Homotopy in the foundations of mathematics (Voevodsky's approach).

Sergei Soloviev (Acadie, IRIT, University of Toulouse)

11/01/2011

A few words about this talk -

This is a kind of survey, intended to outline Voevodsky's approach in a much larger context of modern Type Theory and its problems.
Sources:

- **Voevodsky:**
- A very short note on homotopy lambda calculus (Oct. 2, 2006).
- Notes on type systems (updated Sep. 14, 2010)
- Univalent Foundations Project (Oct.1, 2010)
- Talks at: CMU (04.02.10), Bonn (08.09.10), IAS (10.12.10), WoLLIC (18.05.11), Gothenburg (05.09.11), Bergen, "Types"(11.O9.2011) and UPenn(22.09.11)
- All found at http://www.math.ias.edu/∼vladimir/Site3/Univalent_Foundations.html

A few words about this talk -

- **Others**:
- Peter Aczel, On Voevodsky's Univalence Axiom. 06.07.11, Edinburgh Third European Set Theory Conference.
- R. Brown, Ph.J. Higgins, R. Sivera. Nonabelian Algebraic Topology. Filtered spaces, crossed complexes, cubical homotopy groupoids. EMS,Tracts in Math. 15, 2011.
- Publications on type theory in general (relevant references in the text below).

Voevodsky's approach

Using Voevodsky's own words, his view of foundations may be characterized as follows (WoLLIC's talk):

1. Foundations of mathematics which can be used both for constructive and for non-constructive mathematics.
2. Foundations which naturally include "axiomatization" of the categorical and higher categorical thinking.
3. Foundations which can be conveniently formalized using the well known class of formal languages called Martin-Löf type systems.
4. Foundations which are based on direct axiomatization of the "world" of homotopy types instead of the "world" of sets.

Larger context

With respect to first two items one may notice:

- There is a lot of “splits” (seemingly irreconciliable divisions) in existing approaches to Foundations, *e.g.*,
- Constructive/non-constructive;
- Categorical/set-theoretical.
- Everybody (or almost everybody) is “tired” to fight for the “truth” and is camped on his own positions.

Larger context

With respect to second two items:

- Active development of proof-assistants (Coq, Lego, Plastic, Isabelle...);
- Many of them based on Type Theory and using dependent types, like in Martin-Löf Type Theory;
- Problems with equality in Type Theory - *e.g.*, the problem of extensional versus intensional equality. Decidable intensional equality used in proof-assistants often counter-intuitive from the point of view of everyday mathematics.
- The idea to use homotopy - an attempt to solve difficulties with equality.

Larger context

Type Theory (used in proof-assistants) is an attempt to combine logics, static data and computation in one formalism.

- Logics: derivations, judgements (propositions of several basic forms, for example $\Gamma \vdash t : A$, meaning “term t is of type A in the context Γ). Usually - higher order logics.
- Data - inductive, coinductive types, for example $Nat = Ind(\alpha)(0 : \alpha, succ : \alpha \to \alpha)$.
- Computation rules for terms - reductions (including recursion), possibly user-defined rules.

Larger context

All this makes Type Theory a good candidate to be used for formalization of foundations of mathematics (in the form appropriate for computer-assisted reasoning).

Non-standard reductions and inductive types

- A well known problem with proof-assistants and related type-theoretic frameworks is the problem of extensional versus intentional equality.
- Many equalities (between functions etc.) often considered as granted by the user are not easily obtained and not easy to carry around because they are not intentional.
- Intentional equality is supported by “reduction part” of the system (for example $\beta\eta$).
- Extensional equality requires a proof (like $\forall x.(f(g(x)) = x)$) and the proof-term has to be carried everywhere it is needed.

Non-standard reductions and inductive types

- For illustrative purposes let us consider simply typed λ-calculus with inductive types.

The expression $\mu\alpha(c_1 : \sigma_1(\alpha), ..., c_n : \sigma_n(\alpha))$ will denote an inductive type.
Here $c_1, ..., c_n$ are names of type constructors (introduction operators), $\sigma(\alpha)$ are type-schemes of the constructors, for exemple, $\alpha, \alpha \to \alpha, \mathit{Nat} \to \alpha \to \alpha$.
In general they have the form $\rho_1 \to ... \to \rho_n \to \alpha$ with ρ either with α strictly positive (like in $\mathit{Nat} \to \alpha$) or without α.

Non-standard reductions and inductive types

The recursors (recursion operators) are defined for each pair A, B, where A is an inductive type. They have the form

$$(|t_1, ..., t_n|)^{A,B}$$

if A has n constructors. The corresponding reduction (sometimes noted ι or β_r) is of the form

$$(|\bar{t}|)(c_i\bar{r}) \rightarrow t_i\bar{r}\bar{(}(|\bar{t}|)r^o)(|\bar{t}|) \circ r^1)$$

(r^0 are non-functional arguments, r^1 functional arguments).

Non-standard reductions and inductive types

For example, if we take the type

$$Ord = \mu\alpha(0_{Ord} : \alpha, succ_{ord} : \alpha \to \alpha, lim : (Nat \to \alpha) \to \alpha)$$

we shall have

$$(|t_1, t_2, t_3|)^{Ord,B} lim(f) \to (t_3 f)((|t_1, t_2, t_3|)^{Ord,B} \circ f)$$

Here
$t_3 : (Nat \to Ord) \to (Nat \to B) \to B, t_1 : B, t_2 : Ord \to B \to B,$
$f : Nat \to Ord.$

Non-standard reductions and inductive types

The “non-standard” reductions below correspond to certain “typical” situations where a researcher with standard mathematical training would expect an equality, but in Type Theory (and so, in proof assistants based on Type Theory) there is no intentional equality. In the case(s) considered below the problem may be solved introducing non-standard reductions (and proving their good behaviour) but it is not the case in general.

Non-standard reductions and inductive types

Some “non standard” reductions:

“Copy”.Given an inductive type $A = \mu\alpha(c_1 : \sigma_1(\alpha), ..., c_n : \sigma_n(\alpha))$ its copy $A' = \mu\alpha(c'_1 : \sigma_1(\alpha), ..., c'_n : \sigma_n(\alpha))$ may be defined.

This definition may be generalized, to include change of parameter(s).

For example, we may consider:

$$Nat = \mu\alpha(0 : \alpha, succ : \alpha \to \alpha)$$

and its copy

$$Nat' = \mu\alpha(0' : \alpha, succ' : \alpha \to \alpha)$$

.

Non-standard reductions and inductive types

Or we may consider

$$Ord = \mu\alpha(0_{Ord} : \alpha, succ_{Ord} : \alpha \to \alpha, lim : (Nat \to \alpha) \to \alpha)$$

and its copy

$$Ord' = \mu\alpha(0'_{Ord} : \alpha, succ'_{Ord} : \alpha \to \alpha, lim' : (Nat \to \alpha) \to \alpha)$$

.

Non-standard reductions and inductive types

Function *Copy* is easily defined using recursors (and functions between parameter types A, A' if we need to change the parameter). Let $Copy : A \to A'$, $Copy' : A' \to A$. The corresponding reduction is:

$$Copy \circ Copy' \to_\xi id_{A'}, Copy' \circ Copy \to_\xi id_A$$

This reduction is generalized to the case of parameter change if the parameters are already isomorphic (maybe, w.r.t. the system with non-standard reductions, like Nat, Nat').

Non-standard reductions and inductive types

Theorem. Simply typed λ-calculus with inductive types and $\beta\eta\iota\xi$-reductions is SN and CR.

- David Chemouil, Sergei Soloviev. Remarks on isomorphisms of simple inductive types. - In : Mathematics, Logic and Computation, Eindhoven, 04/07/2003-05/07/2003, Elsevier, ENTCS 85, 7, p. 1-19, july 2003. Abstract - URL : http://dx.doi.org/10.1016/S1571-0661(04)80760-6
- David Chemouil. Isomorphisms of simple inductive types through extensional rewriting.- MSCS, 15(5), 875-915, 2005.

Dependent types

To understand better the place of Voevodsky's ideas we need to consider first the notion of dependent type and outline the structure (and principal differences) of dependent type systems.

Dependent types

In ordinary simply typed lambda-calculus (seen as a Type Theory) it is possible to separate completely types and terms, contexts (and the notion of context validity) and judgements (typing propositions that we try to derive).
The context may be seen as any list of variables with types $x_1 : A_1, ..., x_n : A_n$. It is valid (well defined) iff

- The variables $x_1, ..., x_n$ are distinct.
- The types $A_1, ..., A_n$ are well formed. (Verification of the “well formedness” of types is similar to the verification of logical formulas, and does not involve the verification of derivability of judgements in the theory itself.)

Dependent types

From the point of view of models, it would be useful to say that the context in a “simple typed” system is usually modelled by some sort of product.
If $|A|$ is the interpretation of type A in some model, then
$x_1 : A_1, ..., x_n : A_n$ is modelled by $|A_1| \times ... \times |A_n|$.

Dependent types

When dependent types are considered, one can no more consider context validity independently of other forms of judgements.
Types can depend on terms (and in the context - on the variables defined earlier).

- Let $n : Nat$. It is possible to define the type $Vect(n)$ of vectors of the dimension n depending on n.

Dependent types

As to the structure of contexts, we may have the contexts of the form

$$n : Nat, v : Vect(n), ...$$

It means that one of the principal rules will be

$$\frac{\Gamma \vdash A : Type}{\Gamma, x : A \vdash valid}$$

where x has to be a fresh variable.

Dependent types

So the contexts (context validity judgements) cannot be separated any more from other forms of judgement. All forms of judgements

$$\Gamma \vdash \mathit{valid}$$

$$\Gamma \vdash A : \mathit{Type}$$

$$\Gamma \vdash A = B : \mathit{Type}$$

$$\Gamma \vdash t : A$$

$$\Gamma \vdash t_1 = t_2 : A$$

...

have to be considered simultaneously. Validity of context is verified by derivation, in its turn, it is necessary to prove the validity $\Gamma \vdash \mathit{valid}$ in order to prove $\Gamma \vdash A : \mathit{Type}$ etc.

Dependent types

Two important operations defined for dependent types:

- Dependent product $(\Pi x : A)B(x)$ - type of functions f, $x \mapsto f(x) : B(x)$.
- Dependent sum $(\Sigma x : A)B(x)$ - type of pairs (x, y) for $x : A$, $y : B(x)$.

Dependent types

Usually the systems of dependent types include also

- A logical “nucleus” (unpredicative), and
- An hirerachy of type universes of some sort.

Dependent types

Logical part:

- For example, we have $Prop : Type$ (type of propositions).
- There is $Prf : Prop \rightarrow Type$, *i.e.*, Prf(A):Type for $A : Prop$ (the *proofs* of a proposition A form a type.
- (Just as illustration:)

$$\forall : (A : Type)(B : (x : A)Prop)Prop$$

- Other logical operators (defined in similar way).

Dependent types

Type hierarchy:

- The predicative type universes $Type_i$ ($i \in \omega$) are introduced.
- The predicative universes are types whose objects are names of types. There are also functions $\mathbf{T_i}$ that transform *names* into *types*.
- $Type_i : \mathbf{Type}$, $\mathbf{T_i} : (Type_i)\mathbf{Type}$, $\mathbf{T_i}(a)$ is the type named by a.
- Each predicative universe $Type_i$ has a name $type_i$ in $Type_{i+1}$, $type_i : Type_{i+1}, \mathbf{T_i}(type_i) = Type_i : \mathbf{Type}$.
- The impredicative universe of propositions has a name *prop* in $Type_0$: $prop : Type_0, \mathbf{T_0}(prop) = Prop : \mathbf{Type}$.

Dependent types

There may be means to define also the types like the “identity type” (for the types A, B)

$$Id(A, B)$$

(its elements are witnesses that A, B are identical, for example proofs represented by proof-terms) or the type of isomorphisms

$$A \overset{\sim}{=} B$$

etc

Back to Voevodsky

Now, after this brief outline of certain aspects of Dependent Type Theories we shall go back to Voevodsky's approach. More precisely, it should be called by two names - *Voevodsky and Awodey* approach. Both were at the origins of this approach, but Voevodsky's considerable "mathematical veight" cannot be neglected. Essentially, his approach may be seen as a combination (amalgam) of

- Type Theory
- Higher dimensional groupoid/category theory
- Homotopy theory

His direct contribution to the foundations is the *Univalence Axiom (UA)* to be added to the intensional dependent type theory.

Back to Voevodsky

As far as I see, it may be seen just as a way to make things work together.
Of course there is a deep underlying principle: *SIP (Structure Identity Principle)* promoted by both Voevodsky and Awodey.
I'll try to explain what the principle is, and everybody can judge himself how justified this principle may be in the context of Type Theory (and its difficulties) mentioned above.
But first let me discuss how the elements mentioned above are "blended".

Back to Voevodsky

In next few slides I largely followed the presentation by Peter Aczel (an eminent type theorist who contributed a lot to the development of proof assistants based on type theory).

Remark. I had, though, a different purpose: I tried to “localize” some difficult or dangerous points (from, say, constructive point of view). The introductory part of this talk served to illustrate what kind of difficulties may arise in Type Theory. Below I will use color to indicate the properties that may be difficult to verify.

Back to Voevodsky

Higher dimensional category theory.
Sets have elements/objects *(dim 0)*
Categories have morphisms between objects *(dim 1)*
2-categories have in addition 2-morphisms between morphisms, *i.e.*, the $Hom(A, B)$ is itself a category *(dim 2)*
...
n-categories have morphisms between objects, 2-morphisms between morphisms, up to *n*-morphisms
Identity is part of the picture:
standard equality between elements of a set *(dim 0)*
isomorphism between objects of a category *(dim 1)*
equivalence between objects of a 2-category *(dim 2)*
...

Back to Voevodsky

Groupoids and homotopy theory

A groupoid is a category in which every arrow is invertible.

The n-categories - see above. A weak n + 1-category need only have identity and associative laws up to an n-equivalence.

A weak n + 1-groupoid is a weak n + 1-category in which each arrow is invertible up to an n-equivalence.

With each space X are associated the set $\Pi_0(X)$ of its (path) connected components, its fundamental groupoid $\Pi_1(X)$ and higher dimensional groupoids $\Pi_n(X)$.

A continuous function $f : X \to Y$ is a weak equivalence if it induces isomorphisms between $\Pi_n(X)$ and $\Pi_n(Y)$ ($\forall n \geq 0$).

Homotopies may be seen as paths between paths.

Back to Voevodsky

Homotopy Type Theory

- Interpretation of types as spaces and identity types as path spaces.
- Higher dimensional inductive definitions of the standard spaces.
- Hierarchy of homotopy levels of types.

Back to Voevodsky

Now, what is the Structure Identity Principle?
Isomorphic mathematical structures are structurally identical, i.e., *have the same structural properties.*

- Central question: **What is a structural property?**
- In mathematical practice the notion is usually not precisely defined. In logic there can be a precise answer.
- Another important question: how is the isomorphism defined? (For example - extensional versus intensional.)

For you to judge how are the answers constructive.

Back to Voevodsky

Consider two structures $\mathcal{A}, \mathcal{B}$ of the same signature.

$$\mathcal{A} \cong \mathcal{B} \Rightarrow \mathcal{A} =_{str} \mathcal{B}$$

$\mathcal{A} =_{str} \mathcal{B}$ means by definition $P(\mathcal{A}) \iff P(\mathcal{B})$.
In fact we have to take first a formal language L for the signature in question. Then we consider the sets of L-sentences T (theories). The structural properties P are the properties that can be expressed as

"$\mathcal{A}$ is a model of T".

In particular, this notion depends on the language L

Back to Voevodsky

HoTT - Homotopy Type Theory

- HoTT is intensional dependent type theory with the Univalence Axiom.
- Structure Identity Principle in HoTT:

  Isomorphic Structures are Identical
- If $A \cong B$ means the type of isomorphisms from A to B and $Id_C(A, B)$ the type of witnesses that A, B are identical then
- $A \cong B \to Id_C(A, B)$
- (meaning: the type of functions above is inhabited).

Remark. Verification of typing $i : A \cong B$ may be problematic.

- It is a particular formal language in which only structural properties can be represented. To me it is not clear, how this is guaranteed..
- It uses (as in Martin-Löf Type Theory) the identity types $Id_A(a, a')$. This type is usually denoted $a \sim_A a'$. The elements of $Id_A(a, a')$ are "witnesses" of identity of a, a'.
- There is a predicative type universe U (if one considers the hierarchy of type universes it doesn't change much). The elements of U are (small) types. It is closed w.r.t. standard operations.
- There is a type theoretic version of the axiom of choice AC.

Back to Voevodsky

- Extensional equality of functions $f \stackrel{\sim}{\approx} f'$ means by definition $(\Pi x : A) f(x) \sim f'(x)$.
- It is included the Function Extensionality Axiom

$$(\Pi f, f')(f \stackrel{\sim}{\approx} f' \to f \sim f')$$

- As an immediate consequence one has the Eta Axiom

$$(\Pi f : C) f \sim (\lambda x : A) f(x)$$

- **Remark.** Again, when a certain type A is declared to be axiom in Type Theory it means that it is inhabited.

Back to Voevodsky

Remark. A well known result is that if one tries to add reductions corresponding to all extensional equalities, the resulting type theory is in general “very bad” - no decidable typing, *etc.*

Back to Voevodsky

- A type X is contractible if the type $contr(X)$

$$(\Sigma x : X)(\Pi x' : X)x \sim x'$$

  is inhabited. (This property cannot be verified constructively.)

- If $f : A \to B$ let $A \overset{\sim}{-}_f B$ mean

$$(\Pi y : B)contr(f^{-1}(y)) = (\Pi y : B)contr((\Sigma x : A)f(x) \sim y)))$$

- In HoTT this can be used to express that f is a weak equivalence. Let $AA \overset{\sim}{-} B$ mean

$$(\Sigma f : A \to B)(A \overset{\sim}{-}_f B$$

  Using this one can represent reflectivity, *etc.* ("Express" does not mean "verify".)

Back to Voevodsky

- If A, B, C are types one can define the type of isomorphisms $A \overset{\sim}{=}_f B$ as

$$(\Sigma g : B \to A)[(g \circ f \overset{\sim}{\sim} id_A) \times (f \circ g \overset{\sim}{\sim} id_B)]$$

  and naturally $A \overset{\sim}{=} B$ as $(\Sigma f : A \to B)A \overset{\sim}{=}_f B$. (No chance to have decidability here, but this is not the purpose.)
- In HoTT the type $A \overset{\sim}{=}_f B$ can express that $f : A \to B$ is a homotopy equivalence.
- Proposition. $A \overset{\sim}{=} B \to (A \leftrightarrow B)$.

  Proposition. $(A \overset{\sim}{-} B) \leftrightarrow (A \overset{\sim}{=} B)$.

Back to Voevodsky

- Let U be a type universe. Using the Elimination rule for U there is $EXYZ : (X \overset{\sim}{-} Y)$ for $X, Y : U$ and $Z : (X \sim Y)$
  So $EXY : (X \sim Y) \to (X \overset{\sim}{-} Y)$ for $X, Y : U$.
- The Univalence Axiom $UA(U)$ is defined as

$$(\Pi X, Y : U)[(X \sim Y) \overset{\sim}{-}_{EXY} (X \overset{\sim}{-} Y)]$$

- Theorem. $UA(U) \to [X \overset{\sim}{=} Y \leftrightarrow X \sim Y]$ for $X, Y : U$.
- Theorem.Univalence Axiom and Eta Axiom for U implies Function Extensionality for U

.

Back to Voevodsky

Theorem (Voevodsky)
In ZFC+ a Grothendiek Universe, intensional dependent type theory with $UA(U)$ has an interpretation in the category of simplicial sets.

Back to Voevodsky

Conclusion

My aim was to try to place the suggestions of Voevodsky in the context of Type Theory and its problems. I did not want to go too much into technical details.

- My own impression was that his ideas certainly are interesting.
- The use of the language (proof assitant) Coq to test these ideas and their formalization is very interesting as well.
- Discovery of new models is already an important contribution.
- At the same time I am not sure that Voevodsky himself (or anybody) is able at the moment to estimate the impact of his suggestions.
- Also I do not think that constructive and non-constructive parts are “cleanly separated”.