Univalent Foundations Project

(a modified version of an NSF grant application)

Vladimir Voevodsky

October 1, 2010

1 General outline of the proposed project

While working on the completion of the proof of the Bloch-Kato conjecture I have thought a lot about what to do next. Eventually I became convinced that the most interesting and important directions in current mathematics are the ones related to the transition into a new era which will be characterized by the widespread use of automated tools for proof construction and verification.

I have started to actively learn about the related subjects around 2003/04. A few years ago I have come up with an idea for a new semantics for dependent type theories - the class of formal systems which are widely used in the theory of programming languages. Unlike the usual semantics which interpret types as sets this "univalent semantics" interprets types as homotopy types. The key property of the univalent interpretation is that it satisfies the *univalence axiom* a new axiom which makes it possible to automatically transport constructions and proofs between types which are connected by appropriately defined *weak equivalences*. In 2009/2010 I made a number of presentations on the univalent interpretation which were received with great interest by the type theoretic community. As of today two special events have been planned for the further discussion of the related ideas - a workshop in Oberwolfach in March 2011 and a year long special program at the Institute for Advanced Study in 2012-2013.

Eventually it became clear to me that the univalent semantics is just a first step and that I am really working on new foundations of mathematics. The key features of these "univalent foundations" are as follows:

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

To see what homotopy theory has to do with foundations of mathematics consider the following definitions which take place in the usual homotopy category:

A (homotopy) type T is said to be of h-level 0 if it is contractible,

A (homotopy) type T is said to be of h-level 1 if for any two points of T the space of paths between these two points is contractible,

A (homotopy) type T is said to be of h-level n+1 if for any two points of T the space of paths between these two points is of h-level n.

Then we have:

There is only one (up to a homotopy equivalence) type of h-level 0 - the one point type pt.

There are exactly two types of h-level 1, pt and $\emptyset$ i.e. types of h-level 1 are the truth values.

Types of h-level 2 are types such that the space of paths between any two points is either empty or contractible. Such a type is a disjoint union of contractible components i.e. (up to an equivalence) types of h-level 2 are sets.

Types of h-level 3 are (homotopy types of nerves of) groupoids.

More generally, types of h-level n+2 can be seen as equivalence classes of n-groupoids.

We can now reason as follows:

Set-level mathematics works with structures on sets i.e. on types of h-level 2, the collection of all sets with a given kind of structure in a universe $u0$ forms a groupoid i.e. a type of h-level 3 in any universe $u1$ which contains $u0$ as a member,

Category-level mathematics works with structures on groupoids i.e. on types of h-level 3 (it is easy to see that a category is a groupoid level analog of a partially ordered set). The collection of all groupoids with a given kind of structure in a universe $u0$ forms a 2-groupoid i.e. a type of h-level 4 in any universe $u1$ which contains $u0$ as a member etc.

Through this correspondence mathematics of all levels can be seen as dealing with structures on homotopy types.

I believe that this point of view has been known to many mathematicians but until now it has not been very useful because there was no way to axiomatize the world of homotopy types directly without reducing everything to sets. The key point which makes the present day situation different is that we now understand that

it is possible to directly formalize the world of homotopy types using the class of languages called dependent type systems and in particular Martin-Lof type systems.

As far as I know this was first understood by myself and, independently and from a somewhat different perspective by Steve Awodey, in around 2005.

One of the flavors of Martin-Lof type systems has been implemented in a proof assistant called Coq. It's development was mostly influenced by its role as a tool for the verification of programs. At the moment it is probably the leader among the the general purpose proof assistants and is being taught in the context of the program verification to a large number of computer science students every year.

I am developing the univalent foundations directly in Coq. The actual "foundations" is a body of definitions, theorems and proofs written in the formal language of Coq and, of course, verified by it. These Coq files can be found at `http://github.com/vladimirias/Foundations/`.

In my view, developments in the following three areas are most needed to ensure the adoption of the univalent foundations (with the associated type-theoretic formalization) by mathematicians and the adoption of the univalent semantics by the type-theorists:

1. The univalent foundations are connected with the commonly accepted language of current mathematics through the univalent model of type theory. Most of the work on this model has already been done but some more will be required to prepare the corresponding publications.

2. Basic mathematics has to be developed in univalent foundations (in Coq) up to a level when other people can start to build on the provided platform.

3. In order to prove that the univalence axiom does not compromise the constructive character of the Coq system one has to prove that this axiom is "constructive". The precise meaning of this statement is discussed below.

2 Type systems and their semantics

The approach to type systems and their semantics outlined below is new. I have developed it in 2009 in order to give a rigorous construction of the univalent model for Martin-Lof type theory. It was influenced by the approach of [6]. For another approach see e.g. [3, Ch. 11].

Type systems are syntactic objects which are specified in several steps. First one chooses a formal language L which allows the use of variables and substitution. Then one chooses a collection of relations on the sets of L-expressions with a given set of free variables which is stable under the substitutions. These relations are called the reduction rules and the equivalence relation generated by the reduction rules is called the conversion relation.

A pre-context in L is defined as a sequence of pairs $(x_i, T_i)_{i=1,\dots,n}$ where x_i's are names of variables and T_i is an expression in L with free variables from the set $\{x_1, \dots, x_{i-1}\}$. One conventionally writes such a sequence in the form $(x_1 : T_1, \dots, x_n : T_n)$. A pre-sequent in L is defined as a pair which consists of a pre-context $(x_1 : T_1, \dots, x_n : T_n)$ together with a pair of expressions (t, T) in L with free variables from the set $\{x_1, \dots, x_n\}$. A type system based on L is defined as a pair of subsets BB and $\widetilde{BB}$ in the sets of pre-contexts and pre-sequents respectively which satisfy a number of conditions with respect to reduction and substitution. Elements of BB are called the (valid) contexts of a type system and elements of $\widetilde{BB}$ the (valid) sequents of the type system. They are conventionally written in the form $x_1 : T_1, \dots, x_n : T_n$ for contexts and $x_1 : T_1, \dots, x_n : T_n \vdash t : T$ for sequents. The capital Greek letters Γ, Δ etc. are often used to denote a general (valid) context. A sequent of the form $x_1 : T_1, \dots, x_n : T_n \vdash t : T$ is "translated" into the natural language as the sentence "if x_1 is a variable of type T_1, x_2 is a variable of type T_2 etc. then $T = T(x_1, \dots, x_n)$ is a valid type expressions and $t = t(x_1, \dots, x_n)$ is a valid term expression of type T".

The conversion relation together with the relation defined by the re-naming of the context variables produce equivalence relations on BB and $\widetilde{BB}$. Let B and $\widetilde{B}$ be the corresponding quotients. Let $\partial : \widetilde{B} \to B$ be the map which sends a sequent $x_1 : T_1, \dots, x_n : T_n \vdash t : T$ to the context $x_1 : T_1, \dots, x_n : T_n, x_{n+1} : T_{n+1}$ and $ft : B \to B$ the map which sends a context $x_1 : T_1, \dots, x_n : T_n$ to the context $x_1 : T_1, \dots, x_{n-1} : T_n$ and sends the empty context ($n = 0$) to itself.

A model of a type system $(B, \widetilde{B})$ with values in a category $\mathcal{C}$ is defined by a mapping which sends a context Γ to an object $M(\Gamma)$ together with a morphism $p_\Gamma : M(\Gamma) \to M(ft(\Gamma))$ and a sequent S to a section of $p_{\partial(S)}$ i.e. to a morphism $s(S) : M(ft(\partial(S))) \to M(\partial(S))$ such that $p_{\partial(S)} \circ s(S) = Id$. To define a model this mapping should satisfy some additional conditions, in particular it should send the empty context to a final object pt of the target category. A rigorous construction of a model of realistic type theory is usually a complex undertaking. The idea outlined below allows to subdivide such constructions into two independent parts which require different techniques and can be independently adapted to the changes in the type theory or in the target category of the model.

In practice, the sets BB and $\widetilde{BB}$ are usually defined as the sets generated by some finite collection of rules which are specific to a given type system. The rules are often written using "above the

line/below the line" notation. For example

$$\frac{\Gamma, x : T}{\Gamma, x : T \vdash x : T}$$

expresses the rule which says that if $\Gamma, x : T$ is a valid context then $\Gamma, x : T \vdash x : T$ is a valid sequent. Good examples of definitions of type systems can be found in [1] (see in particular, p. 205).

From the algebraic perspective the context and sequent formation rules may be considered as partially defined operations on the sets B and $\widetilde{B}$. From this perspective a type system can be seen as model of a quasi-algebraic theory (see [5]) with two sorts B and $\widetilde{B}$ and a set of partially defined operations satisfying equational conditions. The notations of the underlying system of expressions and the reduction rules are usually chosen in such a way as to ensure that this is the initial model for a particular theory i.e. the free object with no generators.

The first part of the construction of models of a type system looks as follows. First one has to write down a specification of the quasi-algebraic theory underlying the type system in the form of a set of operations, equations specifying the domains of definition of these operations and the equations specifying the properties of these operations. Then one has to prove that the type system realizes the initial model of this theory. An example of such a proof (which does not use the terminology introduced above) can be found in [6, pp. 181-220]. These proofs can be very long and tedious but they are quite elementary and can be formalized and computer-verified using a proof assistant with very simple underlying type system. In addition, the quasi-algebraic theories which appear in the theory of type systems form an hierarchy by being combinations of different "conservative" extensions of the core theory of contextual structures and in practice adapting a fully formalized proof of this kind for one type system to another one can be quite easy.

Once the correspondence between a type system and a particular quasi-algebraic theory has been established construction of models of the type system becomes equivalent to the construction of models of the quasi-equational theory. Since one wants to obtain a model with values in a category it means that one need to construct a model of a quasi-algebraic theory, which is an object defined up to an isomorphism, from a category with some additional structures, which is an object defined up to an equivalence. One technique for doing this which I found very useful is based on the notion of a universe structure in a category. Let $\mathcal{C}$ be a category. By a universe structure on $\mathcal{C}$ we will mean a collection of data of the following form:

1. a final object pt in $\mathcal{C}$,
2. a morphism $p : \widetilde{U} \to U$,
3. for any morphism $f : X \to U$ a choice of a pull-back square

$$\begin{array}{ccc} (X, f) & \xrightarrow{Q(f)} & \widetilde{U} \\ {\scriptstyle p_{(X,f)}}\downarrow & & \downarrow{\scriptstyle p} \\ X & \xrightarrow{f} & U \end{array}$$

No conditions on the "coherence" of the choices are required. Let us write $(X, f_1, \ldots, f_n)$ for $(\ldots((X, f_1), f_2)\ldots, f_n)$. Let now B be the set of sequences of the form $(pt, f_1, \ldots, f_n)$ (for all $n \geq 0$) and $\widetilde{B}$ be the set of pairs $(pt, f_1, \ldots, f_{n+1}), s$ where s is a section of the morphism $p_{(pt,f_1,\ldots,f_{n+1})}$. These two sets can be equipped with the contextual structure in a canonical way such that an

equivalence between categories with universe structures defines an isomorphism between the corresponding contextual structures. Since changing the choice of the pull-back squares or of the final object creates an equivalent category with universe structure the contextual structure is well defined up to a canonical isomorphism by the pair $(\mathcal{C}, p)$.

When $\mathcal{C}$ is an lccc - a locally cartesian closed category, then many important for type theory structures on $(B, \widetilde{B})$ correspond to structures on the pair $(\mathcal{C}, p)$. In particular this applies to the structures corresponding to such type theoretic constructions as dependent products (with evaluation, abstraction and $\beta\eta$-reduction), dependent sums, universes and, most importantly for us, Martin-Lof identity types.

Combining the two parts one can prove that to construct a model of a given type system in a given lccc $\mathcal{C}$ it is sufficient to construct a morphism $p : \widetilde{U} \to U$ in $\mathcal{C}$ together with some additional structures on this morphism. For example, in order to interpret dependent products one needs to choose two horizontal arrows $\widetilde{P}$ and P in the diagram below which make it into a pull-back square:

$$\begin{array}{ccc} \underline{Hom}_U(\widetilde{U}, U \times \widetilde{U}) & \xrightarrow{\widetilde{P}} & \widetilde{U} \\ \downarrow & & \downarrow \\ \underline{Hom}_U(\widetilde{U}, U \times U) & \xrightarrow{P} & U \end{array}$$

3 Univalent semantics of Martin-Lof type systems

The approach outlined in the previous section reduces the problem of constructing a model for a given type system to the construction of a locally cartesian closed category $\mathcal{C}$, a morphism $p : \widetilde{U} \to U$ and some additional structures on p which depend on the particular type system. To define $\mathcal{C}$ and p which correspond to the univalent model of Martin-Lof type systems we have to assume the consistency of ZFC with at least one unreachable cardinal α. Depending on the particular rules for the existence of universes and for the behavior of the eliminators for inductive definitions relative to these universes in the type system we may have to assume the existence of a more or less extended system of large cardinals which are less than α in the set theory.

As was mentioned above the univalent model takes values in the homotopy category H associated with a given set theory. To construct this model one first chooses a locally cartesian closed model category (in the sense of homotopy theory) for H, constructs a model with values in this category and then projects it to H itself. I prefer to use the category of simplicial sets as the model one.

Let me recall some basic definitions. The simplicial category Δ is the category whose objects are natural numbers (denoted $[n]$) and morphisms from $[m]$ to $[n]$ are order preserving maps from the finite set $\{0, \dots, m\}$ to the finite set $\{0, \dots, n\}$. A simplicial set is a contravariant functor from Δ to $Sets$. The category of simplicial sets is denoted $\Delta^{op}Sets$. It is a Grothendieck topos and in particular an lccc. For a classic introduction to the homotopy theory of simplicial sets see [4]. The key definitions which are needed for the formulation of the main Theorem 3.1 given below are as follows.

One defines Δ^n as the functor represented by $[n]$. Let Λ^n_k, $k = 0, \dots, n$ be the smallest simplicial subset of Δ^n which contains all simplexes of dimension $n-1$ except for the k-th one.

One defines Kan fibrations as morphisms $q : E \to B$ such that for any n and k and any commutative square of the form

$$\begin{array}{ccc} \Lambda^n_k & \longrightarrow & E \\ \downarrow & & \downarrow q \\ \Delta^n & \longrightarrow & B \end{array}$$

where the left had side vertical arrow is the natural inclusion, there exists a morphism $\Delta^n \to E$ which makes both triangles commutative. A simplicial set whose morphism to the point is a Kan fibration is called a Kan simplicial set.

A morphism $i : A \to X$ is called an anodyne morphism if for any Kan fibration $q : E \to B$ and any commutative square of the form

$$\begin{array}{ccc} A & \longrightarrow & E \\ {\scriptstyle i}\downarrow & & \downarrow{\scriptstyle q} \\ X & \longrightarrow & B \end{array}$$

there exists a morphism $X \to E$ which makes both triangles commutative. Clearly, the inclusions $\Lambda^n_k \to \Delta^n$ are anodyne but there are actually many more anodyne morphisms. A morphism which becomes an isomorphism if one inverts all anodyne morphisms is called a weak equivalence. A simplicial set whose projection to the point is a weak equivalence is called (weakly) contractible.

The localization of $\Delta^{op}Sets$ with respect to the class of weak equivalences (or, equivalently the class of anodyne morphisms) is equivalent to the usual homotopy category defined in terms of sufficiently "nice" topological spaces and homotopy equivalences.

By a well-ordered simplicial set I will mean a simplicial set X together with well orderings on all of the sets $X_n = X([n])$ (the well orderings need not be compatible with maps between X_n's defined by the morphisms of Δ). Let us say that an ordered morphism $f : X \to Y$ between two simplicial sets is a morphism together with well-orderings of all its fibers.

Since there is at most one isomorphism between two well ordered sets there is at most one codomain fixing, order preserving isomorphism between two ordered morphisms. This makes it possible to consider the universal ordered morphism $\widetilde{V}_{<\alpha} \to V_{<\alpha}$ with fibers of cardinality $< \alpha$ such that any ordered morphism $f : Y \to X$ with such fibers can be included into a unique, up to a unique isomorphism, pull-back square of the form

$$\begin{array}{ccc} Y & \longrightarrow & \widetilde{V}_\alpha \\ \downarrow & & \downarrow \\ X & \longrightarrow & V_{<\alpha} \end{array}$$

Since the class of Kan fibrations is closed under pull-backs (this is obvious from their definition) there is a sub-morphism $p_{<\alpha} : \widetilde{U}_{<\alpha} \to U_{<\alpha}$ in $\widetilde{V}_{<\alpha} \to V_{<\alpha}$ which is the universal ordered Kan fibration with fibers of cardinality $< \alpha$.

Theorem 3.1 *[2010.2.2.th1] For a sufficiently large α there are structures on $p_{<\alpha}$ corresponding to the products of families of types, dependent sums, impredicative Prop, universes, Martin-Lof identity types and strictly positive inductive types. Moreover, in a well defined sense, the space of such structures on $p_{<\alpha}$ is contractible.*

Note that the second half of the theorem asserts that these structures on $p_{<\alpha}$ are essentially unique. The interpretations of the main constructions of the type system which arise from these structures on $p_{<\alpha}$ are as follows:

Contexts are interpreted as Kan simplicial sets and the "display maps" $M(\Gamma) \to M(ft(\Gamma))$ as Kan fibrations.

Prop is interpreted as $\{0, 1\}$ (note that it is not the sub-object classifier of the topos structure on $\Delta^{op}Sets$).

Universes are interpreted as sub-objects of $U_{<\alpha}$ classifying Kan fibrations with fibers of the size bounded by large cardinals $< \alpha$.

Dependent sum $\Gamma, y : \sum x : X, P(x)$ is interpreted as $M(\Gamma, x : X, y : P)$ with the projection to $M(\Gamma)$ being the composition $M(\Gamma, x : X, y : P) \to M(\Gamma, x : X) \to M(\Gamma)$.

Dependent product $\Gamma, f : \prod x : X.P(x)$ is interpreted as the simplicial sets of sections of the fibration $q : M(\Gamma, x : X, y : P) \to M(\Gamma, x : X)$ over $M(\Gamma)$ i.e. $M(\Gamma, y : \prod x : X, P(x))$ is defined by the pull-back square

$$\begin{array}{ccc} M(\Gamma, y : \prod x : X, P(x)) & \longrightarrow & M(\Gamma) \\ \downarrow & & \downarrow Id \\ \underline{Hom}_{M(\Gamma)}(M(\Gamma, x : X), M(\Gamma, x : X, y : P)) & \xrightarrow{\circ q} & \underline{Hom}_{M(\Gamma)}(M(\Gamma, x : X), M(\Gamma, x : X)) \end{array}$$

The Martin-Lof identity type $Id\, X\, x\, x'$ is interpreted as the simplicial paths space from x to x' in X.

To describe the interpretation of the general strictly-positive inductive types used in Coq some additional explanations are required. However, the usual inductive types such as natural numbers, trees etc. are interpreted as they would in any set-theoretic semantics.

Definition 3.2 *The models of type theory which correspond to the pair $(\Delta^{op}Sets, p_{<\alpha})$ are called standard univalent models.*

These models were foreseen to some extend by Hofmann and Streicher in [2]. They have also foreseen in that paper some of the implications which this view of the semantics will have on the formalization of mathematics.

Two properties of $p_{<\alpha}$ are of the key importance for proving Theorem 3.1. Both are proved using the theory of so called minimal fibrations. The first one is simple:

Theorem 3.3 *For any infinite α the simplicial set $U_{<\alpha}$ is a Kan simplicial set.*

The second one requires an additional definition which plays a key role in the whole picture.

For any morphism $q : E \to B$ consider the simplicial set $\underline{Hom}_{B\times B}(E \times B, B \times E)$. If q is a fibration then it contains, as a union of connected components, a simplicial subset $weq(E\times B, B\times E)$ which corresponds to morphisms which are weak equivalencies. The obvious morphism from the diagonal $\delta : B \to B \times B$ to $\underline{Hom}_{B\times B}(E \times B, B \times E)$ over $B \times B$ factors uniquely through a morphism

$$m_q : B \to weq(E \times B, B \times E)$$

Definition 3.4 *A fibration q is called univalent if m_q is a weak equivalence.*

Theorem 3.5 *For any infinite α, the fibration $p_{<\alpha}$ is univalent.*

While the univalence property of $p = p_{<\alpha}$ is used to show that there are structures on p which are required for the construction of models of Martin-Lof type systems it itself is not required for the construction of such models. However the condition which makes it necessary for the universe structure p to be univalent in order to support models of a given type system can be expressed

in the language this type system if it has a sufficiently rich universe structure (e.g. if any type is a member of a universe which is itself a member of another universe). The corresponding formal statement is called the *univalence axiom*. I will discuss this axiom in more detail below but first I need to explain how to express the fundamental concepts of the univalent foundations in Martin-Lof type systems. I will be using the syntax of Coq proof assistant for my constructions but one can easily see that they can be repeated in any other Martin-Lof type system.

4 Formalization of the basic concepts of the univalent foundations in Coq

The last pages of the proposal reproduce the actual Coq code which contains the formulation of the univalence axiom for a particular universe UU0 as well as some other constructions. As explained in the comment at the beginning of the code the current "universe management" in Coq is somewhat maladapted to our purpose and as a result we need to repeat some code twice. If we ignore this technical complication we get a sequence of definitions and constructions which leads to the formulation of the univalence axiom and looks as follows.

First we re-define the Martin-Lof identity types under the notation "paths". Then, for convenience, we re-define the dependent sums under the notation "total2" which reminds that it is a total space of a fibration (the standard Coq notation for dependent sums is "sigT"). Then we define the notion of contractibility. Then the notion of a homotopy fiber of a function over a term. Then we define a weak equivalence as a function such that all its homotopy fibers are contractible. Then we prove that the identity function is a homotopy equivalence which allows us to define the map *eqweqmap* whose model is m_d. Finally, we formulate the univalence axiom as the condition that *eqweqmap* is a weak equivalence.

We also define two structures which are corollaries of the univalence axiom - the mapping *weqtopaths* from the type of weak equivalences between X and Y to the type of paths between the corresponding points of the universe and the family of paths *weqpathsweq* which essentially shows that the *weqtopaths* is the left inverse to *eqweqmap*. As far as I understand the combination of these two corollaries is equivalent to the axiom itself but I have not checked it yet.

One of the important type-theoretic corollaries of the univalence axiom is that it implies both functional and dependent extensionality i.e. using the univalence axiom one can prove that a family of paths betweens the values of two functions (or, more generally, sections of families) defines a path between the functions. The proof can be found in the Coq files on my web page.

The next page of code contains the formalization of the concept of h-level for types and functions which is central to the univalent "worldview". As was mentioned above, types of h-level 1 correspond to the truth values. In type theory truth values are usually considered to be members of a special universe denoted *Prop*. Univalent approach suggests that there is no need for a special universe called *Prop* and that one can should use types of h-level 1 instead. Such types, by definition, satisfy an analog of the proof irrelevance principle as well as "local impredicativity" - the dependent product of a family of types of h-level 1 with both the base and the fibers being in a universe UU is again a type of h-level 1 in UU.

When one adapts this point of view one should make sure that the types which appear in the definitions which are supposed to express *conditions* as opposed to structures are indeed of h-level 1. For example, to support the assertion that the key definition *iscontr* expresses the *condition* on a type to be contractible one has to prove that for any type X the type $iscontr\,X$ is indeed of h-level 1. The proof (using the univalence axiom) can be also be found in the Coq files on my web page. Once it is shown that *iscontr* is of h-level 1 it is relatively easy to show that other definitions such as *isweq* or $isofhlevel$ are of h-level 1.

According to the general univalent point of view sets in a given universe are defined as types of

level 2 in this universe. The property of being of level 2 is somewhat stronger then the so called Axiom K and in particular all the Coq developments which require this axiom can be reformulated as unconditional constructions on types of level 2.

The concept of h-levels extends from types to functions through the simple definition saying that $f : X \to Y$ is of h-level n if all its homotopy fibers are of h-level n. In particular, f is of h-level 0 if and only if it is a weak equivalence and of h-level 1 if and only if it is "injective". This concept of h-levels of functions can be used in particular to create some order on the forgetting "functors" which connect the moduli types of algebraic structures. For example, the fact that the field structure is actually a property of the underlying ring structure is expressible as the assertion that the corresponding forgetting map is of h-level 1.

5 Constructiveness of the univalence axiom

The type system of Coq is constructive. While the general definition of constructiveness is not so easy to produce this property is well illustrated by the following fact: any term of the type nat (natural numbers) which is in the normal form is of the form $S(S(\ldots(S\ O)\ldots))$ i.e. it is the standard representation of a particular natural number n. Together with a theorem which says that the normalization algorithm terminates and therefore each term has a normal form (which is conditional on the consistency of something like set theory) this implies for example that if one constructs (without the use of any axioms) any term f of the type $T- > nat$ then by typing $eval\,compute\,(ft)$ for a particular $t : T$ and waiting for a while one will obtain a concrete natural number.

To be even more specific consider the following scenario. Suppose we have defined what a group is, what a field is, what a finite set is and constructed a function $numberofelements : FSets- > nat$. Suppose further that we have defined what $GL_n(k)$ is for a field k and $n : nat$ and what the finite fields $\mathbf{F}_q$ are. Finally suppose that we have proved that the underlying set of a general linear group over a finite field is finite. Combining these proofs and definitions together we will define a function $primes- > nat- > nat- > nat$ which sends (p, m, n) to the number of elements in $GL_n(\mathbf{F}_{p^m})$. Applying the previous comment to this function we see that for each (p, m, n) the system will compute the number of elements of $GL_n(\mathbf{F}_{p^m})$ using nothing but the definitions and the proofs of finiteness of this group.

If we add to the system the axiom of excluded middle (forall P:Prop, P$\vee \neg$ P) then this constructiveness will disappear - now there will be terms in the normal form of type nat which are not of the form $S(S(\ldots(S\ O)\ldots))$ and whose "value" as a natural number may be impossible to determine. This shows that the axiom of excluded middle is not constructive.

I expect that the univalence axiom is constructive. In particular, I conjecture the following:

Conjecture 1 *There is a terminating algorithm which for any term nn of type nat constructed with the use of the univalence axiom outputs a pair (nn', pf) where nn' is a term of type nat constructed without the use of the univalence axiom and pf is a term of type paths nat nn nn' (i.e. a proof that $nn' = nn$). The term pf may use the univalence axiom.*

This conjecture seems to be highly non-trivial. The only way to approach it which I can see involves detailed analysis of the structure of terms of different types in the Coq system. It might be easier to start with a more elementary type system especially in relation to the structure of inductive definitions and the universe structure. In any event I find this conjecture to be both very important for the univalent foundations and very interesting.

```
(* What follows is an example of Coq 8.2 code written using the Proof-general interface for Coq based on
emacs. The code consists of three modules u0, u1 and u01. The need for such an organization in this case
arises from the current state of "universe management" in Coq. While there is a mechanism for automatic
universe level assignment to occurrences
of "Type" expression this mechanism does not allow for a possibility of universe polymorphic constructions.
Since to formulate the equivalence axiom properly we need to define paths spaces for types on at least two
universe levels we have to use "manual" universe management. This is done by the creation of two modules u0
and u1 which are identical except for their names. Each starts with a definition UU:=Type which fixes a
universe named UU. When both are imported into the module u01 we find ourselves in the possession of two
universes u0.UU and u1.UU with all the results and definitions of the modules u0, u1 for each of the
universes. When Coq sees a reference to an identifier from u0 or u1 which is used without the explicit use of
"u0." or "u1." as a prefix it assigns to it prefix "u0." since u0 was imported after u1. As a result UU now
refers to u0.UU etc.

In the first four lines of u01 we rename u0.UU into UU0 and u1.UU into UU1. We also use these two universes
as parts of definitions which force Coq to introduce constrains on the corresponding universe variables which
correspond to the conditions that UU0 is both a member and a sub-universe of UU1. After these constrains have
been established any attempt to use these universes which is incompatible with the constraints e.g. to
consider UU1 as a member of UU0 will generate an error message 'Universe inconsistency". *)


Module u0.

Definition UU:=Type.

Inductive paths (X:UU)(x:X): X -> UU := idpath: paths X x x.

Inductive total2 (X:UU) (P:X -> UU) : UU := tpair: forall x:X, forall p: P x, total2 X P.
Definition pr21 (X:UU) (P:X -> UU) (z: total2 X P) : X :=
match z with tpair x p => x end.
Definition pr22 (X:UU) (P:X -> UU) (z: total2 X P) : P (pr21 _ _ z):=
match z with tpair x p => p end.

Definition iscontr (X:UU) : UU := total2 X (fun cntr:X => forall x:X, paths X x cntr).
Definition iscontrpair (X:UU) (cntr: X) (e: forall x:X, paths X x cntr) : iscontr X :=
tpair X (fun cntr:X => forall x:X, paths X x cntr) cntr e.


Definition hfiber (X:UU)(Y:UU)(f:X -> Y)(y:Y) : UU := total2 X (fun pointover:X => paths Y (f pointover) y).
Definition hfiberpair (X:UU)(Y:UU)(f:X -> Y)(y:Y) (x:X) (e: paths Y (f x) y): hfiber _ _ f y :=
tpair X (fun pointover:X => paths Y (f pointover) y) x e.

Definition isweq (X:UU)(Y:UU)(F:X -> Y) : UU := forall y:Y, iscontr (hfiber X Y F y) .

Lemma idisweq (X:UU) : isweq X X (fun t:X => t).
Proof. intros. unfold isweq. intros. assert (y0: hfiber X X (fun t : X => t) y).
apply (tpair X (fun pointover:X => paths X ((fun t:X => t) pointover) y) y (idpath X y)). split with y0.
intros.
destruct y0. destruct x. induction p. induction p0. apply idpath. Defined.

Definition weq (X:UU)(Y:UU) : UU := total2 (X -> Y) (fun f:X->Y => isweq X Y f) .
Definition weqpair (X:UU)(Y:UU)(f:X-> Y)(is: isweq X Y f) : weq X Y :=
tpair (X -> Y) (fun f:X->Y => isweq X Y f) f is.
Definition idweq (X:UU) : weq X X := tpair (X-> X) (fun f:X->X => isweq X X f) (fun x:X => x) ( idisweq X ) .

Definition invmap (X:UU) (Y:UU) (f:X-> Y) (isw: isweq X Y f): Y->X.
Proof. intros. unfold isweq in isw. apply (pr21 _ _ (pr21 _ _ (isw X0))). Defined.

Definition weqfg (X:UU) (Y:UU) (f:X-> Y) (is1: isweq _ _ f): forall t2:Y, paths Y (f ((invmap _ _ f is1) t2))
t2.
Proof. intros. unfold invmap. simpl. unfold isweq in is1. apply (pr22 _ _ (pr21 _ _ (is1 t2))). Defined.

End u0.
```

```
Module u1.

Definition UU:=Type.

Inductive paths (X:UU)(x:X): X -> UU := idpath: paths X x x.

Inductive total2 (X:UU) (P:X -> UU) : UU := tpair: forall x:X, forall p: P x, total2 X P.
Definition pr21 (X:UU) (P:X -> UU) (z: total2 X P) : X :=
match z with tpair x p => x end.
Definition pr22 (X:UU) (P:X -> UU) (z: total2 X P) : P (pr21 _ _ z):=
match z with tpair x p => p end.

Definition iscontr (X:UU) : UU := total2 X (fun cntr:X => forall x:X, paths X x cntr).
Definition iscontrpair (X:UU) (cntr: X) (e: forall x:X, paths X x cntr) : iscontr X :=
tpair X (fun cntr:X => forall x:X, paths X x cntr) cntr e.


Definition hfiber (X:UU)(Y:UU)(f:X -> Y)(y:Y) : UU := total2 X (fun pointover:X => paths Y (f pointover) y).
Definition hfiberpair (X:UU)(Y:UU)(f:X -> Y)(y:Y) (x:X) (e: paths Y (f x) y): hfiber _ _ f y :=
tpair X (fun pointover:X => paths Y (f pointover) y) x e.

Definition isweq (X:UU)(Y:UU)(F:X -> Y) : UU := forall y:Y, iscontr (hfiber X Y F y) .

Lemma idisweq (X:UU) : isweq X X (fun t:X => t).
Proof. intros. unfold isweq. intros. assert (y0: hfiber X X (fun t : X => t) y).
apply (tpair X (fun pointover:X => paths X ((fun t:X => t) pointover) y) y (idpath X y)). split with y0.
intros.
destruct y0. destruct x. induction p. induction p0. apply idpath. Defined.

Definition weq (X:UU)(Y:UU) : UU := total2 (X -> Y) (fun f:X->Y => isweq X Y f) .
Definition weqpair (X:UU)(Y:UU)(f:X-> Y)(is: isweq X Y f) : weq X Y :=
tpair (X -> Y) (fun f:X->Y => isweq X Y f) f is.
Definition idweq (X:UU) : weq X X := tpair (X-> X) (fun f:X->X => isweq X X f) (fun x:X => x) ( idisweq X ) .

Definition invmap (X:UU) (Y:UU) (f:X-> Y) (isw: isweq X Y f): Y->X.
Proof. intros. unfold isweq in isw. apply (pr21 _ _ (pr21 _ _ (isw X0))). Defined.

Definition weqfg (X:UU) (Y:UU) (f:X-> Y) (is1: isweq _ _ f): forall t2:Y, paths Y (f ((invmap _ _ f is1) t2))
t2.
Proof. intros. unfold invmap. simpl. unfold isweq in is1. apply (pr22 _ _ (pr21 _ _ (is1 t2))). Defined.

End u1.


Module u01.
Import u1 u0.

Definition j01:UU -> u1.UU:= fun T:UU => T.
Definition j11:u1.UU -> u1.UU:=fun T:u1.UU => T.

Definition UU0:=j11 UU.
Definition UU1:=u1.UU.

Definition eqweqmap (X:UU0) (Y:UU0) : (u1.paths _ X Y) -> (weq X Y).
Proof. intros. induction X0. apply idweq. Defined.

Axiom univalenceaxiom: forall X:UU0, forall Y:UU0, u1.isweq (u1.paths Type X Y) (weq X Y) (eqweqmap X Y).

Definition weqtopaths (X:UU0)(Y:UU0)(f:X -> Y)(is:isweq _ _ f): u1.paths _ X Y :=
u1.invmap _ _ (eqweqmap X Y) (univalenceaxiom X Y) (weqpair _ _ f is).

Definition weqpathsweq (X:UU0)(Y:UU0)(f:X -> Y)(is:isweq _ _ f): u1.paths _ (eqweqmap _ _ (weqtopaths _ _ f
is))
(weqpair _ _ f is) := u1.weqfg _ _ (eqweqmap X Y) (univalenceaxiom X Y) (weqpair _ _ f is).

End u01.
```

```
Inductive empty:UU := .
Inductive unit:UU := tt:unit.
Inductive bool:UU := true:bool | false:bool.
Inductive nat:UU := O:nat | S: nat -> nat.


Fixpoint isofhlevel (n:nat) (X:UU): UU:=
match n with
O => iscontr X |
S m => forall x:X, forall x':X, (isofhlevel m (paths _ x x'))
end.

Theorem hlevelsincl (n:nat) (T:UU) : isofhlevel n T -> isofhlevel (S n) T.



Definition isaprop (X:UU): UU := isofhlevel (S O) X.

Theorem isapropempty: isaprop empty.

Theorem isapropunit: isaprop unit.

Theorem isapropiscontr (X:UU0): isaprop (iscontr X).

Theorem isapropisweq (X:UU0)(Y:UU0)(f:X-> Y) : isaprop (isweq _ _ f).

Theorem isapropisofhlevel (n:nat)(X:UU0): isaprop (isofhlevel n X).



Definition isaset (X:UU): UU := isofhlevel (S (S O)) X.

Theorem impred (n:nat)(T:UU0)(P:T -> UU0): (forall t:T, isofhlevel n (P t)) -> (isofhlevel n (forall t:T, P
t)).

Theorem isasetifdec (X:UU): (forall (x x':X), coprod (paths _ x x') (paths _ x x' -> empty)) -> isaset X.

Theorem isasetbool: isaset bool.

Theorem isasetnat: isaset nat.



Definition isofhlevelf (n:nat)(X:UU)(Y:UU)(f:X -> Y): UU := forall y:Y, isofhlevel n (hfiber _ _ f y).
```

References

[1] H. P. Barendregt. Lambda calculi with types. In *Handbook of logic in computer science, Vol. 2*, volume 2 of *Handb. Log. Comput. Sci.*, pages 117–309. Oxford Univ. Press, New York, 1992.

[2] Martin Hofmann and Thomas Streicher. The groupoid interpretation of type theory. In *Twenty-five years of constructive type theory (Venice, 1995)*, volume 36 of *Oxford Logic Guides*, pages 83–111. Oxford Univ. Press, New York, 1998.

[3] Bart Jacobs. *Categorical logic and type theory*, volume 141 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Co., Amsterdam, 1999.

[4] Peter May. *Simplicial objects in algebraic topology*. Van Nostrand, 1968.

[5] E. Palmgren and S. J. Vickers. Partial horn logic and Cartesian categories. *Ann. Pure Appl. Logic*, 145(3):314–353, 2007.

[6] Thomas Streicher. *Semantics of type theory*. Progress in Theoretical Computer Science. Birkhäuser Boston Inc., Boston, MA, 1991. Correctness, completeness and independence results, With a foreword by Martin Wirsing.