquad\qquad\qquad\quad \to \Gamma \vdash \tau \\
\quad \lambda \quad\ \ : \Gamma \rhd \tau_1 \vdash \tau_2 \qquad\qquad\qquad \to \Gamma \vdash \Pi\ \tau_1\ \tau_2 \\
\quad (@) \ \ : \Gamma \vdash \Pi\ \tau_1\ \tau_2 \to (t_2 : \Gamma \vdash \tau_1) \to \Gamma \vdash \tau_2\ /\ \mathit{sub}\ t_2 \\
\quad (::^{\equiv}_{\vdash}) : \Gamma \vdash \tau_1 \qquad \to \tau_1 =_{\star} \tau_2 \qquad \to \Gamma \vdash \tau_2 \\
\quad (/_{\vdash}) \ \ : \Gamma \vdash \tau \qquad\ \ \to (\rho : \Gamma \Rightarrow \Delta) \to \Delta \vdash \tau\ /\ \rho
\end{array}$$

$$(::_{\vdash}) : \Gamma_1 \vdash \tau_1 \to \tau_1 =_{\star} \tau_2 \to \Gamma_2 \vdash \tau_2$$

- Note that $(\vdash)$ is indexed by Ty.

Variables

$$\begin{array}{l}
\textbf{data}\ (\ni) : (\Gamma : \mathit{Ctxt}) \to \mathit{Ty}\ \Gamma \to \mathit{Set}\ \textbf{where} \\
\quad \mathit{vz} \quad : \qquad\qquad\quad\ \Gamma \rhd \sigma \ni \sigma\ /\ \mathit{wk}\ \sigma \\
\quad \mathit{vs} \quad : \Gamma \ni \tau \to \Gamma \rhd \sigma \ni \tau\ /\ \mathit{wk}\ \sigma \\
\quad (::^{\equiv}_{\ni}) : \Gamma \ni \tau_1 \to \tau_1 =_{\star} \tau_2 \to \Gamma \ni \tau_2
\end{array}$$

Substitutions

$$\begin{array}{l}
\textbf{data}\ (\Rightarrow) : \mathit{Ctxt} \to \mathit{Ctxt} \to \mathit{Set}\ \textbf{where} \\
\quad \mathit{sub} : \Gamma \vdash \tau \qquad \to \Gamma \rhd \tau \Rightarrow \Gamma \\
\quad \mathit{wk} \ \ : (\sigma : \mathit{Ty}\ \Gamma) \to \Gamma \Rightarrow \Gamma \rhd \sigma \\
\quad \mathit{id} \ \ \ : \Gamma \Rightarrow \Gamma \\
\quad (\odot) : \Gamma \Rightarrow \Delta \to \Delta \Rightarrow X \to \Gamma \Rightarrow X \\
\quad (\uparrow) \ : \quad (\rho : \Gamma \Rightarrow \Delta) \to (\sigma : \mathit{Ty}\ \Gamma) \\
\qquad\quad \to \Gamma \rhd \sigma \Rightarrow \Delta \rhd (\sigma\ /\ \rho)
\end{array}$$

$$\begin{array}{l}
\emptyset \quad : \varepsilon \Rightarrow \Delta \\
(\blacktriangleright) : (\rho : \Gamma \Rightarrow \Delta) \to \Delta \vdash \tau\ /\ \rho \to \Gamma \rhd \tau \Rightarrow \Delta
\end{array}$$

Equality

- β- and η-rules.
- Evaluation rules for $(/_{\vdash})$.
- Casts can be removed.
- Congruence.
- Heterogeneous.

Term equality

data $(=_\vdash) : \Gamma_1 \vdash \tau_1 \to \Gamma_2 \vdash \tau_2 \to \mathit{Set}$ **where**

-- Equivalence.

$\mathit{refl}_\vdash \ : (t : \Gamma \vdash \tau) \to t =_\vdash t$

$\mathit{sym}_\vdash \ : t_1 =_\vdash t_2 \to t_2 =_\vdash t_1$

$\mathit{trans}_\vdash : t_1 =_\vdash t_2 \to t_2 =_\vdash t_3 \to t_1 =_\vdash t_3$

-- Congruence.

$\mathit{var}_{Cong} \ : v_1 =_\ni v_2 \to \mathit{var}\ v_1 =_\vdash \mathit{var}\ v_2$

$\lambda_{Cong} \ : t_1 =_\vdash t_2 \to \lambda\ t_1 =_\vdash \lambda\ t_2$

$(@_{Cong}) : t_1^1 =_\vdash t_1^2 \to t_2^1 =_\vdash t_2^2 \to t_1^1 @ t_2^1 =_\vdash t_1^2 @ t_2^2$

$(/_{\vdash Cong}) : t_1 =_\vdash t_2 \to \rho_1 =_\Rightarrow \rho_2 \to t_1 /_\vdash \rho_1 =_\vdash t_2 /_\vdash \rho_2$

. . .

Term equality

Meta language · Object language (Contexts, Types, Terms, Variables, Substitutions, Equality, Why . . . ?) · Implicit substitutions · Normalisation (Normal forms, Values, Environments, Evaluation, reify + reflect, Normalisation) · Equality · Type checker

data $(=_\vdash) : \Gamma_1 \vdash \tau_1 \to \Gamma_2 \vdash \tau_2 \to \mathit{Set}$ **where**

-- Cast, beta and eta equality.

$\mathit{castEq}_\vdash : (t ::^{\equiv}_{\vdash} \mathit{eq}) =_\vdash t$

$\beta \ : (\lambda\ t_1) @ t_2 =_\vdash t_1 /_\vdash \mathit{sub}\ t_2$

$\eta \ : \ \{t : \Gamma \vdash \Pi\ \tau_1\ \tau_2\}$

$\quad \to \lambda\ ((t /_\vdash \mathit{wk}\ \tau_1) @ \mathit{var}\ \mathit{vz}) =_\vdash t$

. . .

Term equality

Meta language · Object language (Contexts, Types, Terms, Variables, Substitutions, Equality, Why . . . ?) · Implicit substitutions · Normalisation (Normal forms, Values, Environments, Evaluation, reify + reflect, Normalisation) · Equality · Type checker

data $(=_\vdash) : \Gamma_1 \vdash \tau_1 \to \Gamma_2 \vdash \tau_2 \to \mathit{Set}$ **where**

-- Substitution application axioms.

$\lambda\ t \ /_\vdash \rho \ =_\vdash \lambda\ (t /_\vdash \rho \uparrow \tau_1)$

$(t_1 @ t_2) \ /_\vdash \rho \ =_\vdash (t_1 /_\vdash \rho) @ (t_2 /_\vdash \rho)$

$t \ /_\vdash \mathit{id} \ =_\vdash t$

$t \ /_\vdash (\rho_1 \odot \rho_2) =_\vdash t /_\vdash \rho_1 /_\vdash \rho_2$

$\mathit{var}\ v \ /_\vdash \mathit{wk}\ \sigma \ =_\vdash \mathit{var}\ (\mathit{vs}\ v)$

$\mathit{var}\ \mathit{vz} \ /_\vdash \mathit{sub}\ t \ =_\vdash t$

$\mathit{var}\ (\mathit{vs}\ v) \ /_\vdash \mathit{sub}\ t \ =_\vdash \mathit{var}\ v$

$\mathit{var}\ \mathit{vz} \ /_\vdash (\rho \uparrow \sigma) \ =_\vdash \mathit{var}\ \mathit{vz}$

$\mathit{var}\ (\mathit{vs}\ v) \ /_\vdash (\rho \uparrow \sigma) \ =_\vdash \mathit{var}\ v /_\vdash \rho /_\vdash \mathit{wk}\ (\sigma / \rho)$

Why heterogeneous equality?

Meta language · Object language (Contexts, Types, Terms, Variables, Substitutions, Equality, Why . . . ?) · Implicit substitutions · Normalisation (Normal forms, Values, Environments, Evaluation, reify + reflect, Normalisation) · Equality · Type checker

- $\mathit{var}\ \mathit{vz} /_\vdash \mathit{sub}\ t =_\vdash t$.
- $\sigma\ /\ \mathit{wk}\ \sigma\ /\ \mathit{sub}\ t \overset{?}{=} \sigma$.
- With homogeneous equality:
  $\sigma\ /\ \mathit{wk}\ \sigma\ /\ \mathit{sub}\ t =_\star \sigma$ proved or postulated.
- Not proved because:
  Very large mutually recursive definition.
- Not postulated because:
  $\tau\ /\ \rho$ would not evaluate.

Why explicit substitutions?

Meta language · Object language (Contexts, Types, Terms, Variables, Substitutions, Equality, Why . . . ?) · Implicit substitutions · Normalisation (Normal forms, Values, Environments, Evaluation, reify + reflect, Normalisation) · Equality · Type checker

- If $(/_\vdash)$ were a function: similar problems.

Implicit substitutions

Meta language · Object language (Contexts, Types, Terms, Variables, Substitutions, Equality, Why . . . ?) · Implicit substitutions · Normalisation (Normal forms, Values, Environments, Evaluation, reify + reflect, Normalisation) · Equality · Type checker

data $\mathit{Tm}^- : \Gamma \vdash \tau \to \mathit{Set}$ **where**

$\mathit{var}^- \ : (v : \Gamma \ni \tau) \to \mathit{Tm}^-\ (\mathit{var}\ v)$

$\lambda^- \ : \ \{t : \Gamma \rhd \tau_1 \vdash \tau_2\}$

$\quad \to \mathit{Ty}^-\ \tau_1 \to \mathit{Tm}^-\ t$

$\quad \to \mathit{Tm}^-\ (\lambda\ t)$

$(@^-) \ : \mathit{Tm}^-\ t_1 \to \mathit{Tm}^-\ t_2 \to \mathit{Tm}^-\ (t_1 @ t_2)$

$(::^{\equiv}_{\vdash}{}^-) : \mathit{Tm}^-\ t_1 \to t_1 =_\vdash t_2 \to \mathit{Tm}^-\ t_2$

$\mathit{tm}^-\mathit{ToTm} \ : \{t : \Gamma \vdash \tau\} \to \mathit{Tm}^-\ t \to \Gamma \vdash \tau$

$\mathit{tm}^-\mathit{ToTmEq} : (t^- : \mathit{Tm}^-\ t) \to \mathit{tm}^-\mathit{ToTm}\ t^- =_\vdash t$

$\mathit{tmToTm}^- \ : (t : \Gamma \vdash \tau) \to \mathit{Tm}^-\ t$

Normal forms

$$
\begin{array}{l}
\textbf{data } \mathit{Atom} : \Gamma \vdash \tau \to \mathit{Set} \textbf{ where} \\
\quad \mathit{var}_{\mathit{At}} \;:\; (v : \Gamma \ni \tau) \to \mathit{Atom}\ (\mathit{var}\ v) \\
\quad (@_{\mathit{At}}) \;:\; \mathit{Atom}\ t_1 \to \mathit{NF}\ t_2 \to \mathit{Atom}\ (t_1 @ t_2) \\
\quad (::^{\equiv}_{\mathit{At}}) \;:\; \mathit{Atom}\ t_1 \to t_1 =_{\vdash} t_2 \to \mathit{Atom}\ t_2 \\
\\
\textbf{data } \mathit{NF} : \Gamma \vdash \tau \to \mathit{Set} \textbf{ where} \\
\quad \mathit{atom}^{\star}_{\mathit{NF}} \;:\; \{t : \Gamma \vdash \star\} \quad\; \to \mathit{Atom}\ t \to \mathit{NF}\ t \\
\quad \mathit{atom}^{\mathit{El}}_{\mathit{NF}} \;:\; \{t : \Gamma \vdash \mathit{El}\ t'\} \to \mathit{Atom}\ t \to \mathit{NF}\ t \\
\quad \lambda_{\mathit{NF}} \;:\; \mathit{NF}\ t \to \mathit{NF}\ (\lambda\ t) \\
\quad (::^{\equiv}_{\mathit{NF}}) \;:\; \mathit{NF}\ t_1 \to t_1 =_{\vdash} t_2 \to \mathit{NF}\ t_2
\end{array}
$$

Meta language
Object language: Contexts, Types, Terms, Variables, Substitutions, Equality, Why . . . ?
Implicit substitutions
Normalisation: Normal forms, Values, Environments, Evaluation, reify + reflect, Normalisation
Equality
Type checker

Context extensions

$$
\begin{array}{l}
\textbf{data } \mathit{Ctxt}^{+}\ (\Gamma : \mathit{Ctxt}) : \mathit{Set} \textbf{ where} \\
\quad \varepsilon^{+} \quad : \mathit{Ctxt}^{+}\ \Gamma \\
\quad (\triangleright^{+}) : (\Gamma' : \mathit{Ctxt}^{+}\ \Gamma) \to \mathit{Ty}\ (\Gamma +\!\!+ \Gamma') \to \mathit{Ctxt}^{+}\ \Gamma \\
\\
(+\!\!+) : (\Gamma : \mathit{Ctxt}) \to \mathit{Ctxt}^{+}\ \Gamma \to \mathit{Ctxt} \\
\Gamma +\!\!+ \varepsilon^{+} \qquad\quad = \Gamma \\
\Gamma +\!\!+ (\Gamma' \triangleright^{+} \tau) = (\Gamma +\!\!+ \Gamma') \triangleright \tau
\end{array}
$$

Meta language
Object language: Contexts, Types, Terms, Variables, Substitutions, Equality, Why . . . ?
Implicit substitutions
Normalisation: Normal forms, Values, Environments, Evaluation, reify + reflect, Normalisation
Equality
Type checker

Values

$$
\begin{array}{l}
\textbf{data } \mathit{Val} : \Gamma \vdash \tau \to \mathit{Set} \textbf{ where} \\
\quad (::_{\mathit{Val}}) : \mathit{Val}\ t_1 \to t_1 =_{\vdash} t_2 \to \mathit{Val}\ t_2 \\
\quad \star_{\mathit{Val}} \;: \{t : \Gamma \vdash \star\} \qquad \to \mathit{Atom}\ t \to \mathit{Val}\ t \\
\quad \mathit{El}_{\mathit{Val}} : \{t : \Gamma \vdash \mathit{El}\ t'\} \to \mathit{Atom}\ t \to \mathit{Val}\ t \\
\quad \Pi_{\mathit{Val}} \;: \quad \{t_1 : \Gamma \vdash \Pi\ \tau_1\ \tau_2\} \\
\qquad\quad \to (f \quad : \quad (\Gamma' : \mathit{Ctxt}^{+}\ \Gamma) \\
\qquad\qquad\qquad \to \{t_2 : \Gamma +\!\!+ \Gamma' \vdash \tau_1 \mathbin{/} \mathit{wk}^{*}\ \Gamma'\} \\
\qquad\qquad\qquad \to (v : \mathit{Val}\ t_2) \\
\qquad\qquad\qquad \to \mathit{Val}\ ((t_1 \mathbin{/_{\vdash}} \mathit{wk}^{*}\ \Gamma') @ t_2)) \\
\qquad\quad \to \mathit{Val}\ t_1 \\
\\
(@_{\mathit{Val}}) : \mathit{Val}\ t_1 \to \mathit{Val}\ t_2 \to \mathit{Val}\ (t_1 @ t_2) \\
\mathit{wk}^{\star}_{\mathit{Val}} \;: \mathit{Val}\ t \to (\Gamma' : \mathit{Ctxt}^{+}\ \Gamma) \to \mathit{Val}\ (t \mathbin{/_{\vdash}} \mathit{wk}^{*}\ \Gamma')
\end{array}
$$

Meta language
Object language: Contexts, Types, Terms, Variables, Substitutions, Equality, Why . . . ?
Implicit substitutions
Normalisation: Normal forms, Values, Environments, Evaluation, reify + reflect, Normalisation
Equality
Type checker

Environments

$$
\begin{array}{l}
\textbf{data } \mathit{Env} : \Gamma \Rightarrow \Delta \to \mathit{Set} \textbf{ where} \\
\quad \emptyset_{\mathit{Env}} \quad : \mathit{Env}\ \emptyset \\
\quad (\blacktriangleright_{\mathit{Env}}) : \quad \{\rho : \Gamma \Rightarrow \Delta\} \to \{t : \Delta \vdash \sigma \mathbin{/} \rho\} \\
\qquad\qquad \to \mathit{Env}\ \rho \to \mathit{Val}\ t \to \mathit{Env}\ (\rho \blacktriangleright t) \\
\quad (::^{\equiv}_{\mathit{Env}}) : \mathit{Env}\ \rho_1 \to \rho_1 =_{\Rightarrow} \rho_2 \to \mathit{Env}\ \rho_2 \\
\\
\mathit{lookup} : (v : \Gamma \ni \tau) \to \mathit{Env}\ \rho \to \mathit{Val}\ (\mathit{var}\ v \mathbin{/_{\vdash}} \rho)
\end{array}
$$

Meta language
Object language: Contexts, Types, Terms, Variables, Substitutions, Equality, Why . . . ?
Implicit substitutions
Normalisation: Normal forms, Values, Environments, Evaluation, reify + reflect, Normalisation
Equality
Type checker

Evaluation

$$
\begin{array}{l}
[\![\cdot]\!] : \mathit{Tm}^{-}\ t \to \mathit{Env}\ \rho \to \mathit{Val}\ (t \mathbin{/_{\vdash}} \rho) \\
[\![\mathit{var}^{-}\ v]\!]\,\gamma \qquad = \mathit{lookup}\ v\ \gamma \\
[\![t_1^{-}\ @^{-}\ t_2^{-}]\!]\,\gamma \;= ([\![t_1^{-}]\!]\,\gamma\ @_{\mathit{Val}}\ [\![t_2^{-}]\!]\,\gamma)\ ::_{\mathit{Val}}\ \ldots \\
[\![t^{-}\ ::^{\equiv}_{\vdash}\ \mathit{eq}]\!]\,\gamma = [\![t^{-}]\!]\,\gamma\ ::_{\mathit{Val}}\ \ldots \\
[\![\lambda^{-}\ t_1^{-}]\!]\,\gamma \qquad = \Pi_{\mathit{Val}}\ (\backslash\Delta'\ v_2 \to \\
\quad [\![t_1^{-}]\!]\,(\mathit{wk}^{\star}_{\mathit{Env}}\ \gamma\ \Delta' \blacktriangleright_{\mathit{Env}}\ (v_2\ ::_{\mathit{Val}} \ldots)) \\
\quad ::_{\mathit{Val}}\ \ldots \beta \ldots)
\end{array}
$$

Meta language
Object language: Contexts, Types, Terms, Variables, Substitutions, Equality, Why . . . ?
Implicit substitutions
Normalisation: Normal forms, Values, Environments, Evaluation, reify + reflect, Normalisation
Equality
Type checker

reify + reflect

$$
\begin{array}{l}
\mathit{reify} : (\tau : \mathit{Ty}\ \Gamma) \to \{t : \Gamma \vdash \tau\} \to \mathit{Val}\ t \to \mathit{NF}\ t \\
\mathit{reify}\ (\Pi\ \tau_1\ \tau_2)\ (\Pi_{\mathit{Val}}\ f) = \\
\quad \lambda_{\mathit{NF}}\ (\mathit{reify}\ (\tau_2 \mathbin{/} \_ \mathbin{/} \_) \\
\qquad\qquad (f\ (\varepsilon^{+} \triangleright^{+} \tau_1) \\
\qquad\qquad\quad (\mathit{reflect}\ (\tau_1 \mathbin{/} \_)\ (\mathit{var}_{\mathit{At}}\ \mathit{vz})\ ::_{\mathit{Val}} \ldots))) \\
\quad ::_{\mathit{NF}} \ldots \eta \ldots \\
\\
\mathit{reflect} : (\tau : \mathit{Ty}\ \Gamma) \to \{t : \Gamma \vdash \tau\} \to \mathit{Atom}\ t \to \mathit{Val}\ t \\
\mathit{reflect}\ (\Pi\ \tau_1\ \tau_2)\ \mathit{at} = \Pi_{\mathit{Val}}\ (\backslash\Gamma'\ v \to \\
\quad \mathit{reflect}\ (\tau_2 \mathbin{/} \_ \mathbin{/} \_)\ (\mathit{wk}^{\star}_{\mathit{At}}\ \mathit{at}\ \Gamma'\ @_{\mathit{At}}\ \mathit{reify}\ (\tau_1 \mathbin{/} \_)\ v))
\end{array}
$$

Meta language
Object language: Contexts, Types, Terms, Variables, Substitutions, Equality, Why . . . ?
Implicit substitutions
Normalisation: Normal forms, Values, Environments, Evaluation, reify + reflect, Normalisation
Equality
Type checker

Normalisation

$$id_{Env} : (\Gamma : Ctxt) \to Env\ (id\ \Gamma)$$

$$normalise : (t : \Gamma \vdash \tau) \to NF\ t$$
$$normalise\ t = reify\ \_\ (\llbracket tmToTm^{-}\ t \rrbracket\ id_{Env}\ ::_{Val}\ \ldots)$$

$$normaliseEq : (t : \Gamma \vdash \tau) \to nfToTm\ (normalise\ t) =_{\vdash} t$$
$$normaliseEq\ t = nfToTmEq\ (normalise\ t)$$

- The completeness proof is under way.

Equality

Meta language
Object language
Contexts
Types
Terms
Variables
Substitutions
Equality
Why . . . ?
Implicit substitutions
Normalisation
Normal forms
Values
Environments
Evaluation
reify + reflect
Normalisation
Equality
Type checker

NFs Strip casts, check syntactic equality.
Terms Normalise, then check.
Types Check structurally.

$$\textbf{data}\ TyEq?\ (\tau_1 : Ty\ \Gamma_1)\ (\tau_2 : Ty\ \Gamma_2) : Set\ \textbf{where}$$
$$equalTy\ \ : \tau_1 =_{\star} \tau_2 \to TyEq?\ \tau_1\ \tau_2$$
$$notEqualTy : TyEq?\ \tau_1\ \tau_2$$

$$(\stackrel{?}{=}) : (\tau_1, \tau_2 : Ty\ \Gamma) \to TyEq?\ \tau_1\ \tau_2$$

Type checker

Meta language
Object language
Contexts
Types
Terms
Variables
Substitutions
Equality
Why . . . ?
Implicit substitutions
Normalisation
Normal forms
Values
Environments
Evaluation
reify + reflect
Normalisation
Equality
Type checker

- Raw terms (*RawTm*).
- Lambdas annotated with raw types.

$$\textbf{data}\ IsTm^{-}?\ (\Gamma : Ctxt) : RawTm \to Set\ \textbf{where}$$
$$isTm\ \ : (\tau : Ty\ \Gamma) \to (t : \Gamma \vdash \tau) \to (t^{-} : Tm^{-}\ t)$$
$$\to IsTm^{-}?\ \Gamma\ (eraseTm^{-}\ t^{-})$$
$$noTm\ (e : RawTm) : IsTm^{-}?\ \Gamma\ e$$

$$inferTm^{-} : (\Gamma : Ctxt) \to (e : RawTm) \to IsTm^{-}?\ \Gamma\ e$$

Discussion

Meta language
Object language
Contexts
Types
Terms
Variables
Substitutions
Equality
Why . . . ?
Implicit substitutions
Normalisation
Normal forms
Values
Environments
Evaluation
reify + reflect
Normalisation
Equality
Type checker

- Internal method advantage:
  Types give a lot of info.
  Example: No de Bruijn index arithmetic.
- Disadvantage:
  Sometimes things become very dependent on each other.