Towards higher models and syntax of type theory

jww Paolo Capriotti, Ambrus Kaposi, Nicolai Kraus

Thorsten Altenkirch

Functional Programming Laboratory
School of Computer Science
University of Nottingham

May 11, 2018

Type Theory in Type Theory

- Plan: develop the metatheory of type theory.
- What language should we use for this?
- Type Theory!

Extrinsic Syntax

- Common presentation of type theory:
  - Sets of preterms (t), precontexts (Γ) and pretypes (A),...
  - Inductively defined typing relations include
    - Context validity $\vdash \Gamma$
    - Type validity $\Gamma \vdash A$
    - Typing $\Gamma \vdash t : A$
    - Convertibility of terms $\Gamma \vdash t \equiv t' : A$
    - Convertibility of types $\Gamma \vdash A \equiv A'$
- From this we can derive e.g. typable terms

$$\mathrm{Tm}_0(\Gamma, A) = \{t \mid \Gamma \vdash t : A\}$$

- And quotient them by derivable equality

$$\mathrm{Tm}(\Gamma, A) = \mathrm{Tm}_0(\Gamma, A)/(\lambda t, t'.\Gamma \vdash t \equiv t' : A)$$

Intrinsic syntax

- Why do we define untyped objects, if we are only interested in typed ones?
- The extrinsic approach is conceptually misleading and justifies many unnecessary complicated developments.
- Instead, we can use *intrinsic syntax*: we only define the typed terms.
- Even better: using equality constructors we can also build in the conversion relation.
- We use Quotient Inductive Inductive Types (QIITs), that is mutually defined HITs, which are set-truncated.

POPL 2016

Type theory in type theory using quotient inductive types
TA, Ambrus Kaposi

Type Theory in Type Theory as a QIIT

$$
\begin{aligned}
\mathrm{Con} &: \textbf{Set} \\
\mathrm{Ty} &: \mathrm{Con} \to \textbf{Set} \\
\mathrm{Tm} &: \Pi\Gamma : \mathrm{Con}.\mathrm{Ty}(\Gamma) \to \textbf{Set} \\
\mathrm{Tms} &: \mathrm{Con} \to \mathrm{Con} \to \textbf{Set} \\
&\vdots \\
\mathrm{Pi} &: \Pi A : \mathrm{Ty}(\Gamma), B : \mathrm{Ty}(\Gamma.A).\mathrm{Ty}(\Gamma) \\
&\vdots \\
\mathrm{lam} &: \mathrm{Tm}(\Gamma.A, B) \to \mathrm{Tm}(\Gamma, \mathrm{Pi}(A, B)) \\
\mathrm{app} &: \mathrm{Tm}(\Gamma, \mathrm{Pi}(A, B)) \to \mathrm{Tm}(\Gamma.A, B) \\
&\vdots \\
\beta &: \Pi t : \mathrm{Tm}(\Gamma.A, B).\mathrm{app}(\mathrm{lam}(t)) = t
\end{aligned}
$$

Categories with families

A category with families (CwF) is given by:

- A category of contexts and substitutions **Con**.
- A presheaf of types $\mathbf{Ty} : \mathbf{Con}^{\mathrm{op}} \to \mathbf{Type}$
- A presheaf of terms over contexts and types $\int \mathbf{Ty}^{\mathrm{op}} \to \mathbf{Type}$
- A terminal object in **Con**.
- For any $A : \mathbf{Ty}(\Gamma)$, the presheaf

$$\Delta \mapsto \Sigma f : \mathbf{Con}(\Delta, \Gamma).A[f]$$

  is representable.
- For Π-types: ...

The QIIT defines the initial CwF. The *initiality theorem* is trivial.

Decidability

- We can show that all the sets (and families) we define have a *decidable equality*.
- To do this we employ a semantic normalisation proof: normalisation by evaluation (nbe).
- The main idea is to show that evaluation into the CwF of presheaves over the category of contexts with projections is invertible.

FSCD 2016

Normalisation by Evaluation for Dependent Types
TA, Ambrus Kaposi

The truncation problem

- We would like to define the standard semantics of type theory, interpreting types as sets or types.
- However, it is not clear how to do this since we have explicitly truncated the syntax.
- And **Set** is not a set (in the sense of HoTT)!
- In our paper we replace set with an inductive-recursive universe, this is an intensional universe, it is not univalent.
- This is unsatisfying, we would like to interpret the syntax in semantic (i.e. univalent) models.

An analogy using $\mathbb{Z}$

We can model the integers as the following QIT:

$$\begin{aligned}
0 &: \mathbb{Z} \\
\mathrm{suc} &: \mathbb{Z} \to \mathbb{Z} \\
\mathrm{pred} &: \mathbb{Z} \to \mathbb{Z} \\
\mathrm{sucpred} &: \Pi i : \mathbb{Z}. \mathrm{suc}\,(\mathrm{pred}\, i) =_{\mathbb{Z}} i \\
\mathrm{predsuc} &: \Pi i : \mathbb{Z}. \mathrm{pred}\,(\mathrm{suc}\, i) =_{\mathbb{Z}} i \\
\mathrm{isSet} &: \Pi i, j : \mathbb{Z}. \Pi p, q : i =_{\mathbb{Z}} j \to p =_{i =_{\mathbb{Z}} j} q
\end{aligned}$$

- We can show that this set has a decidable equality by normalising into signed integers.
- However, because we truncated we can only eliminate into sets.

An analogy using $\mathbb{Z}$

We can overcome this problem by replacing isSet by a coherence.
(suggested by Paolo Capriotti)

$$
\begin{aligned}
0 &: \mathbb{Z} \\
\mathrm{suc} &: \mathbb{Z} \to \mathbb{Z} \\
\mathrm{pred} &: \mathbb{Z} \to \mathbb{Z} \\
\mathrm{sucpred} &: \Pi i : \mathbb{Z}.\mathrm{suc}\,(\mathrm{pred}\, i) =_{\mathbb{Z}} i \\
\mathrm{predsuc} &: \Pi i : \mathbb{Z}.\mathrm{pred}\,(\mathrm{suc}\, i) =_{\mathbb{Z}} i \\
\mathrm{coh} &: \mathrm{sucpred}\,(\mathrm{suc}\, i) = \mathrm{resp}\,\mathrm{suc}\,(\mathrm{predsuc}\, i)
\end{aligned}
$$

- Effectively we are saying that suc is an equivalence.
- The eliminator is more flexible because we can eliminate into non-sets (we do have to verify the coherence condition).
- We can still normalize, hence our integers are still a set (and indeed equivalent to the truncated definition).

Can we do something like this for type theory?

1. Define higher CwF with coherence conditions.
2. Construct an initial higher CwF using HIITs.
3. Do the NbE construction for the initial higher CwF (the coherence conditions should hold in the presheaf model).
4. As a consequence the contexts and types in the initial CwF are still sets.
5. We have gained a more powerful elimination principle, allowing us to evaluate into semantic (univalent) models.

Higher Categories with families

A higher category with families (HCwF) is given by:

- A $(\infty, 1)$-category of contexts and substitutions **Con**.
- A higher presheaf of types $\mathbf{Ty} : \mathbf{Con}^{\mathrm{op}} \to \mathbf{Type}$, note that **Type** is an $(\infty, 1)$-category.
- A presheaf of terms over contexts and types $\int \mathbf{Ty}^{\mathrm{op}} \to \mathbf{Type}$. We need to explain $\int$ for higher presheaves.
- A terminal object in **Con**.
- For any $A : \mathbf{Ty}(\Gamma)$, the higher presheaf

  $$\Delta \mapsto \Sigma f : \mathbf{Con}(\Delta, \Gamma).A[f]$$

  is representable.
- For Π-types: ...

1st step

What is an $(\infty, 1)$-category in Type Theory?

Semisimplicial types

A semisimplicial type X is an infinite sequence

$$
\begin{aligned}
X_0 &: \textbf{Type} \\
X_1 &: X_0 \to X_0 \to \textbf{Type} \\
X_2 &: \Pi_{x_0, x_1, x_2 : X_0} X_1(x_0, x_1) \to X_1(x_1, x_2) \to X_1(x_0, x_2) \to \textbf{Type} \\
\vdots & \quad \vdots
\end{aligned}
$$

- We don't know how to fill in the $\vdots$ in plain HoTT (open problem).
- However, we can define the approximations upto n in a 2-level system.
- We can then define the type of semisimplicial types as the limit (assuming that the strict natural numbers are fibrant).

CSL 2016

Extending Homotopy Type Theory with Strict Equality
TA, Paolo Capriotti and Nicolai Kraus

$(\infty, 1)$-semicategories

To define $(\infty, 1)$-semicategories we impose the *Segal*-condition:
The canonical map from the n-simplex to the n-spine is an equivalence
By the *n*-spine we mean

$$\Sigma x_0, x_1, \ldots x_n : X_0, X_1(x_0, x_1) \times X_1(x_1, x_2) \times \ldots X_1(x_{n-1}, x_n)$$

So for example we say that the projection

$$\begin{aligned} &\Sigma_{x_0,x_1,x_2:X_1}, x_{01} : X_1(x_0, x_1), x_{12} : X_1(x_1, x_2), x_{02} : X_1(x_0, x_2). \\ &\qquad X_2(x_{01}, x_{12}, x_{02}) \\ \to &\Sigma_{x_0,x_1,x_2:X_1}, x_{01} : X_1(x_0, x_1), x_{12} : X_1(x_1, x_2) \end{aligned}$$

is an equivalence.

$(\infty, 1)$-~~semi-~~categories

- How to add the identities (degeneracies) ?
- It is not obvious how to define even simplicial types upto n.
  We would have to add equalities which trigger higher coherences.
- Instead we can add univalence, which says that

  $$\Sigma x_1 : X_0, f : X_1(x_0, x_1), \mathrm{isEquivalence}(f)$$

  is contractible for any $x_0 : X_1$.
- Univalent $(\infty, 1)$-semicategories have degeneracies (and hence are (univalent) $(\infty, 1)$-categories).

POPL 18

Univalent Higher Categories via Complete Semi-Segal Types
Paolo Capriotti and Nicolai Kraus

Univalence?

- Univalent categories can only have sets of objects if they have no non-trivial equivalences.
- This will not be the case for the initial (higher) CwF.
- E.g. two contexts that are equivalent are not equal in the syntax.

Direct replacement

- The problem is that Δ (the simplicial category) is not inverse unlike Δ^+ (the semisimplicial category).
- A homotopical category has marked equivalences and functors between them have to preserve them.
- Kraus and Sattler present a homotopical category $\mathfrak{D}$ which is inverse and whose homotopy category is Δ (inverting all marked equivalences).
- The replacement of a finite part of Δ is still finite.

arXiv:1704.04543

Space-Valued Diagrams, Type-Theoretically
Nicolai Kraus, Christian Sattler

A sketch of $\mathfrak{D}$

$$
\begin{tikzcd}
(1) \arrow[r, shift left=1] \arrow[r, shift right=1] \arrow[d, dashed] \arrow[dr, shift left=1] \arrow[dr, shift right=1] & (1,1) \arrow[r, shift left=2] \arrow[r] \arrow[r, shift right=2] \arrow[dl] \arrow[d, dashed, shift right=1] \arrow[d, shift left=1] & (1,1,1) \arrow[dl] \\
(2) \arrow[r] \arrow[d, dashed] & (2,1) \arrow[dl] & \\
(3) & (1,2) &
\end{tikzcd}
$$

Simplicial types (Reedy limit of $\mathfrak{D}$)

$$
\begin{aligned}
X_1 &: \textbf{Type} \\
X_{11} &: X_1 \to X_1 \to \textbf{Type} \\
X_{111} &: \Pi_{x_0,x_1,x_2:X_1} X_{11}(x_0,x_1) \to X_{11}(x_1,x_2) \to X_{11}(x_0,x_2) \to \textbf{Type} \\
X_2 &: \Pi_{x_0:X_1} X_{11}(x_0,x_0) \to \textbf{Type} \\
c_2 &: \Pi x_0 : X_1.\mathrm{isContr}(\Sigma x_{00} : X_{11}(x_0,x_0).X_2(x_{00})) \\
X_{21} &: \Pi_{x_0,x_1:X_1} x_{00} : X_{11}(x_0,x_0), x_{01} : X_{11}(x_0,x_1).X_2(x_0) \\
&\quad \to X_{111}(x_{00},x_{01},x_{01}) \to \textbf{Type} \\
c_{21} &: \Pi_{x_0,x_1:X_1} x_{01} : X_{11}(x_0,x_1).\mathrm{isContr}(\Sigma x_{00} : X_{11}(x_0,x_0), \\
&\quad x_2 : X_2(x_0), x_{001} : X_{111}(x_{00},x_{01},x_{01}.X_{21}(x_{01},x_{00},x_2,x_{001})) \\
&\vdots \quad \vdots
\end{aligned}
$$

(non-univalent) $(\infty, 1)$-categories

- As for semisimplicial types we can define simplicial types in a 2-level type theory using $\mathfrak{D}$ instead of Δ.
- We define a $(\infty, 1)$-category to be a simplicial type with the Segal condition.
- **Type** (Types and functions) is a strict category, hence its nerve is a strict diagram over Δ and hence (by fibrant replacement) a simplicial type.
- Morphisms between $(\infty, 1)$-categories are morphisms between the simplicial types which can be defined level-wise.
- Hence we can define higher presheaves over $(\infty, 1)$-categories.

Next steps

- To define the category of elements, we need to define the universe of simplicial types.
- One we have done this we should be able to define higher CwFs.

Higher Syntax

- The idea is to define approximations up to level n as a HIIT.
- We can then take the colimit of these approximations and embeddings as the definition of the syntax.
- We need to show that the constructors in the approximations lift to the colimit.
- This forms a HCwF which is the syntax of higher type theory.
- It would be interesting but not essential to show that this is initial in the $(2, \infty)$-category of HCwFs.