Notes on homotopy λ-calculus

Vladimir Voevodsky

Started Jan. 18, Feb. 11, 2006

Contents

1 An introduction

In this paper we suggest a new approach to the foundations of mathematics. In fact the adjective "new" in the previous sentence may be superfluous since one can argue that pure mathematics as it is practiced today has no foundations. We have come a long way from where we used to be at the beginning of the century when the thesis that mathematics is that which can be formalized in the framework of the Zermelo-Fraenkel theory became generally accepted. Contemporary constructions and proofs can not be translated in any sensible way into the Zermelo-Fraenkel theory. One can (may be!) translate the categorical definition of group cohomology into the ZF-theory using the Grothendieck's idea of universes but it is hardly fair to call an approach which requires one to believe in strongly unreachable cardinals in order to interpret a simple algebraic construction sensible.

Because of the amazing intuitiveness of the basic concepts of category theory this foundational insufficiency is not (yet) a serious problem in our everyday work. In my experience it becomes noticeable only at the level of 2-categories which may be the reason why 2-categorical arguments are not very common in mathematics. The real need of formalized foundations arises when one attempts to use a computer to verify a proof. It is a fact of life that proofs in contemporary mathematics are getting too long, complicated and numerous to be rigorously checked by the community. Moreover it is precisely the most technical parts of the proofs where the probability of a mistake is the highest which are most likely to be skipped in the verification process. Since the ability to build on layers and layers of earlier results is probably *the* most important feature which underlies the success of mathematics as an enterprise this is a very serious problem and without finding its solution mathematics can not move forward. The idea to use computers in proof verification is an old and obvious one. However, despite the fact that the first generation proof verification systems such as Automath or Mizar are more than thirty years old now none of the existing proof verification systems are practically usable in the context of pure mathematics.

Since the computer science is getting very good at working with complex formal systems I would conjecture that the real reason for this situation is not that it is very difficult to create a good proof verification system but that we do not have formalized foundations which we could pass to the computer scientists and say - "please create the software to verify proofs in this system". Mizar does a pretty good job of implementing the ZF-theory but it is nearly impossible to prove even the most basic algebraic theorems with Mizar because it requires one to specify every single isomorphism on a crazily detailed level. Clearly it is not a problem of Mizar but a problem of the ZF-theory.

Attempts to develop alternatives to the ZF-theory and to the first order theories in general have been made since the beginning of the century (at least). From what I know about the history of mathematics the two such attempts which are most relevant here are the type theory of Russel and Whitehead and Church's λ-calculus. Type theory and λ-calculus were later unified in typed λ-calculus. A key development (totally unnoticed by the mathematical community) occurred in the 70-ies when the typed λ-calculus was enriched by the concept of dependent types. Most types in mathematics are dependent on members of other types e.g the class (or type) of algebras over a ring R is a type dependent on R which is a member of the type of rings. However, until fairly recently there were no formal languages supporting dependent types.

Today the dependent type theory is a well developed subject but it belongs almost entirely to the realm of computer scientists and in my experience there are very few mathematicians who have a slightest idea of what dependent type systems are. One of these mathematicians is Makkai - the key insight that dependent type systems are exactly what we need in order to formalize categorical thinking is probably due to him (see []). His papers played the key role in convincing me that it is indeed possible to build much better foundations of mathematics than the ones provided by the ZF-theory.

One of the reasons mathematicians have not been more involved in the development of the dependent type systems is that these systems for most part lack clear semantics. It does not mean there is no semantics at all. The abstract categorical semantics for type systems was developed by Jacobs [**?**] and his book was of great help to me. What was missing was semantics with values in an intuitively accessible category. The concepts of first order logic are easy to explain precisely because there is a straightforward notion of a (set-theoretic) model of a first order theory. The ideas of simple type theory both in the intuitionist and in the Boolean versions are reasonably easy to explain because we may use models in toposes of sheaves on topological spaces. For the homotopy λ-calculus such "standard" models take values in the homotopy category. In the same way as a sort in a first order theory is thought of as a set a type in the homotopy λ-calculus can be thought of as a homotopy type.

Let me try to explain what homotopy category has to do with the foundations of mathematics. First of all I want to suggest a modification of the usual thesis stating that categories are higher level analogs of sets. We will take a slightly different position. We will consider groupoids to be the next level analogs of sets. Consider the following hierarchy of (higher) groupoids (we ignore the large versus small distinction for now):

$$\mathcal{L}_{-1} = pt \quad \mathcal{L}_0 = \{0,1\} \quad \mathcal{L}_1 = \{\text{sets and isomorphisms}\} \quad \mathcal{L}_2 = \{\text{groupoids and equivalences}\} \quad \textit{etc.}$$

Categories may be considered as groupoids $\mathcal{C}$ with an additional structure - a pairing

$$\mathcal{C} \times \mathcal{C} \to \mathcal{L}_1$$

which sends X, Y to $Mor(X, Y)$ (plus some higher level structures which I will ignore at the moment). Note that this is indeed a functor between *groupoids* even though it would not be a functor between categories unless we replaced the first $\mathcal{C}$ with $\mathcal{C}^{op}$. Hence a previous level analog of a category is a set X together with a map

$$X \times X \to \mathcal{L}_0$$

i.e. a set with a relation. The existence of compositions and units for categories corresponds to the reflexivity and transitivity of this relation. We conclude that a category is the next level analog of a partially ordered set.

The key argument for this modification of the basic thesis is the following observation - *not all interesting constructions on sets are functorial with respect to maps but they are all functorial with respect to isomorphisms.* Similarly, not all interesting constructions on groupoids or categories are natural with respect to functors but they all are (by definition!) natural with respect to equivalences.

Once the modification of the basic thesis is accepted the connection between foundations and the homotopy theory becomes obvious since we know that n-groupoids are the same as homotopy n-types. We will see below (in **??**) that the n-types corresponding to the groupoids similar to $\mathcal{L}_n$ have a natural homotopy theoretic description in the elementary terms of univalent fibrations. They show up as models of special types (in special contexts) of the homotopy λ-calculus. Similarly, for a type of mathematical structures (e.g. groups) one can define a type in the homotopy λ-calculus whose models are (the homotopy types of) groupoids or higher groupoids of the corresponding structures and their equivalences. Since all the constructions of homotopy λ-calculus correspond by design to homotopy invariant constructions on models it is impossible to make statements which are not invariant under equivalences. This provides a build-in support in our system for the important principle which says that two isomorphic or otherwise appropriately equivalent objects are interchangeable.

Need to mention: Goedel Theorem, Giraurd(sp?) Theorem.

Mention: Goedel theorem implies that there does not exist a recursive well-pointed topos with a natural numbers object (other than the trivial one).

Mention: In our approach *every* type "is" or at least can be though of as a "subtype" of an equivalence type $eq_{\mathbf{R}}(\mathbf{r}_1, \mathbf{r}_2)$ for appropriate $\mathbf{R}$ and $\mathbf{r}_i$.

Mention(?):Firmer foundations allow for a higher level of abstraction.

Mention(?):Some parts of Section **??** give us hope that the formal language of the homotopy λ-calculus can be integrated well with the usual mathematical discourse.

Mention in the next section: Intuitionist models in Fib/B and Boolean but multi-valued ones in Fib/B where B is $K(\pi, 0)$.

A type system is said to be simple (or pure) if terms and types are distinct and there is no way to produce types from terms. If there is a way to produce types from terms i.e. there are type constructors which take terms of previously defined types as arguments then one says that the type system allows dependent types[1]. If types and terms are mixed i.e. terms of some types are themselves types then the type system is called polymorphic. The homotopy λ-calculus is a dependent type system with (strong) dependent sums and dependent products. There is no polymorphism in the usual sense i.e. terms and types never get mixed up but there are universe constructors which provide a replacement for the traditional polymorphism.

1 Homotopy theory and foundations of mathematics

In this section I will describe several constructions of homotopy theoretic nature which I hope explain the connection between the homotopy theory and the foundations. The foundations used in this section itself are the intuitive ones.

Let us start with the following observation. Fix a large universe $\mathcal{U}$ and let $H = H(\mathcal{U})$ be the homotopy category of spaces (or simplicial sets) in this universe. We will not mention this large universe explicitly anymore. Let U be a universe which is small relative to $\mathcal{U}$.

Let us show that to any type of mathematical structures Γ we can assign an object $\Gamma(U)$ in H such that $\pi_0(\Gamma(U))$ is the set of equivalence classes of structures of type Γ in U. Since we do not have a formal definition of a structure yet we will do it by examples:

[1]There are almost always term constructors which make terms dependent on types e.g. the identity term of the function type $T \to T$.

1. For types Γ such as sets, groups, topological spaces or Grothendieck schemes we define $\Gamma(U)$ as the nerve of the groupoid whose objects are structures of type Γ in U and morphisms are isomorphisms of such structures. We will call types of structures of this kind i.e. such that the notion of an isomorphism is defined, types of level 1.
2. For structures Γ such as groupoids, categories, monoidal categories, etc. we define $\Gamma(U)$ as the nerve of the 2-groupoid whose objects are structures of this type in U, morphisms are equivalences and 2-morphisms are natural isomorphisms between equivalences. We call types of this kind i.e. such that the notion of isomorphism between equivalences is defined, types of level 2.
3. For higher level types of structures such as 2-groupoids, 2-categories etc. we proceed in the same way.

Let now Γ_1, Γ_2 be two types of structures and F be a construction which assigns to a structure of type Γ_1 in U a structure of type Γ_2 in U. For example we can take $\Gamma_1 = \mathit{Pointed\_Sets}$, $\Gamma_2 = \mathit{Sets}$ and F to be the construction which forgets the distinguished point. Or we can take $\Gamma_1 = \mathit{Rings}$ and $\Gamma_2 = \mathit{Categories}$ and F to be the construction which assigns to a ring R the category of schemes of finite type over R. Since any construction must respect equivalences it defines a map $F : \Gamma_1(U) \to \Gamma_2(U)$ which is well defined up to a homotopy. Moreover, any such map will correspond to an equivalence preserving construction producing a structure of type Γ_2 in U from a structure of type Γ_1 in U.

If P is a condition on structures of type Γ then we have a subspace $(\Gamma, P)(U)$ in $\Gamma(U)$ which consists of connected components of $\Gamma(U)$ corresponding to the equivalence classes of structures satisfying this condition. Of course this subspace is itself of the form $\Gamma'(U)$ where $\Gamma' = (\Gamma, P)$ is the type of structures of type Γ satisfying in addition the condition P. Using this correspondence one can express any mathematical theorem as the condition that $\Gamma(U)$ for an appropriately chosen Γ is non-empty. The proof is then a construction which produces a point in this $\Gamma(U)$[2].

We claim that this picture can be used as a basis for the formalization of mathematics.

Let $\Omega = \Omega(U)$ be the object of H which corresponds to ∞-groupoids in U. It contains subobjects $\Omega_{\leq 1} \subset \Omega_{\leq 2} \subset \ldots$ where $\Omega_{\leq 1} = \mathit{Sets}(U)$, $\Omega_{\leq 2} = \mathit{Groupoids}(U)$ etc. One observes easily that each $\Omega_{\leq n}$ is a union of connected components of Ω i.e. these are really subobjects in the homotopy-theoretic sense. Let further $\tilde{\Omega} = \tilde{\Omega}(U)$ be the object corresponding to pointed ∞-groupoids i.e. pairs of an ∞-groupoid and an object in it and $p : \tilde{\Omega} \to \Omega$ be the forgetting map. Our claim is based on the following two facts:

1. Maps of the form $\tilde{\Omega} \to \Omega$ corresponding to universes can be characterized in elementary terms.
2. For a given universe U with the "classifying" map $p : \tilde{\Omega} \to \Omega$ and a given type of structures Γ one can construct $\Gamma(U)$ explicitly from $p : \tilde{\Omega} \to \Omega$.

We will show how to do the first of these two things in Definition 0.14 and will illustrate the second in Examples **??**-**??**.

[2]A construction does not have to be constructive in the sense of logic. It can be based on the argument that if a complement to something in a non-empty something is empty then the original something has a point.

To turn this approach into a real formalization we will define a formal language where one can can build universes i.e. triples $U = (\tilde{\Omega}, \Omega, p)$ satisfying appropriate conditions and where one can further build for any U the types $\Gamma(U)$ corresponding to all the usual types of mathematical structures and maps $\Gamma_1(U) \to \Gamma_2(U)$ corresponding to all usual constructions. Since any mathematical theorem can be expressed as the condition that the space $\Gamma(U)$ for an appropriately chosen Γ is non-empty one can use this language to prove theorems by constructing terms of the appropriate types. The key difficulty one has to overcome is to define the language in such a way that it describes sufficiently many constructions and at the same time all the constructions it describes are homotopy-invariant. Fortunately such languages can be found in the class of languages called dependent type systems. While there is no ready-made dependent type system satisfying all the requirements imposed by our goal we will be able to define such a system as an extension of a known one.

The key observation which allows one to construct $\Gamma(U)$ in terms of $p : \tilde{\Omega} \to \Omega$ is as follows. In general, if $F : \Gamma_1(U) \to \Gamma_2(U)$ is a construction and $X \in \Gamma_2(U)$ then the homotopy fiber $F^{-1}(X)$ of F over X is the space of pairs $(Y \in \Gamma_1(U), \phi : F(Y) \cong X)$. For example, if F is the forgetting construction from groups to sets then $F^{-1}([X])$ is the set of all group structures on X. Applying this observation to the universe map $p : \tilde{\Omega} \to \Omega$ one concludes that the homotopy fiber of p over the point $[\mathcal{G}]$ corresponding to a groupoid $\mathcal{G}$ is the homotopy type associated with $\mathcal{G}$ itself. For example if X is a set in U and $[X]$ is the corresponding point of $\Omega_{\leq 1}$ then $p^{-1}([X])$ is equivalent to X.

Note that we could not do the same in the category of $\mathcal{U}$-sets instead of H. We can assign to Γ the set $\gamma(U)$ of equivalence classes of structures of type Γ in U but then the fiber of the map corresponding to, say, a forgetting construction is not anymore the set of structures which we are forgetting. E.g. the fiber of the map corresponding to the *pointed sets* $\to$ *sets* construction over $[X]$ will be one point if $X \neq \emptyset$ and $\emptyset$ otherwise.

The most important concept which we will introduce is the concept of a univalent map (or univalent fibration). Let us start with the following standard definition.

Definition 0.1 *Let $f : Y \to X$ and $g : Z \to X$ be two continuous maps and $u : Y \to Z$ be a map over X. Then u is called a fiber-wise homotopy equivalence if for any $x \in X$ the corresponding map between the homotopy fibers of f and g is a homotopy equivalence.*

For two maps $f : Y \to X$ and $g : Z \to X$ let $Eq_X(f, g)$ be the space of fiber-wise homotopy equivalences from Y to Z over X. It is fibered over X such that the fiber of $Eq_X(f, g) \to X$ over $x \in X$ is (homotopy equivalent to) the space of homotopy equivalences between the homotopy fibers $f^{-1}(x)$ and $g^{-1}(x)$.

For a map $f : Y \to X$ consider the maps $f \times Id : Y \times X \to X \times X$ and $Id \times f : X \times Y \to X \times X$. Let further

$$E(f) = Eq_{X\times X}(f \times Id, Id \times f).$$

Then $E(f)$ is fibered over $X \times X$ and its fiber over (x, x') is the space of homotopy equivalences between the homotopy fibers of f over x and x'. In particular, $E(f) \to X \times X$ has a canonical section over the diagonal $X \to E(f)$ corresponding to the identity.

Definition 0.2 *[**univ**] A map $p : Y \to X$ is called univalent if the map $X \to E(p)$ is a fiber-wise homotopy equivalence over $X \times X$.*

The homotopy fiber of the diagonal $X \to X \times X$ over (x, x') is the space $P(X; x, x')$ of paths from x to x' in X. Hence a fibration $p : Y \to X$ is univalent if and only if for any x, x' the space of homotopy equivalences between the fibers $p^{-1}(x)$ and $p^{-1}(x')$ is naturally equivalent to the space of paths $P(X; x, x')$.

We say that a map $f : X' \to X$ is a level 0 map if it is fiber-wise equivalent to the embedding of a set of connected components of X to X. Here are some examples of univalent maps:

1. There are only three univalent maps of level 0. They are $\emptyset \to \emptyset$, $\emptyset \to pt$ and $pt \to pt \coprod pt$. Of these three the last one is the universal one since the other two are obtained from it by pull-back.
2. For $n > 0$ the map $BS_{n-1} \to BS_n$ where $S_{n-1} \to S_n$ is the standard embedding of symmetric groups, is univalent. The homotopy fiber of this map is the set with n elements.
3. For $n > 2$ the map $BS_n \to pt$ is univalent. In general, for a group G the map $BG \to pt$ is univalent if its center is trivial and its group of outer automorphisms is trivial.
4. For $n \geq 0$ the inclusion of the distinguished point $pt \to K(\mathbf{Z}/2, n)$ is univalent. For $n = 0$ one gets the map $pt \to pt \coprod pt$ from the first example and for $n = 1$ one gets the map $BS_1 \to BS_2$ of the second example. I do not know at the moment any other examples of univalent maps starting at the point.

Let us state some elementary properties of univalent maps.

Lemma 0.3 *[**unpr1**] Consider a (homotopy) cartesian square*

$$\begin{tikzcd} Y' \arrow[r] \arrow[d, "p'"'] & Y \arrow[d, "p"] \\ X' \arrow[r, "f"] & X \end{tikzcd}$$

such that p is univalent. Then p' is univalent if and only if f is a map of level 0.

Proposition 0.4 *[**class**] If for a given univalent $p : Y \to X$ and a given $p' : Y \to Y$ there exists a (homotopy) cartesian square of the form*

$$\begin{tikzcd} Y' \arrow[r] \arrow[d, "p'"'] & Y \arrow[d, "p"] \\ X' \arrow[r, "f"] & X \end{tikzcd}$$

then such a square is unique up to an equivalence.

Definition 0.5 *[classdef] Let $p : Y \to X$ be a univalent map and $p' : Y' \to X'$ a map. We say that p' is classifiable by p if a cartesian square as in Corollary 0.4 exists.*

Proposition 0.6 *[induced] For any map $f : Y \to X$ there exists a unique (up to an equivalence) homotopy cartesian square*

$$\begin{tikzcd} Y \arrow[r] \arrow[d, "f"'] & \widetilde{Un}(f) \arrow[d, "p"] \\ X \arrow[r, "g"] & Un(f) \end{tikzcd}$$

such that p is univalent and g is surjective on π_0.

Proof: Let us sketch the existence part when $X' = pt$. In this case let $M = Eq(Y', Y')$ be the topological monoid of homotopy auto-equivalences of Y'. Then one takes X to be the classifying space BM of M and $p : Y \to X$ to be the fibration defined by the action of M on Y'.

For $f : Y \to pt$ we will write $\widetilde{Un}(Y) \to Un(Y)$ instead of $\widetilde{Un}(f) \to Un(f)$.

The role of univalent maps in foundations is based in the following theorem.

Theorem 0.7 *[univ1] Given a set of isomorphism classes A in H there exists a unique univalent map $p : \tilde{\Omega}(A) \to \Omega(A)$ such that $X \to pt$ is classifiable by p iff $X \in A$. This correspondence establishes a bijection between isomorphism classes of univalent maps in H and sets of isomorphism classes of objects in H.*

Proof: To prove the existence part let us choose a representative X_a for each isomorphism class $a \in A$. Then set $\Omega(A) = \coprod_{a \in A} Un(X_a)$ and $\tilde{\Omega}(A) = \coprod_{a \in A}(\widetilde{Un}(X_a))$. If there are two such maps $p : \tilde{\Omega}(A) \to \Omega(A)$ and $p' : \tilde{\Omega}'(A) \to \Omega'(A)$ consider $\widetilde{Un}(p \coprod p') \to Un(p \coprod p')$. By Lemma 0.3 the map $\Omega(A) \to Un(p \coprod p')$ is of level 0. On the other hand one can easily see that it is surjective on π_0. Therefore, it is an equivalence. The same holds for $\Omega'(A) \to Un(p \coprod p')$.

Note that if $A \subset A'$ where A, A' are sets of homotopy types then one has a cartesian square

$$\begin{tikzcd} \tilde{U}(A) \arrow[r] \arrow[d] & \tilde{U}(A') \arrow[d] \\ U(A) \arrow[r, "i"] & U(A') \end{tikzcd}$$

where i is a map of level 0.

Define the level of a homotopy type inductively as follows:

1. X is of level -1 iff X is contractible,

2. X is of level $n \geq 0$ iff for any $x, x' \in X$ the paths space $P(X; x, x')$ is of level $n-1$.

There are only two types of level 0 namely $\emptyset$ and pt. A type of level 1 is a set i.e. a space of type $K(\pi, 0)$. More generally one has the following obvious lemma.

Lemma 0.8 *[**levelsob**] A space X is of level $n \geq 1$ iff for all $x \in X$ one has $\pi_i(X, x) = pt$ for $i \geq n$.*

For a type X and $n \geq -1$ let $\Pi_n(X)$ be the i-th stage of its Postnikoff tower which we define for $n = -1$ as $\emptyset$ if $X = \emptyset$ and pt if $X \neq \emptyset$. For any X the space $\Pi_n(X)$ is of level $n+1$ and the functor $X \mapsto \Pi_n(X)$ is the left adjoint to the inclusion of the types of level $n+1$ to all types. We will use the following description of Π_{-1} and Π_0:

Lemma 0.9 *[**pi01**] For any X one has:*

1. $\Pi_{-1}(X) = Hom(Hom(X, \emptyset), \emptyset)$
2. *$\Pi_0(X)$ is the image of the natural map $X \to Hom(Hom(X, \{0,1\}), \{0,1\})$.*

Lemma 0.10 *[**levlev**] For a space X of level n the space $Un(X)$ is of level $n+1$.*

Let us now describe how the condition that a set A is closed under certain operations on types can be described in terms of the properties of the corresponding univalent map $\tilde{\Omega}(A) \to \Omega(A)$. We will consider the following closeness conditions:

Cl_{sum} - asserts that for a fibration $f : Y \to X$ such that X is in A and all fibers of f are in A one has $Y \in A$,

Cl_{prod} - asserts that for $f : Y \to X$ as above the space of sections of f is in A,

Cl_{eq} - asserts that for X in A and $x, x' \in X$ the paths space $P(X; x, x')$ is in A,

Cl_{un} - asserts that for X in A one has $Un(X) \in A$.

Given a fibration $p : Y \to X$ define a space $Fam(p)$ as the space whose points are families of fibers of p parametrized by a fiber of p i.e. $Fam(p)$ is the space of pairs $\{x \in X, f : p^{-1}(x) \to X\}$. More formally, one may define $Fam(p)$ as the space of maps from Y to $X \times X$ over X. It is fibered over X with the fiber over $x \in X$ being the space of (continuous) maps from $p^{-1}(X)$ to X. Consider the following two fibrations over $Fam(p)$:

1. $Sum(p) \to Fam(p)$ whose fiber over (x, f) is $p^{-1}(x) \times_f Y$,

2. $Prod(p) \to Fam(p)$ whose fiber over (x, f) is the space of sections of the projection $p^{-1}(x) \times_f Y \to p^{-1}(x)$.

Proposition 0.11 *[**close1**] Let A be a set of homotopy types and $p : \tilde{\Omega}(A) \to \Omega(A)$ be the corresponding univalent map. Then following conditions are equivalent:*

1. *A satisfies Cl_{sum} (resp. Cl_{prod}),*
2. *the fibration $Sum(p) \to Fam(p)$ (resp. $Prod(p) \to Fam(p)$) is classifiable by p.*

Proposition 0.12 *[**close2**] Let A be a set of homotopy types and $p : \tilde{\Omega}(A) \to \Omega(A)$ be the corresponding univalent map. Then following conditions are equivalent:*

1. *A satisfies Cl_{eq},*
2. *the diagonal map $\tilde{\Omega}(A) \to \tilde{\Omega}(A) \times_{\Omega(A)} \tilde{\Omega}(A)$ is classifiable by p.*

To formulate the Cl_{un} condition define first the following construction. For a map $f : Y \to X$ let $Im_0(f)$ be the set of connected components of X which contain images of points of Y. We can also define $Im_0(f) \to X$ as the universal map of level 0 through which f factors. The space $Im_0(f)$ can be described as $Hom_X(Hom_X(Y, \emptyset), \emptyset)$ where Hom_X denotes the space of maps over X. Let $[X] = Im_0(\Delta_X)$ where $\Delta_X : X \to X \times X$ is the diagonal. One observes easily that if $X = \coprod X_a$ where X_a are connected then $[X] = \coprod X_a^2$.

Proposition 0.13 *[**close3**] Let A be a set of homotopy types and $p : \tilde{\Omega}(A) \to \Omega(A)$ be the corresponding univalent map. Then following conditions are equivalent:*

1. *A satisfies Cl_{un},*
2. *each connected component of $\Omega(A)$ is classifiable by p,*
3. *the projection $[\Omega(A)] \to \Omega(A)$ is classifiable by p.*

We can now give an elementary definition of a universe map and therefore of a universe:

Definition 0.14 *[**univmap**] A map of homotopy types $p : \tilde{\Omega} \to \Omega$ is called a universe map if it satisfies the following conditions:*

1. *p is univalent,*
2. *the fibration $Sum(p) \to Fam(p)$ is classifiable by p,*
3. *the fibration $Prod(p) \to Fam(p)$ is classifiable by p,*

4. *the diagonal map $\tilde{\Omega} \to \tilde{\Omega} \times_{\Omega} \tilde{\Omega}$ is classifiable by p,*

5. *the projection $[\Omega] \to \Omega$ is classifiable by p.*

We say that a set U of homotopy types is a (closed) universe if it is closed under the conditions Cl_{sum}, Cl_{prod}, Cl_{eq} and Cl_{un} or, equivalently, if $p : \tilde{\Omega}(U) \to \Omega(U)$ is a universe map in the sense of Definition 0.14. The following list contains some elementary properties of such closed universes:

1. if $X \in \bar{A}$ and $x \in X$ then the loop spaces $\Omega^n(X, x)$ are in $\bar{A}$,
2. if $X, Y \in \bar{A}$ then $Hom(X, Y) \in \bar{A}$ and $X \times Y \in \bar{A}$,
3. if there exists $X \in \bar{A}$ such that X is not contractible then $\emptyset \in \bar{A}$
4. if $\emptyset \in \bar{A}$, $X \in \bar{A}$ and $f : Y \to X$ is of level 0 then $Y \in \bar{A}$, in particular if X is a set then all subsets of X are in $\bar{A}$,
5. if $\emptyset \in \bar{A}$ and there exists $X \in \bar{A}$ such that $X \neq \emptyset, pt$ then $\{0, 1\} \in \bar{A}$.

These properties imply in particular that there are exactly three closed universes which do not contain $\{0, 1\}$ namely $\emptyset$, $\{pt\}$ and $\{\emptyset, pt\}$. We assume in addition to conditions listed above that $\{0, 1\} \in U$. We have:

1. if $X \in \bar{A}$ is a set then the set PX of all subsets of X is in $\bar{A}$,
2. if $X \in \bar{A}$ then $\Pi_0(X) = \pi_0(X) \in \bar{A}$.

We will formulate other less trivial properties as lemmas.

Lemma 0.15 *[**sur**] Let $f : Y \to X$ is a surjection of sets such that $Y \in \bar{A}$. Then $X \in \bar{A}$.*

Proof: Assuming the axiom of choice we could take a section of f and conclude that X is in $\bar{A}$ as a subset of Y. Without assuming the axiom of choice proceed as follows. Consider $PX \to PY$ where as before P denotes the set-of-subsets functor. This is a mono. On the other hand X is a subset in PX. Hence, X is a subset in PY and therefore is in $\bar{A}$.

Lemma 0.16 *[**fibers**] Let $f : Y \to X$ be a fibration such that $X, Y \in \bar{A}$. Then for any point $x \in X$ the homotopy fiber $f^{-1}(x)$ is in $\bar{A}$.*

Proof: Consider the map $i : f^{-1}(x) \to Y$. The homotopy fiber of this map over $y \in Y$ is the space $P(X; f(y), x)$. Since $\bar{A}$ is closed under Cl_{sum} and Cl_{eq} we conclude that $f^{-1}(x) \in \bar{A}$.

The following lemma is a generalization of the previous one and its proof is similar.

Lemma 0.17 *[**homlim1**] The set $\bar{A}$ is closed under finite homotopy limits. In particular for a pair of maps $f, g : Y \to X$ with $X, Y \in \bar{A}$ the homotopy equalizer $heq(f, g)$ is in $\bar{A}$.*

Lemma 0.18 *[**sumsandproducts**] Let $I \in \bar{A}$ be a set and $(X_i)_{i\in I}$ be a family of types in $\bar{A}$ parametrized by I. Then $\coprod_{i\in I} X_i$ and $\prod_{i\in I} X_i$ are in $\bar{A}$.*

Lemma 0.19 *[**all0**] Let $X \in \bar{A}$. Then $\pi_0(X) \in \bar{A}$ and for any $n > 0$ and any $x \in X$, $\pi_n(X, x) \in \bar{A}$.*

Proof: We have $\pi_0(X) \in \bar{A}$ by Lemma 0.9. By Cl_{eq} we have $\Omega^n(X, x) \in \bar{A}$ where Ω^n is the n-th loop space. Then $\pi_n(X, x) \in \bar{A}$ since $\pi_n(X, x) = \pi_0(\Omega^n(X, x))$.

Proposition 0.20 *[**finitetower**] Let X be a type such that one has:*

1. *$\pi_0(X) \in \bar{A}$,*
2. *for any $x \in X$ and any $n > 0$, $\pi_n(X, x) \in \bar{A}$,*
3. *there exists N such that for any $x \in X$ and any $n > N$ one has $\pi_n(X, x) = 0$.*

Then $X \in \bar{A}$.

Proof: Proceed by induction on N. For $N = 0$ we have $X = \pi_0(X)$ and there is nothing to prove. Using the Cl_{sum} we reduce the problem to a connected X. Let X be connected and let $x \in X$. By induction we know that $M = \Omega^1(X, x) \in \bar{A}$. The space M is a group-like H-space and $X = BM$. It acts on itself by equivalences i.e. there is a map $M \to Eq(M, M)$. Moreover, this map is a mono split by the map which takes an equivalence to its value on the unit (the splitting is not a map of H-spaces). Consider the fibration $f : BM \to BEq(M, M)$. We have $BEq(M, M) = Un(M) \in \bar{A}$ by the inductive assumption and Cl_{un}. It remains to show that the homotopy fiber $F = f^{-1}(*)$ of f over the distinguished point is in $\bar{A}$.

The long exact sequence of homotopy groups defined by f looks as follows (we omit the base points since they are clear):

$$\ldots \to \pi_i(BM) \to \pi_i(BEq) \to \pi_{i-1}(F) \to \ldots \to \pi_1(BM) \to \pi_1(BEq) \to \pi_0(F) \to pt$$

The maps $\pi_i(BM) \to \pi_i(BEq)$ are isomorphic to the maps $\pi_{i-1}(M) \to \pi_{i-1}(Eq)$ and since $M \to Eq(M, M)$ is a split mono they are monomorphisms. By Lemma 0.10 and the inductive assumption we conclude that $\pi_i(F) = 0$ for $i > N - 1$. It remains to check that the non-zero homotopy groups of F are in $\bar{A}$. This follows from our sequence and Lemma 0.15.

Lemma 0.21 *[**homlim2**] Assume that $\mathbf{N} \in \bar{A}$ and let $\ldots X_2 \to X_1 \to X_0$ be a sequence of maps with $X_n \in \bar{A}$. Then $holim X_n \in \bar{A}$.*

Proof: It follows from Lemmas 0.18 and 0.21 since $holim X_n$ is the homotopy equalized of two maps from $\prod_n X_n$ to itself.

Theorem 0.22 *[***all***] Assume that* $\mathbf{N} \in \bar{A}$. *Then the following conditions on* X *are equivalent:*

1. $X \in \bar{A}$
2. $\pi_0(X) \in \bar{A}$ *and for any* $x \in X$ *and any* $n > 0$, $\pi_n(X, x) \in \bar{A}$.

Proof: It follows easily from Lemma 0.21, Lemma 0.19 and Proposition 0.20.

Proposition 0.23 *[***col***] The the following conditions on* $\bar{A}$ *are equivalent:*

1. $\bar{A}$ *is closed under finite homotopy colimits*
2. $\bar{A}$ *contains* $\mathbf{N}$

For a set of homotopy types A let $\bar{A}$ be the universe generated A i.e. the closure of A with respect to Cl_{sum}, Cl_{prod}, Cl_{eq} and Cl_{un}. Consider the following hierarchy of universes and spaces:

1. $U_{-1} = \emptyset$
2. $U_{n+1} = \overline{\{\tilde{\Omega}(U_n), \Omega(U_n)\}}$

Note that since both $\tilde{\Omega}(U_n)$ and $\Omega(U_n)$ are in U_{n+1} all the fibers of the univalent map $\tilde{\Omega}(U_n) \to \Omega(U_n)$ are in U_{n+1} by Lemma 0.16 i.e. $U_n \subset U_{n+1}$. We have the following picture:

1. By definition U_{-1} is empty. Therefore we have $\Omega_{-1} = \tilde{\Omega}_{-1} = \emptyset$.
2. We have $U_0 = \overline{\{\emptyset\}} = \{\emptyset, pt\}$. Therefore $\Omega_0 = \{0, 1\}$ and $\tilde{\Omega}_0 = pt$ which embeds to $\{0, 1\}$ as 1. The map $\tilde{\Omega}_0 \to \Omega_0$ is the universal univalent map of level 0.
3. We have $U_1 = \overline{\{pt, \{0, 1\}\}}$. Form Proposition 0.20 it is easy to deduce that U_1 is the set of homotopy types X such that all π_i of X are finite and there are only finitely many non-trivial π_i's. Let us write:
$$\Omega_1 = \Omega_{1,\leq 0} \subset \Omega_{1,\leq 1} \subset \Omega_{1,\leq 2} \ldots$$
where $\Omega_{1,\leq n}$ is the subtype in Ω_1 corresponding to the types of level n in U_1. We have $\Omega_{1,\leq 0} = \Omega_{0,\leq 0} = \{0, 1\}$. We further have
$$\Omega_{1,\leq 1} = \coprod_{n \geq 0} BS_n$$
in particular $\pi_0(\Omega_{1,\leq 1}) = \mathbf{N}$.

4. The universe U_2 contains $\mathbf{N}$. Therefore by Corollary 0.22 it consists of all types X such that all $\pi_n(X)$ are in U_2. Therefore it is completely determined by its part $U_{2,\le 1}$ of 1-types i.e. sets. This universe contains a lot of sets. It contains $\mathbf{N}$, $\mathbf{R}$ etc. Moreover Lemma 0.18 implies that it contains sets such as $\coprod_{n>0} P^n(\mathbf{N})$. May be this is the whole ZF-universe. In any event it is large enough for all normal mathematics. The universe U_2 can also be described as the closure of the set of finite types with respect to Cl_{sum}, Cl_{prod} and Cl_{eq} or as the closure of $\{pt\}$ with respect to Cl_{sum}, Cl_{prod}, Cl_{eq} and finite homotopy colimits.

5. The higher universes $U_{>2}$ all contain $\mathbf{N}$ and therefore are determined by their subsets of types of level 1. They probably correspond to the universes of ZF with the iterated models of ZF in itself.

Let $U_{n,\le i}$ be the set of types of level i in U_n. We have inclusions $U_{n,\le i} \subset U_{n+1,\le i}$ which are bijections for $i = 0$ and $n \ge 0$. For $i > 0$ it is clear that $U_{0,\le i} \ne U_{1,\le i} \ne U_{2,\le i}$. It seems that one can prove that the same holds for higher n i.e. for $i > 0$ and $n \ge 0$ the set $U_{n+1,\le i}$ is strictly bigger than the universe $U_{n,\le i}$ and any attempt to stabilize this sequence leads to an inconsistency.

Now that we have built the theory of universes of types let us discuss how to define the classifying spaces of types with structures. We fix a universe U or equivalently a univalent map $\tilde{\Omega} \to \Omega$. By definition, the isomorphism classes of types in U are in one to one correspondence with the connected components of Ω. We want to construct spaces whose connected components will be in one to one correspondence with structures in U e.g. groups in U, topological spaces in U, categories in U etc. We define $n - Sets(U)$ to be the class of n-types in U. Instead of $1 - Sets(U)$ we will write $Sets(U)$. The corresponding spaces are $\Omega_{\le n}$.

We start by describing for any first order theory Γ a space $\Gamma(\Omega)$ whose connected components are in one to one correspondence with models of Γ in $Sets(U)$. This will give us spaces such as $Gr(U)$ or $Ring(U)$ for groups and rings in U but not $Top(U)$ or $Cat(U)$. We could skip this stage entirely and consider more general structures right away but this gives some interesting connections with the classical foundations.

Then $Gr(\Omega)$ etc. More generally $\Gamma(\Omega)$ where Γ is a first order theory.

analytic

To different such conditions there correspond different type expressions $\mathbf{Cl}$ and therefore different universe contexts. One of the interesting possibilities is to include the condition that A is closed under homotopy push-outs as well as under homotopy pull-backs. This would imply that A contains all finite homotopy types including in particular S^1. Since any admissible A must be closed under loop functor and $\Omega^1 S^1 = \mathbf{Z}$ the corresponding universe will automatically have integers and it is not hard to deduce from $\mathbf{Z}$ the natural numbers. We see that in the homotopy λ-calculus push-outs i.e. *finite* colimits imply the existence of natural numbers.

1. Let $\Gamma = (T : Type, f : T \to T)$ such that the external models of Γ in Top_A are pairs (X, ϕ) where X is a homotopy type in A and $\phi : X \to X$ is its endomorphism. By assumption A is closed under formation of the internal Hom-objects (spaces of continuous maps). This

implies easily through the universal properties of univalent maps that there is a map $Hom : U \times U \to U$ which takes a point (u_1, u_2) to a point corresponding to the homotopy type $\underline{Hom}(p^{-1}(u_1), p^{-1}(u_2))$. Then the model M of $\Gamma(\Omega)$ is given by the homotopy pull-back square

$$\begin{tikzcd}
M \arrow[r] \arrow[d] & \tilde{U} \arrow[d, "p"'] \\
U \arrow[r, "Hom\circ\Delta"] & U
\end{tikzcd}$$

where $\Delta : U \to U \times U$ is the diagonal.

2. Let $\Gamma = (T : Type; a : \mathbf{Lv}_n)$ where $\mathbf{Lv}_n$ is the type expression explained above. Then the model of $\Gamma(\Omega)$ is the subspace of U which consists of points corresponding to n-types in A. Note the shift of the index compared to the models of the context $(\Omega, a : \mathbf{Lv}_n)$ discussed above.

3. Let Gr be the context of the form $(T : Type; e : T, m : T \times T \to T, a : \textbf{Group\_axioms})$ whose models are groups in A (the group axioms in this case should include $\mathbf{Lv}_1$ since groups must be supported in sets). The model of $Gr(\Omega)$ will be a space $Gr(\tilde{U}/U)$ over U whose homotopy fibers over types which are not sets will be empty and whose fiber over a set X will be the set of all group structures on X. The action of the π_1 of U in the corresponding point in the fiber will be the action of the automorphism group of X on the set of group structures. This space will fit into a pull-back square of the form

$$\begin{tikzcd}
Gr(\tilde{U}/U) \arrow[r] \arrow[d] & \tilde{U} \arrow[d, "p"'] \\
U \arrow[r, "gr"] & U
\end{tikzcd}$$

where $gr : U \to U$ is a map which takes a set to the set of group structures on it and a type which is not a set to the empty set. The connected components of $Gr(\tilde{U}/U)$ are in one-to-one correspondence with the isomorphisms classes of groups in A and for any such group G the corresponding component is the classifying space of the group $Aut(G)$.

4. Extending the previous example consider the construction which assigns to a group its center. This construction will correspond on the one hand to an interpretation of Gr in itself and on the other to an endomorphism of $Gr(\Omega)$ in Ω. The model of this endomorphism will be a map $Gr(\tilde{U}/U) \to Gr(\tilde{U}/U)$ which becomes obvious with respect to the description of $Gr(\tilde{U}/U)$ given at the end of the previous example.

5. Here is a more complicated example. Let Top be the context whose models are topological spaces in A i.e. the sets in A together with a set of subsets satisfying the usual axioms of topology. We will say how to construct such a context below. It is more sophisticated than a context such as Gr since it has several generating types. The main one maps to the underlying set of the corresponding space and the auxiliary one maps to the set $\{0, 1\}$ which is required for the definition of the set of subsets. The model of $Top(\Omega)$ will be the space over U whose components correspond to isomorphism classes (not the homotopy equivalence classes!) of topological spaces whose supporting sets are in A and the component corresponding to a space S will be the classifying space of the group of homeomorphisms from S to itself.

2 Homotopy λ-calculus

1 Type systems

A type system is a formal language. Well formed sentences in this language are called sequents. They are written in the form $\Gamma \vdash \mathcal{G}$ where the part to the left of $\vdash$ is called a context and the part to the right is called a judgement. The collection of all sequents with the given context part Γ is somewhat similar to the collection of all theorems in a given first order theory. In this sense contexts are analogs of theories in the language of a type system. Both the context and the judgement are sequences of words from some initial "vocabulary" V. A pair of such sequences is called a (valid) sequent if it can be obtained from some primitive sequents by means of the constructors i.e. the rules which describe how to construct new sequents from the old ones.

Remark 1.1 Mathematically speaking "sequences of words from a vocabulary V" are elements of the free monoid $F(V)$ generated by V. The underlying language of a type system is therefore defined by a set V, a finite subset S_0 of primitive or generating sequents in $F(V)$ and a finite set R of triples of the form

$$[\mathbf{rules}]\rho = (n_\rho, D_\rho \subset F(V)^{n_\rho}, c_\rho : D_\rho \to F(V)) \tag{1}$$

called rules. The set of sequents S of a type system (V, S_0, R) is the smallest subset in $F(V)$ which contains S_0 and is closed under the rules i.e. given a rule of the form (1) and $s_1, \ldots, s_n \in S$ such that $(s_1, \ldots, s_n) \in D_\rho$ one has $c_\rho(s_1, \ldots, s_n) \in S$. One has to use $F(V)$ as opposed to some abstract set F because some of the rules may involve the substitution maps $F(V) \to F(V)$ given by replacing an element v of V in a word by another word. As far as I understand a type system has no additional structures. However, in order for (V, S_0, R) to be a type system in the usual sense of the word it has to satisfy a lot of conditions and some of these conditions are easier to express as additional structures (they are still conditions in the sense that one can prove uniqueness theorems). To really define a type system in the sense specified above one has to be very careful about the syntax. Two type systems which differ in the number of spaces allowed in a rule are different and not even isomorphic since the only isomorphisms of the combinatorial structures of the kind described above are the ones induced by isomorphisms of the underlying sets V. To get a general theory of type systems one needs to define further the notion of an interpretation of one type system in another and declare two type systems to be equivalent if there are interpretations of the first in the second and vise versa which are in some sense mutually inverse. I do not think such a theory exists and we will not attempt to develop it here. The pedantically formal approach to type systems outlined above is not really followed below.

Let us call a type system decidable if the subset S of sequents is decidable in the set of all sentences over the vocabulary i.e. if there is an algorithm which can mechanically check whether or not a given sentence is a sequent. Decidability of a type system is not directly related to its expressive power. In a sense all sequents in a type system correspond to provable facts. When a decidable type system is used to formalize mathematics each sequent corresponds to something like a theorem together with its proof. The context part encodes all the assumptions, definitions etc. which are necessary for the formulation of the theorem. The judgement part can have different forms. For theorem-like sequents it is a pair $\mathbf{r} : \mathbf{R}$ where $\mathbf{R}$ is the conclusion of the theorem and $\mathbf{r}$ is the proof.

A conjecture is an incomplete sequent having the context Γ and the second half of the judgement $\mathbf{R}$ but not the first part. Proving a conjecture amounts to finding its completion to a full sequent i.e. finding $\mathbf{r}$. Whether or not a conjecture can be proved is not decidable in any sufficiently rich type system. If a usual mathematical proof is known and the type system is well enough adapted to the formalization of mathematics one can obtain $\mathbf{r}$ by translating the steps of the proof into constructors of the system. In many cases a computer program called a proof assistant can be used to translate small steps automatically. If the type system is decidable then a computer program can be used to verify that the result is a valid sequent and therefore the proof is correct.

The meaning of a general sequent of the form $\Gamma \vdash \mathbf{r} : \mathbf{R}$ is that in the context Γ the expression $\mathbf{R}$ describes a type and the expression $\mathbf{r}$ describes a member of this type. Statements of theorems appear as types of a particular kind which are populated by the proofs. Types of this kind are require fairly complex type systems.

To clarify these ideas let us consider first the most fundamental type system called (typed) λ-calculus. In its pure form it can not be used to formalize mathematics since it does not have theorem-like sequents at all. Nevertheless many features of more complex type systems are taken from the λ-calculus and it is a necessary starting point of any type theoretic development.

The vocabulary of the lambda calculus is a disjoint union of three sets Vt, VT and Vsp where Vt and VT are countably infinite sets whose elements are the words reserved for the names of simple terms (term variables and constants) and simple types respectively. The third set Vsp is finite and contains all kinds of special symbols and delimiters appearing in the language. I will not try to explicitly list all its element - they are exactly the ones which appear in the rules given below. I will use lower case letters for elements of Vt and upper case letters for elements of VT. The bold face will be used for expressions i.e. sequences of elements of $V = VT \amalg Vt \amalg Vsp$. For expressions $\mathbf{R}$, $\mathbf{r}$ and $v \in Vt \amalg VT$ one writes $\mathbf{R}(\mathbf{r}/v)$ for the expression obtained from $\mathbf{R}$ by replacing all the occurrences of v in $\mathbf{R}$ by $\mathbf{r}$. The sequents of the lambda calculus are of four forms

$$\Gamma \vdash$$

$$\Gamma \vdash \mathbf{R} : Type \qquad \Gamma \vdash \mathbf{r} : \mathbf{R}$$

and

$$\Gamma \vdash \mathbf{r} = \mathbf{r}' : \mathbf{R}$$

The context part Γ of any sequent is of the form

$$[\mathbf{stcont}] T_1, \ldots, T_n : Type; c_1 : \mathbf{R}_1, \ldots, c_m : \mathbf{R}_m \tag{2}$$

where $T_i \in VT$, $c_j \in Vt$ and $m, n \geq 0$. The generating sequent is the empty one. Traditionally one does not use sequents with the empty judgement part but it seems to make things a little nicer. The rules for generating new sequents are as follows. I may miss some of the more obvious ones. As is usual many different combinations of elementary rules lead to the same final language. I picked the ones which I think are convenient from the explanatory viewpoint even if they are not very convenient from the point of view of implementation.

1. If $T_1, \ldots, T_n : Type; c_1 : \mathbf{R}_1, \ldots, c_m : \mathbf{R}_m \vdash$ is a sequent and $T_{n+1} \in TV - \{T_1, \ldots, T_n\}$ then

$$T_1, \ldots, T_n, T_{n+1} : Type; c_1 : \mathbf{R}_1, \ldots, c_m : \mathbf{R}_m \vdash$$

is a sequent.

2. If $T_1, \ldots, T_n : Type; c_1 : \mathbf{R}_1, \ldots, c_m : \mathbf{R}_m \vdash \mathbf{R} : Type$ is a sequent and $c_{m+1} \in Tv - \{c_1, \ldots, c_m\}$ then
$$T_1, \ldots, T_n : Type; c_1 : \mathbf{R}_1, \ldots, c_m : \mathbf{R}_m, c_{m+1} : \mathbf{R} \vdash$$
is a sequent.

3. If $T_1, \ldots, T_n : Type; c_1 : \mathbf{R}_1, \ldots, c_m : \mathbf{R}_m \vdash$ is a sequent and $i = 1, \ldots, n$ then
$$T_1, \ldots, T_n : Type; c_1 : \mathbf{R}_1, \ldots, c_m : \mathbf{R}_m \vdash T_i : Type$$
is a sequent.

4. If $T_1, \ldots, T_n : Type; c_1 : \mathbf{R}_1, \ldots, c_m : \mathbf{R}_m \vdash$ is a sequent and $j = 1, \ldots, m$ then
$$T_1, \ldots, T_n : Type; c_1 : \mathbf{R}_1, \ldots, c_m : \mathbf{R}_m \vdash c_j : \mathbf{R}_j$$
is a sequent.

5. If $T_1, \ldots, T_n : Type; c_1 : \mathbf{R}_1, \ldots, c_m : \mathbf{R}_m \vdash \mathbf{r} : \mathbf{R}$ is a sequent and $y, y' \in Tv - \{c_1, \ldots, c_m\}$ then
$$T_1, \ldots, T_n : Type; c_1 : \mathbf{R}_1, \ldots, c_m : \mathbf{R}_m \vdash \mathbf{r} = \mathbf{r}(y/y') : \mathbf{R}$$
is a sequent.

To express the rest of the rules we will use the standard notation where one writes
$$\frac{s_1 \; s_2 \; \ldots \; s_n}{s'}$$
to say that if $s_1, \ldots, s_n$ are sequents then s' is a sequent. We set:
$$\frac{\Gamma, y : \mathbf{R} \vdash \mathbf{q} : \mathbf{Q} \qquad \Gamma \vdash \mathbf{r} = \mathbf{r}' : \mathbf{R}}{\Gamma \vdash \mathbf{q}(\mathbf{r}/y) = \mathbf{q}(\mathbf{r}'/y) : \mathbf{Q}}$$

$$[\textbf{typeconstr}]\frac{\Gamma \vdash \mathbf{R} : Type \qquad \Gamma \vdash \mathbf{Q} : Type}{\Gamma \vdash \mathbf{R} \to \mathbf{Q} : Type} \tag{3}$$

$$[\textbf{termconstruct}]\frac{\Gamma, y : \mathbf{R} \vdash \mathbf{q} : \mathbf{Q}}{\Gamma \vdash \lambda y : \mathbf{R}.\mathbf{q} : \mathbf{R} \to \mathbf{Q}} \qquad \frac{\Gamma \vdash \mathbf{r} : \mathbf{R} \qquad \Gamma \vdash \mathbf{f} : \mathbf{R} \to \mathbf{Q}}{\Gamma \vdash ev(\mathbf{f}, \mathbf{r}) : \mathbf{Q}} \tag{4}$$

$$[\textbf{conversions}]\frac{\Gamma, y : \mathbf{R} \vdash \mathbf{q} : \mathbf{Q} \qquad \Gamma \vdash \mathbf{r} : \mathbf{R}}{\Gamma \vdash ev(\lambda y : \mathbf{R}.\mathbf{q}, \mathbf{r}) = \mathbf{q}(\mathbf{r}/y) : \mathbf{Q}} \qquad \frac{\Gamma \vdash \mathbf{f} : \mathbf{R} \to \mathbf{Q}}{\Gamma \vdash \lambda y : \mathbf{R}.ev(\mathbf{f}, y) = \mathbf{f} : \mathbf{R} \to \mathbf{Q}} \tag{5}$$

$$[\textbf{morestr}]\frac{\Gamma \vdash \mathbf{r} = \mathbf{r}' : \mathbf{R}}{\Gamma \vdash \mathbf{r}' = \mathbf{r} : \mathbf{R}} \qquad \frac{\Gamma \vdash \mathbf{r} = \mathbf{r}' : \mathbf{R} \qquad \Gamma \vdash \mathbf{r}' = \mathbf{r}'' : \mathbf{R}}{\Gamma \vdash \mathbf{r} = \mathbf{r}'' : \mathbf{R}} \tag{6}$$

The rules of the first group are called type constructors, the rules of the second group are called term constructors and the rules of the third group are called conversions. The rules of the fourth group should be actually included with the "structural" rules (1)-(5). The first of the term constructors called the function introduction or the abstraction rule the second one is called function elimination or application rule. The first of the two conversions is called the β-conversion and the second the η-conversion. In the η-conversion one should add the additional condition that $y \in Tv - \{c_1, \ldots, c_m\}$. In the classical λ-calculus one writes $\mathbf{f}\, y$ instead of $ev(\mathbf{f}, y)$.

Let us use the word context both for the left hand sides of sequents and for sequents of the form $\Gamma \vdash$. The rules imply that if $\Gamma \vdash \mathcal{G}$ is a sequent then $\Gamma \vdash$ is a sequent so this ambiguity is not problematic. For a context Γ let $Rexp(\Gamma)$ be the set of all $\mathbf{R}$ such that $\Gamma \vdash \mathbf{R} : Type$ is a sequent. For Γ and $\mathbf{R} \in Rexp(\Gamma)$ let $Lexpp(\Gamma, \mathbf{R})$ be the set of all $\mathbf{r}$ such that $\Gamma \vdash \mathbf{r} : \mathbf{R}$. Finally let $Lexp(\Gamma, \mathbf{R})$ be the quotient of $Lexpp(\Gamma, \mathbf{R})$ by the equivalence relation defined by the condition that $\Gamma \vdash \mathbf{r} = \mathbf{r}' : \mathbf{R}$. Elements of $Rexp$ are called (valid) type expressions in Γ and elements of the $Lexp(\Gamma, \mathbf{R})$ are called (valid) term expressions of type $\mathbf{R}$ (in Γ). We do not distinguish term expressions which are convertible to each other. The letters R and L are there because elements of $Lexp$ occur to the left of : and elements of $Rexp$ mostly occur to the right of :.

Contexts are analogs of theories in the languages defined by type systems and the semantics of a type system is based on the notion of a model of a context. In the case of λ-calculus models can take values in any Cartesian closed category but for the illustrative purposes it makes sense to start with set-theoretic models.

One defines models of

$$\Gamma = T_1, \ldots, T_n : Type; c_1 : \mathbf{R}_1, \ldots, c_m : \mathbf{R}_m$$

by induction on m. Such an induction makes sense because the rules of lambda calculus imply that for any context Γ the sequences $\Gamma_{\leq j} = T_1, \ldots, T_n : Type; c_1 : \mathbf{R}_1, \ldots, c_j : \mathbf{R}_j$ are also contexts. Moreover, for a context Γ and an expression $\mathbf{R}$ the sequence $\Gamma_{\leq j}, c : \mathbf{R}$ is a context if and only if $c \in Vt - \{c_1, \ldots, c_m\}$ and $T_1, \ldots, T_n : Type \vdash \mathbf{R} : Type$.

A context with $m = 0$ is called a basic context and the basic context underlying a given Γ is called the base of Γ. A model M_0 of the basic context $T_1, \ldots, T_n : Type$ is just a collection of sets $X_i = M_0(T_i)$ corresponding to the generating types T_i. We assume the model M_0 of the base of Γ fixed. Let $T_1, \ldots, T_n : Type \vdash \mathbf{S} : Type$. Any sequent of this form can be obtained from the sequents $T_1, \ldots, T_n : Type \vdash T_i : Type$ by the constructors (3). Define a set $M(\mathbf{S}) = M(\mathbf{S}, M_0)$ inductively as follows:

1. $M(T_i) = X_i$
2. $M(\mathbf{R} \to \mathbf{Q}) = Hom(M(\mathbf{R}), M(\mathbf{Q}))$

where $Hom(X, Y)$ is the set of maps of sets from X to Y. Then a model M of Γ over M_0 is a sequence of points $M(c_j) \in M(\mathbf{R}_j)$. For example, a model of the context $(T_1, T_2 : Type; f : T_1 \to T_2)$ is a pair of sets X_1, X_2 together with a function $\phi : X_1 \to X_2$.

We already know that a model M of Γ defines for any $\mathbf{S} \in Rexp(\Gamma)$ a set $M(\mathbf{S})$ (which actually depends only on M_0). Let us show now that M further defines for any $\mathbf{s} \in Lexp(\mathbf{S})$ an element $M(\mathbf{s}) \in M(\mathbf{S})$.

We first define $M(\mathbf{s})$ for a sequent $\Gamma \vdash \mathbf{s} : \mathbf{S}$ and then show that if $\Gamma \vdash \mathbf{s} = \mathbf{s}' : \mathbf{S}$ then for any M one has $M(\mathbf{s}) = M(\mathbf{s}')$. For a given base model M_0 the collection of all models of Γ over M_0 is identified with the set

$$E(\Gamma) = E(\Gamma; M_0) = \prod_{j=1}^{m} M(\mathbf{R}_j)$$

All sequents of the form $\Gamma \vdash \mathbf{s} : \mathbf{S}$ are obtained from the sequents $\Gamma \vdash c_j : \mathbf{R}_j$ by the rules (4). Define for each $\Gamma \vdash \mathbf{s} : \mathbf{S}$ a map $e(s) : E(\Gamma) \to M(\mathbf{S})$ inductively as follows:

1. for $\mathbf{R} = \mathbf{R}_j$ and $\mathbf{r} = c_j$ let $e(s)$ be the projection $E(\Gamma) \to M(\mathbf{R}_j)$
2. for $\mathbf{S} = \mathbf{R} \to \mathbf{Q}$ and $\mathbf{s} = \lambda y : \mathbf{R}.\mathbf{q}$ let

   $$s(\lambda y : \mathbf{R}.\mathbf{q}) : E(\Gamma) \to Hom(M(\mathbf{R}), M(\mathbf{Q}))$$

   to be the map adjoint to the map

   $$e(\mathbf{q}) : E(\Gamma, y : \mathbf{R}) = E(\Gamma) \times M(\mathbf{R}) \to M(\mathbf{Q})$$

3. for $\mathbf{S} = \mathbf{Q}$ and $\mathbf{s} = ev(\mathbf{f}, \mathbf{r})$ let

   $$e(ev(\mathbf{f}, \mathbf{r})) : E(\Gamma) \to M(\mathbf{R})$$

   to be the composition

   $$E(\Gamma) \overset{e(\mathbf{f})\times e(\mathbf{r})}{\longrightarrow} Hom(M(\mathbf{R}), M(\mathbf{Q})) \times M(\mathbf{R}) \overset{ev}{\to} M(\mathbf{R})$$

   where ev is the usual evaluation map.

We can now define $M(\mathbf{s})$ for $M \in E(\Gamma)$ as $e(\mathbf{s})(M)$. To verify that this construction indeed agrees with the conversions it is clearly sufficient to check that $e(\mathbf{s}) = e(\mathbf{s}')$ when $\mathbf{s}$ and $\mathbf{s}'$ are as in the rules (5). The β-conversion involves a substitution $\mathbf{q}(y/\mathbf{r})$ and we need first to describe $e(\mathbf{q}(\mathbf{r}/y))$. One has the following lemma.

Lemma 1.2 *[**subst**] Given $\Gamma, y : \mathbf{R} \vdash \mathbf{q} : \mathbf{Q}$ and $\Gamma \vdash \mathbf{r} : \mathbf{R}$ one has $\Gamma \vdash \mathbf{q}(\mathbf{r}/y) : \mathbf{Q}$. The map*

$$e(\mathbf{q}(\mathbf{r}/y)) : E(\Gamma) \to M(\mathbf{Q})$$

is the composition

$$E(\Gamma) \overset{Id\times e(\mathbf{r})}{\longrightarrow} E(\Gamma) \times M(\mathbf{R}) = E(\Gamma, y : \mathbf{R}) \overset{e(\mathbf{q})}{\to} E(\mathbf{Q}).$$

We can now verify the conversions:

1. For the β-conversion we have $\Gamma, y : \mathbf{R} \vdash \mathbf{q} : \mathbf{Q}$ and $\Gamma \vdash \mathbf{r} : \mathbf{R}$ and we need to check that the composition
$$E(\Gamma) \stackrel{Id\times e(\mathbf{r})}{\longrightarrow} E(\Gamma)\times M(\mathbf{R}) \stackrel{e(\mathbf{q})}{\to} E(\mathbf{Q})$$
coincides with
$$e(ev(\lambda y : \mathbf{R}.\mathbf{q}, \mathbf{r})) = ev(e(\lambda y : \mathbf{R}.\mathbf{q}), e(\mathbf{r})).$$
This follows immediately from the definition of $e(\lambda y : \mathbf{R}.\mathbf{q})$ as the function adjoint to $e(\mathbf{q})$.

2. For the η-conversion we have $\Gamma \vdash \mathbf{f} : \mathbf{R} \to \mathbf{Q}$ and we need to check that
$$e(\lambda y : \mathbf{R}.ev(\mathbf{f}, \mathbf{r})) = e(\mathbf{f}).$$
This is a simple exercise in opening up the definitions.

It is clear from the constructions of $M(\mathbf{R})$ and $M(\mathbf{r})$ given above that they can be repeated with the category of sets replaced by any Cartesian closed category i.e. a category with finite products (including the final object) and internal Hom-objects. Given a model M of Γ in such a category $\mathcal{C}$ and a functor $F : \mathcal{C} \to \mathcal{C}'$ which preserves products and internal Hom-objects one gets a model $F(M)$ of Γ in $\mathcal{C}'$. One of the reasons why these generalized models are interesting is that for any context there is a universal generalized model. More precisely to each context Γ one can associate a Cartesian closed category $\mathcal{C}(\Gamma)$ and a model M of Γ in $\mathcal{C}(\Gamma)$ such that for any $\mathcal{C}'$ models of Γ in $\mathcal{C}'$ are in one to one correspondence (up to an isomorphism) with Cartesian functors from $\mathcal{C}(\Gamma)$ to $\mathcal{C}'$. These observation provides a connection between lambda calculus and the theory of Cartesian closed categories which extends in a non-trivial way to other more complex type systems.

The construction of $\mathcal{C}(\Gamma)$ is very simple and can be outlined as follows. The category $\mathcal{C}(\Gamma)$ is a small category in a very strict sense i.e. its objects and morphisms form sets. The set of objects of $\mathcal{C}(\Gamma)$ is $\coprod_{i\geq 0} Rexp(\Gamma)^i$. One denotes its element corresponding to $i = 0$ by $\mathbf{pt}$. The set of morphisms from $(\mathbf{R}_1, \ldots, \mathbf{R}_i)$ to $\mathbf{Q}$ is
$$Mor((\mathbf{R}_1, \ldots, \mathbf{R}_i), \mathbf{Q}) = Lexp((\mathbf{R}_1 \to (\mathbf{R}_2 \to (\ldots \to (\mathbf{R}_i \to \mathbf{Q})\ldots))))$$
in particular the set $Mor(\mathbf{pt}, \mathbf{Q})$ is $Lexp(\mathbf{Q})$. The set of morphisms from $(\mathbf{R}_1, \ldots, \mathbf{R}_i)$ to $(\mathbf{Q}_1, \ldots, \mathbf{Q}_j)$ is
$$Mor((\mathbf{R}_1, \ldots, \mathbf{R}_i), (\mathbf{Q}_1, \ldots, \mathbf{Q}_j)) = \prod_{k=1,\ldots,j} Mor((\mathbf{R}_1, \ldots, \mathbf{R}_i), \mathbf{Q}_j).$$
In particular the set $Mor((\mathbf{R}_1, \ldots, \mathbf{R}_i), \mathbf{pt})$ is the one element set i.e. $\mathbf{pt}$ is the final object and for $j > 0$ the object $(\mathbf{Q}_1, \ldots, \mathbf{Q}_j)$ is the product of objects $\mathbf{Q}_k$ for $k = 1, \ldots, j$.

For a basic context $T_1, \ldots, T_n : Type$ one gets a free Cartesian closed category generated by objects $T_1, \ldots, T_n$. For a more complex context one gets a free Cartesian closed category generated by objects $T_1, \ldots, T_n$ and morphisms $c_1, \ldots, c_m$ from the final object to the corresponding $\mathbf{R}_j$. Since morphisms between two objects are identified with the morphisms from the point to the corresponding internal Hom-object one can obtain in this way a Cartesian closed category freely generated by any finite set of objects and morphisms.

Note also that objects of $\mathcal{C}(T_1, \ldots, T_n : Type)$ are in one to one correspondence with contexts with the base $T_1, \ldots, T_n : Type$ up to the change of names of generating constants c_j. This is a reflection of the fact that for Γ of the usual form (2) the category $\mathbf{C}(\Gamma)$ can be identified with the slice category $\mathcal{C}(T_1, \ldots, T_n : \mathit{Type})/B$ where B is the object of $\mathcal{C}(T_1, \ldots, T_n : \mathit{Type})$ given by $(\mathbf{R}_1, \ldots, \mathbf{R}_j)$ i.e. a category given by generating objects and generating morphisms can be identified with the category of objects over an appropriate base in the category generated by objects only.

What is clearly missing from the lambda calculus as we have described it is the ability to define contexts whose models correspond collections of sets and maps satisfying some equations. Equivalently we can not use this type system to get categories $\mathcal{C}(\Gamma)$ defined by generators and *relations*. This deficiency is in a sense the source of all further developments.

An obvious way to introduce relations into the system is to allow conversion judgments to appear in the context part of the sequents i.e. to add to the structural rules (1)-(5) and (6) the rules

$$\frac{\Gamma \vdash \mathbf{r} : \mathbf{R} \quad \Gamma \vdash \mathbf{r}' : \mathbf{R}}{\Gamma, \mathbf{r} = \mathbf{r}' : \mathbf{R} \vdash}$$

and

$$\frac{T_1, \ldots, T_n : \mathit{Type}; c_1 : \mathbf{R}_1, \ldots, \mathbf{r} = \mathbf{r}' : \mathbf{R}_j, \ldots, c_m : \mathbf{R}_m \vdash}{T_1, \ldots, T_n : \mathit{Type}; c_1 : \mathbf{R}_1, \ldots, \mathbf{r} = \mathbf{r}' : \mathbf{R}_j, \ldots, c_m : \mathbf{R}_m \vdash \mathbf{r} = \mathbf{r}' : \mathbf{R}_j}$$

One can define models of such contexts in the obvious way. For example a set-theoretic model of

$$\Gamma = T : \mathit{Type}; t : T, f : T \to T, ev(f, t) = t : T$$

will be a set X with a point $x \in X$ and an endomorphism $\phi : X \to X$ such that $\phi(x) = x$. In this extended system (which we call the equational λ-calculus) one can define contexts whose models are all kinds of algebraic systems for example one can define a context Gr whose models are groups. This change of the rules however has profound and not very pleasant consequences.

One of the key classical results about λ-calculus is that it is a decidable type system. I.e. given a sentence consisting of the type names, term names and the special symbols used above one can determine in a mechanical way whether or not this sequence represent a valid sequent in the λ-calculus. This is easy to do for sequents of all forms except $\Gamma \vdash \mathbf{r} = \mathbf{r}' : \mathbf{R}$. The fact that such sequents are also decidable i.e. that for two term expressions one can determine mechanically whether or not they are equivalent with respect to conversions is known as the Church-Rosser Theorem.

It is easy to see that the analog of this theorem for the equational λ-calculus fails by constructing for any group given by generators and relations and any two words in this group a context Γ and a candidate sequent of the form $\Gamma \vdash \mathbf{r} = \mathbf{r}' : \mathbf{R}$ which really is a sequent if and only if the corresponding words define the same element of the group.

Another approach to the equality problem is based on the idea of *equality types* which allows one to impose equalities between terms without modifying the conversion rules. Equality types are introduced by the additional type constructor:

$$\frac{\Gamma \vdash \mathbf{r} : \mathbf{R} \quad \Gamma \vdash \mathbf{r}' : \mathbf{R}}{\Gamma \vdash eq_{\mathbf{R}}(\mathbf{r}, \mathbf{r}') : \mathit{Type}}$$

with the semantics that a model M of Γ maps $eq_{\mathbf{R}}(\mathbf{r}, \mathbf{r}')$ to the one point set if $M(\mathbf{r}) = M(\mathbf{r}')$ and to the empty set otherwise. Now one can impose relations between terms by adding to contexts judgments of the usual form $s : \mathbf{S}$ with $\mathbf{S} = eq_{\mathbf{R}}(\mathbf{r}, \mathbf{r}')$. The argument for un-decidability given above does not work anymore since the candidate sequent expressing the equality of elements corresponding to $\mathbf{r}$ and $\mathbf{r}'$ will now take the form $\Gamma \vdash \mathbf{a} : eq_{\mathbf{R}}(\mathbf{r}, \mathbf{r}')$ where $\mathbf{a}$ will be a term expression which contains in it the proof that these elements are equal.

The introduction of the equality types however brings with it a whole new dimension to the type system because the type expressions $eq_{\mathbf{R}}(\mathbf{r}, \mathbf{r}')$ unlike the type expressions we have considered before depend on terms. Another issue which arises is how to ensure that the equality types are mapped by models to *pt* or $\emptyset$ and not to sets with many elements. One can impose the later condition by a rule saying that any two terms of an equality type are equivalent under conversions but this immediately resurrects the un-decidability argument. I do not know of any satisfactory solution of these problems in the usual type theories.

2 Homotopy λ-calculus

There are three layers in the homotopy λ-calculus i.e. strictly speaking we will define three type systems which can be called the basic layer, the logic layer and the universe layer of the homotopy λ-calculus. The logic and the universe layers are obtained from the basic layer by the addition of some new type and term constructors but all three layers share the same basic architecture and the same conversion rules. Up to the moment when we introduce term constructors and conversions for the equality types our system is the standard dependent type system with dependent products, (strong) dependent sums and $\beta\eta$-conversions.

At each step we will describe the syntax and the semantics with respect to the topological models i.e. models with values in the category of (nice enough) topological spaces. It is actually easier to give a rigorous definition of a model with values in the category of Kan simplicial sets but since we do not aim to provided detailed proofs we choose the category of topological spaces as a more familiar one.

In all three layers sequents are of the forms

$$\Gamma \vdash$$

$$\Gamma \vdash \mathbf{R} : Type \qquad \Gamma \vdash \mathbf{r} : \mathbf{R}$$

$$\Gamma \vdash \mathbf{R} = \mathbf{R}' : Type \qquad \Gamma \vdash \mathbf{r} = \mathbf{r}' : \mathbf{R}$$

and contexts are of the form

$$\Gamma = (T_1, \ldots, T_n : Type, c_1 : \mathbf{R}_1, \ldots, c_m : \mathbf{R}_m)$$

The structural rules are the rules (1)-(5) and (6) of the previous section plus the rules:

1. If $T_1, \ldots, T_n : Type; c_1 : \mathbf{R}_1, \ldots, c_m : \mathbf{R}_m \vdash \mathbf{R} : Type$ is a sequent and $y, y' \in Tv - \{c_1, \ldots, c_m\}$ then
$$T_1, \ldots, T_n : Type; c_1 : \mathbf{R}_1, \ldots, c_m : \mathbf{R}_m \vdash \mathbf{R} = \mathbf{R}(y/y') : Type$$
is a sequent

2.
$$\frac{\Gamma \vdash \mathbf{R} = \mathbf{R}' : \mathit{Type}}{\Gamma \vdash \mathbf{R}' = \mathbf{R} : \mathit{Type}} \qquad \frac{\Gamma \vdash \mathbf{R} = \mathbf{R}' : \mathit{Type} \quad \Gamma \vdash \mathbf{R}' = \mathbf{R}'' : \mathit{Type}}{\Gamma \vdash \mathbf{R} = \mathbf{R}'' : \mathit{Type}}$$

3.
$$\frac{\Gamma, y : \mathbf{R} \vdash \mathbf{Q} : \mathit{Type} \quad \Gamma \vdash \mathbf{r} = \mathbf{r}' : \mathbf{R}}{\Gamma \vdash \mathbf{Q}(\mathbf{r}/y) = \mathbf{Q}(\mathbf{r}'/y) : \mathit{Type}}$$

4.
$$\frac{\Gamma \vdash \mathbf{r} : \mathbf{R} \quad \Gamma \vdash \mathbf{R}' = \mathbf{R} : \mathit{Type}}{\Gamma \vdash \mathbf{r} : \mathbf{R}'}$$

3 Basic layer - syntax

There are the following type constructor rules:

$$[\mathbf{sumconstr}]\frac{\Gamma, y : \mathbf{R} \vdash \mathbf{Q} : \mathit{Type}}{\Gamma \vdash \sum y : \mathbf{R}.\mathbf{Q} : \mathit{Type}} \tag{7}$$

$$[\mathbf{prodconstr}]\frac{\Gamma, y : \mathbf{R} \vdash \mathbf{Q} : \mathit{Type}}{\Gamma \vdash \prod y : \mathbf{R}.\mathbf{Q} : \mathit{Type}} \tag{8}$$

$$[\mathbf{eqconstr}]\frac{\Gamma \vdash \mathbf{y}_1 : \mathbf{Q}, \mathbf{y}_2 : \mathbf{Q}}{\Gamma \vdash eq_{\mathbf{Q}}(\mathbf{y}_1, \mathbf{y}_2) : \mathit{Type}} \tag{9}$$

Following the usual convention we will write $\mathbf{R} \to \mathbf{Q}$ instead of $\prod y : \mathbf{R}.\mathbf{Q}$ and $\mathbf{R} \times \mathbf{Q}$ instead of $\sum y : \mathbf{R}.\mathbf{Q}$ when $\Gamma \vdash \mathbf{Q} : \mathit{Type}$ i.e. when $\mathbf{Q}$ does not depend on y.

Note that $eq(-,-)$ is the only type constructor which allows one to produce types dependent on terms. If any type expression $\mathbf{Q}$ depends on a term variable v it means that somewhere in this expression there appears $eq_{\mathbf{S}}(\mathbf{s}_1(v), \mathbf{s}_2(v))$ where $\mathbf{S}$ is a type expression which does not dependent on v.

There are the following term constructor rules:

1. One has the following "introduction" and "elimination" rules for the sum

$$[\mathbf{sumintro}]\frac{\Gamma \vdash \mathbf{R} : \mathit{Type} \quad \Gamma, y : \mathbf{R} \vdash \mathbf{Q} : \mathit{Type}}{\Gamma, y : \mathbf{R}, z : \mathbf{Q} \vdash \langle y, z\rangle : \sum y : \mathbf{R}.\mathbf{Q}} \tag{10}$$

$$[\mathbf{sumelim}]\frac{\Gamma \vdash \mathbf{u} : \sum y : \mathbf{R}.\mathbf{Q}}{\Gamma \vdash \pi\mathbf{u} : \mathbf{R}} \qquad \frac{\Gamma \vdash \mathbf{u} : \sum y : \mathbf{R}.\mathbf{Q}}{\Gamma \vdash \pi'\mathbf{u} : \mathbf{Q}[\pi\mathbf{u}/y]} \tag{11}$$

2. One has the following "introduction" and "elimination" rules for the product

$$[\mathbf{prodintro}]\frac{\Gamma, y : \mathbf{R} \vdash \mathbf{q} : \mathbf{Q}}{\Gamma \vdash \lambda y : \mathbf{R}.\mathbf{q} : \prod y : \mathbf{R}.\mathbf{Q}} \tag{12}$$

$$[\mathbf{prodelim}]\frac{\Gamma \vdash \mathbf{f} : \prod y : \mathbf{R}.\mathbf{Q} \quad \Gamma \vdash \mathbf{r} : \mathbf{R}}{\Gamma \vdash ev(\mathbf{f}, \mathbf{r}) : \mathbf{Q}[\mathbf{r}/y]} \tag{13}$$

3. One has the following five rules for the equivalence types:

$$[\mathbf{idrule}]\frac{\Gamma \vdash \mathbf{r} : \mathbf{R}}{\Gamma \vdash id(\mathbf{r}) : eq_{\mathbf{R}}(\mathbf{r},\mathbf{r})} \tag{14}$$

$$[\mathbf{smart0}]\frac{\Gamma \vdash \mathbf{Q} : Type \quad \Gamma, y : \mathbf{R} \vdash \mathbf{q} : \mathbf{Q} \quad \Gamma \vdash \mathbf{h} : eq_{\mathbf{R}}(\mathbf{r},\mathbf{r}')}{\Gamma \vdash \theta y : \mathbf{R}.(\mathbf{h},\mathbf{q}) : eq_{\mathbf{Q}}(\mathbf{q}(\mathbf{r}/y), \mathbf{q}(\mathbf{r}'/y))} \tag{15}$$

$$[\mathbf{smart2}]\frac{\Gamma, y : \mathbf{R} \vdash \mathbf{e} : eq_{\mathbf{Q}}(\mathbf{q},\mathbf{q}')}{\Gamma \vdash ex(\mathbf{e}) : eq_{\prod y:\mathbf{R}.\mathbf{Q}}(\lambda y : \mathbf{R}.\mathbf{q}, \lambda y : \mathbf{R}.\mathbf{q}')} \tag{16}$$

$$\frac{\Gamma \vdash \mathbf{R} : Type}{\Gamma, x, y, z : \mathbf{R}, \phi : eq_{\mathbf{R}}(x,y), \psi : eq_{\mathbf{R}}(x,z) \vdash s(\phi,\psi) : eq_{\sum u:\mathbf{R}.eq_{\mathbf{R}}(x,u)}(\langle y,\phi\rangle, \langle z,\psi\rangle)} \tag{17}$$

$$\frac{\Gamma \vdash \mathbf{R} : Type}{\Gamma, x, y : \mathbf{R}, \phi : eq_{\mathbf{R}}(x,y), \vdash \epsilon(\phi) : eq_{eq_{\mathbf{R}}(x,y)}(\pi(s(id(x),\phi)),\phi)} \tag{18}$$

Our introduction and elimination rules for the dependent sum and the dependent product are the same as in other dependent type systems with the elimination rule for the sums being the "strong" version (see e.g. [**?**,]).

Let me make a few comments about the rules for the equivalences. The first of these is the usual introduction rule which provides the canonical identity term in the equivalences between a term and itself. The second rule essentially says that equivalences can be pushed through functions i.e. given a function $y \mapsto \mathbf{q}$, two terms $\mathbf{r}$ and $\mathbf{r}'$ in the source and an equivalence between these two terms one gets an equivalence between images of these terms. The notation $\theta y : \mathbf{R}.(\mathbf{h},\mathbf{q})$ is chosen to emphasize that y becomes a bound variable in this expression.

Here is an example. Consider $f, f' : R \to Q$ and $g : Q \to S$. The composition $g \circ f$ is written in the λ-calculus as $\lambda r : R.ev(g, ev(f,r))$ and similarly for $g \circ f'$. If now $\phi : eq_{R\to Q}(f,f')$ then

$$\theta f : \mathbf{R} \to \mathbf{Q}.(\phi, \lambda r : R.g\, f\, r) : eq_{\mathbf{R}\to\mathbf{S}}(\lambda r : R.ev(g, ev(f,r)), \lambda r : R.ev(g, ev(f',r)))$$

i.e. we got what in the language of 2-categories one would denote by $g * \phi : eq_{R\to S}(g \circ f, g \circ f')$.

The third equivalence rule is known as functional extensionality. In its original form it was meant to encode the fact if two functions give the same result when applied to any input then they are equal. In our semantics it corresponds to the fact that a homotopy between two maps is the same as a path from the point corresponding to the first map to the point corresponding to the second in the space of maps.

The forth rule asserts that for two equivalences starting at the same term x we are given an equivalence between them in the space of equivalences starting in x. The projection $\pi(s(\phi,\psi))$ is a member of $eq_{\mathbf{R}}(y,z)$ which corresponds on the intuitive level to the composition of the inverse to the first equality with the second. The fifth rule asserts that the composition of an equivalence with the identity is equivalent to the original equivalence. These two rules imply in particular that the inhabitation of the types $eq_R(-,-)$ defines an equivalence relation on terms of R. They are related on the model level to the extension of covering homotopy property for fibrations and play important role in many basic constructions. I am not completely sure at the moment that the rules (17) and (18) are sufficient to cover all the cases where some analog of the extension of covering homotopy property is required but there is a chance that they are.

The last two rules are distinct from the first three since there are conversions relating the first three rules but there are no conversions related to the last two.

In the usual dependent type systems there is also the equality elimination rule. It ensures that if we have a type expression $\mathbf{Q} = \mathbf{Q}(y)$ where y is a variable of type R and if we have an equivalence $\phi : eq_R(y, y')$ then there is a way to produce members of $\mathbf{Q}(y')$ from members of $\mathbf{Q}(y)$ (one in fact considers the ability to produce members of $\mathbf{Q}(y, y')$ from members in $\mathbf{Q}(y, y)$ where $\mathbf{Q}$ depends on two variables from $\mathbf{R}$). As was mentioned above in our case the only way to create a type expression dependent on a term variable is through the use of eq-constructor. Since equivalences can be "pushed through" all term expressions with the help of rule (15) this implies that we only need an analog of the equality elimination rule for the expressions $\mathbf{Q} = eq_R(x, y)$. This is achieved by our rule (17) which therefore may be considered as an analog of the equality elimination rules in other dependent type systems. From this point of view rule (18) corresponds to the β-conversion for the equality. We could have introduced a conversion instead of the equivalence $\epsilon(-)$ but this approach allows more flexibility in the models.

There are the following conversion rules:

1. The β and η conversions for the product:

$$\frac{\Gamma, y : \mathbf{R} \vdash \mathbf{q} : \mathbf{Q} \qquad \Gamma \vdash \mathbf{r} : \mathbf{R}}{\Gamma \vdash ev(\lambda y : \mathbf{R}.\mathbf{q}, \mathbf{r}) = \mathbf{q}(\mathbf{r}/y) : \mathbf{Q}(\mathbf{r}/y)} \qquad \frac{\Gamma \vdash \mathbf{f} : \prod y : \mathbf{R}.\mathbf{Q}}{\Gamma \vdash \lambda y : \mathbf{R}.ev(\mathbf{f}, y) = \mathbf{f} : \prod y : \mathbf{R}.\mathbf{Q}}$$

   where in the second (i.e. η) conversion one needs to assume in addition that y is not among the generating constants of Γ i.e. that y "does not occur freely in $\mathbf{f}$".

2. The β conversions and the η conversion for the sum:

$$\frac{\Gamma \vdash \langle \mathbf{r}, \mathbf{q} \rangle : \sum y : \mathbf{R}.\mathbf{Q}}{\Gamma \vdash \pi \langle \mathbf{r}, \mathbf{q} \rangle = \mathbf{r} : \mathbf{R}} \qquad \frac{\Gamma \vdash \langle \mathbf{r}, \mathbf{q} \rangle : \sum y : \mathbf{R}.\mathbf{Q}}{\Gamma \vdash \pi' \langle \mathbf{r}, \mathbf{q} \rangle = \mathbf{q} : \mathbf{Q}(\mathbf{r}/y)}$$

$$\frac{\Gamma \vdash z : \sum y : \mathbf{R}.\mathbf{Q}}{\Gamma \vdash \langle \pi\, z, \pi' z \rangle = z : \sum y : \mathbf{R}.\mathbf{Q}}$$

3. The following conversions for the equality types:

$$\frac{\Gamma \vdash \mathbf{h} : eq_{\mathbf{R}}(\mathbf{r}, \mathbf{r}')}{\Gamma \vdash \theta y : \mathbf{R}.(\mathbf{h}, y) = \mathbf{h} : eq_{\mathbf{R}}(\mathbf{r}, \mathbf{r}')}$$

$$\frac{\Gamma \vdash \theta z : \mathbf{Q}.(\theta y : \mathbf{R}.(\mathbf{h}, \mathbf{q}), \mathbf{s}) : eq_{\mathbf{S}}(\mathbf{s}(\mathbf{q}(\mathbf{r}/y)/z), \mathbf{s}(\mathbf{q}(\mathbf{r}'/y)/z))}{\Gamma \vdash \theta z : \mathbf{Q}.(\theta y : \mathbf{R}.(\mathbf{h}, \mathbf{q}), \mathbf{s}) = \theta y : \mathbf{R}.(\mathbf{h}, \mathbf{s}(\mathbf{q}/z)) : eq_{\mathbf{S}}(\mathbf{s}(\mathbf{q}(\mathbf{r}/y)/z), \mathbf{s}(\mathbf{q}(\mathbf{r}'/y)/z))}$$

4 Basic layer - semantics in Top

A model M of Γ in Top is given by a sequence of topological spaces $X_i = M(T_i)$ one for each of the generating types $T_1 \dots T_n$ and points $x_i = M(c_i)$ in the spaces $M(\mathbf{R}_i)$ corresponding to the type expressions $\mathbf{R}_i$.

The key feature of the homotopy λ-calculus is the "invariance" of models with respect to homotopy equivalences. Consider for example the context $\Gamma = (T_1, \dots, T_n : \mathit{Type})$. A model M of Γ is a collection of topological spaces $X_i = M(T_i)$ for $i = 1, \dots, n$. Let $X'_i = M'(T_i)$ be another model of Γ and let $f_i : X_i \to X'_i$ be homotopy equivalences. The invariance property in this case means that for any type expression $\mathbf{R}$ in Γ there exists a homotopy equivalence $f(\mathbf{R}) : M(\mathbf{R}) \to M'(\mathbf{R})$. Hence, while we speak of models in Top the real target category is the homotopy category H. It is important to note that models are not functorial with respect to maps $X_i \to X'_i$ (or even with respect to homotopy equivalences). For example, it is not difficult to define in the context $(T : \mathit{Type})$ a type expression $\mathbf{End}$ such that for a model $X = M(T)$ the space $M(\mathbf{End})$ will be homotopy equivalent to the space $End(X)$ of endomorphisms of X. Clearly, $End(X)$ is not functorial with respect to X. However, if $f : X \to X'$ is a homotopy equivalence then there exists a homotopy equivalence $End(X) \to End(X')$.

We let $\Gamma_{\leq i}$ denote the context

$$\Gamma_{\leq i} = (T_1, \dots, T_n : \mathit{Type}; c_1 : \mathbf{R}_1, \dots, c_i : \mathbf{R}_i).$$

Then $\Gamma_{\leq i+1} = (\Gamma_{\leq i}, c_{i+1} : \mathbf{R}_{i+1})$ where $\mathbf{R}_{i+1}$ is a valid type expression in $\Gamma_{\leq i}$.

A model of a basic context is just a collection of spaces $X_1, \dots, X_n$ corresponding to the generating types. We will further speak of models of a context over a given model of its base i.e. we will assume the spaces $X_1, \dots, X_n$ fixed. For a context of the form (**??**) and a given base model $(X_1, \dots, X_n)$ we will define a sequence of fibrations of the form

$$E(\Gamma) \to E(\Gamma_{\leq m-1}) \to \dots \to E(\Gamma_{\leq 1}) \to pt$$

such that the models of Γ over $(X_1, \dots, X_n)$ will be identified with the points of $E(\Gamma)$. The space $E(\Gamma_{\leq 1})$ will be the space obtained from $X_1, \dots, X_n$ by the construction corresponding to the expression $\mathbf{R}_1 = \mathbf{R_1}(T_1, \dots, T_n)$. Points of this space will then be in one to one correspondence with the models of $(T_1, \dots, T_n; c_1 : \mathbf{R}_1)$ over $(X_1, \dots, X_n)$ according to the rule that a point x_1 corresponds to the model M with $M(c_1) = x_1$. The fiber of the space space $E(\Gamma_{\leq 2})$ over $x_1 \in E(\Gamma_{\leq 1})$ is obtained from $X_1, \dots, X_n$ and x_1 by the construction corresponding to $\mathbf{R}_2 = \mathbf{R}_2(T_1, \dots, T_n; c_1)$ etc.

In order to make it work we will be doing inductively the following:

1. Assuming that $E(\Gamma)$ is constructed and $\mathbf{R}$ is a valid type expression in Γ we will construct a fibration $E(\Gamma, \mathbf{R}) \to E(\Gamma)$.
2. Assuming the same as above and in addition that $\mathbf{r}$ is a valid term expression of type $\mathbf{R}$ in Γ we will construct a section $s(\mathbf{r}) : E(\Gamma) \to E(\Gamma, \mathbf{R})$.

3. Assuming the same as above and in addition that $\mathbf{r}' : \mathbf{R}$ is another type expression which is convertible into $\mathbf{r}$ we will show that the sections $s(\mathbf{r})$ and $s(\mathbf{r}')$ coincide.

To define $E(\Gamma_{i+1})$ from $E(\Gamma_i)$ we set $E(\Gamma_{i+1}) = E(\Gamma, \mathbf{R}_{i+1})$.

In a context Γ of the form (**??**) the class of valid type expressions is defined in the following recursive way.

1. $T_1, \ldots, T_n$ are valid type expressions
2. If $\mathbf{R}$ is a valid type expression in Γ and $\mathbf{Q}$ is a valid type expression in $(\Gamma, y : \mathbf{R})$ then $\sum y : \mathbf{R}.\mathbf{Q}$ is a valid type expression in Γ.
3. If $\mathbf{R}$ is a valid type expression in Γ and $\mathbf{Q}$ is a valid type expression in $(\Gamma, y : \mathbf{R})$ then $\prod y : \mathbf{R}.\mathbf{Q}$ is a valid type expression in Γ.
4. If $\mathbf{Q}$ is a valid type expression in Γ and $\mathbf{y}_1, \mathbf{y}_2$ are valid term expressions of type $\mathbf{Q}$ in context Γ (see below) then $eq_{\mathbf{Q}}(\mathbf{y}_1, \mathbf{y}_2)$ is a valid type expression in Γ

In order to simplify the notation one writes $\Gamma \vdash \mathbf{R} : Type$ to signify that $\mathbf{R}$ is a valid type expression in the context Γ. One also writes $\Gamma \vdash \mathbf{r} : \mathbf{R}$ to signify that in the context Γ, $\mathbf{R}$ is a valid type expression and $\mathbf{r}$ is a valid term expression of type $\mathbf{R}$. The last three type forming rules described above take in this notation the following form

Let us describe the semantics of type constructors (1)-(4) i.e. given $E(\Gamma)$ and a type expression $\mathbf{S}$ formed according to one of these four rules we will describe the fibration $E(\Gamma, \mathbf{S}) \to E(\Gamma)$.

1. In the case $\mathbf{S} = T_i$ we set $E(\Gamma, S) = E(\Gamma) \times X_i$.
2. In the case $\mathbf{S} = \sum y : \mathbf{R}.\mathbf{Q}$ we proceed as follows. Since $\mathbf{R}$ is a valid type expression in Γ we have by the inductive assumption a fibration $p_{\mathbf{R}} : E(\Gamma, \mathbf{R}) \to E(\Gamma)$. Since $\mathbf{Q}$ is a valid type expression in $(\Gamma, y : \mathbf{R})$ we further have a fibration $p_{\mathbf{Q}} : E(\Gamma, \mathbf{R}, \mathbf{Q}) \to E(\Gamma, \mathbf{R})$. We set
$$E(\Gamma, \mathbf{S}) = E(\Gamma, \mathbf{R}, \mathbf{Q})$$
with the projection to $E(\Gamma)$ being the composition of the projections $p_{\mathbf{Q}}$ and $p_{\mathbf{R}}$.
3. In the case $\mathbf{S} = \prod y : \mathbf{R}.\mathbf{Q}$ we proceed as follows. As in the previous case we have two fibrations:
$$E(\Gamma, \mathbf{R}, \mathbf{Q}) \overset{p_{\mathbf{Q}}}{\to} E(\Gamma, \mathbf{R}) \overset{p_{\mathbf{R}}}{\to} E(\Gamma)$$
We define $E(\Gamma, \mathbf{S})$ to be the fibration over $E(\Gamma)$ whose fiber over $x \in E(\Gamma)$ is the space of (continuous) sections of the fibration
$$p_{\mathbf{Q}}^{-1} p_{\mathbf{R}}^{-1}(x) \to p_{\mathbf{R}}^{-1}(x).$$
More formally, this space is defined by the universal property saying that for any Z over $E(\Gamma)$ the set of maps from Z to $E(\Gamma, \mathbf{S})$ over $E(\Gamma)$ is naturally isomorphic to the set of maps from $Z \times_{E(\Gamma)} E(\Gamma, \mathbf{R})$ to $E(\Gamma, \mathbf{R}, \mathbf{Q})$ over $E(\Gamma, \mathbf{R})$.

4. In the case $\mathbf{S} = eq_{\mathbf{Q}}(\mathbf{y}_1, \mathbf{y}_2)$ we proceed as follows. Since $\mathbf{Q}$ is a valid type expression in Γ we have a fibration $p_{\mathbf{Q}} : E(\Gamma, \mathbf{Q}) \to E(\Gamma)$. Since $\mathbf{y}_i$, $i = 1, 2$ are valid term expressions of type $\mathbf{Q}$ in Γ we have two sections
$$s_i = s(\mathbf{y}_i) : E(\Gamma) \to E(\Gamma, \mathbf{Q}).$$
We define $E(\Gamma, \mathbf{S})$ to be the fibration over $E(\Gamma)$ whose fiber over $x \in E(\Gamma)$ is the space of (continuous) paths from $s_1(x)$ to $s_2(x)$ in the fiber $p_{\mathbf{Q}}^{-1}(x)$.

Let us describe now the semantics of our term constructors. Given a context Γ, a type expression $\mathbf{S}$ in Γ and a term expression $\mathbf{s}$ of type $\mathbf{R}$ which is produced by one of our term constructors we need to describe the corresponding section $s(\mathbf{s})$ of the fibration $E(\Gamma, \mathbf{R}) \to E(\Gamma)$.

1. In the case $\mathbf{R} = \mathbf{R}_i$ and $\mathbf{s} = c_i$ we proceed as follows. Since $\mathbf{R}_i$ is independent on c_j for $j > i - 1$ we have a pull-back square
$$\begin{tikzcd} E(\Gamma, \mathbf{R}_i) \arrow[r] \arrow[d] & E(\Gamma_{\leq i}, \mathbf{R}_i) \arrow[r] \arrow[d] & E(\Gamma_{<i}, \mathbf{R}_i) \arrow[d] \\ E(\Gamma) \arrow[r] & E(\Gamma_{\leq i}) \arrow[r] & E(\Gamma_{<i}) \end{tikzcd}$$
where $\Gamma_{\leq i} = (T_1, \ldots, T_n; c_1 : \mathbf{R}_1, \ldots, c_i : \mathbf{R}_i)$. Since $E(\Gamma_{\leq i}) = E(\Gamma_{<i}, \mathbf{R}_i)$ there is the diagonal section of the middle vertical arrows which pulls back to a section of the left hand side vertical arrow. We define $s(\mathbf{s})$ as this pull-back.

2. The sum constructors:

   (a) In the case $\mathbf{s} = \langle y, z \rangle$ we proceed as follows. We have the fiber square:
$$\begin{tikzcd} E(\Gamma, \mathbf{R}, \mathbf{Q}, \sum y : \mathbf{R}.Q) \arrow[r] \arrow[d] & E(\Gamma, \sum y : \mathbf{R}.Q) \arrow[d] \\ E(\Gamma, \mathbf{R}, \mathbf{Q}) \arrow[r] & E(\Gamma) \end{tikzcd}$$
   By definition $E(\Gamma, \sum y : \mathbf{R}.Q) = E(\Gamma, \mathbf{R}, \mathbf{Q})$. Therefore we again have the diagonal section Δ of the left hand side arrow. We set $s(\mathbf{s}) = \Delta$.

   (b) In the case $\mathbf{s} = (\pi \mathbf{u})$ and $\mathbf{s}' = (\pi' \mathbf{u})$ we proceed as follows. We have fibrations
$$E(\Gamma, \sum y : \mathbf{R}.\mathbf{Q}) = E(\Gamma, \mathbf{R}, \mathbf{Q}) \overset{p_{\mathbf{Q}}}{\to} E(\Gamma, \mathbf{R}) \overset{p_{\mathbf{R}}}{\to} E(\Gamma)$$
   and a section $s(\mathbf{u}) : E(\Gamma) \to E(\Gamma, \mathbf{R}, \mathbf{Q})$. We set $s(\pi \mathbf{u})$ to be the composition $s(\pi \mathbf{u}) = p_{\mathbf{Q}} s(\mathbf{u})$. Then we get a pull-back square
$$\begin{tikzcd} E(\Gamma, \mathbf{Q}[\pi \mathbf{u}/y]) \arrow[r] \arrow[d] & E(\Gamma, \mathbf{R}, \mathbf{Q}) \arrow[d] \\ E(\Gamma) \arrow[r, "s(\pi \mathbf{u})"] & E(\Gamma, \mathbf{R}) \end{tikzcd}$$
   and we set $s(\pi' \mathbf{u})$ to be the map $E(\Gamma) \to E(\Gamma, \mathbf{Q}[\pi \mathbf{u}/y])$ which is the product of Id and $s(\mathbf{u})$.

3. The product constructors:

   (a) In the case $\mathbf{s} = \lambda y : \mathbf{R}\mathbf{q}$ we proceed as follows. We have a section $s(\mathbf{q})$ of the fibration

   $$p_{\mathbf{Q}} : E(\Gamma, \mathbf{R}, \mathbf{Q}) \to E(\Gamma, \mathbf{R})$$

   We need a section of the fibration

   $$[\mathbf{prodsem1}] E(\Gamma, \prod y : \mathbf{R}.\mathbf{Q}) \to E(\Gamma) \tag{19}$$

   By definition the fiber of the later projection over $x \in E(\Gamma)$ is the space of sections of the fibration $p_{\mathbf{Q}}^{-1} p_{\mathbf{R}}^{-1}(x) \to p_{\mathbf{R}}^{-1}(x)$. Using this description or the universal property of $E(\Gamma, \prod y : \mathbf{R}.\mathbf{Q})$ one concludes that sections of (19) are in natural one to one correspondence with sections of $p_{\mathbf{Q}}$. We define $s(\mathbf{s})$ as the image of $s(\mathbf{q})$ under this correspondence.

   (b) In the case $\mathbf{s} = \mathbf{f}\,\mathbf{r}$ we proceed as follows. We have sections $s(\mathbf{f})$ and $s(\mathbf{r})$ of the fibrations

   $$E(\Gamma, \prod y : \mathbf{R}\mathbf{Q}) \to E(\Gamma)$$

   and

   $$E(\Gamma, \mathbf{R}) \to E(\Gamma)$$

   respectively. Observe that there is a pull-back square

   $$[\mathbf{anothersq}] \quad \begin{tikzcd} E(\Gamma, \mathbf{Q}[\mathbf{r}/y]) \arrow[r] \arrow[d] & E(\Gamma, \mathbf{R}, \mathbf{Q}) \arrow[d] \\ E(\Gamma) \arrow[r, "s(\mathbf{r})"] & E(\Gamma, \mathbf{R}) \end{tikzcd} \tag{20}$$

   and we need to get a section of the left hand side vertical arrow.

   By definition of $E(\Gamma, \prod y : \mathbf{R}\mathbf{Q})$ the section $s(\mathbf{f})$ takes a point $x \in E(\Gamma)$ to a section of $p_{\mathbf{Q}}^{-1} p_{\mathbf{R}}^{-1}(x) \to p_{\mathbf{R}}^{-1}(x)$. Evaluating this section on $s(\mathbf{r})(x)$ we get an element of $p_{\mathbf{Q}}^{-1} p_{\mathbf{R}}^{-1}(x)$. This means we got a map $g : x \mapsto s(\mathbf{f})(s(\mathbf{r})(x))$ from $E(\Gamma)$ to $E(\Gamma, \mathbf{R}, \mathbf{Q})$. By construction this map has the property $p_{\mathbf{Q}} \circ g = s(\mathbf{r})$ and therefore defines a section of the left hand side arrow in (20). We define $s(\mathbf{s})$ as this section.

4. The equality constructors.

   (a) In the case $\mathbf{s} = \mathbf{r}[\phi/v]$ we proceed as follows. We have fibrations $p_{\mathbf{R}} : E(\Gamma, \mathbf{R}) \to E(\Gamma)$ and $p_{\mathbf{Q}} : E(\Gamma, \mathbf{Q}) \to E(\Gamma)$. The term expression $\mathbf{r}$ defines a map $E(\Gamma, \mathbf{Q}) \to E(\Gamma, \mathbf{R})$ over $E(\Gamma)$. By abuse of notation we will denote this map by $s(\mathbf{r})$. We further have two sections $s(\mathbf{q}_i)$, $i = 1, 2$ of $p_{\mathbf{Q}}$. We need to get a section of

   $$[\mathbf{needsec}] E(\Gamma, eq_{\mathbf{Q}}(\mathbf{q}_1, \mathbf{q}_2), eq_{\mathbf{R}}(\mathbf{r}[\mathbf{q}_1/v], \mathbf{r}[\mathbf{q}_2/v])) \to E(\Gamma, eq_{\mathbf{Q}}(\mathbf{q}_1, \mathbf{q}_2)) \tag{21}$$

   The map $s(\mathbf{r})$ defines a map on the spaces of paths

   $$E(\Gamma, eq_{\mathbf{Q}}(\mathbf{q}_1, \mathbf{q}_2) \to E(\Gamma, eq_{\mathbf{R}}(\mathbf{r}[\mathbf{q}_1/v], \mathbf{r}[\mathbf{q}_2/v]))$$

   over $E(\Gamma)$. Such maps are in one to one correspondence with sections of (21).

(b) In the case $\mathbf{s} = c(\phi, \psi)$ we proceed as follows. We have a fibration $p_{\mathbf{R}} : E(\Gamma, \mathbf{R}) \to E(\Gamma)$. Since everything will be happening fiber by fiber over $E(\Gamma)$ let us fix a point $p \in E(\Gamma)$ and look at fibers over this point. Let $L = p_{\mathbf{R}}^{-1}(p)$. The fiber of $E(\Gamma, x, y, z : \mathbf{R}, \phi : eq_{\mathbf{R}}(x, y), \psi : eq_{\mathbf{R}}(x, z))$ over p is the fibration F over $L \times L \times L$ whose fiber over (l_1, l_2, l_3) is the space of pairs γ_{12}, γ_{13} where γ_{12} is a path from l_1 to l_2 and γ_{13} is a path from l_1 to l_3. The fiber over p of the fibration

$$\begin{array}{c} E(\Gamma, x, y, z : \mathbf{R}, \phi : eq_{\mathbf{R}}(x, y), \psi : eq_{\mathbf{R}}(x, z), eq_{\sum u:\mathbf{R}.eq_{\mathbf{R}}(x,u)}(\langle y, \phi\rangle, \langle z, \psi\rangle)) \\ \downarrow \\ E(\Gamma, x, y, z : \mathbf{R}, \phi : eq_{\mathbf{R}}(x, y), \psi : eq_{\mathbf{R}}(x, z)) \end{array}$$

is a fibration $w : E \to F$ such that the fiber of w over a point $(l_1, l_2, l_3, \gamma_{12}, \gamma_{13})$ of F is the space of paths from γ_{12} to γ_{13} in the space of paths in X starting in l_1. To construct $s(\mathbf{s})$ we need to get a section of w. In other words for any pair of paths starting in l_1 we need to assign in a continuous way a path from the first path to the second in the space of paths starting in l_1. There are clearly many way of doing this and we pick any one. Intuitively we can say that we first contract the first path to the source point l_1 and then take the inverse to the contraction of the second path to l_1.

(c) In the case $\mathbf{s} = \epsilon(\phi)$ we use any homotopy which relates the previous construction to identity when the first of two passes is degenerate.

(d) In the case $\mathbf{s} = ex(\mathbf{e})$ we proceed as follows. Everything again happens fiber by fiber over $E(\Gamma)$. We fix a point $p \in E(\Gamma)$ and look at fibers over this point. Let $w : F \to L$ be the fiber over p of the fibration

$$E(\Gamma, \mathbf{R}, \mathbf{Q}) \to E(\Gamma, \mathbf{R}).$$

We have two sections s_1, s_2 of this fibration (which are fibers of $s(\mathbf{r}_i)$, $i = 1, 2$) and the section γ (which is the fiber of $s(bfe)$) of the fibration of paths from s_1 to s_2 i.e. a map which assigns to any $l \in L$ a path $\gamma(l)$ from $s_1(l)$ to $s_2(l)$ in the fiber $w^{-1}(l)$. We need to construct a point in the fiber M of

$$E(\Gamma, eq_{\prod y:\mathbf{R}.\mathbf{Q}}(\lambda y : \mathbf{R}.\mathbf{r}_1, \lambda y : \mathbf{R}.\mathbf{r}_2)) \to E(\Gamma)$$

over p. By construction this fiber is the space of paths from the section s_1 to the section s_2 in the space of sections of w. This space is the same as the space of sections of the space of paths where we already have a point γ.

I am not ready to describe all the conversions in the system yet. It is clear that β-conversions both for the sum and for the product should be included. There may also be a need for a conversion related to the functional extensionality rule. There does not appear to a need in any other conversions.

Example 4.1 [**function**] For the context $\Gamma = (T_1, T_2 : Type; f : T_1 \to T_2)$ the space $E(\Gamma)$ corresponding to the given model X_1, X_2 of (T_1, T_2) is the space $\underline{Hom}(X_1, X_2)$ of continuous maps from X_1 to X_2. An individual model of Γ is therefore, as one would expect, a triple $(X_1, X_2; f : X_1 \to X_2)$.

5 Levels

We will not be entirely formal both because the complete formality is only possible to achieve in a computer implementation and because it is important to develop some sort of semi-formal language which then can be used as a higher level language of the implementation. We will usually write

$$\{y : \mathbf{R}, z : \mathbf{Q}\}$$

instead of $\sum y : \mathbf{R}.\mathbf{Q}$ and similarly use notations such as

$$\{y : \mathbf{R}, z : \mathbf{Q}, u : \mathbf{S}\}$$

for iterated sums (in this case $\sum y : \mathbf{R}. \sum z : \mathbf{Q}.\mathbf{S}$). When we do use sums we may write $\sum y, y' : \mathbf{R}$ instead of $\sum y : \mathbf{R}. \sum y' : \mathbf{R}$ and similarly for products. To make notation shorter we will sometimes write contexts in the form $(T_1, \ldots, T_n; \ldots)$ instead of $(T_1, \ldots, T_n : Type; \ldots)$

We start with the most important type expression $\mathbf{Contr}(T)$. It is defined in the context $T : Type$ by the formula:

$$\mathbf{Contr}(T) = \{t_0 : T, \phi : eq_{T \to T}(\lambda t : T.t_0, \lambda t : T.t)\}$$

Models of $(T; a : \mathbf{Contr}(T))$ are contractible spaces. Indeed, a model $Contr(X)$ of $\mathbf{Cont}(T)$ over a model X of T is the space of pairs (x_0, h) where x_0 is a point of X and h is a homotopy from the map $X \to X$ which is identically equal to x_0 to the identity map. Clearly a model of $(T; a : \mathbf{Contr}(T))$ is a space with a point and a contraction to this point. This is the same as a contractible space since for any X such that $Cont(X)$ is not empty it is contractible. We will prove the last fact on the level of the type system in Theorem 5.3 i.e. we will show that in the context $(T; a : \mathbf{Contr}(T))$ (I am using an abbreviated expression for the context) there is a term expression $\mathbf{c}$ of type $\mathbf{Contr}(\mathbf{Contr}(T))$. This theorem provides a good demonstration of how one uses the constructors and conversions of the previous section to translate homotopy-theoretic arguments into the formal language of our type system.

Define now by induction type expressions $\mathbf{Lv}_n$ for all $n \geq -1$. We set:

$$\mathbf{Lv}_{-1}(T) = \mathbf{Contr}(T)$$

$$\mathbf{Lv}_n(T) = \prod t, t' : T.\mathbf{Lv}_{n-1}(eq_T(t, t'))$$

We already know that a model of $(T; a : \mathbf{Lv}_{-1}(T))$ is a contractible space.

Lemma 5.1 *[**lvmodels**] For $n \geq 0$ a model of $(T : Type; a : \mathbf{Lv}_n(T))$ is a space X such that for all $x \in X$ one has $\pi_i(X, x) = 0$ for $i \geq n$. In particular, for $n = 0$ there are only two models the empty space and the contractible space.*

Proof: For $n = 0$ a model is a space X together with a point in

$$Lv_0(X) = \prod_{x, x' \in X} Contr(P(X; x, x'))$$

where $P(X; x, x')$ is the space of paths from x to x' in X. There are only two spaces for which such a point exist - the empty space and the contractible space. Note also that if the space $Lv_0(X)$ is non-empty it is contractible. Proceed now by induction on n assuming that for any X the space

$$Lv_n(X) = \prod_{x,x' \in X} Lv_{n-1}(P(X; x, x'))$$

is non-empty if and only if for all $x \in X$ one has $\pi_i(X, x) = 0$ for $i \geq n$ and in this case it is contractible. Consider $Lv_{n+1}(X)$. One can easily see that it is non-empty if and only if for any $x \in X$ the loop space $\Omega_{X,x}$ of X in x has the property Lv_n and that in this case it is contractible. This clearly implies the inductive step.

Before proving the basic properties of the level expressions on the system level we need to establish a few basic facts about equivalences. Let us start with the following notations. In the context $(T; x, y : T, a : eq_T(x, y))$ set

$$\mathbf{inv}(a) = \pi s(a, x) : eq_T(y, x)$$

Recall that our rule (15) allows us to write x for the identity equivalence from x to x. Recall further that π is the projection from a dependent sum to its index type. One verifies easily that for a model $(X; x, y \in X, \gamma \in P(X; x, y))$ of our context the model $inv(\gamma)$ of $\mathbf{inv}(a)$ is a path from y to x which represents up to homotopy the inverse to γ. In the context $(T; x, y, z : T, a : eq_T(x, y), b : eq_T(y, z))$ set

$$\mathbf{comp}(a, b) = \pi s(\mathbf{inv}(a), b) : eq_T(x, z)$$

Again one verifies easily that on models this corresponds to the composition of paths. The following lemma shows that identities are identities with respect to **comp** and **inv** is an inverse with respect to **comp**.

Lemma 5.2 *[***invcomp***]*

1. *There are term expression* $\mathbf{c}$*,* $\mathbf{c}'$ *such that*

$$T : Type; x, y : T, a : eq_T(x, y) \vdash \mathbf{c} : eq_{eq_T(x,y)}(a, \mathbf{comp}(x, a))$$

$$T : Type; x, y : T, a : eq_T(x, y) \vdash \mathbf{c}' : eq_{eq_T(x,y)}(a, \mathbf{comp}(a, y))$$

2. *There are term expressions* $\mathbf{c}$*,* $\mathbf{c}'$ *such that*

$$T : Type; x, y, z : T, a : eq_T(x, y), b : eq_T(y, z) \vdash \mathbf{c} : eq_{eq_T(x,y)}(a, \mathbf{comp}(\mathbf{comp}(a, b), \mathbf{inv}(b)))$$

$$T : Type; x, y, z : T, a : eq_T(x, y), b : eq_T(y, z) \vdash \mathbf{c}' : eq_{eq_T(x,y)}(b, \mathbf{comp}(\mathbf{inv}(a), \mathbf{comp}(a, b)))$$

Proof: Later.

We have already seen that on the model level the space $Lv_n(X)$ is of level 0 for any X. The following result proves this statement on the type system level.

Theorem 5.3 *[**main1**] For any $n \geq -1$ there exists a term expression* $\mathbf{c}$ *such that*

$$T : Type \vdash \mathbf{c} : \mathbf{Lv}_0(\mathbf{Lv}_n(T)).$$

Proof: The proof will be considered later.

On the model level it is obvious that any Lv_n space is Lv_{n+1} space. The following theorem asserts the same fact on the type system level.

Theorem 5.4 *[**main2**] For any $n \geq -1$ there exists a term expression* $\mathbf{c}$ *such that*

$$T : type; a : \mathbf{Lv}_n(T) \vdash \mathbf{c} : \mathbf{Lv}_{n+1}(T).$$

We shall say that a type expression $\mathbf{R}$ in a context Γ is of level n if there exists a term expression in the same context of type $\mathbf{Lv}_n(\mathbf{R})$.

Here is another useful construction. It asserts that a product of types of level n is itself of level n and that the product of types is contractible if and only if each type is contractible.

Proposition 5.5 *[**lvprod**] Let $\Gamma, y : \mathbf{R} \vdash \mathbf{Q} : Type$. Then for any $n \geq -1$ there exists is a term expression* $\mathbf{c}$ *such that*

$$\Gamma \vdash \mathbf{c} : (\prod y : \mathbf{R}.\mathbf{Lv}_n(\mathbf{Q})) \to \mathbf{Lv}_n(\prod y : \mathbf{R}.\mathbf{Q})$$

and if $n = -1$ then there is a term expression $\mathbf{d}$ *such that*

$$\Gamma \vdash \mathbf{d} : \mathbf{Lv}_{-1}(\prod y : \mathbf{R}.\mathbf{Q}) \to (\prod y : \mathbf{R}.\mathbf{Lv}_{-1}(\mathbf{Q})).$$

Proof: Later.

It is clear from Lemma 5.1 that models of types of level 0 are truth values (i.e. either empty or equivalent to a point). Form this point of view an addition to a context Γ of a constant c of type $\mathbf{R}$ where $\mathbf{R}$ is a type expression of level zero means that we add an axiom to Γ while an addition of a constant c of type whose level is greater than zero means that we add a structure.

Before describing the type expression $\mathbf{Un} = \mathbf{Un}(f)$ (defined in the context $(T_1, T_2; f : T_1 \to T_2)$) which corresponds to the univalence of f we need to do some preliminary work. We start with the following expression defined in $\Gamma = (T_1, T_2; f : T_1 \to T_2, v : T_2)$:

$$\mathbf{Fb}(f, v) = \{u : T_1, \phi : eq_{T_2}(f\, u, v)\}$$

we will further abbreviate $\mathbf{Fb}(f, v)$ to $f^{-1}(v)$. Note that for am model $(X_1, X_2; p : X_1 \to X_2, y : X_2)$ of our context the model of $f^{-1}(v)$ is the homotopy fiber of p over y. In the context $(\Gamma, v' : T_2, a : eq_T(v, v'), z : f^{-1}(v))$ define

$$\mathbf{ha}(f, a, z) = (\text{unpack } z \text{ as } \langle u, \phi\rangle \text{ in } \langle u, \mathbf{comp}(\phi, a)\rangle) : f^{-1}(v')$$

On the model level this construction gives us the usual action of the paths of the base on the fibers of a fibration.

In the context $(T_1, T_2; f : T_1 \to T_2)$ set

$$\mathbf{Eq}(f) = \prod v : T_2.\mathbf{Contr}(f^{-1}(v)).$$

One verifies easily that $\mathbf{Eq}(f)$ is a level 0 expression. Its model is non-empty if and only if the model of f is a homotopy equivalence. The following lemma show that the paths of the base act on the fibers by equivalences.

Lemma 5.6 *[**haeq**] There is a term expression* $\mathbf{c}$ *such that*

$$T_1, T_2 : Type; f : T_1 \to T_2, v, v' : T_2, a : eq_{T_2}(v, v') \vdash \mathbf{c} : Eq(\lambda z : f^{-1}(v).\mathbf{ha}(f, a, z))$$

Proof: Later.

In the basic context T_1, T_2 set:

$$\mathbf{Eq}(T_1, T_2) = \{f : T_1 \to T_2, e : \mathbf{Eq}(f)\}.$$

Since $\mathbf{Eq}(f)$ is a type expression of level 0, $\mathbf{Eq}(T_1, T_2)$ is in a sense a subtype of $T_1 \to T_2$. For a model (X_1, X_2) of the context the model $Eq(X_1, X_2)$ of this type is the subspace of homotopy equivalences in $\underline{Hom}(X_1, X_2)$. Note also that since $\mathbf{Contr}(T)$ projects to T and since $f^{-1}(v)$ projects to T_1 any member $\langle f, e\rangle$ of $\mathbf{Eq}(T_1, T_2)$ defines a member of $\prod v : T_2.T_1 = (T_1 \to T_2)$ which corresponds on the model level to the equivalence inverse to f.

In $(T_1, T_2; f : T_1 \to T_2, v, v' : T_2)$ we have

$$\mathbf{haeq}(f, v, v') = \lambda a : eq_{T_2}(v, v').\langle \lambda z : f^{-1}(v).\mathbf{ha}(f, a, z), \mathbf{c}\rangle : eq_{T_2}(v, v') \to \mathbf{Eq}(f^{-1}(v), f^{-1}(v'))$$

where $\mathbf{c}$ is the term expression of Lemma 5.6. This gives a mapping from paths to the equivalences between the homotopy fibers.

We can now translate the definition of a univalent map given in Section **??** into the type system. In the context $(T_1, T_2; f : T_1 \to T_2)$ set

$$[\mathbf{univdef}]\mathbf{Un}(f) = \prod v, v' : T_2.\mathbf{Eq}(\mathbf{haeq}(f, v, v')) \tag{22}$$

Clearly, $\mathbf{Un}(f)$ is a level 0 type which is non-empty iff for each pair $v, v' : T_2$ the map $\mathbf{haeq}(f)$ from the paths between v and v' to the equivalences between the corresponding homotopy fibers of f is an equivalence. Models of $(T_1, T_2; f : T_1 \to T_2, a : \mathbf{Un}(f))$ are exactly the univalent maps $p : X_1 \to X_2$ in the sense of Definition 0.2.

To complete the definition of a universe context we need to find a way to express in the system the condition that the class of types defined by a univalent map is closed under enough operations to serve as a universe where models of homotopy λ-calculus may be considered. Fix a context

$\Gamma = (\mathcal{U},\tilde{\mathcal{U}}; v : \tilde{\mathcal{U}} \to \mathcal{U})$ which we will eventually extend to create a universe context. We assume that this context is included into all the contexts considered below unless the opposite is explicitly mentioned. We also fix a model $p : \tilde{U} \to U$ of Γ.

Let $\mathbf{R}$ be a valid type expression in Γ and $\mathbf{S}$ be a valid type expression in the context $(\Gamma, y : \mathbf{R})$. Set

$$\mathbf{Class}(\mathbf{R}, \mathbf{S}, v) = \{f : \mathbf{R} \to \mathcal{U}, coh : \prod y : \mathbf{R}.\mathbf{Eq}(v^{-1}(f\,y), \mathbf{S})\}$$

Theorem 5.7 *[**main2**] For any* $\mathbf{R}$ *and* $\mathbf{S}$ *as above there is a term expression* $\mathbf{c}$ *such that*

$$a : \mathbf{Un}(v) \vdash \mathbf{c} : \mathbf{Lv}_0(\mathbf{Class}(\mathbf{R}, \mathbf{S}, v))$$

In other words for any univalent v *and any* $\mathbf{R}$, $\mathbf{S}$ *the type* $\mathbf{Class}(\mathbf{R}, \mathbf{S}, v)$ *is of level* 0.

Proof: Later.

Theorem 5.7 shows that for univalent v we may treat $\mathbf{Class}(\mathbf{R}, \mathbf{S}, v)$ as a condition on $\mathbf{R}$ and $\mathbf{S}$. Let us see what it means on the model level.

For a fixed model $p : \tilde{U} \to U$ of Γ the sequent $(\Gamma, y : \mathbf{R} \vdash \mathbf{S})$ defines a fibration $p_{\mathbf{S}} : E(\mathbf{R}, \mathbf{S}) \to E(\mathbf{R})$. The model of $\mathbf{Class}(\mathbf{R}, \mathbf{S}, v)$ is the space of all possible ways of inducing this fibration from p. The first component gives a map $E(\mathbf{R}) \to U$ and the second an equivalence of $\tilde{U} \times_U E(\mathbf{R})$ with $E(\mathbf{R}, \mathbf{S})$ over $E(\mathbf{R})$. In particular it is non-empty if and only if $p_{\mathbf{S}}$ can be obtained as a pull-back of p i.e. if and only if it is classifiable by p. Due to the universal property of univalent maps this is equivalent to the condition that the fibers of $p_{\mathbf{S}}$ belong to the set $A(p)$ of homotopy types occurring as fibers of p. Theorem 5.7 asserts that in this case the space of all possible ways to achieve such a classification it is contractible.

Set

$$\mathbf{Fam}(v) = \{u : \mathcal{U}, f : v^{-1}(u) \to \mathcal{U}\}$$

If a model of p is a univalent map $p : \tilde{U} \to U$ corresponding to a set of homotopy types $A = A(p)$ then a point in the model of $\mathbf{Fam}(p)$ is a homotopy type I from A together with a map to U i.e. it is a family of homotopy types from A indexed by I. It is further the same as a fibration $J \to I$ which is classifiable by p i.e. such that all its fibers are in A.

In $(\Gamma, y : \mathbf{Fam}(v))$ consider the type expressions:

$$\mathbf{Prod} = \prod z : v^{-1}(\pi\,y).v^{-1}(\pi' y\,z)$$

$$\mathbf{Sum} = \sum z : v^{-1}(\pi\,y).v^{-1}(\pi' y\,z)$$

where π and π' are the term constructors of (11) which take $\langle u, f\rangle$ to u and f respectively.

Set in Γ:

$$\mathbf{Cl\_prod} = \mathbf{Class}(\mathbf{Fam}(v), \mathbf{Prod}, v)$$

$$\mathbf{Cl\_sum} = \mathbf{Class}(\mathbf{Fam}(\upsilon), \mathbf{Sum}, \upsilon)$$

According to Theorem 5.7 in the presence of $a : \mathbf{Un}(\upsilon)$ these types are of level 0 and one verifies immediately that for a univalent model $p : \tilde{U} \to U$ of Γ the model of **Cl_Prod** (resp. **Cl_Sum**) is non-empty if and only if $A(p)$ is closed under products (resp. sums) of families. We also need to ensure that $A(p)$ is closed under the formation of the space of paths. On the level of models this means that the diagonal $\tilde{U} \to \tilde{U} \times_U \tilde{U}$ is classifiable i.e. that there is a homotopy pull-back square

$$\begin{tikzcd}
\tilde{U} \arrow[r] \arrow[d, "\Delta"'] & \tilde{U} \arrow[d, "p"] \\
\tilde{U} \times_U \tilde{U} \arrow[r, "Eq"] & U
\end{tikzcd}$$

In on the type system level this amounts to the inhabitation condition for a **Class** expression defined as follows. Set in Γ:

$$\mathbf{Path} = \{u : U, x : \upsilon^{-1}(u), x' : \upsilon^{-1}(u)\}$$

Set in $(\Gamma, y : \mathbf{Path})$:

$$\mathbf{Diag} = eq_{\upsilon^{-1}(\pi\, y)}(\pi' y, \pi'' y)$$

where $\pi\langle u, x, x'\rangle = u$, $\pi'\langle u, x, x'\rangle = x$, $\pi''\langle u, x, x'\rangle = x'$. Finally set in Γ:

$$\mathbf{Cl\_eq} = \mathbf{Class}(\mathbf{Path}(\upsilon), \mathbf{Diag}, \upsilon).$$

We can now define the standard universe context as follows.

Definition 5.8 *[***standuniv***] The standard universe context Ω is given by:*

$$\Omega = (\tilde{\mathcal{U}}, \mathcal{U} : Type; \upsilon : \tilde{\mathcal{U}} \to \mathcal{U}, a_{prod} : \mathbf{Cl\_prod}, a_{sum} : \mathbf{Cl\_sum}, a_{eq} : \mathbf{CL\_eq}).$$

It is my current understanding that this indeed defines a universe context i.e. that we may construct vertical models of any context Γ in Ω. In particular no further structures or axioms are needed in order to deal with term constructors and conversions. Let me describe now what I mean by that in detail.

Mention: not all $(\mathbf{R}, \mathbf{S})$ are classifiable e.g. $(\mathcal{U}, \mathcal{U})$ whose model is $U \times U \to U$ is not classifiable. Similarly, the pair whose model is $Fam(p) \to U$ is not classifiable or "large". If we make $U \times U \to U$ classifiable then only the trivial and the empty models survive (Th. Girard).

Mention: $\mathbf{Class}(\mathbf{S})$. Show that $\mathbf{Class}(\mathbf{R}, \mathbf{S})$ is equivalent to $\prod y : \mathbf{R}.\mathbf{Class}(\mathbf{S})$?

???? **Cl_eq** is actually provable? ????

6 Basic layer - generalized models

I do not have a good name at the moment for the full class of categories where models for contexts in the homotopy λ-calculus can take values. What is clear is that this class includes the categories of fibrant objects in Cartesian closed *Quillen* model categories. To illustrate the main ideas of our constructions we will use the category Top of (nice enough) topological spaces as the standard target category for our models.

7 Homotopy λ-calculus - logic layer

8 Homotopy λ-calculus - universe constructors

Going back to the homotopy λ-calculus we see that models of the context $(T_1, T_2 : Type; f : T_1 \to T_2, a : \mathbf{Un})$ are in one to one correspondence with sets of isomorphism classes of homotopy types. We will describe below (see Section **??**) type expressions $\mathbf{Lv}_n$ in the context $(T : Type)$ such that models of $(T : Type; a : \mathbf{Lv}_n)$ are exactly spaces with $\pi_i = 0$ for $i \geq n$. In particular models of $(T : Type; a : \mathbf{Lv}_{-1})$ are contractible spaces i.e. there is essentially only one model - the point, models of models of $(T : Type; a : \mathbf{Lv}_0)$ are truth values i.e. there are essentially two models the empty space and the point, models of $(T : Type; a : \mathbf{Lv}_1)$ are sets etc. Combining this with the expression $\mathbf{Un}$ we see that models of the context $(T_1, T_2 : Type; f : T_1 \to T_2, a : \mathbf{Un}, b : \mathbf{Lv}_{n+1}(T_2))$ are exactly subsets in the superset of the isomorphism classes of n-types. In particular, models of $(T_1, T_2 : Type; f : T_1 \to T_2, a : \mathbf{Un}, b : \mathbf{Lv}_2(T_2))$ are the subsets in the superset of isomorphism classes of sets.

So far the empty context of our type system is exactly that - empty. One can follow two approaches in the further development of the system. In one approach one would leave the empty context empty and set up a context which is rich enough to be able to encode mathematics in it. In another approach one introduces new type constructors which allow one to populate the empty context. I will take the first approach. The context I want to consider is the universe context of the preceding section together with a number of additional axioms. One can vary these additional axioms obtaining for example boolean or intuitionist universes. The main reason for choosing this approach over the other one is that there is no consensus over the exact properties a universe should posses. Some may want to work with weaker universes which therefore will have a wider class of external models and some with stronger ones. Because of this it seems to be a good idea to keep the type system itself as simple as possible and introduce the additional bells and whistles on the level of the universe context.

1. The basic universe structure i.e. $v : \tilde{\mathcal{U}} \to \mathcal{U}$, $a : \mathbf{Un}(v)$
2. The basic closeness axioms $a_{prod} : \mathbf{Cl\_prod}(v)$, $a_{sum} : \mathbf{Cl\_sum}(v)$, $a_{eq} : \mathbf{Cl\_eq}(v)$.
3. Define the empty type $\emptyset = \emptyset(v) = \prod y : \mathcal{U}.v^{-1}(y)$. Require $\emptyset$ to be small i.e. $a_\emptyset : \mathbf{Class}(\emptyset, v)$.
4. Define $\mathcal{U}_n = \{u : \mathcal{U}, a : \mathbf{Lv}_n(v^{-1}(u))\}$ i.e. $\mathcal{U}_n$ is the part of the universe span by n-types. Require $\mathcal{U}_0$ to be small i.e. $a_{prop} : \mathbf{Class}(\mathcal{U}_0, v)$. This is an analog of "impredicativity of Prop".
5. If desired add the boolean axiom $a_{bool} : \prod u : \mathcal{U}_0.(((v^{-1}(u) \to \emptyset) \to \emptyset) \to v^{-1}(u))$.
6. Impose an analog of Proposition 0.6 combined with the fact that any set of types is contained in a set of types closed under the sum, product and path operations. To do it add to our universe context the following. For each pair $u : \mathcal{U}$, $f : v^{-1}(u) \to \mathcal{U}$ fix

   $$univ(u, f) : \mathcal{U}, \quad \xi(u, f) : v^{-1}(univ(u, f)) \to \mathcal{U}.$$

   Set

   $$U(u, f) = v^{-1}(univ(u, f))$$

$$\tilde{U}(u,f) = \{z : U(u,f), v : \tilde{\mathcal{U}}, \phi : eq_{\mathcal{U}}(\xi(u,f)(z), v)\}$$

and $\upsilon(u,f) : \tilde{U}(u,f) \to U(u,f)$ let be the projection. Fix further:

$$a_0 : \mathbf{Un}(\upsilon(u,f))$$

$$a_1 : \mathbf{Cl\_prod}(\upsilon(u,f)), \quad a_2 : \mathbf{Cl\_sum}(\upsilon(u,f)), \quad a_3 : \mathbf{Cl\_eq}(\upsilon(u,f))$$

$$a_4 : \mathbf{Class}(\upsilon^{-1}(u), \upsilon^{-1}(f\,y), \upsilon(u,f)).$$

In human language it means that any family with small fibers and small base can be induced from a univalent family such that the corresponding class of fibers is closed under the standard operations while both the base and the fibers are again small.

We will sometimes call models of the kind discussed above external models. They are extremely useful at the stage of type system development. However, if one wants to use a type system to build foundations of mathematics one has to be able to speak of models defined entirely inside the type system. One may consider two types of such internal models - the horizontal models or interpretations and the vertical models. Both types of "models" exist in the first order logic. Interpretations of one theory in another assign sorts to the second theory to sorts of the first, formulas of the second theory to predicate symbols of the first and possibly complex function-like expressions of the second theory to functional symbols of the first. Interpretations are "horizontal" in a sense that they provide a correspondence between entities of the same kinds in two theories. Models on the other hand assign constants of the second theory to sorts of the first. For example $\mathbf{Z}/2$ may be treated as a model of group theory in the set theory which assigns to the only generating sort of group theory a constant corresponding to a set with two elements. In order to be able to speak about models of one theory in another the target theory should have special properties because otherwise it is unclear how to extend the correspondence from sorts to functional and predicate symbols. Since the notion of a map between constants is essential for the construction of such an extension and because there is no sensible way to say what properties or structures the target theory should have to enable one to discuss maps between constants, models in the first order logic are considered only with values in versions of the set theory.

The situation in the homotopy λ-calculus looks as follows. Horizontal models of one context in another can again be defined for any pair of contexts and correspond naturally to interpretations of one first order theory in another. As the adjective "horizontal" suggests such models assign a type expression in the target context to each generating type in the source context and similarly map term constants to appropriately typed term expressions.

Vertical models of a context can only be considered with values in special contexts which I will call *universe contexts*. Vertical models in a given universe context Ω are quantifiable i.e. for a context Γ the vertical models of Γ in Ω can be identified (in an appropriate sense) with terms of a type $\Gamma(\Omega)$ defined in Ω. For example, taking the context Gr which corresponds to the group theory (formulated as a first order theory) one gets for any universe context Ω a type $Gr(\Omega)$ whose members can be thought of as groups in the universe Ω.

In particular, any universe context has a type $\mathcal{U}(\Omega)$ which corresponds to vertical models of the basic context $\mathcal{U} = (T : Type)$ in Ω. The adjective "vertical" comes from the fact that vertical models assign to type expressions in Γ term expressions of type $\mathcal{U}(\Omega)$ in Ω. Since a type expression in Γ is the same as an interpretation of the basic context $T : Type$ in Γ this is a particular case of the fact that interpretations of Γ' in Γ defines functions from $\Omega(\Gamma)$ to $\Omega(\Gamma')$.

Let me explain the semantics of universe contexts with respect to models in Top. Any universe context has among its generating types two distinguished ones - the type $\mathcal{U}$ mentioned above and the type $\tilde{\mathcal{U}}$ corresponding to the models of the context $(T : Type; t : T)$. In addition there is a constant $v : \tilde{\mathcal{U}} \to \mathcal{U}$ which corresponds to the obvious interpretation of $(T : Type)$ in $(T : Type; t : T)$. In the most simple case there are no other generating types and all other generating constants are in an appropriate sense axioms rather than structures. Therefore, a model of Ω is given by two spaces $\tilde{U}$ and U and a continuous map $p : \tilde{U} \to U$ satisfying certain conditions. By the invariance principle $\tilde{U}$, U and p can be replaced by any homotopy equivalent triple $(\tilde{U}', U', p')$ and we may assume that p is a fibration.

The definition of univalent maps given above can be directly translated into the homotopy λ-calculus such that we get a type expression $\mathbf{Un}$ in the context $(T_1, T_2 : Type; f : T_1 \to T_2)$ with the property that models of the context $(T_1, T_2 : Type; f : T_1 \to T_2, a : \mathbf{Un})$ are exactly the univalent maps (see (22)).

9 Comparison with the Martin-Lof's type system

Let me discuss briefly the equality issues in the (intensional) Martin-Lof's type system which seems to be the closest relative of the homotopy λ-calculus among the known type systems. In this system equality shows up in two ways. There are so called equality judgments which are of the form

$$\mathbf{a} = \mathbf{b} : \mathbf{R}.$$

This judgement translates into the human language as - "$\mathbf{R}$ is a valid type expression, $\mathbf{a}$ and $\mathbf{b}$ are valid term expressions of type $\mathbf{R}$ and these expressions are definitionally equal". There are also equality types (originally called identity types) which are introduced in the same way as our types eq i.e. by the rule

$$\frac{\Gamma \vdash \mathbf{T} : Type}{\Gamma, x, y : \mathbf{T} \vdash eq_{\mathbf{T}}(x, y) : Type}$$

The validity of the equality judgement is known as definitional equality and the inhabitation of the equality type as propositional equality.

In our type system we do have analogs of both. Our equivalence types are clearly analogs of Martin-Lof's equality types. The definitional equality of two terms in our system means that these two terms are convertible into each other. While we do not have a special form of judgement reserved for it, the rules of our system show that the judgement

$$\mathbf{a} : eq_{\mathbf{R}}(\mathbf{a}, \mathbf{b}) \tag{23}$$

will be valid if and only if $\mathbf{R}$ is a valid type expression, $\mathbf{a}$ and $\mathbf{b}$ are valid term expressions of type $\mathbf{R}$ and these two term expressions can be converted to each other. A somewhat strange form of (23) is a consequence of the fact that in the current syntax of homotopy λ-calculus we use the same symbol for a term and the corresponding identity equivalence from it to itself. Convertibility is expected to have two main properties (which are at the moment conjectural):

1. Convertibility should be decidable
2. Convertibility should be context independent i.e. if two term expressions **a** and **b** are well defined in Γ then they are convertible to each other in Γ if and only if they are convertible to each other in Γ, Δ where Γ, Δ is an extension of Γ.

The first of these two properties implies that convertibility does not require a proof - it can be checked automatically. The second one implies that the convertibility can not be imposed by adding something to the context. Due to these two properties we do not treat definitional equality (= convertibility) as a part of the language but in a sense as a property of the language. As far as I understand the same can be said about the definitional equality in the intensional version of Martin-Lof's type theory except that there it is made more explicit through the equality judgments.

The properties of the equality types however differ considerably between the Martin-Lof's system and our system. First of all in Martin-Lof's system the equality (identity) types are actually *defined* as special instances of the inductive types. Since at the moment I do not understand how to introduce general inductive types into the homotopy λ-calculus I do not know whether or not something like this is possible there.

There is another approach to the definition of equality based on the Leibniz idea that two things are equal if they have the same properties. To make this into a formal definition one needs a distinguished type $Prop$ (discussed in []). Then one says that for $x, y : T$ one has $x =_L y$ if for all $P : T \to Prop$ one has $P(x) = P(y)$. The equality in $Prop$ is defined as equivalence of propositions. This can be made precise in any context which has $Prop$ and it seems it should be equivalent to the inhabitation of our equality type $eq_T(x, y)$. However even if we fix a Leibniz equality between x and y it does not give us enough information to replace x by y in constructions since it does not tell us anything about which equivalence to choose.

10 The leftovers

Remark 10.1 Doing foundations for a mathematician is a little like doing mathematics for a physicist. One has intuitive ideas of what should be right and what should be wrong but does not know exactly how to formalize these ideas.

The standard example of a type system is the (pure) typed λ-calculus. It is a very general but not a very rich type system. Consider for example the context $(T : Type, f : T \to T)$. A set-theoretic model M of this context is given by a set $X = M(T)$ together with an endomorphism $\phi = M(f)$. Suppose now that we want to define a context whose model is a set with an involution i.e. with an endomorphism ϕ such that $\phi^2 = 1$. In order to do so we need to be able to require that $f^2 = Id$. This "axiom" should be a part of the context so it has to be expressed in the form $e : Rexp(T, f)$ where $Rexp(T, f)$ is some type expression of T band f and e is a new variable. On the level of models it means that we must express the condition that $\phi^2 = Id$ in terms of non-emptiness of some set constructed in the language of λ-calculus out of X and ϕ. One observes that there is not way of doing this. In the classical λ-calculus one deals with this problem by adding the axiom $f^2 = Id$ as a new conversion rule. This clearly contradicts the philosophy outlined above which considers conversions to be a part of the type system and the type system to be fixed. In the human language analogy one would say that one does not modify the grammar of the language each time one wants to describe a new scene.

Another problem which one encounters in the λ-calculus is the following one. Let us again consider a set-theoretic model $M = (X, \phi)$ of the context $(T : Type, f : T \to T)$. The pair (X, ϕ) "generates" many other sets, for example one may consider the set of fixed points of ϕ i.e. the set $\{x \in X | \phi(x) = x\}$. There is however no way to produce a type R in our context such that $M(R) = \{x \in X | \phi(x) = x\}$ which shows that even with equations introduced on the conversion level the usual typed λ-calculus lacks enough constructors to emulate the most basic set-theoretic operations.

In oder to use a type system to formalize pure mathematics we need it to have, for any type of mathematical structure, a context whose models are exactly the structures of this type. Let us see what this meta-condition means. In order to even start thinking about it we have to answer two questions. What do we mean by a type of mathematical structure? Where our models take values? Since these questions have no mathematical sense outside of an already chosen formalization of mathematics we need to address them on the intuitive level.

We can deal with the first question by choosing a few basic types of structure and hoping that if we can find contexts to represent these types then we will also manage to find contexts to represent all other types. To get started let us consider for example finite sets. Thus we want to see what is required from a type system so that we can find a context Γ whose models are finite sets. In this case we probably should not reflect too much on where our models take values and consider models in sets.

Suppose now that we want to construct a theory (in a given type system) where we can conveniently express pure mathematics. Since the notion of a set is central to contemporary pure math there has to be a type S in this theory whose members we want to think of as sets. One can achieve this to some extend in classical type systems by creating a context which provides an encoding in terms of this type system of the Zermelo theory or some version of it (see e.g. [**?**] where this is done in the type system of Coq). Doing such a thing however recreates all the problems with the Zermelo approach to set theory the major one of which from my point of view is the fact that in this theory one can formulate and prove theorems about sets which are not invariant under isomorphisms of sets.

The version I am looking into right now is based on the idea that along with the usual dependent sum and dependent product there is a group of additional type constructors of the following form. First, for any finite (labeled) simplicial set B and any type T there is a new type $T(B)$. Second for any B as above, any simplicial subset A of B and any term $x : T(A)$ there is a type $T(B, A, x)$. One has term constructors and conversions to ensure among other things that $T(pt) = T$ and $T(A \coprod B) = T(A) \times T(B)$. The basic example is that for $< x, y >: T \times T = T(\{0\} \coprod \{1\})$ the type

$$eq(x, y) = T(\Delta^1, \partial\Delta^1, < x, y >)$$

is the type of equivalences between x and y in T. There are also term constructors and conversions which ensure that $T(B, A, x)$ is contravariantly functorial with respect to maps of pairs $(B, A) \to (B', A')$ and that $T(A)(A') = T(A \times A')$. Together with a sort of Kan axiom these structures allow one to define things like compositions of equivalences and prove that these compositions have good properties.

What is outlined is a language. As always in type theory a theory in this language is given by a context i.e. a series of declarations of generating types and terms of the form $T_1, ...T_n : Type$ and $c_1 : R_1, ..., c_n : R_n$ where R_i is a type expression of $T_1, .., T_n$ and $c_1, ...c_{i-1}$. Given a context Γ it makes sense to speak of models of the theory which Γ defines. Standard models in my approach take values not in the (a) category of sets but in the (a) homotopy category which one can think of as the category of ∞-groupoids.

It can still be defined for members of types but is not reflexive unless the type is of level 1 i.e. is mapped to a set (as opposed to a general infinity groupoid) by any model.

It is crucial to understand what we mean by a model. We will distinguish three kinds of models: external models, horizontal (internal) models and vertical (internal) models.

If one wants to use a type system to build foundations of mathematics external models can only be used for illustrative purposes. Indeed, one of the major problems in any approach to foundations is to formalize the notion of a set and before speaking about sets in the definition of a model or in any other context we need to say what a set is first. Therefore on the formal level we can only consider models of one context in another. As it turns out there are two possible notions of such models. I shall call them horizontal models and vertical models. Horizontal models correspond to the logical notion of interpretation while vertical models correspond to models proper.

Suppose first that I have a set-theoretic model of a context $\Gamma = (T : Type, t : R(T))$ in the intuitive sense outlined above. How to translate this model into some structure defined entirely in the framework of our type system? First we should choose a formalization of set theory in our system. It means that we have to define a context *SetTheory* whose set-theoretic models are set-theoretic models of set theory. In particular there should be a type *Sets* in *SetTheory* whose members we think of as sets and enough structures on this type to emulate the usual operations with sets. Our intuitive model M assigned a set X to T and an element x in $R(X)$ to t where $R(X)$ refers to the set obtained by applying the type constructor $R(-)$ to the set X according to some procedure for doing so. Hence, the formal version of M should assign a member X of *Sets* to T and "an element x of $R(X)$" to t. In order to make sense of the part of the sentence in the quotes we should be able to do two things. First we should be able to define a new member $r(X)$ of sets by translating somehow the type constructor R into a term constructor r for terms of *Sets*. Second, we should be able to assign a type $\tau(A)$ to each member A of *Sets* in a manner compatible with our constructor translation. Then the quoted phrase above can be replaced by "and a member x of the type $\tau(r(X))$".

This will be called a (slanted) model of Γ in *Sets*. The name slanted comes from the fact that our model assigns a term to a type. Horizontal models discussed above assign a type to a type. The key advantage of slanted models is that they are classifiable i.e. for any context Γ and any universe Ω the set of all models of Γ in Ω can be identified with the set of all terms of a type $\Gamma(\Omega)$ which is defined in the context Ω. For example, if Gr is the context encoding the notion of a group then $Gr(\Omega)$ will be the type of groups in Ω.

To describe the second problem suppose that we want to have a context (theory) useful for the formalization of our intuitive dealings with finite sets. In particular this means that we want to have a type S in this context whose members we want to think of as finite sets.

The next thing to understand is what happens given $(\Gamma, v : Q \vdash R(v) : Type)$ and $(\Gamma, \phi : eq_Q(x,y))$. It is clear that ultimately one should have then $R(\phi) : Eq_{Type}(R(x), R(y))$. The question is can we then provide all necessary conversions in a concise way. $Eq(T_1, T_2)$ can be defined as:

$$[\mathbf{eqdef1}]\{f : T_1 \to T_2, b : T_2 \to T_1, \phi : eq_{T_1 \to T_1}(Id, bf), \psi : eq_{T_2 \to T_2}(Id, fb), \alpha : eq_{eq_{T_1 \to T_2}(f, fbf)}(\psi * f, f * \phi)\} \tag{24}$$

Another approach is to define $Eq(T_1, T_2)$ as having infinitely many components. One could also define it through the use of $\exists$ but this looks like bad idea since $\exists$ normally should appear in the theory much later (together with $\emptyset$). Let's see if we can rewrite the definition (24) more explicitly i.e. without so far undefined compositions. Instead of bf we have to write $\lambda x : T_1.bfx$, instead of fb should write $\lambda y : T_2.fby$. Instead of fbf should write $\lambda x : T_1.fbfx$. So far so good. What should we write instead of $\psi * f$? Looks like $\lambda x : T_1.\psi fx$. This works by our rule (15). According to this rule the expression $\lambda x : T_1.\psi fx$ lies in

$$eq_{T_1 \to T_1}(\lambda x : T_1.(\lambda y : T_2.y)fx, \lambda x : T_1.(\lambda y : T_2.fby)fx)$$

which through β-conversion will give us

$$eq_{T_1 \to T_1}(\lambda x : T_1.fx, \lambda x : T_1.fbfx).$$

Good. Now $f * \phi$. This seems to be $\lambda x : T_1.f\phi x$. It lies in

$$eq_{T_1 \to T_1}(\lambda x : T_1.f(\lambda x' : T_1.x')x, \lambda x : T_1.f(\lambda x' : T_1.bfx')x)$$

It is fine provided we know that $\lambda x : T_1.f(\lambda x' : T_1.x')x : T_1 \to T_1$. Do we? Looks OK. Summarizing:

$$\psi * f = \lambda x : T_1.\psi fx$$

$$f * \phi = \lambda x : T_1.f\phi x.$$

Going back to what happens given $(\Gamma, v : Q \vdash R(v) : Type)$ and $(\Gamma, \phi : eq_Q(x,y))$. What if we simply require that $R(\phi) : R(x) \to R(y)$. Can we then construct $R(y) \to R(x)$ and the rest of the equivalence structure?

We will most likely need to introduce the inverses for equivalences first. E.g. lets state the rule

$$[\mathbf{inv1}]\frac{\Gamma \vdash \phi : eq_Q(x,y)}{\Gamma \vdash \phi^{-1} : eq_Q(y,x), \xi : eq_{eq_Q(x,x)}(\phi^{-1}\phi, Id)} \tag{25}$$

This should be enough for everything. First need to formulate the rule properly i.e. expand/explain $\phi^{-1}\phi$. In general given $\phi : eq_Q(x,y)$, $\psi : eq_Q(y,z)$ what is $\psi\phi$? First it seems that we can do even better with (25) by requiring both left and right inverses with the corresponding homotopies and providing no relation between the two. Of course one can be constructed from another but may be more notationally convenient to have both? Back to the composition $\psi\phi$. This composition is obtained from our "$R(\phi)$" rule. Note first that we should be able to write

$$eq(\phi, y) : eq(x,y) \to eq(y,y)$$

$$eq(\phi, x) : eq(x,x) \to eq(y,x)$$

$$eq(x,\phi) : eq(x,x) \to eq(x,y)$$

$$eq(y,\phi) : eq(y,x) \to eq(y,y).$$

Clearly, our idea is that $eq(\phi, x)Id_x = \phi^{-1}$. Strange that we are getting only one inverse. For ϕ, ψ we have:

$$eq(x,\psi) : eq(x,y) \to eq(x,z)$$

and $\psi\phi := eq(x,\psi)\phi$.

Sum associated term expressions:

1. if Q is a valid type expression in Γ and S is a valid type expression in $(\Gamma, y : Q)$ if further $l1$ is a valid term expression of type Q in Γ and $l2$ is a valid term expression of type S in $(\Gamma, y : Q)$ then $\langle l1, l2\rangle$ is a valid type expression in Γ of type $\sum_{y:Q} S$.