Formalising the metatheory of type theory using quotient inductive types

Ambrus Kaposi
Eötvös Loránd University, Budapest

j.w.w. Thorsten Altenkirch and András Kovács

Workshop on Foundations for the
Practical Formalization of Mathematics,
Nantes,
27 April 2017

Motivation

Why type theory in type theory?

- Study the metatheory of type theory in a nice language
- Type-safe template type theory (metaprogamming)
  - generic programming
  - extensions of type theory justified by models

Plan

1. Extrinsic vs. intrinsic syntax for simple type theory
2. Extrinsic vs. intrinsic syntax for type theory
3. Defining functions from the intrinsic syntax
4. Relationship of extrinsic and intrinsic sytnax
5. Models

Extrinsic vs. intrinsic syntax for simple type theory

Extrinsic syntax for simple type theory

4 inductive sets + 2 inductive relations.

$$\begin{aligned} x &::= \mathsf{zero} \mid \mathsf{suc}\, x \\ t &::= x \mid \mathsf{lam}\, t \mid \mathsf{app}\, t\, t' \\ A &::= \iota \mid A \Rightarrow A' \\ \Gamma &::= \cdot \mid \Gamma, A \end{aligned}$$

$$\frac{}{\Gamma, A \vdash_v \mathsf{zero} : A} \qquad \frac{\Gamma \vdash_v x : A}{\Gamma, B \vdash_v \mathsf{suc}\, x : A}$$

$$\frac{\Gamma \vdash_v x : A}{\Gamma \vdash x : A} \quad \frac{\Gamma, A \vdash t : B}{\Gamma \vdash \mathsf{lam}\, t : A \to B} \quad \frac{\Gamma \vdash t : A \to B \quad \Gamma \vdash u : A}{\Gamma \vdash \mathsf{app}\, t\, u : B}$$

Intrinsic syntax for simple type theory

4 inductively defined families of sets.

$$
\begin{aligned}
&\mathsf{Ty} && : \mathsf{Set} \\
&\quad \iota && : \mathsf{Ty} \\
&\quad - \Rightarrow - && : \mathsf{Ty} \to \mathsf{Ty} \to \mathsf{Ty} \\
&\mathsf{Con} && : \mathsf{Set} \\
&\quad \cdot && : \mathsf{Con} \\
&\quad -, - && : \mathsf{Con} \to \mathsf{Ty} \to \mathsf{Con} \\
&\mathsf{Var} && : \mathsf{Con} \to \mathsf{Ty} \to \mathsf{Set} \\
&\quad \mathsf{zero} && : \mathsf{Var}\,(\Gamma, A)\,A \\
&\quad \mathsf{suc} && : \mathsf{Var}\,\Gamma\,A \to \mathsf{Var}\,(\Gamma, B)\,A \\
&\mathsf{Tm} && : \mathsf{Con} \to \mathsf{Ty} \to \mathsf{Set} \\
&\quad \mathsf{var} && : \mathsf{Var}\,\Gamma\,A \to \mathsf{Tm}\,\Gamma\,A \\
&\quad \mathsf{lam} && : \mathsf{Tm}\,(\Gamma, A)\,B \to \mathsf{Tm}\,\Gamma\,(A \Rightarrow B) \\
&\quad \mathsf{app} && : \mathsf{Tm}\,\Gamma\,(A \Rightarrow B) \to \mathsf{Tm}\,\Gamma\,A \to \mathsf{Tm}\,\Gamma\,B
\end{aligned}
$$

Extrinsic vs. intrinsic syntax for type theory

Extrinsic syntax

4 inductive sets + 8 inductive relations (we can't avoid talking about conversion).

$$
\begin{aligned}
&\mathsf{PCon}, \mathsf{PTy}, \mathsf{PTm}, \mathsf{PTms} : \mathsf{Set} \\
&\vdash_{\mathsf{Con}} : \mathsf{PCon} \to \mathsf{Prop} \\
&\vdash_{\mathsf{Ty}} : \mathsf{PCon} \to \mathsf{PTy} \to \mathsf{Prop} \\
&\vdash_{\mathsf{Tm}} : \mathsf{PCon} \to \mathsf{PTy} \to \mathsf{PTm} \to \mathsf{Prop} \\
&\vdash_{\mathsf{Tms}} : \mathsf{PCon} \to \mathsf{PCon} \to \mathsf{PTms} \to \mathsf{Prop} \\
&\sim_{\mathsf{Con}} : \mathsf{PCon} \to \mathsf{PCon} \to \mathsf{Prop} \\
&\sim_{\mathsf{Ty}} : \mathsf{PCon} \to \mathsf{PTy} \to \mathsf{PTy} \to \mathsf{Prop} \\
&\sim_{\mathsf{Tm}} : \mathsf{PCon} \to \mathsf{PTy} \to \mathsf{PTm} \to \mathsf{PTm} \to \mathsf{Prop} \\
&\sim_{\mathsf{Tms}} : \mathsf{PCon} \to \mathsf{PCon} \to \mathsf{PTms} \to \mathsf{PTms} \to \mathsf{Prop}
\end{aligned}
$$

Relations are given by rules for ER, coercion, congruence, conversion.

Extrinsic syntax, PER variant (Dybjer: Undec... LCCC)

4 inductive sets + 4 inductive relations.

$$
\begin{aligned}
&\mathsf{PCon}, \mathsf{PTy}, \mathsf{PTm}, \mathsf{PTms} : \mathsf{Set} \\
&\sim_{\mathsf{Con}} : \mathsf{PCon} \to \mathsf{PCon} \to \mathsf{Prop} \\
&\sim_{\mathsf{Ty}} : \mathsf{PCon} \to \mathsf{PTy} \to \mathsf{PTy} \to \mathsf{Prop} \\
&\sim_{\mathsf{Tm}} : \mathsf{PCon} \to \mathsf{PTy} \to \mathsf{PTm} \to \mathsf{PTm} \to \mathsf{Prop} \\
&\sim_{\mathsf{Tms}} : \mathsf{PCon} \to \mathsf{PCon} \to \mathsf{PTms} \to \mathsf{PTms} \to \mathsf{Prop}
\end{aligned}
$$

Recovering typing relations as reflexive cases:

$$
\begin{aligned}
\vdash_{\mathsf{Con}} \Gamma &:= \Gamma \sim_{\mathsf{Con}} \Gamma \\
\Gamma \vdash_{\mathsf{Ty}} A &:= \Gamma \vdash A \sim_{\mathsf{Ty}} A \\
\Gamma \vdash_{\mathsf{Tm}} t : A &:= \Gamma \vdash t \sim_{\mathsf{Tm}} t : A \\
\Gamma \vdash_{\mathsf{Tms}} \sigma : \Delta &:= \Gamma \vdash \sigma \sim_{\mathsf{Tms}} \sigma : \Delta
\end{aligned}
$$

Congruence rules and typing rules are identified. E.g.

$\dfrac{\vdash_{\mathsf{Con}} \Gamma \qquad \Gamma \vdash_{\mathsf{Ty}} A}{\vdash_{\mathsf{Con}} \Gamma, A}$ is expressed by $\dfrac{\Gamma \sim_{\mathsf{Con}} \Gamma \qquad \Gamma \vdash A \sim_{\mathsf{Ty}} A'}{\Gamma, A \sim_{\mathsf{Con}} \Gamma', A'}$.

Intrinsic syntax (James Chapman: TT should eat itself)

An inductive inductive definition of 4 families of sets + 4 families of relations.

$$
\begin{aligned}
\mathsf{Con} &: \mathsf{Set} \\
\mathsf{Ty} &: \mathsf{Con} \to \mathsf{Set} \\
\mathsf{Tm} &: (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \to \mathsf{Set} \\
\mathsf{Tms} &: \mathsf{Con} \to \mathsf{Con} \to \mathsf{Set} \\
\sim_{\mathsf{Con}} &: \mathsf{Con} \to \mathsf{Con} \to \mathsf{Prop} \\
\sim_{\mathsf{Ty}} &: (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \to \mathsf{Ty}\,\Gamma \to \mathsf{Prop} \\
\sim_{\mathsf{Tm}} &: (\Gamma : \mathsf{Con})(A : \mathsf{Ty}\,\Gamma) \to \mathsf{Tm}\,\Gamma\,A \to \mathsf{Tm}\,\Gamma\,A \to \mathsf{Prop} \\
\sim_{\mathsf{Tms}} &: (\Gamma\,\Delta : \mathsf{Con}) \to \mathsf{Tms}\,\Gamma\,\Delta \to \mathsf{Tms}\,\Gamma\,\Delta \to \mathsf{Prop}
\end{aligned}
$$

No more separation of pre-things and things. One can only talk about well-typed terms.

Quotient intrinsic syntax

A quotient inductive inductive definition of 4 families of sets.

$$
\begin{aligned}
\mathsf{Con} &: \mathsf{Set} \\
\mathsf{Ty} &: \mathsf{Con} \to \mathsf{Set} \\
\mathsf{Tm} &: (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \to \mathsf{Set} \\
\mathsf{Tms} &: \mathsf{Con} \to \mathsf{Con} \to \mathsf{Set}
\end{aligned}
$$

- Conversion relation is the identity type for each set. Conversion rules (e.g. β, η) are given as equality constructors.
- Rules for ER, coercion and congruence are properties of the identity type.
- No more separation of convertible things. One can only do constructions on the syntax up to equality.

The syntax for a type theory with Π and an empty universe

$$
\begin{array}{llll}
\cdot & : \mathsf{Con} & [\mathsf{id}] & : A[\mathsf{id}] \equiv A \\
-,- & : (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \to \mathsf{Con} & [][] & : A[\sigma][\nu] \equiv A[\sigma \circ \nu] \\
-[-] & : \mathsf{Ty}\,\Delta \to \mathsf{Tms}\,\Gamma\,\Delta \to \mathsf{Ty}\,\Gamma & \mathsf{id}\circ & : \mathsf{id} \circ \sigma \equiv \sigma \\
\mathsf{id} & : \mathsf{Tms}\,\Gamma\,\Gamma & \circ\mathsf{id} & : \sigma \circ \mathsf{id} \equiv \sigma \\
- \circ - & : \mathsf{Tms}\,\Theta\,\Delta \to \mathsf{Tms}\,\Gamma\,\Theta \to \mathsf{Tms}\,\Gamma\,\Delta & \circ\circ & : (\sigma \circ \nu) \circ \delta \equiv \sigma \circ (\nu \circ \delta) \\
\epsilon & : \mathsf{Tms}\,\Gamma\,\cdot & \epsilon\eta & : \{\sigma : \mathsf{Tms}\,\Gamma\,\cdot\} \to \sigma \equiv \epsilon \\
-,- & : (\sigma : \mathsf{Tms}\,\Gamma\,\Delta) \to \mathsf{Tm}\,\Gamma\,A[\sigma] \to \mathsf{Tms}\,\Gamma\,(\Delta, A) & \pi_1\beta & : \pi_1\,(\sigma, t) \equiv \sigma \\
\pi_1 & : \mathsf{Tms}\,\Gamma\,(\Delta, A) \to \mathsf{Tms}\,\Gamma\,\Delta & \pi\eta & : (\pi_1\,\sigma, \pi_2\,\sigma) \equiv \sigma \\
-[-] & : \mathsf{Tm}\,\Delta\,A \to (\sigma : \mathsf{Tms}\,\Gamma\,\Delta) \to \mathsf{Tm}\,\Gamma\,A[\sigma] & ,\circ & : (\sigma, t) \circ \nu \equiv (\sigma \circ \nu), ({}_{[][]*}\,t[\nu]) \\
\pi_2 & : (\sigma : \mathsf{Tms}\,\Gamma\,(\Delta, A)) \to \mathsf{Tm}\,\Gamma\,A[\pi_1\,\sigma] & \pi_2\beta & : \pi_2\,(\sigma, t) \equiv^{\pi_1\beta}\, t \\
\hline
\Pi & : (A : \mathsf{Ty}\,\Gamma) \to \mathsf{Ty}\,(\Gamma, A) \to \mathsf{Ty}\,\Gamma & \Pi[] & : (\Pi\,A\,B)[\sigma] \equiv \Pi\,A[\sigma]\,B[\sigma\uparrow] \\
\mathsf{lam} & : \mathsf{Tm}\,(\Gamma, A)\,B \to \mathsf{Tm}\,\Gamma\,(\Pi\,A\,B) & \Pi\beta & : \mathsf{app}\,(\mathsf{lam}\,t) \equiv t \\
\mathsf{app} & : \mathsf{Tm}\,\Gamma\,(\Pi\,A\,B) \to \mathsf{Tm}\,(\Gamma, A)\,B & \Pi\eta & : \mathsf{lam}\,(\mathsf{app}\,t) \equiv t \\
 & & \mathsf{lam}[] & : (\mathsf{lam}\,t)[\sigma] \equiv^{\Pi[]} \mathsf{lam}\,(t[\sigma\uparrow]) \\
\hline
\mathsf{U} & : \mathsf{Ty}\,\Gamma & \mathsf{U}[] & : \mathsf{U}[\sigma] \equiv \mathsf{U} \\
\mathsf{El} & : \mathsf{Tm}\,\Gamma\,\mathsf{U} \to \mathsf{Ty}\,\Gamma & \mathsf{El}[] & : (\mathsf{El}\,\hat{A})[\sigma] \equiv \mathsf{El}\,({}_{\mathsf{U}[]*}\,\hat{A}[\sigma])
\end{array}
$$

Defining functions from the intrinsic syntax

Nondependent eliminator (recursor)

The inductive type of natural numbers:

$$\mathbb{N} : \mathsf{Set}$$
$$\mathsf{zero} : \mathbb{N}$$
$$\mathsf{suc} : \mathbb{N} \to \mathbb{N}$$

Arguments of the recursor (a natural number algebra):

$$\mathbb{N}_1 : \mathsf{Set}$$
$$\mathsf{zero}_1 : \mathbb{N}_1$$
$$\mathsf{suc}_1 : \mathbb{N}_1 \to \mathbb{N}_1$$

The recursor is a function which respects the operations (an algebra morphism).

$$\mathsf{Rec}\mathbb{N} : \mathbb{N} \to \mathbb{N}_1$$
$$\mathsf{Rec}\mathbb{N}\,\mathsf{zero} = \mathsf{zero}_1$$
$$\mathsf{Rec}\mathbb{N}\,(\mathsf{suc}\,n) = \mathsf{suc}_1\,(\mathsf{Rec}\mathbb{N}\,n)$$

Recursor for a higher inductive type

Example:

Constructors:		Arguments of the recursor:	
I	$: \mathsf{Set}$	I_1	$: \mathsf{Set}$
left	$: \mathsf{I}$	left_1	$: \mathsf{I}_1$
right	$: \mathsf{I}$	right_1	$: \mathsf{I}_1$
$\mathsf{segment}$	$: \mathsf{left} \equiv \mathsf{right}$	$\mathsf{segment}_1$	$: \mathsf{left}_1 \equiv \mathsf{right}_1$

The recursor:

$$\begin{aligned}
&\mathsf{RecI} : \mathsf{I} \to \mathsf{I}_1 \\
&\mathsf{RecI}\,\mathsf{left} = \mathsf{left}_1 \\
&\mathsf{RecI}\,\mathsf{right} = \mathsf{right}_1 \\
&\mathsf{ap}\,\mathsf{RecI}\,\mathsf{segment} = \mathsf{segment}_1
\end{aligned}$$

Recursor for intrinsic type theory

- An algebra for the quotient intrinsic syntax is a categories with families (CwF, a notion of model of type theory).
  - An algebra for the intrinsic syntax is a more relaxed CwF, where conversion can be interpreted by relations other than equality.
- The recursor is a strict morphism of models from the initial model (the syntax) to the model given by the arguments of the recursor.
- The recursor for an inductive inductive type is recursive recursive (Forsberg).

$$
\begin{array}{ll}
\mathsf{Con} : \mathsf{Set} & \mathsf{Ty} : \mathsf{Con} \to \mathsf{Set} \\
\mathsf{Con}_1 : \mathsf{Set} & \mathsf{Ty}_1 : \mathsf{Con}_1 \to \mathsf{Set} \\
\mathsf{RecCon} : \mathsf{Con} \to \mathsf{Con}_1 & \mathsf{RecTy} : \mathsf{Ty}\,\Gamma \to \mathsf{Ty}_1\,(\mathsf{RecCon}\,\Gamma)
\end{array}
$$

Recursor for quotient inductive types in Agda

```
{-# OPTIONS --rewriting #-}

postulate
  Con : Set
  Ty  : Con → Set
  _,_ : (Γ : Con) → Ty Γ → Con

  RecCon : Con → Con₁
  RecTy  : Ty Γ → Ty₁ (RecCon Γ)

  β, : RecCon (Γ , A) ≡ RecCon Γ ,₁ RecTy A

{-# REWRITE β, #-}
...
```

Relationship of extrinsic and intrinsic sytnax

From intrinsic to extrinsic syntax (work in progress)

- Inductive inductive types can be represented by normal inductive types and "typing relations" (noticed by Altenkirch and Capriotti). Similar to representing indexed W-types by plain ones.
  - If we start with intrinsic syntax, we get back an extrinsic syntax which is
    - fully annotated (e.g. $- \circ -$ has 5 arguments)
    - paranoid (e.g. typing for lam needs well-formedness of Γ)
  - The usual syntax comes after some ad-hoc constructions (removing assumptions that are admissible).
- Going from quotient intrinsic syntax to intrinsic is doing an internal setoid-interpretation. For example, the inductively defined setoid equality relation for $\mathbb{N}$:

$$
\begin{aligned}
\sim_{\mathbb{N}} &: \mathbb{N} \to \mathbb{N} \to \mathsf{Prop} \\
\sim_{\mathsf{zero}} &: \mathsf{zero} \sim_{\mathbb{N}} \mathsf{zero} \\
\sim_{\mathsf{suc}} &: (n_0 \sim_{\mathbb{N}} n_1) \to \mathsf{suc}\, n_0 \sim_{\mathbb{N}} \mathsf{suc}\, n_1
\end{aligned}
$$

From extrinsic to intrinsic syntax (Streicher: Semantics of TT)

A model is given by a category with families $\mathcal{C}$.
A preterm carries enough information to reconstruct its precontext (ctx) and pretype (ty). Similarly for the other pre-things.
Partial functions by recursion on the presyntax:

$$\begin{aligned}
\llbracket - \rrbracket_{\mathsf{Con}} &: \mathsf{PCon} \rightharpoonup |\mathcal{C}| \\
\llbracket - \rrbracket_{\mathsf{Ty}} &: (A : \mathsf{PTy}) \rightharpoonup \mathsf{Ty}_{\mathcal{C}}\big(\llbracket \mathsf{ctx}\, A \rrbracket_{\mathsf{Con}}\big) \\
\llbracket - \rrbracket_{\mathsf{Tm}} &: (t : \mathsf{PTm}) \rightharpoonup \mathsf{Tm}_{\mathcal{C}}\big(\llbracket \mathsf{ctx}\, t \rrbracket_{\mathsf{Con}}, \llbracket \mathsf{ty}\, t \rrbracket_{\mathsf{Ty}}\big) \\
\llbracket - \rrbracket_{\mathsf{Tms}} &: (\sigma : \mathsf{PTms}) \rightharpoonup \mathcal{C}\big(\llbracket \mathsf{dom}\, \sigma \rrbracket_{\mathsf{Con}}, \llbracket \mathsf{cod}\, \sigma \rrbracket_{\mathsf{Con}}\big)
\end{aligned}$$

By induction on the typing and conversion relations we have:

$$\begin{aligned}
&\Gamma \vdash_{\mathsf{Tm}} t : A \rightarrow \llbracket t \rrbracket_{\mathsf{Tm}} \text{ is defined} \\
&\Gamma \vdash t \sim_{\mathsf{Tm}} t' : A \rightarrow \llbracket t \rrbracket_{\mathsf{Tm}} \text{ and } \llbracket t' \rrbracket_{\mathsf{Tm}} \text{ are defined and are equal}
\end{aligned}$$

Similarly for contexts, types and subsitutions.
This can be used to map extrinsic syntax to quotient intrinsic syntax.

Conjecture

$$\mathsf{Con} \;\cong\; \Big((\Gamma : \mathsf{PCon}) \times \vdash \Gamma\Big)/{\sim_{\mathsf{Con}}}$$

$$\begin{aligned} &(\Gamma : \mathsf{Con}) \;\times\; \mathsf{Ty}\,\Gamma \\ \cong\; &\Big((\Gamma : \mathsf{PCon}) \;\times \vdash \Gamma \;\times\; (A : \mathsf{PTy}) \;\times\; \Gamma \vdash A\Big)/{\sim_{\mathsf{Con}}}/{\sim_{\mathsf{Ty}}} \end{aligned}$$

...

Typechecking (not yet formalised)

$$\mathsf{inferTm} : (\Gamma : \mathsf{Con})(t : \mathsf{PTm}) \to (A' : \mathsf{Ty}\,\Gamma) \times \mathsf{Tm}\,\Gamma\,A'$$

$$\mathsf{checkTm} : (\Gamma : \mathsf{Con})(t : \mathsf{PTm}) \to (A' : \mathsf{Ty}\,\Gamma) \to \mathsf{Tm}\,\Gamma\,A'$$

$$\mathsf{checkTy} : (\Gamma : \mathsf{Con}) \to \mathsf{PTy} \to \mathsf{Ty}\,\Gamma$$

$$\mathsf{inferTm}\,\Gamma\,x := \mathsf{lookup}\,\Gamma\,x$$

$$\begin{aligned}
&\mathsf{inferTm}\,\Gamma\,(t\;u) := \mathsf{case}\,\mathsf{inferTm}\,\Gamma\;t\,\mathsf{of}\\
&\qquad\big((x : A') \to B', t'\big) \mapsto \mathsf{case}\,\mathsf{checkTm}\,\Gamma\;u\,A'\mathsf{of}\\
&\qquad\qquad\quad u' \mapsto \big(B'[x \mapsto u'], t'\;u'\big)
\end{aligned}$$

$$\begin{aligned}
&\mathsf{inferTm}\,\Gamma\,(t : A) := \mathsf{case}\,\mathsf{checkTy}\,\Gamma\,A\,\mathsf{of}\\
&\qquad A' \mapsto \mathsf{case}\,\mathsf{checkTm}\,\Gamma\;t\,A'\mathsf{of}\\
&\qquad\qquad\quad t' \mapsto (A', t')
\end{aligned}$$

$$\mathsf{checkTm}\,\Gamma\,(\lambda x.t)\,\big((x : A') \to B'\big) := \mathsf{lam}\,(\mathsf{checkTm}\,(\Gamma, x : A')\;t\;B')$$

$$\begin{aligned}
&\mathsf{checkTm}\,\Gamma\;t\,A := \mathsf{case}\,\mathsf{inferTm}\,\Gamma\;t\,\mathsf{of}\\
&\qquad (A', t') \mapsto \mathsf{if}\;A \stackrel{?}{=} A'\;\mathsf{then}\;t'
\end{aligned}$$

Models

Models formalised in Agda (with K and funext)

For a theory with Π, a base type and a family over the base type.

- Non-dependent eliminator:
  - standard model
  - presheaf model
  - setoid model
- Dependent eliminator:
  - logical predicate translation of Bernardy
  - presheaf logical predicate interpretation

Standard model

- A sanity check
- Every syntactic construct is interpreted as the corresponding metatheoretic construction.

$$
\begin{aligned}
&\mathsf{Con}_1 && := \mathsf{Set} \\
&\mathsf{Ty}_1\,[\![\Gamma]\!] && := [\![\Gamma]\!] \to \mathsf{Set} \\
&[\![\Gamma]\!],_1\,[\![A]\!] && := (\gamma : [\![\Gamma]\!]) \times [\![A]\!]\,\gamma \\
&\ldots && \\
&\Pi_1\,[\![A]\!]\,[\![B]\!]\,\gamma && := (x : [\![A]\!]\,\gamma) \to [\![B]\!]\,(\gamma, x) \\
&\mathsf{lam}_1\,[\![t]\!]\,\gamma && := \lambda x \to [\![t]\!]\,(\gamma, x) \\
&\ldots && \\
&\Pi\beta_1 && := \mathsf{refl}
\end{aligned}
$$

- We defined this for a syntax with Σ, $\bot$, $\top$, Bool, $\mathbb{N}$, Id as well.

Logical predicate interpretation

Parametricity expressed as a syntactic translation.

$$\frac{\Gamma \vdash}{\Gamma^{\mathsf{P}} \vdash} \qquad \frac{\Gamma \vdash A : \mathsf{U}}{\Gamma^{\mathsf{P}} \vdash A^{\mathsf{P}} : A \to \mathsf{U}} \qquad \frac{\Gamma \vdash t : A}{\Gamma^{\mathsf{P}} \vdash t^{\mathsf{P}} : A^{\mathsf{P}}\, t}$$

All of the following equations need to be well-typed (and preserve conversion).

$$\begin{aligned}
(\Gamma, x : A)^{\mathsf{P}} &:= \Gamma^{\mathsf{P}}, x : A, x^{M} : A^{\mathsf{P}}\, x \\
x^{\mathsf{P}} &:= x^{M} \\
\mathsf{U}^{\mathsf{P}}\, A &:= A \to \mathsf{U} \\
\big((x : A) \to B\big)^{\mathsf{P}}\, f &:= (x : A)(x^{M} : A^{\mathsf{P}}\, x) \to B^{\mathsf{P}}\, (f\, x) \\
(\lambda x.t)^{\mathsf{P}} &:= \lambda x\, x^{M}.t^{\mathsf{P}} \\
(f\, a)^{\mathsf{P}} &:= f^{\mathsf{P}}\, a\, a^{\mathsf{P}}
\end{aligned}$$

NBE for dependent types

Presheaf logical predicate

$$
\begin{aligned}
\mathsf{P}_\Delta &: \forall\Psi.\mathsf{Tms}\,\Psi\,\Delta \to \mathsf{Set}\\
\mathsf{P}_A &: \forall\Psi.(\rho : \mathsf{Tms}\,\Psi\,\Gamma) \to \mathsf{P}_\Gamma\,\rho \to \mathsf{Tm}\,\Psi\,A[\rho] \to \mathsf{Set}\\
\mathsf{P}_\sigma &: \forall\Psi.(\rho : \mathsf{Tms}\,\Psi\,\Gamma) \to \mathsf{P}_\Gamma\,\rho \to \mathsf{P}_\Delta\,(\sigma \circ \rho)\\
\mathsf{P}_t &: \forall\Psi.(\rho : \mathsf{Tms}\,\Psi\,\Gamma)(p : \mathsf{P}_\Gamma\,\rho) \to \mathsf{P}_A\,\rho\,p\,(t[\rho])
\end{aligned}
$$

At the base type:

$$\mathsf{P}_\iota\,\rho\,t = \mathsf{isNf}\,\Psi\,\iota\,t$$

Quote and unquote:

$$
\begin{aligned}
\mathsf{q}_A &: (p : \mathsf{P}_\Gamma\,\rho)(t : \mathsf{Tm}\,\Psi\,A[\rho]) \to \mathsf{P}_A\,\rho\,p\,t \to \mathsf{isNf}\,\Psi\,A[\rho]\,t\\
\mathsf{u}_A &: (p : \mathsf{P}_\Gamma\,\rho)(t : \mathsf{Tm}\,\Psi\,A[\rho]) \to \mathsf{isNe}\,\Psi\,A[\rho]\,t \to \mathsf{P}_A\,\rho\,p\,t
\end{aligned}
$$

NBE for a universe and Bool with large elimination (not yet formalised)

$$
\begin{aligned}
\mathsf{P}_\Delta &: \forall \Psi.(\rho : \mathsf{Tms}\,\Psi\,\Delta) \to (\mathsf{r} : \mathsf{Set}) \times (\mathsf{u} : \mathsf{isNes}\,\Psi\,\Delta\,\rho \to \mathsf{r}) \\
\mathsf{P}_A &: \forall \Psi.(\rho : \mathsf{Tms}\,\Psi\,\Delta) \to \mathsf{P}_\Gamma\,\rho.\mathsf{r} \to (t : \mathsf{Tm}\,\Psi\,A[\rho]) \\
&\qquad\qquad \to (\mathsf{r} : \mathsf{Set}) \times (\mathsf{q} : \mathsf{r} \to \mathsf{isNf}\,\Psi\,A[\rho]\,t) \\
&\qquad\qquad\qquad \times (\mathsf{u} : \mathsf{isNe}\,\Psi\,A[\rho]\,t \to r) \\
\mathsf{P}_t &: \forall \Psi.(\rho : \mathsf{Tms}\,\Psi\,\Delta) \to \mathsf{P}_\Gamma\,\rho.\mathsf{r} \to \mathsf{P}_A\,\rho\,q\,(t[\rho]).\mathsf{r}
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{P}_\mathsf{U}\,\Psi\,(\rho : \mathsf{Tms}\,\Psi\,\Gamma)(p : \mathsf{P}_\Gamma\,\rho)(\hat{A} : \mathsf{Tm}\,\Psi\,\mathsf{U}).\mathsf{r} \\
&:= \mathsf{isNf}\,\Psi\,\mathsf{U}\,\hat{A} \times \forall \Omega.(\beta : \mathsf{REN}(\Omega, \Psi))(t : \mathsf{Tm}\,\Omega\,(\mathsf{El}\,\hat{A}[\beta])) \\
&\quad \to (\mathsf{r} : \mathsf{Set}) \times (\mathsf{q} : \mathsf{r} \to \mathsf{isNf}\,\Omega\,(\mathsf{El}\,\hat{A}[\beta])\,t) \times (\mathsf{u} : \mathsf{isNe}\,\Omega\,(\mathsf{El}\,\hat{A}[\beta])\,t \to \mathsf{r})
\end{aligned}
$$

Summary

Quotient intrinsic syntax has the following properties:

- more abstract: close to categorical models; analogy with HITs and setoids (c.f. Peter Dybjer's talk)
- get back old-style syntax using general methods (WIP)
- typechecking and normalisation fit well
- definition of operations on the syntax in a type-safe way

Future work:

- finish unfinished things
- extend the syntax with QIITs to do full internalisation
- formalisation is hard
- we need cubical type theory or similar to compute with quotient types

Formalisation: `http://bitbucket.org/akaposi/tt-in-tt`