Mathematical theory of type theories and the initiality conjecture

Research proposal to Templeton Foundation for 2016-2019

Project Description

Vladimir Voevodsky (PI) April 2016.

1 Introduction

The ultimate goal of the proposed research is to make computer assisted formalization and verification a norm in the work of pure mathematicians while preserving the highly abstract and novel character of pure mathematics that developed in the last several decades.

This goal, when it is achieved, is expected to have very serious transformational consequences both for the social practice of mathematics and for the content of mathematics.

Computer proof assistants are indispensable allies of those of us who want mathematics to be absolutely precise and transparent and I believe that in each case where it might look like a concession must be made to the representatives of the opposite camp there is hidden either a mistake or a large amount of new mathematics that is still to be developed.

The main focus of the current proposal - the *initiality conjecture*, is an example of a mathematical problem that grew out of the refusal to accept the "obvious" arguments employed in type theory.

At the moment the most important, in my opinion, problem that is facing the formalization movement is to ensure that the proof assistants that we are using and that will be appearing in the future can be trusted.

When Thomas Hales was choosing a proof assistant for the formalization of his proof of Kepler's Conjecture he chose HOL Light, a proof assistant that is often difficult to work with. One of the main reasons for this choice was the fact that the validity of proofs verified with HOL Light was beyond reasonable doubt modulo our believe in the consistency of ZFC. The reason for such a high level of trust in the correctness of HOL Light is that it is based on a very simple formal system called HOL (Higher order Logic) that is a small modification of the Church's type theory [4] and that the part of the proof assistant that actually verifies proofs is very small and can itself be reliably verified by humans[1].

The proof assistants used for the univalent formalization such as Coq, Agda or Lean are different. They are different first of all in that they are based on much more complex formal systems called dependent type theories. The difference between Church's type theory and, for example, the Calculus of Inductive Constructions, which is the foundation of Coq, is not simply in the number of rules. These are systems that belong to two distinct classes. Church's type theory is a so called simple type theory. Calculus of Inductive Constructions (CIC) is a dependent type theory.

Types (sorts) in a simple type theory are formed according to more or less complex rules but there are no type families parametrized by elements of other types. In the CIC sequences of dependence of arbitrary length are possible - there can be a type expression $T(x_0, \ldots, x_{n-1})$ where x_{n-1} has type

[1]After Thomas completed his proof the correctness of the HOL Light kernel was also formally verified using another proof assistant.

$T_{n-1}(x_0, \dots, x_{n-2})$ that depends on the previous $n-1$ variables, x_{n-2} has type $T_{n-2}(x_0, \dots, x_{n-3})$ that depends on $x_0, \dots, x_{n-3}$ and only x_0 has a type T_0 that does not depend on elements of any other types.

Mathematical theory of dependent type theories is underdeveloped, not because of the lack of trying but because it is very complex. Correspondingly, the proof assistants that are based on such type theories are not known to be always correct and indeed inconsistencies are found from time to time both in Coq and in Agda. Fortunately, these inconsistencies require the use of the parts of the type theories that have always been excluded from the UniMath language [12],[8] that the author of this proposal uses for his formalizations.

Martin-Löf 1992

UniMath, which is implemented currently in Coq, uses a minimal subset of the CIC that allows one to formalize the main concepts of the univalent foundations combined with the univalence axiom and an additional construct called the resizing rules.

Mathematically rigorous verification of the consistency of the UniMath language may be considered to be the main immediate practical goal of the research that I intend to conduct.

Achieving this practical goal in a way that will allow for the future extensions of this language to be verified using the same mathematical infrastructure requires careful development of the parts of the theory of type theories that are missing and in many cases reworking of the parts of the theory that are considered to be known at a new level of mathematical rigor.

2 Mathematical theory of expressions with free and bound variables

Type theories are formal deduction systems. They consist~~s~~ of rules for the formation of syntactic expressions or sequences of such expressions with some of such sequences representing mathematical theorems and some of such sequences representing proofs of theorems. A type theory may have a decidable type checking in which case the sequence of rules used in the construction of a proof need not be remembered because the correctness of a proof can be decidedly verified by a computer from the proof term that roughly speaking represents the text of the proof. Some type theories do not have decidable type checking. In that case the proof term is not always sufficient to verify whether or not it is correct and either a full deduction sequence or some hints about how to construct such a sequence need to be remembered.

In any case mathematical theory of type theories need to start with a mathematical theory of syntactic expressions of the kind that is used in type theories. Fortunately, a satisfactory mathematical theory of such expressions now exists due to the work of Fiore, Plotkin and Turi [5], Hirschowitz and Maggesi [6], Ahrens [1] and many others.

In this theory one wants to describe expressions such as, for example,

$$\Sigma(x0 : X), \Pi(x : X), Id\, X\, x\, x0$$

usually considered up to the α-equivalence, i.e., the renaming of bound variables. In the expression that we gave as an example $x0$ and x are bound variables and X may be considered as a free variable.

If one ignores the peculiarities of the exact syntactic representations then the possible shape of expressions in a theory is specified by a "binding signature".

Let us recall that a usual, algebraic, signature is a set Op of operations together with a function $Ar : Op \to \mathbf{N}$ that assigns to any operation its arity, i.e., the number of its arguments. A binding signature permits one to specify operations that bound some of the variables in their arguments. For example, the quantifier $\forall$ is usually written as $\forall\, x, P$ with the name of a variable x put forward to say that the occurrences of x in P will be bound by this quantifier (unless they are shadowed by another bound x). In the theory of expressions with bound variables $\forall$ is an operation with the binding arity (1). In general, a binding arity is a sequence of natural numbers $(n_1, \ldots, n_d)$ where n_i is the number of variables that an operations of this arity bounds in its i-th argument. An example where $d > 1$ can be seen in multi-sorted logic where $\forall$ has the form $\forall\, x : S, P(x)$. Here the binding arity is $(0, 1)$ since no variables are bound in S and one variable is bound in P.

The theory of algebraic expressions can be made independent of the details of the syntax through the paradigms of universal algebra. The set of expressions with variables from a set V corresponding to the algebraic signature $(Op, Ar : Op \to \mathbf{N})$ can be mathematically defined as the set of elements of the free algebraic structure with the set of operations Op where each operation $O \in Op$ is a function of the form

$$O : X^{Ar(O)} \to X$$

that is generated by V.

'irschowitz + Maggesi' A similar description exists now for the expressions, considered modulo the α-equivalence, whose form is determined by a binding signature Σ. The most accessible form of such a description can be found in [6, p.555]. There one considers the category of monads R on the category of sets $Sets$ that are equipped with operations corresponding to the elements of Op. Given an element $O \in Op$ such that $Ar(O) = (n_1, \ldots, n_d)$ one should consider operation on monads of the form

$$R^{(n_1)} \times \ldots \times R^{(n_d)} \to R \tag{1}$$

The explanation for this formula is as follows. Given a monad R one defines the notion of a left module over R and the notion of a linear morphism of such modules. For a module M one denotes by M' the functor $X \mapsto M(X \amalg pt)$ where pt is the chosen one point set. On this functor one constructs, quite easily, a structure of a left module over R. The notation $M^{(n)}$ refers to the left module obtained from M by applying the $M \mapsto M'$ construction n times. Similarly, one constructs the structure of a left module on the product of functors with left module structure. The monad is a left module over itself just as a ring is a module over itself. Therefore, both sides of (1) are left modules over R and an operation is a linear morphism of these modules.

We may consider now the category of monads equipped with operations of the form (1) for each of the elements of Op. It turns out that this category has an initial object R_Σ and, for a finite set X, the set $R_\Sigma(X)$ is in a bijective correspondence with the set of expressions whose shape is determined by Σ with free variables from X and considered modulo α-equivalence. The functor and the monad structures are given by substituting free variables for expressions in the way that is known in computer science as "capture-free substitution".

This gives us mathematical understanding of Σ-expressions. From this point on we may ignore the details of the syntax or complex algorithms for capture-free substitution and build our theory in such a way that all that we need from the sets of expressions and the substitution operation is encoded in the universal property of R_Σ.

The goal of the mathematical theory of type theories that I am working on is to obtain a similar characterization for a subset of dependent type theories that in particular contains a version of the

$\Gamma, A, B, o, f.$
metavariables

$$\frac{\begin{array}{l}\Gamma \vdash \\ \Gamma \vdash A \\ \Gamma, A \vdash B \\ \Gamma \vdash o : A \\ \Gamma \vdash f : \Pi(A, B)\end{array}}{\Gamma \vdash ap(f, o) : B[}$$

UniMath language.

3 An outline of my approach to the mathematical theory of type theories

Type theories are given to us most often in the form of lists of "inference rules" that are schematic expressions that usually have the form such as this one

$$\frac{\Gamma, x : A, y : B \rhd \quad \Gamma \rhd o : A \quad \Gamma \rhd f : \Pi(x : A), B}{\Gamma \rhd f\, o : B[o/x]} \tag{2}$$

which is the standard form of the inference rule for function application except for the use of the symbol $\rhd$ instead of the symbol $\vdash$. We prefer to use $\rhd$ because the use of $\vdash$ in the inference rules of type theories conflicts with its use in other places in logic. We also write $\Gamma, x : T \rhd$ instead of the more usual $\Gamma \rhd T$ *type*.

There are four types of sentences in type theories

$$\Gamma \rhd \qquad \Gamma \rhd o : T$$

$$\Gamma \rhd T = T' \qquad \Gamma \rhd o = o' : T$$

where Γ is a sequence of the form $x_0 : T_0, \ldots, x_{n-1} : T_{n-1}$, where T_i is an expression in variables $x_0, \ldots, x_{i-1}$, and o, o', T and T' are expressions in variables $x_0, \ldots, x_{n-1}$.

We will, in the next few pages, ignore the fact that there are sentences of the third and the fourth kind and proceed as if only sentences of the first two kinds existed.

One can interpret inference rules in mathematical terms as follows.

Firstly, one re-writes the expressions such as $\Pi(x : A), B$ and $f\, o$ in the standard form that makes the underlying binding signature visible. For Π it is $\Pi(A, x.B)$ with arity $(0, 1)$ and for $f\, o$, we need to introduce a name for this operation, for example, ap and then re-write $f\, o$ as $ap(f, o)$ with arity $(0, 0)$. One can now apply the general theory outlined in the previous section and say that by the word "expression" we mean an element of $R_\Sigma(X)$ for some X.

Secondly, for every monad R on *Sets* one considers the sets:

$$B(R, R) = \coprod_{n \geq 0} R(stn(0)) \times \ldots \times R(stn(n-1))$$

and

$$\widetilde{B}(R, R) = \coprod_{n \geq 0} R(stn(0)) \times \ldots \times R(stn(n-1)) \times R(stn(n)) \times R(stn(n))$$

where

$$stn(n) = \{i \in \mathbf{N} \mid i < n\}$$

is our choice for the standard set with n elements and the product of the empty sequence is taken to be $stn(1)$.

Elements of $B(R, R)$ can be written as

$$T_0, \ldots, T_{n-1} \rhd$$

elements of $\widetilde{B}(R,R)$ as

$$T_0,\ldots,T_{n-1}\rhd o:T$$

and elements of $stn(n)$ may be written as x_i instead of simply i.

We can now say that sentences of the first kind are elements of $B=B(R,R)$ and sentences of the second kind are elements of $\widetilde{B}=\widetilde{B}(B,R)$ and that inference rules are schemes that describe partially defined operations on pairs of sets of the form $B,\widetilde{B}$.

For partially defined operations on any set or collection of sets there is a notion of dependency where we say that operation Op_1 depends on the operation Op_2 if Op_2 is required for the description of the domain of definition of Op_1. We can analyze the tree of operations on which the application Ap defined by the inference rule (2) depends.

The domain of definition of Ap is the subset in $B\times\widetilde{B}\times\tilde{B}$ that consists of triples (X,r,s) such that

$$\partial(r)=ft(X)$$

and

$$\partial(s)=\Pi(X)$$

where ∂ and ft are the two most basic operations on $B,\widetilde{B}$

- operation ft on B takes $T_0,\ldots,T_{n-1}\rhd$ to $T_0,\ldots,T_{n-2}\rhd$ and

- operation ∂ from $\widetilde{B}$ to B takes $T_0,\ldots,T_{n-1}\rhd o:T$ to $T_0,\ldots,T_{n-1},T\rhd$.

Both ∂ and ft are everywhere defined and so do not depend on any other operations.

Operation Π is an operation that is given by the inference rule

$$\frac{\Gamma,x:A,y:B\rhd}{\Gamma,y:\Pi(A,x.B)\rhd}$$

that is defined on the subset of B of elements X such that $l(X)\geq 2$ where $l:B\to\mathbf{N}$ is the length function.

For Π we can say that it is an operation that depends on l ignoring the fact that l itself is not an operation. This quirk arises from our preference to present B as one set with a function l to $\mathbf{N}$ instead of presenting B and $\widetilde{B}$ each as a family of sets B_n and $\widetilde{B}_n$ parametrized by $\mathbf{N}$. In the latter case we could say that the structures that we discuss are (models of) essentially algebraic theories with infinitely many sorts. With the choice we make we can not say that but we gain the advantage of much greater simplicity of actual proofs. This choice, unexpected from some perspectives, was made as a result of formalization work. We may use the term essentially l-algebraic structures for the structures that arise in this way.

The scheme (2) actually describes more than just a partially defined operation. It also describes its syntactic realization.

Since substitution in (2) can be expressed through the monad operations this inference rule and the rules that it depends on may be translated into partially defined operations on $B(R,R)$ and $\widetilde{B}(R,R)$ for any monad R that is itself equipped with the usual, everywhere defined, operations of the form

$$\Pi:R\times R'\to R$$

and

$$ap : R \times R \to R$$

More generally, one can always "read of" any set of type-theoretic inference rules used in the literature the minimal binding signature Σ required to realize these rules as partial operations on $B(R_\Sigma, R_\Sigma)$ and $\widetilde{B}(R_\Sigma, R_\Sigma)$.

Given a set of inference rules Inf consider the corresponding minimal signature Σ and the smallest pair of subsets $B(Inf), \widetilde{B}(Inf)$ in $B(R_\Sigma, R_\Sigma)$, $\widetilde{B}(R_\Sigma, R_\Sigma)$ that is closed under the operations corresponding to elements of Inf.

The sets $\underline{B(Inf), \widetilde{B}(Inf)}$ provide a rigorous mathematical interpretation for the sets of derivable sentences of the type theory that is "generated" by the set of inference rules Inf. A proof assistant that is based on such a type theory will provide tools for the generation of elements of these two sets and for the verification that a given element of the set $B(R_\Sigma, R_\Sigma)$ (resp. $\widetilde{B}(R_\Sigma, R_\Sigma)$) belongs to $B(Inf)$ (resp. $\widetilde{B}(Inf)$).

The veracity of a proof assistant relative to the type theory is in whether it correctly judges whether a sequence of expressions belongs to $B(Inf)$ or $\widetilde{B}(Inf)$.

To use a type theory as a basis for formalization of mathematics, elements of $B(Inf)$ and $\widetilde{B}(Inf)$ must be provided with mathematical meaning. This is usually achieved by constructing a representation of the type theory into a *model* that is build out of objects and morphisms of a mathematically meaningful category such as the category of sets or homotopy types.

To construct such a representation two ingredients are required.

One ingredient is the model itself. For a moment consider a model to be any pair of sets $B, \widetilde{B}$ equipped with operations $ft : B \to B$, $\partial : \widetilde{B} \to B$ and a length function $l : B \to \mathbf{N}$ and partially defined operations corresponding to elements of Inf.

The second ingredient is a theorem saying that the syntactic model $B(Inf), \widetilde{B}(Inf)$ is an <u>initial</u> object in the category of all models.

From these two ingredients one obtains an interpretation of the type theory in $B, \widetilde{B}$. The construction of $B, \widetilde{B}$ is usually chosen such that the representation can be used to derive properties of the sets $B(Inf), \widetilde{B}(Inf)$ that are hard or impossible to see otherwise such as, for example, that $\widetilde{B}(Inf)$ does not contain any element of the form $\rhd o : \emptyset$, that is, that the type theory generated by Inf is "consistent".

The second ingredient in the construction that we have described is called the Initiallity Theorem. It is possible to normalize the form of inference rules such that the inference rules themselves will not depend on the choices one can make in the representation of (2) such as whether to use the syntax $ap(f, o)$, $ap(f, o, A)$ or $ap(f, o, A, x.B)$ for the application. Then the initiallity theorem is a theorem with two "arguments" - the set *Opp* of "normalized" inference rules and the set *Synt* of their syntactic representations.

There is a simple example that shows that not all pairs $(Inr, Synt)$ lead to initial syntactic models. Without going into detail for a moment of what elements of Inr are in general, consider the type theories (A) and (B) defined as follows.

Both (A) and (B) have five inference rules. The first four rules are the same for both type theories:

$$(I)\quad \frac{\Gamma\rhd}{\Gamma, x:T0\rhd} \qquad (II)\quad \frac{\Gamma, x:T\rhd}{\Gamma, x:R(T)\rhd} \qquad (III)\quad \frac{\Gamma\rhd o:T}{\Gamma\rhd r(o):R(T)} \qquad (IV)\quad \frac{\Gamma, x:T\rhd}{\Gamma, x:T\rhd x:T}$$

and the fifths rules are different. For the type theory (A) the rule is

$$(Va)\quad \frac{\Gamma\rhd o:T}{\Gamma\rhd o:R(T)}$$

and for the type theory (B) the rule is

$$(Vb)\quad \frac{\Gamma\rhd o:T}{\Gamma\rhd s(o):R(T)}$$

It is clear that the rules for both type theories represent the same partial operations on the sets $B, \widetilde{B}$, that is, they correspond to the same element of Inr. ?

However, in the first type theory there are three derivable "sequents" of the form

$$x:T0\rhd t:R(R(T0))$$

namely the ones with

$$t = x, r(x), r(r(x))$$

while in the second there are four, with

$$t = r(r(x)), r(s(x)), s(r(x)), s(s(x)).$$

Since any two initial models should be isomorphic both type theories can not satisfy the initiallity theorem. Further consideration suggests that the second one is more likely to be initial than the first one. However, we do not have an Initiallity Theorem that would be general enough even to obtain this very simple example as a particular case.

The inference rules for Ap and Π are special in many ways. We will have to consider two more examples to be able to explain why we need B- and C-systems that are the essential l-algebraic structures that much of the work that we propose to do is concerned with.

Firstly, consider the following three rules, which give the definition of the one element type in the Martin-Lof type theories:

$$\frac{\Gamma\rhd}{\Gamma, x:unit\rhd} \qquad \frac{\Gamma\rhd}{\Gamma\rhd tt:unit} \qquad \frac{\begin{array}{l}\Gamma\rhd u:unit\\ \Gamma, x:unit, y:P\rhd\\ \Gamma\rhd stt:P[tt/x]\end{array}}{\Gamma\rhd unit_{rect}(u, x.P, stt):P[u/x]} \tag{3}$$

They introduce three operations of which the third depend on the second and the second on the first. More importantly, the third operation depends on substitution. There are two substitution operations S and $\widetilde{S}$. In this case we need S that can be described by the inference rule

$$\frac{\Gamma\rhd o:T \qquad \Gamma, x:T, \Delta\rhd}{\Gamma, \Delta[o/x]\rhd}$$

where Δ is a continuation of the context $\Gamma, x : T$.

Secondly, consider the following four rules that introduce the natural numbers in the Martin-Lof type theories. The first three rules are

$$\frac{\Gamma \rhd}{\Gamma, x : nat \rhd} \qquad \frac{\Gamma \rhd}{\Gamma \rhd O : nat} \qquad \frac{\Gamma \rhd}{\Gamma, x : nat \rhd S(x) : nat} \tag{4}$$

and the fourth rule is

$$\frac{\begin{array}{l}\Gamma \rhd n : nat\\ \Gamma, x : nat, p : P \rhd\\ \Gamma \rhd s0 : P[0/x]\\ \Gamma, x : nat, p : P \rhd r : P[S(x)/x]\end{array}}{\Gamma \rhd nat_{rect}(n, x.P, s0, x.p.r)} \tag{5}$$

The condition that the fourth argument of the fourth rule must satisfy can not be expressed without another operation that is called *weakening*. There are two weakening operations T and $\widetilde{T}$. In this case the operation that is required is T that can be described by the inference rule

$$\frac{\Gamma, x : T \rhd \qquad \Gamma, \Delta \rhd}{\Gamma, x : T, \Delta \rhd}$$

One should also take into account the operation δ that is always added to the list of type theory rules and that is expressed by

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

These five operations - two substitution operations S and $\widetilde{S}$, two weakening operations T and $\widetilde{T}$ and operation δ, are called the structural operations. They are required to be included in the lists of operations of all more complex type theories to make these lists closed under the rule dependency.

These operations, as can be easily seen on simple examples, are not free in the syntactic models but satisfy in all of these models the same system of standard equations. As a consequence of this fact we can not hope to have the Initiallity Theorem to hold in the class of all Inf-structures when Inf includes the structural operations. Instead we must study it in the class of structures where the structural operations satisfy the standard equations.

Definition 3.1 *A pair of sets $(B, \widetilde{B})$ equipped with the length function $l : B \to \mathbf{N}$, operations $ft : B \to B$ and $\partial : \widetilde{B} \to B$ such that $l(\partial(r)) > 0$ for all $r \in \widetilde{B}$ and structural operations satisfying the system of standard equations is called a* B-system.

For a precise form of the domain of definition of structural operations and for the precise form of the standard equations see [?].

The theory of B-systems is conjecturally equivalent to the theory of C-systems that were introduced by John Cartmell under the name "contextual categories" in [2],[3]. Proving this equivalence is among the first goals of the proposed research.

This equivalence is important because there is a convenient way of constructing C-systems $CC(\mathcal{C}, p)$ from the so called "universe categories" (see [7]). Moreover there are convenient ways of equipping C-systems so produced with some of the systems of operations that are often present in modern type theories such as the dependent product system (see [10]), the dependent sum system (see [11]) and the system of rules for the Martin-Lof identity types (see [9]).

The categorical B-systems $(B(\mathcal{C},p), \widetilde{B}(\mathcal{C},p))$ obtained by application of the inverse equivalence to the C-systems $CC(\mathcal{C},p)$ have the property that $B(\mathcal{C},p)$ maps to objects of $\mathcal{C}$ and $\widetilde{B}(\mathcal{C},p)$ to morphisms of $\mathcal{C}$ satisfying a special condition.

4 What is an abstract system of inference rules

In this section we will outline a definition of the set Inr_0 of primary inference rules. Here the word "primary" means that we will consider the rules that only depend on the structural operations but not on any previously defined rules. How to define sequences of inference rules is another goal of the proposed research. Due to the complexity of the inductive constructions that need to be used we plan to develop a full UniMath formalization of the set of primary rules before moving to the systems of rules.

We fix a universe U and let $BSys$ denote the set of B-systems in U as well as the category whose set of objects is this set and the set of morphisms is the disjoint union the sets of obviously defined homomorphisms between B-systems (see [?] for details).

We will need the following construction.

Problem 4.1 *Let $BB = (B, \widetilde{B})$ be a B-system.*

1. *Let $\Delta \in B$. To construct a representation for the functor on $BSys$ of the form*

$$BB' \mapsto \{(f, X) \,|\, f : BB \to BB',\ X \in B',\ l(X) > l(\Delta),\ ft(X) = f(\Delta)\} \tag{6}$$

2. *Let $T \in B$ be such that $l(T) > 0$. To construct a representation for the functor on $BSys$ of the form*

$$BB' \mapsto \{(f, r) \,|\, f : BB \to BB',\ r \in \widetilde{B}',\ \partial(r) = f(T)\} \tag{7}$$

To provide constructions for these problems is another goal of the proposed research. One can show that such representations exist by using the equivalence between B-systems and C-systems, followed by the application of the theorem of Cartmell [2] asserting that the category of C-systems is equivalent to the category of generalized algebraic theories, followed by the application of the theorem of Richard Garner asserting that the category of generalized algebraic theories is monadic over the category of presheaves on the category of B-system carriers. However, we expect there to be a more direct approach to the construction of these representations.

We let $E(BB, \Delta)$ denote the object representing functor (6) and $\widetilde{E}(BB, T)$ the object representing the functor (7).

To describe the set of normalized primary inference rules we first need to consider the set of forms of premises of a primary rule. Here we may use the syntax

$$\Gamma \rhd A\ type$$

keeping in mind that for us it is just another way to write $\Gamma, x : A\rhd$.

There is the special premise $\Gamma\rhd$ that must be a part of the list of premises for any inference rule and that does not introduce any arguments of the corresponding operation. We consider it to be the "zeroth" premise.

It is intuitively clear that the first argument must always be introduced by a premise of the form

$$\Gamma \rhd A_1 \ type$$

Γ meta-variable ?

The second premise might be of one of two forms

$$\Gamma, \Delta_2(A_1) \rhd A_2 \ type \qquad \text{or} \qquad \Gamma, \Delta_2(A_1) \rhd a_2 : T_2(A_1)$$

Δ_2 ?

The third premise will be of one of the two forms

$$\Gamma, \Delta_3(A_1, A_2) \rhd A_3 \ type \qquad \text{or} \qquad \Gamma, \Delta_3(A_1, A_2) \rhd a_3 : T_3(A_1, A_2) \ type$$

in the first case and of one of the two forms

$$\Gamma, \Delta_3(A_1, a_2) \rhd A_3 \ type \qquad \text{or} \qquad \Gamma, \Delta_3(A_1, a_2) \rhd a_3 : T_3(A_1, a_2) \ type$$

in the second. We will provide an exact meaning for these expressions in a moment.

Define by induction on n pairs (P_n, tC_n) where P_n is a set and $tC_n : P_n \to BSys$ a function as follows:

1. $P_0 = stn(1)$ and $tC_0(0) = Pt$ where Pt is the B-system with $B = stn(1)$ and $\widetilde{B} = \emptyset$.
2. $P_1 = stn(1)$ and $tC_1(0) = \underline{E(Pt, 0)}$.
3. For $n > 0$ define P_{n+1} as $P_{n+1,0} \amalg P_{n+1,1}$ where

$$P_{n+1,0} = \coprod_{fX \in P_n} B(tC_n(fX)) \qquad P_{n+1,1} = \coprod_{fX \in P_n} B_{>0}(tC_n(fX))$$

   $B(tC_n(fX))$ is the B-set of the B-system $tC_n(fX)$ and $B_{>0}(tC_n(fX))$ is the subset of elements of length > 0 in $B(tC_n(fX))$.
   The function tC_{n+1} is given on $(fX, \Delta) \in P_{n+1,0}$ by

$$tC_{n+1}(fX, \Delta) = \underline{E(tC_n(fX), \Delta)}$$

   and on $(fX, T) \in P_{n+1,1}$ by

$$tC_{n+1}(fX, T) = \underline{\widetilde{E}(tC_n(fX), T)}$$

Definition 4.2 *The set of forms of premises of a primary inference rule is defined as the set* $\amalg_{n \in \mathbf{N}} P_n$.

There are inference rules of two kinds - the ones that introduce new types and the ones that introduce new elements. We will call them type inference rules and element inference rules respectively.

The form of a normalized type rule is determined by the form of its premises. Correspondingly, the set of forms of such rules is the set

$$\coprod_{n\in\mathbf{N}} P_n.$$

The form of a normalized element rule must specify in addition to its premises the type of the element it introduces. Correspondingly, the set of forms of such rules is

$$\coprod_{n\in\mathbf{N},X\in P_n} B_1(tC_n(X))$$

where $B_1(tC_n(X))$ is the set of elements of the B-set of $tC_n(X)$ of length 1.

The union of these two sets is the set of forms of normalized primary inference rules.

Consider the cases of rules with few arguments.

If the rule has 0 arguments then the set of possible forms of premises of such rules is $stn(1)$, i.e., there is only one form. The corresponding B-system $tC_0(X)$ is Pt. For Pt we have $B_1(Pt) = \emptyset$. Correspondingly there is one form of a rule that has 0 arguments and it is necessarily a type rule. Its schematic representation is

$$\frac{\Gamma\rhd}{\Gamma \rhd C\ type}$$

Such rules introduce type constants of the type system.

If the rule has 1 argument then the set of possible forms of premises of such a rule is $stn(1)$, i.e., there is again only one form. The corresponding B-system $tC_0(X)$ is $E(Pt,0)$. It is the B-system freely generated by one element in B of length 1. Using the conjectural equivalence of B-systems with C-systems one can see that $B(E(Pt,0)) = \mathbf{N}$ such that l is the identity function and in particular $B_1(E(Pt,0))$ is a set with one element. Therefore there are two forms of such rules, one for a type rule and one for an element rule. The schematic representation for these forms is

$$\frac{\begin{array}{c}\Gamma\rhd\\ \Gamma\rhd A\ type\end{array}}{\Gamma\rhd Op(A)\ type} \quad \text{and} \quad \frac{\begin{array}{c}\Gamma\rhd\\ \Gamma\rhd A\ type\end{array}}{\Gamma\rhd op(A) : A}$$

The next case are rules with two arguments. We will again have the "zeroth" premise $\Gamma\rhd$ and the premise that introduces the first argument of the same form as in the rule with one argument. The possible forms of the second premise are given by elements of $B(E(Pt,0))$ in the case of a type argument and $B_{>0}(E(Pt,0))$ in the case of an element argument. As we have already remarked $B(E(Pt,0)) = \mathbf{N}$. In the schematic notation this corresponds to premises of the form

$$\Gamma, x_1 : A, \ldots, x_n : A \rhd B\ type \tag{8}$$

and

$$\Gamma, x_1 : A, \ldots, x_n : A \rhd b : A \tag{9}$$

There is only one form of a type inference rule for each form of premises. For the element inference rule the forms are numbered by $B_1(E(E(Pt,0),n))$ in the case when the second premise is of the form (8) and $B_1(\widetilde{E}(E(Pt,0),n+1))$ when the second premise is of the form (9). Intuitively it seems clear that there is still only one possibility with the conclusion being of the form

$$\Gamma \rhd op(A, x.B) : A$$

when $n > 0$ and two possibilities

$$\Gamma \rhd op(A, B) : A \qquad \Gamma \rhd op(A, B) : B$$

when $n = 0$ but a real proof of it would require proving results about $B_1(E(E(Pt, 0), n))$ and $B_1(\widetilde{E}(E(Pt, 0), n + 1))$.

Here is an example of a primary inference rule with three arguments that shows that starting with three arguments the complexity increases considerably and analyzing the possibilities without having some theory becomes difficult.

$$\frac{\begin{array}{l}\Gamma \rhd \\ \Gamma \rhd A_1\ type \\ \Gamma, x : A_1 \rhd A_2\ type \\ \Gamma, x_1 : A_1, x_2 : A_1, x_3 : A_2[x_1/x] \rhd b_3 : A_1\end{array}}{\Gamma \rhd op(A_1, x_1.A_2, x_1.x_2.x_3.A_3) : A_1} \tag{10}$$

$$\text{cf } op : (A_1 : Set) \to (A_2 : A_1 \to Set) \to (b_3 : (x_1 : A_1) \to (x_2 : A_2) \to A_2 x_1 \to A_1) \to A_1$$

Next, let us show how the form of a normalized primary inference rule defines the form of a partial operation on any B-system $BB = (B, \widetilde{B})$.

Let $BB = (B, \widetilde{B})$ be a B-system. For $X, Y \in B$ let $X \geq Y$ if $l(X) \geq l(Y)$ and $ft^{l(X)-l(Y)}(X) = Y$. Let $X > Y$ if $X \geq Y$ and $l(X) > l(Y)$.

Let $\Gamma \in B$ and let $B(\Gamma)$ be the set of $A \in B$ such that $A \geq \Gamma$. Let $\widetilde{B}(\Gamma) = \{r \in \widetilde{B} \mid \partial(r) > \Gamma\}$. The B-system operations can be restricted to the subsets $B(\Gamma), \widetilde{B}(\Gamma)$ and setting $pt_\Gamma = \Gamma$ one obtains a new B-system that is denoted by $BB(\Gamma)$.

The inclusions $B(\Gamma) \to B$ and $\widetilde{B}(\Gamma) \to \widetilde{B}$ form "almost" a homomorphism of B-systems with the only axiom that does not hold being the compatibility with pt's. We denote these inclusions by rel_Γ.

Define a B-system arity as a sequence $(\epsilon_1, \ldots, \epsilon_n)$ where $\epsilon_i \in \{0, 1\}$. We let SAr denote the set of arities and by $la : Ar \to \mathbf{N}$ the function that takes $(\epsilon_1, \ldots, \epsilon_n)$ to n.

Given a B-system $BB = (B, \widetilde{B})$ and $e \in SAr$ define $B \times BB^e$ as the product $B \times BB_{\epsilon_1} \times \ldots \times BB_{\epsilon_{la(e)}}$ where $BB_0 = B$ and $BB_1 = \widetilde{B}$ and the product is defined by applying the binary product operation such that on $(i + 1)$-step with $i > 0$ the previously defined product is multiplied by $BB_{\epsilon_{i+1}}$ on the right. For the empty e one takes B.

For $X \in P_n$ define $Ar(X) \in SAr$ by induction on n as follows.

1. For $n = 0$, $Ar(X)$ is the empty sequence.
2. For $n = 1$, $Ar(X) = (0)$.
3. For the successor of $n > 0$, we set
   (a) for $(fX, \Delta) \in P_{n+1,0}$, $Ar(fX, \Delta) = (Ar(fX), 0)$,
   (b) for $(fX, T) \in P_{n+1,1}$, $Ar(fX, T) = (Ar(fX), 1)$.

For each n, each $X \in P_n$ and each B-system $BB = (B, \widetilde{B})$ define

$$X_{dom,0}(BB) = \coprod_{\Gamma \in B} Hom_{Bsys}(tC_n(X), BB(\Gamma))$$

Next, define a function

$$h = h(X, BB) : X_{dom,0}(BB) \to B \times BB^{Ar(X)}$$

by induction on n as follows.

1. For $n = 0$,
$$h(\Gamma, f) = \Gamma$$
Note that in this case $Hom_{Bsys}(tC_n(X), BB(\Gamma))$ is a set with one element,

2. For $n = 1$, h maps the element
$$(\Gamma, (f, A)) \in \coprod_{\Gamma \in B} Hom_{Bsys}(E(Pt, 0), BB(\Gamma)) =$$
$$\coprod_{\Gamma \in B} Hom_{Bsys}(Pt, BB(\Gamma)) \times \{A \in B(\Gamma) \mid l_\Gamma(A) \geq 1,\ ft(A) = \Gamma\}$$
to
$$h(\Gamma, (f, A)) = (\Gamma, rel_\Gamma(A))$$

3. For the successor of $n > 0$ we set:

   (a) for $X = (fX, \Delta) \in P_{n+1,0}$, h maps the element
$$(\Gamma, (f, A)) \in \coprod_{\Gamma \in B} Hom_{Bsys}(E(tC_n(fX), \Delta), BB(\Gamma)) =$$
$$\coprod_{\Gamma \in B} \{(f, A) \mid f \in Hom_{Bsys}(tC_n(fX), BB(\Gamma)),\ A \in B(\Gamma),\ l(A) > l(f(\Delta)),\ ft(A) = f(\Delta)\}$$
   to
$$h(\Gamma, (f, A)) = (h(\Gamma, f), rel_\Gamma(A))$$

   (b) for $X = (fX, T) \in P_{n+1,1}$, h maps the element
$$(\Gamma, (f, r)) \in \coprod_{\Gamma \in B} Hom_{Bsys}(\widetilde{E}(tC_n(fX), Y), BB(\Gamma)) =$$
$$\coprod_{\Gamma \in B} \{(f, r) \mid f \in Hom_{Bsys}(tC_n(fX), BB(\Gamma)),\ r \in \widetilde{B}(\Gamma),\ \partial(r) = f(T)\}$$
   to
$$h(\Gamma(f, r)) = (h(\Gamma, f), rel_\Gamma(r))$$

We define, for $X \in P_n$ the domain of definition of the operations with the form of premises X as

$$X_{dom}(BB) = Im(h(X, BB)) \subset B \times BB^{Ar(X)}$$

For $A \in X_{dom}$ we define $\Gamma_A \in B$ as the projection of A to the first B in the product. One can easily see that h is an injection and therefore we can also define

$$f_A \in Hom_{Bsys}(tC_n(X), BB(\Gamma_A)) \text{ such that } f_{h(\Gamma,f)} = f$$

If R is the form of a normalized rule that introduces a type we have $R = (n, X)$ for $n \in \mathbf{N}$ and $X \in P_n$. Let $R_{dom} = X_{dom}$. We define the form of the corresponding operation to be a function

$$Op(R) : R_{dom}(BB) \to B$$

and the relation that it satisfies as

$$ft(Op(R)(A)) = \Gamma_A.$$

If R is a normalized rule that introduces an element we have $R = (n, X, T)$ for $n \in \mathbf{N}$, $X \in P_n$ and $T \in B_1(Ct_n(X))$. Let $R_{dom} = X_{dom}$. We define the form of the corresponding operation to be a function

$$Op(R) : R_{dom}(BB) \to \widetilde{B} \tag{11}$$

and the relation that it satisfies as

$$\partial(Op(R)(A)) = rel_{\Gamma_A}(f_A(T)). \tag{12}$$

Operations that correspond to inference rules on the syntactic B-systems satisfy, in addition to the relation (11) or (12) two equations that express that they commute with T and S in the case of type operation or $\widetilde{T}$ and $\widetilde{S}$ in the case of an element operation.

A form of sequences of inference rules of length n is defined by induction on n. For $n = 1$ one considers the primary inference rules. For the successor of $n > 0$ one considers a sequence S of length n together with an operation of the form defined in the same way as above but considering instead of $E(X, \Delta)$ and $\widetilde{E}(X, T)$ the B-systems that represent similar functors in the category of B-systems equipped with operations of forms corresponding to S and satisfying the relations outlined above.

It should be more or less clear from our discussion and example (10) how to define the binding signature $\Sigma(Op)$ corresponding to the form of a primary inference rule and to the form of a sequence of inference rules.

There are operations corresponding to the sequence S on the syntactic B-system $BB(R_{\Sigma(S)}, R_{\Sigma(S)})$ corresponding to the binding signature corresponding to S and the initiallity conjecture says that the the smallest B-subsystem of $BB(R_{\Sigma(S)}, R_{\Sigma(S)})$ that is closed under these operations is initial among B-systems with such sequences of operations and satisfying the triples of relations explained above.

References

[1] Benedikt Ahrens. Initiality for typed syntax and semantics. In *Logic, language, information and computation*, volume 7456 of *Lecture Notes in Comput. Sci.*, pages 127–141. Springer, Heidelberg, 2012.

[2] John Cartmell. Generalised algebraic theories and contextual categories. *Ph.D. Thesis, Oxford University*, 1978. `https://uf-ias-2012.wikispaces.com/Semantics+of+type+theory`.

[3] John Cartmell. Generalised algebraic theories and contextual categories. *Ann. Pure Appl. Logic*, 32(3):209–243, 1986.

[4] Alonzo Church. A formulation of the simple theory of types. *J. Symbolic Logic*, 5:56–68, 1940.

[5] Marcelo Fiore, Gordon Plotkin, and Daniele Turi. Abstract syntax and variable binding (extended abstract). In *14th Symposium on Logic in Computer Science (Trento, 1999)*, pages 193–202. IEEE Computer Soc., Los Alamitos, CA, 1999.

[6] André Hirschowitz and Marco Maggesi. Modules over monads and initial semantics. *Inform. and Comput.*, 208(5):545–564, 2010.

[7] Vladimir Voevodsky. A C-system defined by a universe category. *Theory Appl. Categ.*, 30:No. 37, 1181–1215, 2015.

[8] Vladimir Voevodsky. An experimental library of formalized mathematics based on the univalent foundations. *Math. Structures Comput. Sci.*, 25(5):1278–1294, 2015.

[9] Vladimir Voevodsky. Martin-L'of identity types in the C-systems defined by a universe category. *arXiv 1505.06446, under review in Publication IHES*, pages 1–51, 2015.

[10] Vladimir Voevodsky. Products of families of types in the C-systems defined by a universe category. *arXiv 1503.07072, submitted*, pages 1–30, 2015.

[11] Vladimir Voevodsky. Interpretation of the rules for dependent sums on the C-system defined by a universe category. *In preparation*, pages 1–38, 2016.

[12] Vladimir Voevodsky, Benedikt Ahrens, Daniel Grayson, et al. *UniMath*: Univalent Mathematics. Available at `https://github.com/UniMath`.