Weak ω-Groupoids in Type Theory

Based on joint work with Ondrej Rypacek

Thorsten Altenkirch

Functional Programming Laboratory
School of Computer Science
University of Nottingham

April 2, 2012

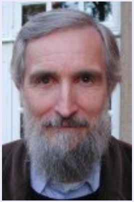
Question

Can we understand a concept by describing it categorically?

- Category theory helps us to structure and relate concepts.
- True understanding must come form somewhere else.
- Set theory?
- There is something better!

Type Theory

Per Martin-Löf



- Propositions are types
- Basic constructions on types: functions, tuples, enumerations, ...
- Implementations of Type Theory: Coq, Agda, ...
- Informal understanding of Type Theory!

Basic ingredients of Type Theory

Π-types *dependent function types*
functions, implication, universal quantification

Σ-types *dependent pair types*
tuples, conjunction, existential quantification

Finite types $0, 1, 2$

Equality types Given $a, b : A$, $a \equiv b$ is the type of proofs that a is equal to b.

Inductive and coinductive types Finite and infinite trees.

Universes $\mathbf{Set}_0 : \mathbf{Set}_1 : \ldots$

Example: Axiom of choice

$$
\begin{aligned}
ac &: ((a : A) \longrightarrow \Sigma\,[b : B]\; R\; a\; b) \\
&\quad \longrightarrow \Sigma\,[f : (A \longrightarrow B)]\;((a : A) \longrightarrow R\; a\; (f\; a)) \\
ac\; g &= (\lambda\; a \longrightarrow \mathit{proj}_1\; (g\; a)), (\lambda\; a \longrightarrow \mathit{proj}_2\; (g\; a))
\end{aligned}
$$

- Follows from the constructive explanation of connectives.
- *ac* is actually an isomorphism, i.e. there is an inverse:

$$
\begin{aligned}
ac' &: \Sigma\,[f : (A \longrightarrow B)]\;((a : A) \longrightarrow R\; a\; (f\; a)) \\
&\quad \longrightarrow ((a : A) \longrightarrow \Sigma\,[b : B]\; R\; a\; b) \\
ac'\; (f, g) &= \lambda\; a \longrightarrow f\; a, g\; a
\end{aligned}
$$

Propositional equality

$$
\begin{array}{l}
\textbf{data}\ \_ \equiv \_\ : A \longrightarrow A \longrightarrow \mathit{Set}\ \textbf{where} \\
\quad \mathit{refl} : \quad a \equiv a
\end{array}
$$

Using pattern matching we can show that $\_ \equiv \_$ is an equivalence relation:

$$
\begin{array}{l}
\_^{-1} : a \equiv b \longrightarrow b \equiv a \\
\mathit{refl}^{-1} = \mathit{refl} \\
\_ \circ \_ : b \equiv c \longrightarrow a \equiv b \longrightarrow a \equiv c \\
\mathit{refl} \circ q = q
\end{array}
$$

The eliminator J

Instead of pattern matching we can use the eliminator:

$$
\begin{aligned}
&J : (P : \{a\; b : A\} \longrightarrow a \equiv b \longrightarrow \mathit{Set}) \\
&\quad \longrightarrow (\{a : A\} \longrightarrow P\; \{a\}\; \mathit{refl}) \\
&\quad \longrightarrow \{a\; b : A\} \longrightarrow (p : a \equiv b) \longrightarrow P\; p \\
&J\; P\; m\; \mathit{refl} = m
\end{aligned}
$$

$$
\begin{aligned}
&\_^{-1} : a \equiv b \longrightarrow b \equiv a \\
&\_^{-1} = J\; (\lambda\; \{a\}\; \{b\}\; \_ \longrightarrow b \equiv a)\; \mathit{refl} \\
&\_ \circ \_ : b \equiv c \longrightarrow a \equiv b \longrightarrow a \equiv c \\
&\_ \circ \_\; \{a\} = J\; (\lambda\; \{b\}\; \{c\}\; \_ \longrightarrow a \equiv b \longrightarrow a \equiv c)\; (\lambda\; p \longrightarrow p)
\end{aligned}
$$

Uniqueness of Identity Proofs ?

Question

Can all pattern matching proofs done using the eliminator?

UIP

Can we prove that all identity proofs are equal?

$$\begin{aligned}&\mathit{uip} : (p\ q : a \equiv b) \longrightarrow p \equiv q\\&\mathit{uip}\ \mathit{refl}\ \mathit{refl} = \mathit{refl}\end{aligned}$$

Groupoids

Groupoid

A groupoid is a category where every morphism is an isomorphism.

- Categories are the generalisation of preorders and monoids.
- Groupoids are the generalisation of equivalence relations and groups.

Groupoid laws

Laws

$$\begin{aligned}
&\mathit{refl} \circ p \equiv p \\
&p \circ \mathit{refl} \equiv p \\
&p \circ (q \circ r) \equiv (p \circ q) \circ r \\
&p \circ p^{-1} \equiv \mathit{refl} \\
&p^{-1} \circ p \equiv \mathit{refl}
\end{aligned}$$

- Using only J we can establish the groupoid laws.

$$\rho : (p : a \equiv b) \longrightarrow p \circ \mathit{refl} \equiv p$$
$$\rho = J\ (\lambda\ p \longrightarrow p \circ \mathit{refl} \equiv p)\ (\lambda\ \{\_\} \longrightarrow \mathit{refl})$$

Hofmann/Streicher

Hofmann/Streicher 94 

Groupoids form a model of Type Theory in which *uip* doesn't hold.
Hence *uip* is not derivable from *J* only.

Consider the functions

$$f : \mathbb{N} \longrightarrow \mathbb{N}$$
$$f = \lambda\, n \longrightarrow n + 0$$
$$g : \mathbb{N} \longrightarrow \mathbb{N}$$
$$g = \lambda\, n \longrightarrow n$$

We can show

$$\mathit{exteq} : (n : \mathbb{N}) \longrightarrow f\ n \equiv g\ n$$
$$\mathit{exteq}\ n = \mathit{add0lem}\ n$$

but we cannot show

$$\mathit{eq} : f \equiv g$$

because if such a proof exists.
Then there is one in normal from (*refl*).
And f and g would have to be convertible (same normal form).
However, $n + 0$ and n are not convertible.

Extensionality

This shows that the principle:

$$
\begin{aligned}
&ext : (f\ g : A \longrightarrow B) \\
&\quad \longrightarrow ((a : A) \longrightarrow f\ a \equiv g\ a) \longrightarrow f \equiv g
\end{aligned}
$$

is not provable in Type Theory.

Equality of functions

- What should be equality of functions?
- All operations in Type Theory preserve extensional equality of functions.
  The only exception is intensional propositional equality.
- We would like to define propositional equality as extensional equality.

Setoids

- Setoids are sets with an equivalence relation.

$$
\begin{array}{l}
\mathit{record}\ \mathit{Setoid} : \mathit{Set}_1\ \textbf{where} \\
\quad \mathit{field} \\
\quad\quad \mathit{set} : \mathit{Set} \\
\quad\quad \mathit{eq} : \mathit{set} \longrightarrow \mathit{set} \longrightarrow \mathit{Prop} \\
\quad\quad \ldots
\end{array}
$$

- I write *Prop* to indicate that all proofs should be identified.
- This seems necessary for the construction.

Function setoids

- A function between setoids has to respect the equivalence relation.

$$
\begin{array}{l}
\mathit{record}\ \_\Rightarrow\mathit{set}\_\ (A\ B : \mathit{Setoid}) : \mathit{Set}\ \textbf{where} \\
\quad \mathit{field} \\
\quad\quad \mathit{app} : \mathit{set}\ A \longrightarrow \mathit{set}\ B \\
\quad\quad \mathit{resp} : \forall \{a\}\ \{a'\} \longrightarrow \mathit{eq}\ A\ a\ a' \longrightarrow \mathit{eq}\ B\ (\mathit{app}\ a)\ (\mathit{app}\ a')
\end{array}
$$

- Equality between functions is extensional equality:

$$
\begin{array}{l}
\_\Rightarrow\_ : \mathit{Setoid} \longrightarrow \mathit{Setoid} \longrightarrow \mathit{Setoid} \\
A \Rightarrow B = \mathit{record}\ \{ \\
\quad \mathit{set} = A \Rightarrow \mathit{set}\ B; \\
\quad \mathit{eq} = \lambda\ f\ f' \longrightarrow \\
\quad\quad \forall \{a\} \longrightarrow \mathit{eq}\ B\ (\mathit{app}\ f\ a)\ (\mathit{app}\ f'\ a) \}
\end{array}
$$

Eliminating extensionality

- Adding principles like *ext* as constants destroys basic computational properties of Type Theory.
- E.g. there are natural numbers not reducible to a numeral.
- We can eliminate *ext* by translating every type as a setoid see my LICS 99 paper: *Extensional Equality in Intensional Type Theory*.
- This construction only works for a proof-irrelevant equality (UIP holds).

Equality of types

- When should two types be provably equal?
- All operations in Type Theory preserve isomorphisms.
- Unlike Set Theory, e.g. $\{0,1\} \simeq \{1,2\}$ but $\{0,1\} \cup \{0,1\} \not\simeq \{0,1\} \cup \{1,2\}$.
- Indeed, isomorphic types are propositionally indistinguishable in Type Theory.
- Leibniz principle: isomorphic sets should be equal!?

Univalent Type Theory

- Vladimir Voevodsky proposed a
  new principle for Type Theory: the univalence principle.
- This is inspired by models of Homotopy theoretic models of Type Theory.
- He defines the notion of *weak equivalence* of types.

Voevodsky's Univalence Principle

Equality of types is weakly equivalent to weak equivalence

- Using this principle we can show that isomorphic types are equal.
- It also implies *ext*.
- However, it is incompatible with *uip*.

The question

- Can we eliminate univalence?
- We cannot use setoids because they rely on UIP.
- Groupoids are better.
- But Groupoids still rely on proof-irrelevance for the equality of equality proofs ...
- Hence we need ω-groupoids.
- Since the equalities are not all strict we need **weak** ω-groupoids.

What are weak ω-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 ω-groupoids?

Globular sets

We define a *globular set* G : Glob coinductively:

$$\begin{array}{rcl} \mathsf{obj}_G & : & \mathsf{Set} \\ \mathsf{hom}_G & : & \mathsf{obj}_G \to \mathsf{obj}_G \to \infty\,\mathsf{Glob} \end{array}$$

Given globular sets A, B a morphism f : Glob(A, B) between them is given by

$$\begin{array}{rcl} \mathrm{obj}^{\to}_f & : & \mathsf{obj}_A \to \mathsf{obj}_B \\ \mathrm{hom}^{\to}_f & : & \Pi a, b : \mathsf{obj}_A. \\ & & \quad \mathsf{Glob}(\mathsf{hom}_A\ a\, b, \mathsf{hom}_B(\mathrm{obj}^{\to}_f\ a, \mathrm{obj}^{\to}_f\ b)) \end{array}$$

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

$$\begin{array}{rcl} \mathsf{obj}_{\mathbf{1}_{\mathsf{Glob}}} & = & \mathbf{1}_{\mathsf{Set}} \\ \mathsf{hom}_{\mathbf{1}_{\mathsf{Glob}}}\ x\, y & = & \mathbf{1}_{\mathsf{Glob}} \end{array}$$

The Identity Globular set

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

$$
\begin{aligned}
\mathsf{obj}_{\mathrm{Id}^\omega A} &= A \\
\mathsf{hom}_{\mathrm{Id}^\omega A}\, a\, b &= \mathrm{Id}^\omega\,(a = b)
\end{aligned}
$$

Globular sets as a presheaf

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

$$0 \underset{t_0}{\overset{s_0}{\rightrightarrows}} 1 \underset{t_1}{\overset{s_1}{\rightrightarrows}} 2 \dots n \underset{t_n}{\overset{s_n}{\rightrightarrows}} (n+1) \dots$$

with the globular identities:

$$\begin{aligned} t_{i+1} \circ s_i &= s_{i+1} \circ t_i \\ t_{i+1} \circ t_i &= s_{i+1} \circ t_i \end{aligned}$$

A syntactic approach

- When is a globular set a weak ω-groupoid?
- We define a syntax for objects in a weak ω-groupoid.
- A globular set is a weak ω-groupoid, if we can interpret the syntax.
- This is reminiscient of environment λ-models.

The syntactical framework

Contexts

$$\mathrm{Con} : \mathrm{Set}$$

$$\frac{}{\varepsilon : \mathrm{Con}} \qquad \frac{C : \mathrm{Cat}\,\Gamma}{(\Gamma, C) : \mathrm{Con}}$$

Categories

$$\frac{\Gamma : \mathrm{Con}}{\mathrm{Cat}\,\Gamma : \mathrm{Set}}$$

$$\frac{}{\bullet : \mathrm{Cat}\,\Gamma} \qquad \frac{C : \mathrm{Cat}\,\Gamma \quad a, b : \mathrm{Obj}\,C}{C[\,a\,,\,b\,] : \mathrm{Cat}\,\Gamma}$$

Objects

$$\frac{C : \mathrm{Cat}\,\Gamma}{\mathrm{Obj}\,C, \mathrm{Var}\,C : \mathrm{Set}}$$

Interpretation

1. An assignment of sets to contexts:

$$\frac{\Gamma : \mathrm{Con}}{[\![\Gamma]\!] : \mathrm{Set}}$$

2. An assignment of globular sets to category expressions:

$$\frac{C : \mathrm{Cat}\ \Gamma \qquad \gamma : [\![\Gamma]\!]}{[\![C]\!]\ \gamma : \mathrm{Glob}}$$

3. Assignments of elements of object sets to object expressions and variables

$$\frac{C : \mathrm{Cat}\ \Gamma \qquad A : \mathrm{Obj}\ C \qquad \gamma : [\![\Gamma]\!]}{[\![A]\!]\ \gamma : \mathrm{obj}_{[\![C]\!]\ \gamma}}$$

subject to some (obvious) conditions such as:

$$\begin{aligned} [\![\bullet]\!]\ \gamma &= G \\ [\![C[a,b]]\!]\ \gamma &= \mathrm{hom}_{[\![C]\!]\gamma}\ ([\![a]\!]\ \gamma)\ ([\![b]\!]\ \gamma) \end{aligned}$$

Composition

$$\begin{tikzcd} a \arrow[r, "f"] & b \arrow[r, "g"] & c \end{tikzcd} \quad \mapsto \quad \begin{tikzcd} a \arrow[r, "gf"] & c \end{tikzcd}$$

$$\begin{tikzcd} a \arrow[r, bend left, "f"] \arrow[r, bend right, "f'"'] \arrow[r, phantom, "\alpha\Downarrow"] & b \arrow[r, bend left, "g"] \arrow[r, bend right, "g'"'] \arrow[r, phantom, "\beta\Downarrow"] & c \end{tikzcd} \quad \mapsto \quad \begin{tikzcd} a \arrow[r, bend left, "gf"] \arrow[r, bend right, "g'f'"'] \arrow[r, phantom, "\beta\alpha\Downarrow"] & c \end{tikzcd}$$

$$\begin{tikzcd}[column sep=huge] a \arrow[r, bend left=60, "f"] \arrow[r, "f'" description] \arrow[r, bend right=60, "f''"'] \arrow[r, phantom, shift left=3, "\alpha\Downarrow \xrightarrow{\gamma} \Downarrow\alpha'"] \arrow[r, phantom, shift right=3, "\beta\Downarrow \xrightarrow[\delta]{} \Downarrow\beta'"] & b \end{tikzcd} \quad \mapsto \quad \begin{tikzcd}[column sep=huge] a \arrow[r, bend left, "f"] \arrow[r, bend right, "f''"'] \arrow[r, phantom, "\beta\cdot\alpha \Downarrow \xrightarrow{\delta\cdot\gamma} \Downarrow \beta'\cdot\alpha'"] & c \end{tikzcd}$$

Telescopes

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

$$\frac{C : \mathrm{Cat}\,\Gamma \qquad n : \mathbb{N}}{\mathrm{Tel}\; C\; n : \mathrm{Set}}$$

We can turn telescopes into categories:

$$\frac{t : \mathrm{Tel}\; C\; n}{C +\!\!+\, t \;:\; \mathrm{Cat}\,\Gamma}$$

Formalizing composition

$$\frac{\alpha : \mathrm{Obj}(t \Downarrow) \qquad \beta : \mathrm{Obj}(u \Downarrow)}{\beta \circ \alpha : \mathrm{Obj}(u \circ t \Downarrow)}$$

is a new constructor of Obj where

$$\frac{t : \mathrm{Tel}\ (C[a,b])\ n \qquad u : \mathrm{Tel}\ (C[b,c])\ n}{u \circ t : \mathrm{Tel}\ (C[a,c])}$$

is a function on telescopes defined by cases

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

Laws

For example the left unit law in dimension 1:

$$\mathrm{id}_b \circ f \quad = \quad f \ , \tag{1}$$

and in dimension 2.

$$\mathrm{id}^2_b \circ \alpha \quad = \quad \alpha \ ,$$

where $\mathrm{id}^2_b = \mathrm{id}_{\mathrm{id}_b}$

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 explicitely.

$$\lambda_\alpha \;:\; \begin{tikzcd}[column sep=huge]
a \arrow[r, bend left, "f"] \arrow[r, bend right, "f'"'] \arrow[r, phantom, "\alpha\Downarrow"] \arrow[rr, bend left=60, "f"] \arrow[rr, bend right=60, "f'"'] \arrow[rr, phantom, bend left=40, "\Downarrow\lambda_f^{-1}"] \arrow[rr, phantom, bend right=40, "\Downarrow\lambda_{f'}"] & b \arrow[r, bend left, "\mathrm{id}\, f"] \arrow[r, bend right, "\mathrm{id}\, b"'] \arrow[r, phantom, "\mathrm{id}^2 b\Downarrow"] & b
\end{tikzcd}
\quad \Rrightarrow \quad
\begin{tikzcd}[column sep=huge]
a \arrow[r, bend left=50, "f"] \arrow[r, bend right=50, "f'"'] \arrow[r, phantom, "\alpha\Downarrow"] & b
\end{tikzcd}$$

Coherence

Example:

$$
\begin{tikzcd}
(g\,\mathrm{id}_b)\,f \arrow[r, "\alpha_{g,\mathrm{id}_g,f}"] \arrow[dr, "\rho\,\mathrm{id}_f"'] & g\,(\mathrm{id}_b\,f) \arrow[d, "\mathrm{id}_g\,\lambda_f"] \arrow[dl, phantom, "p \Rrightarrow q"] \\
& g\,f
\end{tikzcd}
$$

In summary and full generality:

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

Formalizing coherence

$$\frac{x : \mathrm{Obj}\ C}{\mathrm{hollow}\ x\ :\ \mathrm{Set}}$$

$$\mathrm{hollow}\,(\lambda\,\_\,\_) = \top\ \cdots$$

$$\frac{f\ g : \mathrm{Obj}\ C[a,b] \qquad p : \mathrm{hollow}\ f \qquad q : \mathrm{hollow}\ g}{\mathrm{coh}\ p\ q : \mathrm{Obj}\ C[a,b][f,g]}$$

$$\mathrm{hollow}\,(\mathrm{coh}\,p\,q) = \top$$

Summary

- To be able to eliminate univalence we want to interpret Type Theory in a weak ω-groupoid in Type Theory.
- As a first step we need to define what is a weak ω-groupoid.
- Our approach is to define a syntax for objects in a weak ω groupoid.
- A globular set is a weak ω groupoid if we can interpret this syntax.
- See our draft paper for details: *A Syntactical Approach to Weak ω-Groupoids*

Further work

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