Towards an $\omega$-groupoid model of Type Theory
Based on joint work with Ondrej Rypacek

Thorsten Altenkirch

Functional Programming Laboratory
School of Computer Science
University of Nottingham

August 14, 2012
Background

- In Type Theory for any $A : Type$ and $a \, b : A$ we can form a new type $a = b : Type$, the set of proofs that $a$ is equal to $b$.
- The canonical way to prove an equality is $refl : a = a$.
- Using the standard eliminator (J) we can show that equality is a congruence.
- Since $refl$ is the only constructor we would assume that all equality proofs are equal (*uniqueness of equality proofs*).
- However, this is not provable using the standard eliminator (J).
- This was shown by Hofmann and Streicher using the *Groupoid model* of Type Theory.
Voevodsky proposed an interpretation of Type Theory using Homotopy Theory.

Types are interpreted as topological spaces and equality proofs as paths (homotopies).

This interpretation doesn’t support uniqueness of equality proofs, i.e. \((\alpha : a = a) \rightarrow \alpha = \text{refl}\) is not provable.

However, it does support the standard eliminator (J), in particular we can prove: that given \(a : A\) for all \(p : (b : A) \times (a = b)\) we have \(p = (a, \text{refl})\).
Dimensions (Homotopy levels)

- We say that a type is *contractible* or 0-dimensional, if it contains exactly one element, i.e. there is \((a : A) \times (b : A) \rightarrow a = b\).
- A type is \(n + 1\)-dimensional, if all its equalities are \(n\)-dimensional.
- We arrive at the following hierarchy:
  - 0 contractible types
  - 1 propositions
  - 2 sets
  - 3 ???
- We can show that if a type is \(n\)-dimensional then it is also \(n + 1\)-dimensional.
- Uniqueness of equality proofs means that all types are 2-dimensional.
Weak equivalence

- The notion of *weak equivalence* can be expressed in Type Theory.
- A function $f : A \rightarrow B$ is a weak equivalence if the type $(a : B) \times f \ a = b$ is contractible for $b : B$.
- $A$ and $B$ are weakly equivalent.
- Weak equivalence in different dimensions:
  0 contractible types trivial
  1 propositions logical equivalence
  2 sets isomorphism
  3 ??? weak equivalence
- Univalence axiom (Voevodsky): Weak equivalence is weakly equivalent to equality.
- Univalence implies functional extensionality (Voevodsky): Any two functions which are pointwise equal are equal.
Why are we interested in this?

- We have found a fascinating connection between Type Theory and Homotopy Theory.
- We can use Type Theory to formalize constructions in Homotopy Theory.
- However, most Computer Scientists don’t care about Homotopy theory.
- Is there a way to motivate the univalence axiom which has nothing to do with homotopy theory?
Extensionality

Leibniz principle

Any two objects should either have a property which distinguishes them or they should be equal.

- This principle justifies functional extensionality (black box view of functions).
- Isomorphic sets cannot be distinguished in Type Theory - hence they should be equal.
- Isomorphism is not the correct notion from dimension 3 because it lacks a coherence property.
- This is fixed by weak equivalence (being a weak equivalence is propositional while being an isomorphism is not.
- Note that the Leibniz principle is not satisfied by Extensional Type Theory.
Open problem

Canonicity

Any closed term inhabiting a datatype (like \( \mathbb{N} \)) should be definitionally (strictly) equal to a term in constructor form (starting with a constructor).

- This is justified by Intensional Type Theory due to the normalisation property.
- Assuming univalence destroys canonicity.
- How can we have the univalence principle and keep canonicity?
- How can we eliminate univalence?
Deja vue

- This is reminiscent to the problem of eliminating functional extensionality in Type Theory.

\[
\begin{align*}
\text{ext} : & (f, g : (x : A) \to B x) \\
& \to (x : A \to f x = g x) \to f = g
\end{align*}
\]

- We have proposed a solution to this problem (LICS 99) which relies on a translation using the *Setoid model*.
- This was later (PLPV 08) refined in joint work with Conor McBride and others (*Observational Type Theory*).
- However, the construction relies on a strong form of proof irrelevance.
Sketch of the construction

- We define a translation from a source type theory to a target type theory.
- The target type theory doesn’t have a equality types.
- The source type theory does have equality types and ext is inhabited. This is explained by the translation.
- The translation preserves datatypes (like $\mathbb{N}$) and hence canonicity holds.
- The target type theory features a universe of (strictly) proof-irrelevant propositions $Prop$. 
Prop

- Prop is a subuniverse of Set, i.e. $P : Prop$ implies $P : Set$.
- If $P : Prop$ and $p \ q : P$ then $p$ and $q$ are definitionally equal.
- Prop is closed under Π- and Σ-types. That is
  - If $A : Set$ and $P : A \to Prop$ then $(x : A) \to P x : Prop$
  - If $P : Prop$ and $Q : P \to Prop$ then $(x : P) \times Q x : Prop$.
  - $\top : Prop$ and $\bot : Prop$. 
The translation

- A closed type $A : \text{Set}$ in the source theory is translated as a setoid in the target theory, i.e.
  - $A.\text{set} : \text{Set}$ a type in the target theory
  - $\sim A : A.\text{set} \rightarrow A.\text{set} \rightarrow \text{Prop}$ a relation
  - Proofs of $\text{refl}$, $\text{sym}$, $\text{trans}$.

- A family of types $B : A \rightarrow \text{Set}$ is modelled as a functor from $(A.\text{set}, \sim A)$ into the category of Setoids, i.e.
  - A family $B.\text{fam} : A.\text{set} \rightarrow \text{Setoid}$ in the target theory.
  - A term $\text{subst} : (a \sim A a') \rightarrow (B.\text{fam} a) .\text{set} \rightarrow (B.\text{fam} a') .\text{set}$.
  - Proofs that $\text{subst}$ is functorial upto setoid equality.
Translating \( \Pi \)-types

- Given a setoid \( A : \text{Set} \) and a family of setoids \( B : A \rightarrow \text{Set} \) we construct a setoid \( \Pi A B : \text{Set} \)
  - \( (\Pi A B) \cdot \text{set} \) is the set of functions which preserve equality, i.e.
    \[
    (f : (x : A.\text{set}) \rightarrow (B.\text{fam } x) \cdot \text{set}) \times ((p : x \sim A y) \rightarrow B.\text{subst } p (f x) \sim (B.\text{fam } y) f y)
    \]
  - Equality \( f \sim (\Pi A B) g \) is extensional equality, i.e.
    \[
    (x : A.\text{set}) \rightarrow f x \sim (B.\text{fam } x) g x
    \]
- In fact we have to generalize this construction to the case \( B : A \rightarrow \text{Set} \) and \( C : (a : A) \times B a \rightarrow \text{Set} \) leading to \( \Pi B C : A \rightarrow \text{Set} \).
Interpreting equality

- In the source type theory we interpret equality using $\sim A$.
- Equality for $\Pi$-types is extensional by definition.
- We can derive the eliminator from $\text{subst}$ and the fact that equality is definitionally proof-irrelevant.
- $\mathbb{N}$ is translated by itself using the definable recursive equality on natural numbers.
- We can also interpret quotient types.
- We can use logical equivalence as the equality for propositions hence we can eliminate univalence for propositions.
We need equations between equality proofs at several points of the construction, e.g. when verifying the functor laws for $\text{subst}$ for $\Pi$.

All these equations hold trivially because of definitional proof-irrelevance.

In the end we need to derive $J$ from $\text{subst}$.

This also requires definitional proof-irrelevance.
Observational Type Theory

- Instead of defining the Source Type Theory by a translation, we can define it directly.
- We define $= : A \rightarrow A \rightarrow Prop$ by recursion over the type $A$ and $\text{subst} : (P : A \rightarrow \text{Type}) \rightarrow a = b \rightarrow P a \rightarrow P b$ by recursion over the family $P : A \rightarrow \text{Type}$.
- The other constants don’t need to be defined because they live in $Prop$.
- Using a clever trick we can also address the problem that $\text{subst} P \text{ refl}$ is not definitionally equal to the identity.
- For details see our PLPV 2008 paper (jointly with Conor McBride and Wouter Swierstra).
From Setoids to Groupoids

- The setoid construction allows us to interpret types upto dimension 2.
- Replacing setoids by groupoids we can interpret types upto dimension 3.
- We have to use Groupoids enriched over Setoids: 
  \[ \sim A : A.\text{set} \rightarrow A.\text{set} \rightarrow \text{Prop} \] is replaced by 
  \[ \sim A : A.\text{set} \rightarrow A.\text{set} \rightarrow \text{Setoid}. \]
- The groupoid equations hold up to setoid equality.
- We can define equality of the universe of sets as isomorphism - hence we can interpret univalence at dimension 2.
- Carrying out this construction in detail would provide an alternative to Harper’s and Licata’s proof of canonicity of 2-dimensional Type Theory.
- Our proposal would also address the issue that they have been using an extensional Type Theory at dimension 2.
We would like to eliminate the $Prop$-universe.
If we can construct a groupoid model enriched over itself we should be able to do this.
And we should be able to interpret univalence at any level.
This would require to construct an $\omega$-Groupoid model of Type Theory.
As a first step we need to define what is a $\omega$-Groupoid in Type Theory.
What are weak $\omega$-groupoids?

- There are a number of definitions in the literature, e.g. based on contractible globular operads.
- We need to formalize them in Type Theory . . .
- Formalizing the required categorical concepts creates a considerable overhead.
- Also it is not always clear how to represent them in the absence of UIP.
- E.g. what are strict $\omega$-groupoids?
Globular sets

We define a *globular set* $G : \text{Glob}$ coinductively:

- $\text{obj}_G : \text{Set}$
- $\text{hom}_G : \text{obj}_G \rightarrow \text{obj}_G \rightarrow \infty \text{Glob}$

Given globular sets $A, B$ a morphism $f : \text{Glob}(A, B)$ between them is given by

- $\text{obj}_f : \text{obj}_A \rightarrow \text{obj}_B$
- $\text{hom}_f : \prod a, b : \text{obj}_A.
  \text{Glob}(\text{hom}_A a b, \text{hom}_B(\text{obj}_f a, \text{obj}_f b))$

As an example we can define the terminal object in $1_{\text{Glob}} : \text{Glob}$ by the equations

- $\text{obj}_{1_{\text{Glob}}} = 1_{\text{Set}}$
- $\text{hom}_{1_{\text{Glob}}} x y = 1_{\text{Glob}}$
The Identity Globular set

More interestingly, the globular set of identity proofs over a given set $A$, $\text{Id}^\omega A : \text{Glob}$ can be defined as follows:

$$\text{obj}_{\text{Id}^\omega A} = A$$

$$\text{hom}_{\text{Id}^\omega A} a b = \text{Id}^\omega (a = b)$$
Globular sets as a presheaf

Our definition of globular sets is equivalent to the usual one as a presheaf category over the diagram:

\[
\begin{array}{c}
0 \\ t_0 \\
\end{array} 
\xRightarrow{s_0} 
\begin{array}{c}
1 \\ t_1 \\
\end{array} 
\xRightarrow{s_1} 
\begin{array}{c}
2 \ldots n \\ t_n \\
\end{array} 
\xRightarrow{s_n} 
\begin{array}{c}
(n + 1) \ldots \\
\end{array} 
\]

with the globular identities:

\[
\begin{align*}
t_{i+1} \circ s_i &= s_{i+1} \circ t_i \\
t_{i+1} \circ t_i &= s_{i+1} \circ t_i
\end{align*}
\]
A syntactic approach

- When is a globular set a weak $\omega$-groupoid?
- We define a syntax for objects in a weak $\omega$-groupoid.
- A globular set is a weak $\omega$-groupoid, if we can interpret the syntax.
- This is reminiscent of environment $\lambda$-models.
The syntactical framework

**Contexts**

Con : Set

\( \varepsilon : \text{Con} \)

(\( \Gamma, C \) : Con)

**Categories**

\( \Gamma : \text{Con} \)

\( \text{Cat} \Gamma : \text{Set} \)

\( \bullet : \text{Cat} \Gamma \)

\( C : \text{Cat} \Gamma \)

\( a, b : \text{Obj} C \)

\( C[a, b] : \text{Cat} \Gamma \)

**Objects**

\( C : \text{Cat} \Gamma \)

\( \text{Obj} C, \text{Var} C : \text{Set} \)
Interpretation

1. An assignment of sets to contexts:
   \[ \Gamma : \text{Con} \]
   \[ \llbracket \Gamma \rrbracket : \text{Set} \]

2. An assignment of globular sets to category expressions:
   \[ C : \text{Cat} \Gamma \gamma : \llbracket \Gamma \rrbracket \]
   \[ \llbracket C \rrbracket \gamma : \text{Glob} \]

3. Assignments of elements of object sets to object expressions and variables
   \[ C : \text{Cat} \Gamma \quad A : \text{Obj} C \gamma : \llbracket \Gamma \rrbracket \]
   \[ \llbracket A \rrbracket \gamma : \text{obj}_{\llbracket C \rrbracket} \gamma \]

   Subject to some (obvious) conditions such as:
   \[ \llbracket \bullet \rrbracket \gamma = G \]
   \[ \llbracket C[a, b] \rrbracket \gamma = \text{hom}_{\llbracket C \rrbracket} \gamma (\llbracket a \rrbracket \gamma) (\llbracket b \rrbracket \gamma) \]
Composition

\[ a \xrightarrow{f} b \xrightarrow{g} c \quad \iff \quad a \xrightarrow{gf} c \]

\[ a \xrightarrow{f} b \xrightarrow{g} c \quad \iff \quad a \xrightarrow{\beta \alpha} c \]

\[ a \xrightarrow{f} b \xrightarrow{g} c \quad \iff \quad a \xrightarrow{\beta \cdot \alpha} c \]

\[ a \xrightarrow{f} b \xrightarrow{g} c \quad \iff \quad a \xrightarrow{\beta \cdot \alpha} c \]
Telescopes

A telescope $t : \text{Tel } C n$ is a path of length $n$ from a category $C$ of to one of its (indirect) hom-categories:

$$
\begin{array}{c}
C : \text{Cat } \Gamma \\
N
\end{array}
\Rightarrow
Tel \ C n : \text{Set}
$$

We can turn telescopes into categories:

$$
\begin{array}{c}
t : \text{Tel } C n \\
C \leftrightarrow t : \text{Cat } \Gamma
\end{array}
$$
Formalizing composition

\[
\begin{align*}
\alpha : \text{Obj}(t \downarrow) & \quad \beta : \text{Obj}(u \downarrow) \\
\beta \circ \alpha : \text{Obj}(u \circ t \downarrow)
\end{align*}
\]

is a new constructor of Obj where

\[
\begin{align*}
t : \text{Tel}(C[a, b]) n & \quad u : \text{Tel}(C[b, c]) n \\
u \circ t : \text{Tel}(C[a, c])
\end{align*}
\]

is a function on telescopes defined by cases

\[
\bullet \circ \bullet C = \bullet \\
(\bullet \circ t)[a' \circ a, b' \circ b] = u[a', b'] \circ t[a, b]
\]
Laws

For example the left unit law in dimension 1:

\[ \text{id}_b \circ f = f, \]  \hspace{1cm} (1)

and in dimension 2.

\[ \text{id}^2_b \circ \alpha = \alpha, \]

where \( \text{id}^2_b = \text{id}_{\text{id}_b} \)

In the strict case the 2nd equation only type-checks due to the first. In the weak case we have to apply the previous isomorphism explicitly.
Coherence

Example:

\[(g \text{id}_b) f \xrightarrow{\alpha_{g, \text{id}_g, f}} g (\text{id}_b f)\]

\[\rho \text{id}_f \xrightarrow{p} q \xrightarrow{\text{id}_g \lambda_f} g f\]

In summary and full generality:

*For any pair of coherence cells with the same domain and target, there must be a mediating coherence cell.*

**Problem**

This definition of coherence is too strong!
Formalizing coherence

\[
\begin{align*}
  x : \text{Obj } C \\
  \text{hollow } x : \text{Set} \\
  \text{hollow } (\lambda \_\_ ) = \top \quad \ldots \\
  f \; g : \text{Obj } C[a, b] \\
  p : \text{hollow } f \\
  q : \text{hollow } g \\
  \text{coh } p \; q : \text{Obj } C[a, b][f, g] \\
  \text{hollow } (\text{coh } p \; q) = \top
\end{align*}
\]
Summary

To be able to eliminate univalence we want to interpret Type Theory in a weak $\omega$-groupoid in Type Theory.

As a first step we need to define what is a weak $\omega$-groupoid.

Our approach is to define a syntax for objects in a weak $\omega$-groupoid.

A globular set is a weak $\omega$ groupoid if we can interpret this syntax.

See our draft paper for details: *A Syntactical Approach to Weak $\omega$-Groupoids*
Further work

- We need to fix our definition of coherence!
- The current definition is quite complex - can we simplify it?
- Can we actually show that the identity globular set is a weak $\omega$-groupoid, internalizing results by Lumsdaine and Garner/van de Berg?
- What is a model of Type Theory in a weak $\omega$-groupoid.
- Can we use this construction to eliminate univalence?