Semisimplicial types

Ambrus Kaposi

FP Lab Away Day
2013

Goal

- Make dependently typed languages easy to use for programmers, mathematicians
- Ony aspect of this:
  - Find the right type theory
  - A candidate is Homotopy Type Theory

Why HoTT?

- Extensionality: $\forall x \,.\, f\,x = g\,x \to f = g$
- Transport of structures along isomorphisms:
  - Fin n $\cong \Sigma\,(i : \mathbb{N})\,(i < n) \to$ Fin n $= \Sigma\,(i : \mathbb{N})\,(i < n)$
    - If we have a monoid structure on Fin n, we get a monoid structure on $\Sigma\,(i : \mathbb{N})\,(i < n)$ as well
- Higher Inductive Types
  - Quotients
- This extension of TT seems natural and supported by Voevodsky

Let's implement it!

- MLTT

$$\frac{\Gamma \vdash}{1 : \Gamma \to \Gamma} \qquad \frac{\sigma : \Delta \to \Gamma \quad \delta : \Theta \to \Delta}{\sigma\delta : \Theta \to \Gamma}$$

$$\frac{\Gamma \vdash A \quad \sigma : \Delta \to \Gamma}{\Delta \vdash A\sigma} \qquad \frac{\Gamma \vdash t : A \quad \sigma : \Delta \to \Gamma}{\Delta \vdash t\sigma : A\sigma} \qquad \frac{\Gamma \vdash F : (A)\mathsf{Type} \quad \sigma : \Delta \to \Gamma}{\Delta \vdash F\sigma : (A\sigma)\mathsf{Type}}$$

$$\frac{}{() \vdash} \qquad \frac{\Gamma \vdash \quad \Gamma \vdash A}{\Gamma.A \vdash} \qquad \frac{\Gamma \vdash A}{\mathsf{p} : \Gamma.A \to \Gamma} \qquad \frac{\Gamma \vdash A}{\Gamma.A \vdash \mathsf{q} : A\mathsf{p}}$$

$$\frac{\sigma : \Delta \to \Gamma \quad \Gamma \vdash A \quad \Delta \vdash u : A\sigma}{(\sigma, u) : \Delta \to \Gamma.A}$$

$$\frac{\Gamma \vdash A \quad \Gamma.A \vdash B}{\Gamma \vdash \lambda B : (A)\mathsf{Type}} \qquad \frac{\Gamma \vdash F : (A)\mathsf{Type} \quad \Gamma \vdash a : A}{\Gamma \vdash \mathsf{app}(F, a)}$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash F : (A)\mathsf{Type}}{\Gamma \vdash \mathsf{Fun}\ A\ F} \qquad \frac{\Gamma.A \vdash b : \mathsf{app}(F\mathsf{p}, \mathsf{q})}{\Gamma \vdash \lambda b : \mathsf{Fun}\ A\ F} \qquad \frac{\Gamma \vdash w : \mathsf{Fun}\ A\ F \quad \Gamma \vdash u : A}{\Gamma \vdash \mathsf{app}(w, u) : \mathsf{app}(F, u)}$$

$$1\sigma = \sigma = \sigma 1 \qquad (\sigma\delta)\nu = \sigma(\delta\nu) \qquad 1 = (\mathsf{p}, \mathsf{q})$$

$$(\sigma, u)\delta = (\sigma\delta, u\delta) \qquad \mathsf{p}(\sigma, u) = \sigma \qquad \mathsf{q}(\sigma, u) = u$$

$$(A\sigma)\delta = A(\sigma\delta) \qquad A1 = A \qquad (a\sigma)\delta = a(\sigma\delta) \qquad a1 = a$$

$$\mathsf{app}(w, u)\delta = \mathsf{app}(w\delta, u\delta) \qquad \mathsf{app}(F, u)\delta = \mathsf{app}(F\delta, u\delta) \qquad (\mathsf{Fun}\ A\ F)\sigma = \mathsf{Fun}(A\sigma)(F\sigma)$$

$$\mathsf{app}((\lambda b)\sigma, u) = b(\sigma, u) \qquad \mathsf{app}((\lambda B)\sigma, u) = B(\sigma, u)$$

- Canonicity, type checking terminates

- Adding new rules

$$\frac{\Gamma \vdash p : \mathsf{Fun}\ A\ (\lambda \mathsf{Eq}_{\mathsf{app}(F\mathsf{p},\mathsf{q})}\ \mathsf{app}(f\mathsf{p}, \mathsf{q})\ \mathsf{app}(g\mathsf{p}, \mathsf{q}))}{\Gamma \vdash \mathsf{ext}\ p : \mathsf{Eq}_{\mathsf{Fun}\ A\ F}\ f\ g}$$

  …

- Makes canonicity go away, we no longer have that if n : ℕ, then n ≡ zero or n ≡ suc m, it might be
  n ≡ fun (subst (ext p))
- (But we have consistency)

Towards a solution

- Try to figure out what the elimination rules might be
- Find a model in MLTT!
  - Object theory (HoTT) $\longrightarrow$ Metatheory (MLTT)
    Bool $\longmapsto$ $[\![\ \text{Bool}\ ]\!] :\equiv \text{Maybe Bool}$
    $\lambda x.t : A \rightarrow B$ $\longmapsto$ $[\![\ \lambda x.t\ ]\!] :\equiv \text{Just}\ [\![\ t\ ]\!]$
    ext $\longmapsto$ ...
    …
    $a \equiv b$ $\longmapsto$ $[\![\ a\ ]\!] \equiv [\![\ b\ ]\!]$
  - The last rule ensures canonicity of the object theory

Example (Takeuti, Gandy)

- Simple type theory with extensionality

  $[\![\text{Type}]\!] :\equiv \Sigma\ (A : \text{Type})\ (\_\sim\_ : A \to A \to \text{Type})$

  $[\![A \to B]\!] :\equiv \Sigma\ (f : [\![A]\!] \to [\![B]\!])$
  $\quad (\forall\ x\ u : [\![A]\!]\ .\ x \sim u \to f\ x \sim f\ u)$

  $(f,f') \sim (g,g') :\equiv \forall\ x\ u : [\![A]\!]\ .\ x \sim u \to f\ x \sim g\ u$

- Equality is defined recursively as _~_
- Reflexivity, symmetry, transitivity of _~_ can be proved
- This is a proof that every function is extensional viewed as model construction

Generalisation

A : Type

A : Type

$\_\sim_A\_ : A \to A \to \mathrm{Type}$

$\_\sim_{x\sim y}\_ : x\sim_A y \to x\sim_A y \to \mathrm{Type}$

$\_\sim_{p\sim q}\_ : p\sim_{x\sim y} q \to p\sim_{x\sim y} q \to \mathrm{Type}$

…

laws of $\_\sim_A\_$: refl, sym, trans

laws of $\_\sim_{x\sim y}\_$: [left|right]-id, assoc

laws of $\_\sim_{p\sim q}\_$: complicated

...

Kan Simplicial sets

- Model by Voevodsky
- Consistency of HoTT with regards ZFC
- Non constructive

Semisimplicial types

- Another kind of generalisation:
  A0 : Type

  A1 : A0 → A0 → Type

  A2 : {a0 a1 a2 : A0} → A1 a0 a1 → A1 a0 a2
  → A1 a1 a2 → Type

  A3 : {a0 a1 a2 a3 : A0}
  {a01 : A1 a0 a1} {a02 : A1 a0 a2} {a03 : A1 a0 a3}
  {a12 : A1 a1 a2} {a13 : A0 a1 a3} {a23 : A1 a2 a3}
  → A2 a01 a02 a12 → A2 a01 a03 a13
  → A2 a02 a03 a23 → A2 a12 a13 a23 → Type

  ...

Kan Semisimplicial types

- Draw!
- Completion operators, filling operators
- Model by Thierry Coquand
  - The untruncated version is not yet formalized
  - The truncated version was formalized

small types $\mapsto$ type of points A, edges ηA, completion, filling for the first level, completion for second level (setoids)

Weak MLTT

- Extensionality and univalence are valid in this model, however the rule
  $t \equiv u$ entails $\lambda x.t \equiv \lambda x.u$
  doesn't hold
- Martin-Lof argues that this does not formalise the informal notion of definitional equality correctly.
  - unfolding definitions
  - refl, sym, trans
  - Preservation under substitution:
    $a \equiv b$ entails $u[x<-a] \equiv u[x<-b]$

TODO

- Formalise the truncated version in Agda
- Formalise semisimplicial types (Nicolai, Nuo, Paolo)
- Understand
- Weakness?
- MLTT in MLTT with definitional equality?