A general definition of DTT's

HoTTEST

October 9, 2011

Andrej Bauer

joint work with
Philipp Haselwarter
Peter LeFanu Lumsdaine

What are type theories?

A DTT is a formal system ....

- syntax expressions & types
- binding, dependencies
- judgments
- derivation

...

Methods for dealing with such formalisms:

- concrete syntax: words — obsolete
- abstract syntax: well-founded trees — reasonable ←
- intrinsic syntax: objects of meta-level formal system (logical framework, QIIT and type theory)

Steps:

1. Mathematical preliminaries
2. Raw syntax
3. Raw type theories
4. Acceptable TT
5. Well-founded TT

} definition

---

6. Results:
   - meta theorems ←
   - examples ←→
   - semantics
   - implementation

Martin-Löfian:

→ $\vdash A$ type $\vdash A \equiv B$

$\Gamma \vdash t : A$ $\vdash t \equiv_A s$

↓

Examples: MLTT84 MLTT .....
CIC? HoTTbook TT
(universes a la Tarski)

Counter-examples:

→ Cubicall TT

→ $\Gamma ; \Delta \vdash$ ....
↑ linear

Raw syntax

- Free & bound variables? Pick some standard way, e.g. de Bruijn indices

Γ $x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n$

γ $x_1 : \square, x_2 : \square, \ldots, x_n : \square$ shape

$|\gamma|$ positions

$0 : \square, 1 : \square, \ldots, n : \square$

Signature: describe symbols of TT

arity

Π : (ty, [(ty,0), (ty,1)])

λ : (tm, [(ty,0), (ty,1), (tm,1)])

app : (tm, [(ty,0), (ty,1), (tm,0), (tm,0)])

Syntactic classes: ty, tm

+ : 2

1 : 0

Symbols $S \in \{tm, ty\}$

$S \mapsto (cl_S, \alpha_S)$

$\Pi(A, B)$

$\lambda x.x$

$\lambda\ A\ B\ e$

app A B e_1 e_2

$$\frac{\vdash A\ \text{type} \qquad x : A \vdash B(x)\ \text{type} \qquad y : \text{List}(A) \vdash e(y) : B(y)}{\vdash \lambda\ A\ (\{x\}.B(x))\ (\{y\}.e(y)) : \Pi(A, \{z\}B(z))}$$

$$\Pi \mapsto (\mathsf{ty}, [(\mathsf{ty}, 0), (\mathsf{ty}, 1)]), \qquad \lambda\ \mathbb{N}\ (\{x\}.\mathrm{Vec}(x+3))\ (\{y\}.\mathrm{zero}(y+7))$$

$$\lambda \mapsto (\mathsf{tm}, [(\mathsf{ty}, 0), (\mathsf{ty}, 1), (\mathsf{tm}, 1)]),$$

$$\mathsf{app} \mapsto (\mathsf{tm}, [(\mathsf{ty}, 0), (\mathsf{ty}, 1), (\mathsf{tm}, 0), (\mathsf{tm}, 0)])$$

signature

Definition 3.5. The *raw syntax* over Σ consists of the collections of *raw type expressions* $\mathrm{Expr}^{\mathsf{ty}}_{\Sigma}(\gamma)$ and *raw term expressions* $\mathrm{Expr}^{\mathsf{tm}}_{\Sigma}(\gamma)$, which are generated inductively for any shape γ as follows:

(1) for every position $i \in |\gamma|$, there is a *variable* expression $\mathrm{var}_i \in \mathrm{Expr}^{\mathsf{tm}}_{\Sigma}(\gamma)$;

(2) for every symbol $S \in \Sigma$ of syntactic class c_S, and a map

$$e \in \textstyle\prod_{i \in \arg S} \mathrm{Expr}^{\mathrm{cl}_S\, i}_{\Sigma}(\gamma \oplus \mathrm{bind}_S\, i),$$

there is an expression $S(e) \in \mathrm{Expr}^{\mathrm{c}_S}_{\Sigma}(\gamma)$, the *application* of S to arguments e.

Raw type theories

$\mathbb{T}$ (raw type theory) : over a signature Σ
family of rules

What is a rule?

$$\frac{\vdash A\ \mathrm{type} \qquad x{:}A \vdash B(x)\ \mathrm{type}}{\vdash \Pi(A,B)\ \mathrm{type}}$$

premises → ; conclusion →

$\mathcal{R}$:
- info about meta-variables
- a family of premises (judgements)
- conclusion (judgement)

$\gamma \quad i \in \gamma \quad \Gamma_i \in \mathrm{Expr}^{ty}_{\Sigma + \alpha_R}(\gamma)$

$\Gamma \vdash \mathcal{J}$ $\qquad [0 : B(\mathrm{var}_0)]$

(5)
$$\frac{\vdash A\ \mathrm{type} \qquad [0{:}\,A] \vdash B(\mathrm{var}_0)\ \mathrm{type}}{\vdash f : \Pi(A, B(\mathrm{var}_0)) \qquad \vdash a : A \qquad \vdash a : \Pi(A, B(\mathrm{var}_0))}$$
$$\vdash \mathrm{app}(A, B(\mathrm{var}_0), f, a) : B(a)$$

$A \mapsto (ty, [\,])$
$B \mapsto (ty, [(tm, 0)])$
$f \mapsto (tm, [\,])$
$a \mapsto (tm, [\,])$

$\Sigma + \alpha_R$ ↑ meta-variables as symbols "local to the rule"

(2)
$$\frac{\vdash A\ \mathrm{type} \qquad [0{:}\,A] \vdash B(\mathrm{var}_0)\ \mathrm{type} \qquad \vdash f : \Pi(A, B(\mathrm{var}_0)) \qquad \vdash a : A}{\vdash \mathrm{app}(A, B(\mathrm{var}_0), f, a) : B(a)}$$

(3)
$$\frac{\vdash A\ \mathrm{type} \qquad [0{:}\,A] \vdash B(\mathrm{var}_0)\ \mathrm{type} \qquad \vdash f : A \qquad \vdash a : A}{\vdash \mathrm{app}(A, B(\mathrm{var}_0), f, a) : B(a)}$$

(4)
$$\frac{\vdash A\ \mathrm{type} \qquad [0{:}\,A] \vdash B(\mathrm{var}_0)\ \mathrm{type} \qquad \vdash f : \Pi(A, B(\mathrm{var}_0)) \qquad \vdash a : \Pi(A, B(\mathrm{var}_0))}{\vdash \mathrm{app}(A, B(\mathrm{var}_0), f, a) : B(a)}$$
A

$$\left.\begin{array}{l}\Gamma \vdash A \text{ type}\\ \Gamma \vdash B \text{ type}\\ \Gamma \vdash t : A\\ \Gamma \vdash t : B\end{array}\right\} \quad \Gamma \vdash A \equiv B$$

uniqueness of typing

$$\frac{}{\vdash u : \mathrm{El}(u)} \qquad \frac{\vdash a : \mathrm{El}(u)}{\vdash \mathrm{El}(a) \text{ type}}$$

u $\quad$ El

u = the code of the universe

El decodes codes to types

$$\frac{}{\vdash u_i : \mathrm{El}_{i+1}(u_{i+1})} \qquad \frac{\vdash a : \mathrm{El}_i(u_i)}{\vdash \mathrm{El}_i(a) \text{ type}}$$

$$\frac{\vdash A \text{ type}}{\vdash \mathrm{List}(A) \text{ type}}$$

$$\frac{\vdash A \equiv B}{\vdash \mathrm{List}\ A \equiv \mathrm{List}\ B}$$

$$\left.\begin{array}{l}\vdash A_1 \text{ type} \quad \vdash A_2 \text{ type} \qquad x_1 : A_1 \vdash B_1(x_1) \text{ type} \quad x_2 : A_2 \vdash B_2(x_2) \text{ type}\\ \vdash A_1 \equiv A_2 \qquad\qquad x_1 : A_1 \vdash B_1(x_1) \equiv B_2(x_1)\end{array}\right\} \text{ congruence rule}$$

$$\vdash \Pi(x_1 : A_1).\, B_1(x_1) \equiv \Pi(x_2 : A_2).\, B_2(x_2)$$

$$\frac{\mathcal{D}}{\Gamma \vdash t : A} \quad \rightsquigarrow \quad \frac{\mathcal{D}'}{\Gamma \vdash A \text{ type}}$$

Substitution elimination

Theorem: Suppose `T` is a congruous type theory. If `T` derives a judgement `Γ ⊢ J`, then it also derives `J` without any applications of the substitution rules.

Derivability of presuppositions

Theorem: Suppose `T` is presuppository. If `T` derives a judgement `Γ ⊢ J`, then it also derives its presuppositions.

Uniqueness of typing

Theorem: Suppose `T` is tight. If `T` derives `Γ ⊢ A type`, `Γ ⊢ B type`, `Γ ⊢ t : A`, and `Γ ⊢ t : B`, then it also derives `Γ ⊢ A ≡ B`.

Inversion principles.