Notes on homotopy λ-calculus

Vladimir Voevodsky

Started Jan. 18, Feb. 11, 2006

Contents

1 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):

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

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.

2 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.

1 Univalent maps

We fix a universe $\mathcal{U}$ and consider the homotopy category $H = H(\mathcal{U})$ of spaces (or simplicial sets) in $\mathcal{U}$. Let us recall first recalling the "levels" structure on the homotopy category in a convenient for us form.

Definition 1.1 *[**levels**] Define the level of a homotopy type inductively as follows:*

1. *X is of level -2 iff X is contractible,*

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

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 $n-1$.

By definition there is only one type of level -2 - the point. One can also see easily that there are only two types of level -1 namely $\emptyset$ and pt. For a non-negative n a type of level n is the same as an n-type in the usual homotopy-theoretic sense i.e. one has:

Lemma 1.2 *[levelsob] A space X is of level $n \geq 0$ iff for all $x \in X$ one has $\pi_i(X, x) = pt$ for $i \geq n+1$.*

In particular equivalence classes of 0-types are in one to one correspondence with the isomorphism classes of sets in $\mathcal{U}$, equivalence classes of 1-types are in one to one correspondence with the equivalence classes of groupoids in $\mathcal{U}$ and equivalence classes of n-types are in one to one correspondence with the equivalence classes of n-groupoids in $\mathcal{U}$ if the later notion is properly defined (see e.g. [**?**]).

Let us extend the notion of a level from types to maps of types as follows.

Definition 1.3 *[levelmap] A map $f : Y \to X$ is called a map of level $n \geq -2$ if for any $x \in X$ the homotopy fiber $f^{-1}(x)$ is a type of level n.*

Examples:

1. A map is of level -2 iff it is an equivalence.
2. A map is of level -1 iff it is equivalent to a map of the form $Y \to Y \coprod X'$.
3. A map is of level 0 iff it is equivalent to an unramified covering.
4. Consider a functor $F : \mathcal{G}_1 \to \mathcal{G}_2$ between two groupoids and let $N(F) : N(\mathcal{G}_1) \to N(\mathcal{G}_2)$ be the corresponding map between the nerves. Then one has the following:
   (a) $N(F)$ is of level -2 iff F is an equivalence of categories,
   (b) $N(F)$ is of level -1 iff F is a full embedding,
   (c) $N(F)$ is of level 0 iff F is a faithful functor,
   (d) $N(F)$ is of level n for any F and any $n \geq 1$.

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

Definition 1.4 *Let $f : Y \to X$ and $f' : Y' \to X$ be two continuous maps and $g : Y \to Y'$ be a map over X. Then g is called a fiber-wise homotopy equivalence if for any $x \in X$ the corresponding map between the homotopy fibers of f and f' is a homotopy equivalence.*

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

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 1.5 *[univ] A map $u : E \to B$ is called univalent if the map $B \to E(u)$ is a fiber-wise homotopy equivalence over $B \times B$.*

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

Here are some examples of univalent maps:

1. There are only four univalent maps of level -1. They are $\emptyset \to \emptyset$, $\emptyset \to pt$, $pt \to pt$ and $pt \to pt \coprod pt$. Of these four the last one is the universal one since the other three are obtained from it by pull-back. These four maps are also the only univalent maps between types of level 0 i.e. between sets.
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 $n \geq 0$ 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.
4. A map $X \to pt$ is univalent iff X has no symmetries i.e. iff the space of homotopy auto-equivalences of X is contractible. For a group G it means that the map $BG \to pt$ is univalent if the center and the group of outer automorphisms of G are trivial. In particular, for $n > 2$ the map $BS_n \to pt$ is univalent.

Let us state some elementary properties of univalent maps.

Lemma 1.6 *[lvun] Consider a (homotopy) cartesian square*

$$\begin{array}{ccc} E' & \longrightarrow & E \\ {\scriptstyle u'}\downarrow & & \downarrow{\scriptstyle u} \\ B' & \xrightarrow{f} & B \end{array}$$

such that u is univalent. Then u' is univalent if and only if f is a map of level -1.

Proposition 1.7 *[**class**] If for a given univalent $u : E \to B$ and a given $f : Y \to X$ there exists a (homotopy) cartesian square of the form*

$$\begin{array}{ccc} Y & \longrightarrow & E \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle u} \\ X & \longrightarrow & B \end{array}$$

then such a square is unique up to an equivalence.

Theorem 1.8 *[**existun**] For any map $f : Y \to X$ there exists a unique homotopy cartesian square*

$$\begin{array}{ccc} Y & \longrightarrow & \widetilde{Un}(f) \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle un(f)} \\ X & \xrightarrow{g} & Un(f) \end{array}$$

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

Proof: We will only sketch a proof. Assume first that X is connected and let $x \in X$ be a point of X. Let further $F = f^{-1}(x)$ be the homotopy fiber of f over x. Let $M = Eq(F, F)$ be the topological monoid of homotopy auto-equivalences of F We set $Un(f) = BM$ and let $\widetilde{Un}(f) \to Un(f)$ to be the fibration with the fiber F defined by the obvious action of M on F. To get a map $X \to Un(f)$ note that $X \cong BM'$ where $M' = \Omega^1(X, x)$ is the topological monoid of (Moore) loops on X starting in x. If we use Moore paths in the construction of the homotopy fiber we get an action of M' on $f^{-1}(x)$ i.e. a homomorphism of monoids $M' \to M$ and therefore a map $X = BM' \to Un(f) = BM$. One verifies in the standard manner that this construction extends to a construction of a homotopy cartesian square of the required form.

For a general X let $X = \coprod_{a \in A} X_a$ be the decomposition of X into the union of its connected components. Choose a point $x_a \in X_a$ in each component and let F_a be the homotopy fibers of f over these points. Define an equivalence relation $\cong$ on A by the condition that $a \cong a'$ iff the fibers F_a and $F_{a'}$ are homotopy equivalent and let $A' \subset A$ be a subset which contains exactly one element of each equivalence class. Set

$$Un(f) = \coprod_{a \in A'} Un(f_a)$$

$$\widetilde{Un}(f) = \coprod_{a \in A'} \widetilde{Un}(f)$$

$$un(f) = \coprod_{a \in A'} un(f_a)$$

where f_a is the the map $f^{-1}(X_a) \to X_a$. One verifies immediately that $un(f)$ is univalent and that a square of the required for exists. It is also clear that the map $X \to Un(f)$ is surjective on π_0.

To prove the uniqueness part suppose that there are two homotopy cartesian squares of the required form with the univalent maps $u : E \to B$ and $u' : E' \to B'$. Consider the map $u \coprod u'$ and let us apply the existence part of the poof to it creating a univalent map $un(u \coprod u')$. We get a square:

$$\begin{array}{ccc} X & \xrightarrow{g} & B \\ {\scriptstyle g'}\downarrow & & \downarrow{\scriptstyle j} \\ B' & \xrightarrow[j']{} & Un(u \coprod u') \end{array}$$

The pull-back of $\widetilde{Un}(u \coprod u')$ to X with respect to the two paths in the square are equivalent to Y. Therefore, by Proposition 1.7 the square commutes up to homotopy. By our assumption g, g' and $j \coprod j'$ are surjective on π_0. Therefore, j and j' each are surjective on π_0. On the other hand by Lemma 1.6, j and j' are maps of level -1. We conclude that j and j' are equivalences.

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

Remark 1.9 We used a very non-canonical construction in the proof of the existence part of Theorem 1.8. The following canonical construction should work as well but it is less obvious. Let $f : Y \to X$ be a map. Then $E(f)$ together with the two maps to X, the section $X \to E(f)$ and the obvious composition of equivalences form an internal category $\mathcal{C}(f)$ in Top. Its nerve is a simplicial topological space. Let B be its geometric realization. Consider now the first projection $p : Y \times_X Y \to Y$ and its section Δ given by the diagonal. Let $\tilde{\mathcal{C}}(f)$ be the subcategory in $\mathcal{C}(p)$ which has the same objects (i.e. Y) and whose maps are equivalences preserving Δ. Let E be the geometric realization of the nerve of $\tilde{\mathcal{C}}(f)$ and let $u : E \to B$ be the map corresponding to the obvious functor. It is my understanding that $u \cong un(f)$.

Lemma 1.10 *[**levlev**] For a space X of level n the space $Un(X)$ is of level $n+1$.*

Proof: If $n = -2$ then X is contractible and $Un(X) = X$ is of level -2 and therefore of level -1. If $n = -1$ then $X = pt$ or $X = \emptyset$. We have already considered the first case. In the case $X = \emptyset$ we again have $Un(X) = pt$ since $\emptyset \to pt$ is univalent. Suppose that $n \geq 0$. The construction used in the proof of Theorem 1.8 shows that $Un(X) = BM$ where $M = Eq(X, X)$ is the topological monoid of homotopy auto-equivalences of X. Since this monoid is group-like we have $\Omega^1(Un(X)) \cong M$. It remains to note that $Eq(X, X)$ is a union of connected components of $End(X) = Hom(X, X)$ and that for a space X of level $n \geq 0$ and any space Y the space of continuous maps $Hom(Y, X)$ is of level n.

Definition 1.11 *[**classdef**] Let $u : E \to B$ be a univalent map and $f : Y \to X$ a map. We say that f is classifiable by u if a cartesian square as in Corollary 1.7 exists.*

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

Theorem 1.12 *[univ1] Given a set of isomorphism classes A in H there exists a unique univalent map $\omega(A) : \tilde{\Omega}(A) \to \Omega(A)$ such that $X \to pt$ is classifiable by $\omega(A)$ 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 $\Omega(A) = \coprod_{a\in A} Un(X_a)$, $\tilde{\Omega}(A) = \coprod_{a\in A}(\widetilde{Un}(X_a))$ and $\omega(A) = \coprod_{a\in A} un(X_a)$. If there are two such maps $\omega : \tilde{\Omega} \to \Omega$ and $\omega' : \tilde{\Omega}' \to \Omega'$ consider $\widetilde{Un}(\omega \coprod \omega') \to Un(\omega \coprod \omega')$. By Lemma 1.6 the map $\Omega \to Un(\omega \coprod \omega')$ is of level -1. On the other hand one can easily see that it is surjective on π_0. Therefore, it is an equivalence. The same holds for $\Omega' \to Un(\omega \coprod \omega')$.

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

$$\begin{array}{ccc} \tilde{\Omega}(A) & \longrightarrow & \tilde{\Omega}(A') \\ \downarrow & & \downarrow \\ \Omega(A) & \xrightarrow{i} & \Omega(A') \end{array}$$

where i is a map of level -1. Let A be a set of types which contains $\emptyset$ and pt. Let further $A_{\leq n}$ be the subset of types of level $n-1$ in A and $\Omega_{\leq n}(A) = \Omega(A_{\leq n})$ be the corresponding subspace in $\Omega(A)$. We get a diagram

$$\begin{array}{ccccccccc} \tilde{\Omega}_{\leq -1}(A) & \longrightarrow & \tilde{\Omega}_{\leq 0}(A) & \longrightarrow & \tilde{\Omega}_{\leq 1}(A) & \longrightarrow & \ldots & \longrightarrow & \tilde{\Omega}(A) \\ {\scriptstyle \omega_{\leq -1}}\downarrow & & {\scriptstyle \omega_{\leq 0}}\downarrow & & {\scriptstyle \omega_{\leq 1}}\downarrow & & & & {\scriptstyle \omega}\downarrow \\ \Omega_{\leq -1}(A) & \longrightarrow & \Omega_{\leq 0}(A) & \longrightarrow & \Omega_{\leq 1}(A) & \longrightarrow & \ldots & \longrightarrow & \Omega(A) \end{array}$$

where the squares are homotopy cartesian and horizontal maps are of level -1. The maps $\omega_{\leq n}$ for $n \leq 2$ can be described as follows:

1. $\tilde{\Omega}_{\leq -1}(A) = \Omega_{\leq -1}(A) = pt$ since there is only one type of level -2 and we have assumed that it lies in A,
2. $\tilde{\Omega}_{\leq 0}(A)$ is the one point set, $\Omega_{\leq 0}(A) = \{0, 1\}$ is the two point set, $\omega(A)_{\leq 0} : \tilde{\Omega}_{\leq 0} \to \Omega_{\leq 0}$ is the embedding whose image is 1,
3. $\tilde{\Omega}_{\leq 1}(A)$ is the nerve of the groupoid of pointed sets in A and their isomorphisms, $\Omega_{\leq 1}$ is the nerve of the groupoid of (free) sets in A and their isomorphisms and $\omega(A)_{\leq 1} : \tilde{\Omega}_{\leq 1} \to \Omega_{\leq 1}$ is the map corresponding to the forgetting functor from pointed sets to sets,
4. $\tilde{\Omega}_{\leq 2}(A)$ is the nerve of the 2-groupoid of pointed groupoids in A (i.e. pairs of a groupoid and an object in it) and their equivalences, $\Omega_{\leq 2}$ is the nerve of the 2-groupoid of (free) groupoids in A and their equivalences and $\omega(A)_{\leq 2} : \tilde{\Omega}_{\leq 2} \to \Omega_{\leq 2}$ is the map corresponding to the forgetting functor.

This kind of descriptions can be extended to higher n . In particular, $\Omega = \Omega(A)$ is the nerve of the ∞-groupoid of ∞-groupoids in A and their equivalences. However, since we do not have a commonly accepted theory of ∞-groupoids we will not concentrate on this interpretation.

2 Universes and universe maps

Let U be a set of homotopy types. In order for U to be a "universe" it should satisfy some closeness conditions. In this section we will consider the following conditions:

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

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

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

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

We will first describe some elementary corollaries of different combinations of these conditions and then show how to express Cl_{sum}, Cl_{prod}, Cl_{eq} and Cl_{un} as properties of the univalent map $\omega(U)$ corresponding to U. We say "a set in U" instead of "a type of level 0 in U".

Lemma 2.1 *[**cl1**] Let U be a set of types satisfying Cl_{sum}. Then one has:*

1. *for any $X, Y \in U$ one has $X \times Y \in U$,*
2. *for any set $I \in U$ and any family of types $(X_i)_{i \in I}$ such that all X_i are in U one has $\coprod_{i \in I} X_i \in U$.*

Proof: The space $X \times Y$ is the total space of the fibration $X \times Y \to X$ with the base and the fibers in U. The space $\coprod_{i \in I} X_i$ is the total space of the fibration $\coprod_{i \in I} X_i \to I$ with the base and fibers in U.

Lemma 2.2 *[**cl2**] Let U be a set of types satisfying Cl_{prod}. Then one has:*

1. *for any $X, Y \in U$ one has $Hom(X, Y) \in U$,*
2. *for any set $I \in U$ and any family of types $(X_i)_{i \in I}$ such that all X_i are in U one has $\prod_{i \in I} X_i \in U$.*

Proof: The space $Hom(X, Y)$ is the space of sections of the fibration $X \times Y \to X$. The space $\prod_{i \in I} X_i$ is the space of sections of the fibration $\coprod_{i \in I} X_i \to I$.

Lemma 2.3 *[**fibers**] Let U be a set of types satisfying Cl_{sum} and Cl_{eq}. Let further $f : Y \to X$ be a map such that $X, Y \in U$. Then for any point $x \in X$ the homotopy fiber $f^{-1}(x)$ is in U.*

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 U satisfies Cl_{sum} and Cl_{eq} we conclude that $f^{-1}(x) \in U$.

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

Lemma 2.4 *[**homlim1**] Let U be a set of types satisfying Cl_{sum} and Cl_{eq}. Then U is closed under finite homotopy limits. In particular for a pair of maps $f, g : Y \to X$ with $X, Y \in U$ the homotopy equalizer $heq(f, g)$ is in U.*

Lemma 2.5 *[**point**] Let U be a set of types satisfying Cl_{sum} and Cl_{eq}. If there exists $X \in U$ such that $X \neq \emptyset$ then $pt \in U$.*

Proof: Let x be a point of X. The homotopy fiber of the map $x : pt \to X$ is $\Omega^1(X, x)$ which is in U by Cl_{eq}. Therefore $pt \in U$ by Cl_{sum}.

Lemma 2.6 *[**cl5**] Let U be a set of types satisfying Cl_{eq}. Then if there exists $X \in U$ such that $X \neq pt$ then $\emptyset \in U$.*

Proof: If X is not connected then $X = \emptyset$ or $P(X; x, x') = \emptyset$ for some pair of points $x, x' \in X$. By Cl_{eq} we have $\Omega^n(X, x) \in U$ where Ω^n is the n-th loop space. If X is not contractible then for some n the space $\Omega^n(X, x)$ is not connected.

Lemma 2.7 *[**cl3**] Let U be a set of types satisfying Cl_{sum} and such that $pt \in U$ and $\emptyset \in U$. Then one has:*

1. *if $f : Y \to X$ is a map of level -1 such that $X \in U$ then $Y \in U$,*
2. *if $X \in U$ is a set then for any subset $Y \subset X$ one has $Y \in U$.*

Proof: The second statement is a particular case of the first. To prove the first one observe that the map f is a fibration with base in U and fibers being $\emptyset$ and pt which are also in U.

Lemma 2.8 *[**cl6**] Let U be a set of types satisfying Cl_{sum} and Cl_{eq}. If there exists $X \in U$ such that $X \neq \emptyset, pt$ then $\{0, 1\} \in U$.*

Proof: Since we assume that $X \neq pt$ we conclude by Lemma 2.6 that $\emptyset \in U$. Since we assume that $X \neq \emptyset$ we conclude by Lemma 2.5 that $pt \in U$. If X is a set then $\{0, 1\} \in U$ as a subset of X by Lemma 2.7. If X is not a set it has a connected component which is not contractible. Let

X' be such a component and let $\pi_n(X', x')$ be the first non-trivial homotopy group of X'. We have $X' \in U$ by Lemma 2.7. Let X'' be the connected component of x' in $\Omega^n(X', x')$. Since all connected components of $\Omega^n(X', x')$ are equivalent to each other we have a fibration $\Omega^n(X', x') \to X''$ whose fiber is $\pi_n(X', x')$. We conclude that $\pi_n(X', x') \in U$ by Lemma 2.3. Since it is a non-trivial set we have $\{0, 1\} \in U$ as a subset of $\pi_n(X', x')$.

Lemma 2.9 *[**cl4**] Let U be a set of types satisfying Cl_{prod} and such that $\{0, 1\} \in U$. Let further $X \in U$ be a set. Then the set PX of all subsets of X is in U.*

Lemma 2.10 *[**sur**] Let U be a set of types satisfying Cl_{sum} and Cl_{prod} and such that $\{0, 1\} \in U$. Let further $f : Y \to X$ be a surjection of sets such that $Y \in U$. Then $X \in U$.*

Proof: Assuming the axiom of choice we could take a section of f and conclude that X is in U 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 U.

Lemma 2.11 *[**cl4**] Let U be a set of types satisfying Cl_{sum} and Cl_{prod} and such that $\{0, 1\} \in U$. Then for any $X \in U$ one has $\pi_0(X) \in U$.*

Proof: It follows from Lemmas 2.2 and 2.7 since $\pi_0(X)$ is a subset of the set $Hom(X, \{0, 1\})$.

Lemma 2.12 *[**all0**] Let U be a set of types satisfying Cl_{sum}, Cl_{prod}, Cl_{eq} and such that $\{0, 1\} \in U$. Let $X \in U$. Then $\pi_0(X) \in U$ and for any $n > 0$ and any $x \in X$, $\pi_n(X, x) \in U$.*

Proof: We have $\pi_0(X) \in U$ by Lemma 2.11. By Cl_{eq} we have $\Omega^n(X, x) \in U$ where Ω^n is the n-th loop space. Then $\pi_n(X, x) \in U$ since $\pi_n(X, x) = \pi_0(\Omega^n(X, x))$.

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}. Lemma 2.8 implies that there are exactly three closed universes which do not contain $\{0, 1\}$ namely $\emptyset$, $\{pt\}$ and $\{\emptyset, pt\}$.

Proposition 2.13 *[**finitetower**] Let U be a closed universe. Let further X be a type such that one has:*

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

Then $X \in U$.

Proof: If $X = \emptyset$ or $X = pt$ then $X = \pi_0(X)$ and there is nothing to prove. Assume that $X \neq \emptyset, pt$. By Lemma 2.8 we have $\{0,1\} \in U$. Proceed by induction on N. For $N = 0$ we have $X = \pi_0(X)$. Applying Cl_{sum} to the projection $X \to \pi_0(X)$ we reduce the problem to a connected X. Let $x \in X$. By the inductive assumption we have $M = \Omega^1(X, x) \in U$. 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 $BEq(M, M) = Un(M) \in U$ 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 U.

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 1.10 and the inductive assumption we conclude that $\pi_n(F) = 0$ for $n > N - 1$. It remains to check that the non-zero homotopy groups of F are in U. This follows from our sequence and Lemma 2.10.

Lemma 2.14 *[**homlim2**] Let U be a set of types satisfying Cl_{sum}, Cl_{prod}, Cl_{eq} and such that $\mathbf{N} \in U$. Let further $\ldots X_2 \to X_1 \to X_0$ be a sequence of maps with $X_n \in U$. Then $holim X_n \in U$.*

Proof: It follows from Lemmas 2.2 and 2.4 since $holim X_n$ is the homotopy equalized of two maps from $\prod_n X_n$ to itself.

Theorem 2.15 *[**all**] Let U be a closed universe such that $\mathbf{N} \in U$. Then the following conditions on X are equivalent:*

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

Proof: It follows easily from Lemma 2.14, Lemma 2.12 and Proposition 2.13.

Let us also note the following fact.

Proposition 2.16 *[**col**] Let U be a closed universe. Then the following conditions are equivalent:*

1. *U is closed under finite homotopy colimits*

2. *U contains* $\mathbf{N}$

Let us show now how the conditions Cl_{sum}, Cl_{prod}, Cl_{eq} and Cl_{un} on U can be formulated in terms of the properties of the corresponding univalent map $\omega(U) : \tilde{\Omega}(U) \to \Omega(U)$.

Given a fibration $p : Y \to X$ define $Fam(p)$ 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 2.17 *[**close1**] Let U be a set of homotopy types and $\omega : \tilde{\Omega} \to \Omega$ be the corresponding univalent map. Then following conditions are equivalent:*

1. *U satisfies Cl_{sum} (resp. Cl_{prod}),*
2. *the fibration $Sum(\omega) \to Fam(\omega)$ (resp. $Prod(\omega) \to Fam(\omega)$) is classifiable by ω (see Definition 1.11).*

Proposition 2.18 *[**close2**] Let U be a set of homotopy types and $\omega : \tilde{\Omega} \to \Omega$ be the corresponding univalent map. Then following conditions are equivalent:*

1. *U satisfies Cl_{eq},*
2. *the diagonal map $\tilde{\Omega} \to \tilde{\Omega} \times_{\Omega} \tilde{\Omega}$ is classifiable by ω.*

To describe Cl_{un} let us consider 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 2.19 *[**close3**] Let U be a set of homotopy types and $\omega : \tilde{\Omega} \to \Omega$ be the corresponding univalent map. Then following conditions are equivalent:*

1. *U satisfies Cl_{un},*

2. *each connected component of* Ω *is classifiable by* ω*,*

3. *the projection* $[\Omega] \to \Omega$ *is classifiable by* ω*.*

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

Definition 2.20 *[***univmap***] A map of homotopy types* $\omega : \tilde{\Omega} \to \Omega$ *is called a universe map if it satisfies the following conditions:*

1. ω *is univalent,*

2. *the map* $Sum(\omega) \to Fam(\omega)$ *is classifiable by* ω*,*

3. *the map* $Prod(\omega) \to Fam(\omega)$ *is classifiable by* ω*,*

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

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

The condition that the maps corresponding to Cl_{sum}, Cl_{prod}, Cl_{eq} and Cl_{un} are classifiable by ω implies in particular that there are canonical morphisms:

$$sum : Fam(\omega) \to \Omega$$

$$prod : Fam(\omega) \to \Omega$$

$$eq : \tilde{\Omega} \times_{\Omega} \tilde{\Omega} \to \Omega$$

$$un : \Omega \to \Omega$$

Let $\Omega \times \Omega \to Fam(\omega)$ be the map which sends (x, x') to the pair $\{x, f : \omega^{-1}(x) \to \Omega\}$ where f is the constant map equal to x'. Composing this map with $\sum$ and $\prod$ we get two maps

$$\times : \Omega \times \Omega \to \Omega$$

$$hom : \Omega \times \Omega \to \Omega$$

and one verifies immediately that these maps correspond to direct product and internal hom for types in U i.e. there are canonical equivalences:

$$\omega^{-1}(\times(x, x')) \cong \omega^{-1}(x) \times \omega^{-1}(x')$$

$$\omega^{-1}(hom(x, x')) \cong Hom(\omega^{-1}(x), \omega^{-1}(x')).$$

Similarly, the composition of eq with the diagonal is a map

$$\tilde{\Omega} \to \Omega$$

which corresponds to the loop space construction $(X, x) \to \Omega^1(X, x)$ on types in U.

Suppose now that our ambient universe $\mathcal{U}$ is large enough such that for any type X there exists a a universe which contains X and which is small relative to $\mathcal{U}$. Equivalently, it means that for any

type X there exists a universe map ω such that $X \to pt$ is classifiable by ω. Observe that if this condition holds for any type X then it also holds for any set of types X_i. We let $\bar{A}$ denote the universe generated by a set of types A.

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 2.3 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 2.13 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:
$$\Omega_1 = \Omega_{1,\leq -1} \subset \Omega_{1,\leq 0} \subset \Omega_{1,\leq 1} \dots$$
where $\Omega_{1,\leq n}$ is the subtype in Ω_1 corresponding to the types of level $n-1$ 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 2.15 it consists of all types X such that all $\pi_n(X)$ are in U_2. Therefore it is completely determined by its part $U_{2,\leq 1}$ of 1-types i.e. sets. This universe contains a lot of sets. It contains $\mathbf{N}$, $\mathbf{R}$ etc. Moreover Lemma 2.1 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 $U_{n,\leq i}$ be the set of types of level $i-1$ in U_n. We have inclusions $U_{n,\leq i} \subset U_{n+1,\leq i}$ which are bijections for $i = 0$ and $n \geq 0$. For $i > 0$ it is clear that $U_{0,\leq i} \neq U_{1,\leq i} \neq U_{2,\leq i}$. It seems that one can prove that the same holds for the higher n i.e. for $i > 0$ and $n \geq 0$ the set $U_{n+1,\leq i}$ is strictly bigger than the universe $U_{n,\leq i}$ and any attempt to stabilize this sequence leads to an inconsistency.

3 Classifying spaces of types with structures

Let U be a small (relative to $\mathcal{U}$) universe of types. The equivalence classes of types in U are in one to one correspondence with the connected components of the base Ω of the univalent map ω corresponding to U i.e. Ω is the "classifying space" for types in U. One can also see that $\tilde{\Omega}$ corresponds in the same sense to pointed types in U.

Let us show on examples how one can get similar "classifying spaces" for types with structures in U. We can not do it in full generality since we do not have a good definition of a structure. The formal language which we describe in the following sections will give us a more systematic approach to this problem.

Let us consider sets with structures. For any kind K of structures on sets we have a groupoid $\mathcal{G}(K)$ whose objects are sets with a structure of kind K and morphisms are structure preserving isomorphisms. For example, if $K = group$ then $\mathcal{G}(K)$ is the groupoid of groups in U and their isomorphisms. This groupoid has a nerve $N(\mathcal{G}(K))$ which we call the classifying space of sets with a structure of kind K in U. It is naturally associated with U and K and its connected components are in one to one correspondence with isomorphism classes of sets with a structure of kind K in U. Let us show how to construct this space in terms of $\omega : \tilde{\Omega} \to \Omega$.

First of all observe that any K defines a functor F_K from sets and isomorphisms to sets and isomorphisms which takes a set to the set of structures of kind K on this set. For example F_{group} sends a set X to the subset in the set of maps $Hom(X \times X, X)$ which satisfy the axioms of multiplication in a group. Similarly, F_{top} sends X top a subset in $Hom(Hom(X, \{0,1\}), \{0,1\})$ which consists of sets of subsets of X satisfying the axioms of a topology. A set with a structure of kind K is a pair $(X, s \in F_K(X))$ and isomorphisms of such pairs are defined in the obvious way.

Let us translate this description of $\mathcal{G}(K)$ now into our topological language. A functor from sets and isomorphisms to itself is a map $K : \Omega_{\leq 1} \to \Omega_{\leq 1}$. Given such a map define $\Omega[K]$ as the fiber product

$$\Omega[K] = \Omega_{\leq 1} \times_K \tilde{\Omega_{\leq 1}}.$$

One has the following obvious lemma.

Lemma 3.1 *[**str0**] Let F_K be a functor from sets and isomorphisms to itself. Then the space $\Omega[K]$ defined above is equivalent to the nerve of the groupoid of pairs $(X, s \in F_K(X))$ and their isomorphisms.*

This is basically the only observation which is needed to construct classifying spaces for types with a structure from ω. The map K corresponding to a kind of structures can be obtained constructively from the maps $sum, prod, eq$ and un. We could do it by hand by it seems more efficient to discuss this after we introduce the formal language to deal with this kind of issues.

One other example we can give here is the space of models of a first order theory T. Given such a theory we may consider the groupoid $\mathcal{G}(T)$ of models of T in U-sets and their isomorphisms. The nerve of this groupoid is again a space. If T is multi-sorted with sorts $S_1, \ldots, S_n$ then one

can speak about models of T on a given collection of sets $X_1, \ldots, X_n$ corresponding to these sorts. Such models will form a set $T(X_1, \ldots, X_n)$. The correspondence $(X_1, \ldots, X_n) \mapsto T(X_1, \ldots, T_n)$ is functorial with respect to isomorphisms and therefore defines a map $T : (\Omega_{\leq 1})^n \to \Omega_{\leq 1}$. The nerve of the groupoid of models will again be equivalent to the fiber product

$$\Omega[T] = (\Omega_{\leq 1})^n \times_T \tilde{\Omega}_{\leq 1}$$

The map T corresponding to a theory T can be constructed explicitly from the "generating" maps $sum, prod, eq$ and un.

It is clear what we get when we take T to be *group* or *ring* or something like that. If T is the Peano arithmetic we get $\Omega[T] = pt$ or $\Omega[T] = \emptyset$ depending on whether or not $\mathbf{N} \in U$. If $\Omega[T] = pt$ the the image of the projection $\Omega[T] \to \Omega_{\leq 1}$ is the component corresponding to the type $\mathbf{N}$. If T is the Zermelo-Fraenkel theory ZF we get the following. Note first that if ZF is consistent it has a countable model (just any any other first order theory). Note also that ZF is formulated in such a way that its models have no automorphisms. Therefore the space $\Omega[ZF]$ will be a set. It definitely has many points and its "structure" is complicated. It is naturally a subset in the set of all trees with no symmetries in U.

All the examples given above gave us at most 2-types in U as classifying spaces. To get a 3-type as a classifying space consider the nerve of the 2-groupoid of categories in U and their equivalences. This space $\Omega[Cat]$ projects to $\Omega_{\leq 2}$ by the map which sends a category to the underlying groupoid with the same objects and isomorphisms as morphisms. The fiber of this map over a groupoid $\mathcal{G}$ is the groupoid of (reduced) category structures on $\mathcal{G}$. This groupoid is not in U i.e. the projection $\Omega[Cat] \to \Omega$ is not classifiable by ω. For example, the reduced category structures on a point are reduced monoids i.e. monoids without non-trivial invertible elements. The groupoid of such monoids is too large. To classify the projection $p : \Omega[Cat] \to \Omega$ we need to extend the universe setting $U^{[1]} = \overline{\{\tilde{\Omega}, \Omega\}}$ and $\omega^{[1]} = \omega(U^{[1]})$. Now the projection p is classifiable by a map $Cat : \Omega_{\leq 2} \to \Omega^{[1]}_{\leq 2}$ i.e.

$$\Omega[Cat] = \Omega_{\leq 2} \times_{Cat} \Omega^{[1]}_{\leq 2}$$

The map Cat takes a groupoid $\mathcal{G}$ to the groupoid of (reduced) category structures on $\mathcal{G}$ and can again be described explicitly in terms of the generating maps. We will consider this example in more detail after we describe the language of homotopy λ-calculus.

3 Homotopy λ-calculus

1 Expressions with variables

To define the syntax of the homotopy λ-calculus and to prove a number of syntax related results we will need some basic theory of abstract expressions. It is especially important in our case because homotopy λ-calculus is an unfinished theory and there is a good chance that the syntax described in this paper will change later. To be able to carry over the proofs of the syntactic lemmas in the case of such changes or extensions we need to formulate them in a sufficiently general form. There exists a basic theory of "formal languages" going back to Chomsky but it is inconvenient for our

purposes since it does not take into the account the rules for dealing with variables and since it makes everything too dependent on the details of the syntax.

We will use the following definitions. Let M be a set and let $T(M)$ be the set of finite rooted trees whose vertices are labeled by elements of M and such that for any vertex the set of edges leaving this vertex is ordered. Note that such ordered trees have no symmetries and therefore $T(M)$ is indeed a set. We will use the following notations. For $T \in T(M)$ let $v(T)$ be the set of vertices of T and for $v \in v(T)$ let $l(v) = l(v)_T \in M$ be the label on v. We will sometimes write $v \in T$ instead of $v \in v(T)$. For $v \in v(T)$ let $[v] = [v]_T \in T(M)$ be the subtree in T which consists of v and all the vertices under v. Let $val(v)$ be the valency of v i.e. the number of edges leaving v and $ch_1(v), \dots, ch_{val(v)}(v)$ be the "children" of v i.e. the end points of these edges in the order defined by T. Let further $br_i(v) = [ch_i(v)]$ be the branches of $[v]$. We write $v \leq w$ (resp. $v < w$) if $v \in [w]$ (resp. $v \in [w] - w$). We say that two vertices v and w are independent if $v \notin [w]$ and $w \notin [v]$.

Let Con be a finite set and var be a countably infinite one. The elements of Con are called "constructors" of our language and elements of var are called (names of) variables. Consider the set $T(Con \coprod var \coprod Con \times var)$. Its elements are (ordered, finite, rooted) trees whose vertices are labeled by an element of Con or an element of var or a pair (s, x) where $s \in Con$ and $x \in var$. For $T \in T(Con \coprod var \coprod Con \times var)$ let $bnd(T) \subset var$ be the set of "bound variables" of T i.e. elements of var which occur among the labels of T of the form (s, x) and $var(T) \subset var$ be the set of all variables of T i.e. elements of var which occur in the labels of T both by themselves and in the pairs (s, x). Let further $free(T) = var(T) - bnd(T)$ be the set of "free" variables of T.

We define an (abstract) expression over Con with variables from var as an element

$$T \in T(Con \coprod var \coprod Con \times var)$$

which satisfies the following conditions:

1. if $l(v) \in var$ then $var(v) = 0$
2. if $v \neq v'$, $l(v) = (s, x)$ and $l(v') = (s', x')$ then $x \neq x'$
3. if $l(v) = (s, x)$ and $l(v') = x$ then $v' \in [v]$

The first conditions says that a vertex labeled by a variable is a leaf. The second one says that all the bound variables in T are different. The third one says that if a variable is bound then all the leaves labeled by this variable lie under the vertex where it is boud. One can weaken the second condition by allowing the same variable name to be bound by several vertices if they are independent but this seems to make exposition more involved and does not matter at the end. We let $Exp(Con) = Exp(Con, var)$ denote the set of expressions in Con with variables from var. Note that for any expression T and $v \in v(T)$ the subtree $[v]$ is again an expression. We will sometimes use the same notation for a label and the expression which consists of one vertex labeled by this label.

Example 1.1 /**propositional**/Formulas of the propositional calculus form a subset in $Exp(C_0)$ where $C_0 = \{\vee, \wedge, \neg, \Rightarrow\}$. This subset is completely characterized by the following "local" conditions:

1. $l(v) \in C_0 \coprod var$
2. if $l(v) \in \{\vee, \wedge, \Rightarrow\}$ then $val(v) = 2$
3. if $l(v) = \neg$ then $val(v) = 1$.

All the labels are in Con and so there are no bound variables.

Example 1.2 [**predicate**] Formulas of the predicate calculus form a subset in $Exp(C_1 \coprod GP)$ where $C_1 = C_0 \coprod \{\forall, \exists\}$ and GP is the set of names of generating predicates. This subset is completely characterized by the conditions:

1. $l(v) \in C_0 \coprod C_1 \times var \coprod GP \coprod var$
2. if $l(v) \in \{\vee, \wedge, \Rightarrow\}$ then $val(v) = 2$
3. if $l(v) = \neg$ then $val(v) = 1$
4. if $l(v) = (\forall, x)$ or $l(v) = (\exists, x)$ then $val(v) = 1$
5. if $l(v) \in GP$ then $val(v)$ is the number of arguments of the corresponding predicate.

Example 1.3 [**lambda**] The terms of the untyped λ-calculus can be identified with a subset in $Exp(\{\lambda, ev\})$ which consists of expressions satisfying the conditions:

1. $l(v) \in \{\lambda\} \times var \coprod \{ev\} \coprod var$
2. if $l(v) = (\lambda, x)$ then $val(v) = 1$
3. if $l(v) = ev$ then $val(v) = 2$.

Example 1.4 [**multisorted**] Consider again the predicate logic but suppose now that we have several sorts $GS = \{S_1, \ldots, S_n\}$. Then we can identify formulas with a subset in $Exp(C_1 \coprod GP \coprod GS)$ with labels as in Example 1.2 plus labels from GS but with the vertices labeled by $(\forall, x)$ and $(\exists, x)$ having valency two. For such a vertex v the first branch of $[v]$ is one vertex labeled by an element of GS giving the sort over which the quantification occurs and the second branch is the expression which is quantified. Now however the local conditions on the vertices does characterize the well-formed formulas completely and one needs to add "global" conditions which express the consistency of the declared sorts of bound variables with the sorts of the predicates where they occur as well as the condition that the names of sorts only occur where they should.

For $T, T' \in Exp(Con)$ we say that T is α-equivalent to T' and write $T \cong T'$ if T and T' differ only by the names of bound variables i.e. if $T = T'$ as ordered trees, $free(T) = free(T')$ and there is a bijection $\phi : bnd(T) \to bnd(T')$ such that for any $v \in T$ one has:

1. if $l_T(v) \in Con$ then $l_{T'}(v) = l_T(v)$,
2. if $l_T(v) \in free(T)$ then $l_{T'}(v) = l_T(v)$,
3. if $l_T(v) \in bnd(T)$ then $l_{T'}(v) = \phi(l_T(v))$,
4. if $l_T(v) = (s, x)$ then $l_{T'}(v) = (s, \phi(x))$.

For most part we want to consider expressions up to the α-equivalence. Unfortunately we can not pass to the equivalence classes completely because for two α-equivalent expressions T_1, T_2 and a vertex $v \in v(T_1) = v(T_2)$ the expressions $[v]_{T_1}$ and $[v]_{T_2}$ need not be α-equivalent since some of the variables which are bound in T_1 may be free in $[v]$.

The following operations on expressions are well defined up to the α-equivalence:

1. If $m \in Con \amalg Con \times var$ is a possible label and $T_1, \dots, T_n \in Exp(Con)$ we will write $m(T_1, \dots, T_n)$ for the expression whose root v is labeled by m, $val(v) = n$ and $br_i(v) = T'_i$ where T'_i is obtained from T_i by the change of bound variables such that the bound variables of T'_i do not conflict with each other and with the possible variable name in m.
2. For $T_1, T_2 \in Exp(Con)$ and $v \in T_1$ we let $T_1(T_2/[v])$ be the expression obtained by replacing $[v]$ in T_1 with T'_2 where T'_2 is obtained from T_2 by the change of bound variables such that the bound variables of T'_2 do not conflict with the variables of T_1.
3. For $T_1, T_2 \in Exp(Con)$ and $y \in free(T)$ we let $T_1(T_2/y)$ denote the expression obtained by replacing all the leaves of T_1 marked by y by copies of T_2 where the names of bound variables have been changed to ensure that they do not conflict with each other and with the names of the remaining variables of T_1.

In all the examples considered above these operations correspond to the usual operations on formulas. The first operation can be used to directly associate expressions in our sense with the formulas. For example, the expression associated with the formula $\forall x : S.P(x, y)$ in a multi-sorted predicate calculus is $(\forall, x)(S, P(x, y))$ where as was mentioned above we use the same notation for an element of $Con \coprod var$ and the one vertex tree with the corresponding label.

2 An overview of homotopy λ-calculi

The homotopy λ-calculus belongs to the class of syntactic constructs known as type systems. Since there is no general definition of a type system this characterization is not very useful from the practical point. We will give a definition of homotopy λ-calculus in the framework of abstract expressions with variables introduced in the previous section. We will proceed in several steps defining a series of type systems $H\lambda_*$ with $* = 00, 01, 02, 1, 2$ such that in the appropriate sense one has

$$H\lambda_{00} \subset H\lambda_{01} \subset H\lambda_{02} \subset H\lambda_1 \subset H\lambda_2$$

Informally speaking, $H\lambda_0 = H\lambda_{02}$ is the basic stage which is strictly constructive, $H\lambda_1$ is obtained from $H\lambda_0$ by the addition of the empty type and the "Boolean rule" and $H\lambda_2$ is obtained from $H\lambda_1$

by the addition of universe constructors. It is quite possible that further development will show a need to extend $H\lambda_2$ further. One could of course hide the incremental nature of our construction and speak only about $H\lambda = H\lambda_2$ but for a number of reasons it seems unwise to do so. There is also an additional parameter namely a finite set

$$GT = T_1, \ldots, T_n$$

whose elements are called generating types. We will for most part consider this set fixed. In the application of $H\lambda$ to the formalization of mathematics the key role is played by the theory with $GT = \emptyset$. In what follows we use lower case letters for elements of var, upper case letters for elements of GT and boldface for abstract expressions.

Each of the systems $H\lambda_*$ has the following components:

1. A set of constructors $Con = Con(H\lambda_*)$. We will write below Exp for $Exp(Con \coprod GT, var)$.
2. Subsets:

$$S_0 \subset \coprod_{m \geq 0} (var \times Exp)^m$$

$$S_1 \subset (\coprod_{m \geq 0} (var \times Exp)^m) \times Exp$$

$$S_0 \subset (\coprod_{m \geq 0} (var \times Exp)^m) \times Exp \times Exp$$

Elements of these subsets are called sequents. They represent "complete sentences" of the type system. Elements of S_0 are written as

$$c_1 : \mathbf{R}_1, \ldots, c_m : \mathbf{R}_m \vdash$$

elements of S_1 are written as

$$c_1 : \mathbf{R}_1, \ldots, c_m : \mathbf{R}_m \vdash \mathbf{S} : Type$$

and elements of S_2 as

$$c_1 : \mathbf{R}_1, \ldots, c_m : \mathbf{R}_m \vdash \mathbf{s} : \mathbf{S}$$

We use capital Greek letters for elements of $\coprod_{m \geq 0} (var \times Exp)^m$. For example a generic element of S_1 may be written as $\Gamma \vdash \mathbf{S} : Type$. The part of a sequent to the left of $\vdash$ is called a context and the part to the right is called a judgement. In all systems one has the following implications:

if $\Gamma \vdash \mathbf{s} : \mathbf{S}$ is a sequent then $\Gamma \vdash \mathbf{S} : Type$ is a sequent

if $\Gamma \vdash \mathbf{S} : Type$ is a sequent then $\Gamma \vdash$ is a sequent

In addition $c_1 : \mathbf{R}_1, \ldots, c_m : \mathbf{R}_m \vdash$ is a sequent iff $c_1 : \mathbf{R}, \ldots, , c_{m-1} : \mathbf{R}_{m-1} \vdash \mathbf{R}_m : Type$ is and $c_m \in var - \{c_1, \ldots, c_{m-1}\} - var(\mathbf{R}_1) - \ldots - var(\mathbf{R}_m)$.

Sequents of the homotopy λ-calculus are obtained from the empty sequent $\vdash \in S_0$ by the constructor rules. To describe the rules we will use the standard notation where one writes

$$\frac{s_1 \;\; s_2 \;\; \dots \;\; s_n}{s'}$$

to say that if $s_1, \dots, s_n$ are sequents then s' is a sequent. The rules which generate sequents of the form $\Gamma \vdash$ are called context constructors, the rules which generate sequents of the form $\Gamma \vdash \mathbf{S} : Type$ are called type constructors and the rules which generate sequents of the form $\Gamma \vdash \mathbf{s} : \mathbf{S}$ are called term constructors. For example $H\lambda_{00}$ has a constructor $\sum$ in its vocabulary and the associated constructor rule:

$$\frac{\Gamma \vdash \mathbf{R} : Type \quad \Gamma, y : \mathbf{R} \vdash \mathbf{Q} : Type}{\Gamma \vdash (\sum, y)(\mathbf{R}, \mathbf{Q}) : Type}$$

What it means is that for any

$$\Gamma \in \coprod_{m \geq 0} (var \times Exp)^m \qquad \mathbf{R}, \mathbf{Q} \in Exp$$

and any $y \in var$ such that $\Gamma \vdash \mathbf{R} : Type$ and $\Gamma, y : \mathbf{R} \vdash \mathbf{Q} : Type$ are in S_1 we have

$$\Gamma \vdash (\sum, y)(\mathbf{R}, \mathbf{Q}) : Type \in S_1.$$

Our semantics of $H\lambda$ is based on the idea that its sequents describe constructions in the homotopy category. We take topological spaces in a given universe $\mathcal{U}$ as our standard model category. It is actually easier to provide a rigorous description of the correspondence between sequents and constructions for the category of Kan simplicial sets but we decided to consider topological spaces because they are more familiar.

Let $GT \to Top$ be any map from the set GT to the class of topological spaces i.e. a collection $X_1, \dots, X_n$ of spaces corresponding to the generating types $T_1, \dots, T_n$. Then we associate

to any sequent $\Gamma \vdash$ in S_0 a space $E(\Gamma) = E(\Gamma)_{X_1,\dots,X_n}$,

to any sequent $\Gamma \vdash \mathbf{S} : Type$ in S_1 a fibration $p_{\mathbf{S}} : E(\Gamma, \mathbf{S}) \to E(\Gamma)$,

to any sequent $\Gamma \vdash \mathbf{s} : \mathbf{S}$ in S_2 a section $s(\mathbf{s}) : E(\Gamma) \to E(\Gamma, \mathbf{S})$ of $p_{\mathbf{S}}$.

In particular, any sequent of the form $\vdash \mathbf{S} : Type$ defines a space $E(\mathbf{S})$ and any sequent of the form $\vdash \mathbf{s} : \mathbf{S}$ defines a point $s(\mathbf{s})$ of $E(\mathbf{S})$.

The key feature of the homotopy λ-calculus is that all the constructions it describes are homotopy invariant. Given a context Γ and a collection of spaces $X_1, \dots, X_n$ we get a space $E(\Gamma) = E(\Gamma)_{X_1,\dots,X_n)}$. If we replace $X_1, \dots, X_n$ by homotopy equivalent spaces $X'_1, \dots, X'_n$ the space $E(\Gamma)$ gets replaced by a homotopy equivalent one and the same applies to the objects described by sequents of the form $\Gamma \vdash \mathbf{S} : Type$ and $\Gamma \vdash \mathbf{s} : \mathbf{S}$. Hence, while we speak of semantics in Top the real target category is the homotopy category H. It is important to note that constructions 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 $\Gamma = T : Type$ a type expression **End** such that

$E(\Gamma, \mathbf{End})_X$ 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 $End(X) \to End(X')$.

It will be clear from our description that sequents of $H\lambda_0$ can be used to describe homotopy-inavriant constructions not only in Top but also in a wide class of cartesian closed Quillen model categories. The next stage $H\lambda_1$ is stronger and the constructions described by its sequents can only be defined in rather special model categories and further restrictions apply to $H\lambda_2$. I do not know at the moment how the theory of general categorical models of $H\lambda_i$'s will look like.

When one uses $H\lambda$ to formalize mathematics one considers the case $GT = \emptyset$. Sequents of the form $\Gamma \vdash \mathbf{s} : \mathbf{S}$ are used both to encode particular representatives of types and theorems. For a given mathematical statement F (e.g. Poincare conjecture) one can write a type expression $\mathbf{F}$ in the empty context (i.e. a sequent of the form $\vdash \mathbf{F} : Type$) such that the statement is true in $\mathcal{U}$ if and only if the space $E(\mathbf{F})$ is non-empty. Therefore, any sequent of the form $\vdash \mathbf{p} : \mathbf{F}$ would give a proof of T. Conversely, it is expected that any mathematical proof of F can be translated into a sequent of the form $\vdash \mathbf{p} : \mathbf{F}$. As far as I understand, the full homotopy λ-calculus is incomplete in the same way as any sufficiently complex theory i.e. there exist sequents $\vdash \mathbf{F} : Type$ such that $E(\mathbf{F})$ is non-empty but there is no sequent of the form $\vdash \mathbf{p} : \mathbf{F}$.

This approach to the semantics is easy to relate to the usual semantics of the first order logic. There, one starts with a theory with some sorts $S_1, \dots, S_n$, some predicate and functional symbols and some axioms. Given a theory T a model of T is given by a collection of sets $X_1, \dots, X_n$ and some subsets of (products of) these sets and maps between (products of) these sets which together satisfy the conditions corresponding to the axioms. All models of T with the given $X_1, \dots, X_n$ form a set $E(T) = E(T)_{X_1,\dots,X_n}$. A closed formula F in T defines a subset $E(T, F) \subset E(T)$ of $E(T)$ which consists of models where F holds. If F is a theorem i.e. it has a proof then then $E(T, F) \to E(T)$ is a bijection i.e. it has a section. If we consider contexts to be type-theoretic analogs of theories this description of the semantics for first-order theories agrees with our semantics for $H\lambda$. We will see in Section **??** that this analogy can be made precise in $H\lambda_1$ where there is a formal way to assign to any first order theory T a context Γ such that for any collection of sets $X_1, \dots, X_n$ the set of models of T "over" $X_1, \dots, X_n$ is equivalent to the space $E(\Gamma)$ corresponding to $X_1, \dots, X_n$.

Another feature common to all the $H\lambda$'s is an equivalence relation on sequents which is called convertibility. It is my understanding that the homotopy λ-calculus has the "strong normalization" theorem which asserts that any type or term expression has a unique "normal form" and that two expressions are convertible to each other iff their normal forms coincide. In addition any expression can be reduced to its normal form mechanically in a finitely many steps i.e. convertibility is decidable.

In general one may call a type system decidable if the subsets S_i of sequents are decidable in the set corresponding sets of (sequences of) expressions. It is expected that $H\lambda$ is such a system. Since proving theorems in a type system amounts to finding a completion of an incomplete sequent and not to the verification of validity for a complete one the decidability of a type system is not directly related to its expressive power. It is however a convenient property from the point of view of computer implementation since it means that one can have a simple program which certifies that a

sentence submitted to it is a valid sequent. Once such a program (or programs) exists one can write many different proof assistants which help to compose would-be proofs and these proof assistants need not be rigorously checked for correctness.

3 The syntax of $H\lambda_0$

The vocabulary of $H\lambda_0$ consists of constructor names $\sum, \prod, eq, pair, \pi, \pi', \lambda, ev, id, ev'$ and fex plus the names of generating types GT and the names of variables var. The sets of sequents S_0, S_1, S_2 are defined as the smallest subsets in the corresponding ambient sets which are closed under the constructor rules listed below. As was mentioned above all the rules are divided into context constructor rules which produce elements of S_0, type constructor rules which produce elements of S_1 and term constructor rules which produce elements of S_2.

Out of several equivalent forms of the rules we have chosen the ones which are more convenient for the proof of the "syntactic lemmas" below. These rules are not the shortest possible ones since some of the sequents appearing in the "nominator" parts of the rules are superfluous i.e. they can be recovered from the remaining sequents. However, the "cleaner" rules would require more complicated inductions in the syntactic lemmas which does not seem to be worth it.

There are the following context constructor rules in $H\lambda$:

- There is the "empty" sequent $\vdash$ i.e. we have the rule:

$$[\mathbf{tr0}]\frac{}{\vdash} \tag{1}$$

- One has

$$[\mathbf{tr3}]\frac{\Gamma \vdash \mathbf{R} : Type}{\Gamma, c : \mathbf{R} \vdash} \tag{2}$$

  if $c \in var - var(\Gamma) - var(\mathbf{R})$.

There are the following four type constructor rules in $H\lambda_0$:

- One has

$$[\mathbf{tr2}]\frac{\Gamma \vdash}{\Gamma \vdash T : Type} \tag{3}$$

  if $T \in GT$.

- $$[\mathbf{sumconstr}]\frac{\Gamma \vdash \mathbf{R} : Type \quad \Gamma, y : \mathbf{R} \vdash \mathbf{Q} : Type}{\Gamma \vdash (\sum, y)(\mathbf{R}, \mathbf{Q}) : Type} \tag{4}$$

- $$[\mathbf{prodconstr}]\frac{\Gamma \vdash \mathbf{R} : Type \quad \Gamma, y : \mathbf{R} \vdash \mathbf{Q} : Type}{\Gamma \vdash (\prod, y)(\mathbf{R}, \mathbf{Q}) : Type} \tag{5}$$

- $$[\mathbf{eqconstr}]\frac{\Gamma \vdash \mathbf{R} : Type \quad \Gamma \vdash \mathbf{r} : \mathbf{R} \quad \Gamma \vdash \mathbf{r}' : \mathbf{R}}{\Gamma \vdash eq(\mathbf{R}, \mathbf{r}, \mathbf{r}') : Type} \tag{6}$$

In the type theory one usually writes $\sum y : \mathbf{R}.\mathbf{Q}$ and $\prod y : \mathbf{R}.\mathbf{Q}$ instead of $(\sum, y)(\mathbf{R}, \mathbf{Q})$ and $(\prod, y)(\mathbf{R}, \mathbf{Q})$. We will follow this type-theoretic notation below. One also writes $\mathbf{R} \to \mathbf{Q}$ instead of $\prod y : \mathbf{R}.\mathbf{Q}$ and $\mathbf{R} \times \mathbf{Q}$ instead of $\sum y : \mathbf{R}.\mathbf{Q}$ when $\Gamma \vdash \mathbf{Q} : Type$ i.e. when $\mathbf{Q}$ does not depend on y. Note that when a sequent of the form $\Gamma \vdash \mathbf{f} : \mathbf{R} \to \mathbf{Q}$ appears in the nominator of a rule it should be treated as the pair of sequents $\Gamma \vdash \mathbf{Q} : Type$ and $\Gamma \vdash \mathbf{f} : \prod y : \mathbf{R}.\mathbf{Q}$.

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.

In the description of term constructors it will be convenient for us to further subdivide $H\lambda_0$ into three stages $H\lambda_{00}$, $H\lambda_{01}$ and $H\lambda_{02}$. To simplify the notation we will write $eq(\mathbf{r}, \mathbf{r}')$ instead of $eq(\mathbf{R}, \mathbf{r}, \mathbf{r}')$ when the ambient type $\mathbf{R}$ is clear. We will also use the syntax of term constructors which is standard in the type theory writing for example $\lambda y : \mathbf{R}.\mathbf{q}$ instead of $(\lambda, y)(\mathbf{R}, \mathbf{q})$.

There are the following term constructor rules in $H\lambda_{00}$:

- One has
$$[\mathbf{tr4}]\frac{c_1 : \mathbf{R}_1, \dots, c_m : \mathbf{R}_m \vdash}{c_1 : \mathbf{R}_1, \dots, c_m : \mathbf{R}_m \vdash c_j : \mathbf{R}_j} \tag{7}$$
if $j = 1, \dots, m$.

- One has the following "introduction" and "elimination" rules for the sum
$$[\mathbf{sumintro}]\frac{\Gamma, y : \mathbf{R} \vdash \mathbf{Q} : Type \quad \Gamma \vdash \mathbf{r} : \mathbf{R} \quad \Gamma \vdash \mathbf{q} : \mathbf{Q}(\mathbf{r}/y)}{\Gamma \vdash (pair, y)(\mathbf{Q}, \mathbf{r}, \mathbf{q}) : (\sum, y)(\mathbf{R}, \mathbf{Q})} \tag{8}$$
$$[\mathbf{sumelim}]\frac{\Gamma \vdash \mathbf{u} : \sum y : \mathbf{R}.\mathbf{Q}}{\Gamma \vdash \pi(\mathbf{u}) : \mathbf{R}} \quad \frac{\Gamma \vdash \mathbf{u} : \sum y : \mathbf{R}.\mathbf{Q}}{\Gamma \vdash (\pi', y)(\mathbf{Q}, \mathbf{u}) : \mathbf{Q}[\pi(\mathbf{u})/y]} \tag{9}$$
Unlike all other term constructors formation of a pair requires an explicit specification of the target type. We will abbreviate $pair(\mathbf{Q}, y; \mathbf{r}, \mathbf{q})$ to $\langle \mathbf{r}, \mathbf{q} \rangle$ when $\mathbf{Q}$ and y are clear. Note that it is not always so because the type of $\mathbf{q}$ is not Q but $Q(\mathbf{r}/y)$ and it is unclear how to recover the former from the later in particular because $\mathbf{r}$ might be already present in $\mathbf{Q}$. We also need to carry y and $\mathbf{Q}$ explicitly in the π' constructor for syntactic reasons related to the reduction lemmas proved below.

- One has the following "introduction" and "elimination" rules for the product
$$[\mathbf{prodintro}]\frac{\Gamma, y : \mathbf{R} \vdash \mathbf{q} : \mathbf{Q}}{\Gamma \vdash (\lambda, y)(\mathbf{R}, \mathbf{q}) : (\prod, y)(\mathbf{R}, \mathbf{Q})} \tag{10}$$
$$[\mathbf{prodelim}]\frac{\Gamma \vdash \mathbf{f} : (\prod, y)(\mathbf{R}, \mathbf{Q}) \quad \Gamma \vdash \mathbf{r} : \mathbf{R}}{\Gamma \vdash ev(\mathbf{f}, \mathbf{r}) : \mathbf{Q}[\mathbf{r}/y]} \tag{11}$$

- One has the following two elementary rules for the equivalence types:

$$[\mathbf{idrule}]\frac{\Gamma \vdash \mathbf{r} : \mathbf{R}}{\Gamma \vdash id(\mathbf{r}) : eq(\mathbf{r}, \mathbf{r})} \tag{12}$$

$$[\mathbf{smart0}]\frac{\Gamma \vdash \mathbf{f} : \mathbf{R} \to \mathbf{Q} \quad \Gamma \vdash \mathbf{u} : eq(\mathbf{R}; \mathbf{r}, \mathbf{r}')}{\Gamma \vdash ev'(\mathbf{R}, \mathbf{f}, \mathbf{u}) : eq(\mathbf{Q}; ev(\mathbf{f}, \mathbf{r}), ev(\mathbf{f}, \mathbf{r}'))} \tag{13}$$

The next term constructor rule is know as a conversion rule in some type theories. It is a simple rule with a complicated domain of definition which we need to describe first. In order to do it we will define a partial order on Exp which is called the reducibility relation and which plays an important role in the theory. Since we want our description to remain valid as we extend the set of constructors of the theory we will do it in the following general situation.

Let Con be a set of constructors which contains the constructors of $H\lambda_{00}$ i.e.

$$\sum, \prod, eq, pair, \pi, \pi', \lambda, ev, id, ev' \in Con.$$

Consider $\mathbf{T} \in Exp = Exp(Con, var)$ and $v \in v(\mathbf{T})$. Consider further the following conditions on v:

1. $r_1(v) = true$ iff $[v]$ is of the form $ev((\lambda, y)(\mathbf{R}, \mathbf{q}), \mathbf{r})$,
2. $r_2(v) = true$ iff $[v]$ is of the form $(\lambda, y)(\mathbf{R}, ev(\mathbf{f}, y))$,
3. $r_3(v) = true$ iff $[v]$ is of the form $\pi((pair, y)(\mathbf{Q}, \mathbf{r}, \mathbf{q}))$,
4. $r_4(v) = true$ iff $[v]$ is of the form $(\pi', y)(\mathbf{Q}, (pair, y)(\mathbf{Q}, \mathbf{r}, \mathbf{q}))$,
5. $r_5(v) = true$ iff $[v]$ is of the form $(pair, y)(\mathbf{Q}, \pi(\mathbf{z}), (\pi', y)(\mathbf{Q}, \mathbf{z}))$,
6. $r_6(v) = true$ iff $[v]$ is of the form $ev'(\mathbf{R}, \mathbf{f}, ev'(\mathbf{Q}, \mathbf{g}, \mathbf{u}))$,
7. $r_7(v) = true$ iff $[v]$ is of the form $ev'(\mathbf{R}, \mathbf{f}, id(\mathbf{r}))$,
8. $r_8(v) = true$ iff $[v]$ is of the form $ev'(\mathbf{R}, (\lambda, y)(\mathbf{R}, y), \mathbf{u})$.

One define these conditions in a more formal way avoiding "is of the form". For example, the first condition means that $l(v) = ev$ and $l(ch_1(v)) = (\lambda, y)$ which is how it should be formulated when the system is programmed. We will use our "semi-formal" approach since it is easier to digest.

Let $\mathbf{T}$ and v be as above and suppose that $r_i(v)$ holds. Then we define a new expression $r_i(\mathbf{T}, v)$ which is called the reduction of $\mathbf{T}$ in v by replacing the subexpression $[v]$ with $\mathbf{T}'$ where $\mathbf{T}'$ is is a certain rearrangement of $[v]$. We write below the form of $[v]$ and the form of its reduction $\mathbf{T}'$ as $[v] \mapsto \mathbf{T}'$. Note that for $i = 1$ and $i = 6$ the expression $r_i(T, v)$ is defined only up to an α-equivalence.

1. if $r_1(v)$ then $ev((\lambda, y)(\mathbf{R}, \mathbf{q}), \mathbf{r}) \mapsto \mathbf{q}(\mathbf{r}/y)$,
2. if $r_2(v)$ then $(\lambda, y)(\mathbf{R}, ev(\mathbf{f}, y)) \mapsto \mathbf{f}$,

3. if $r_3(v)$ then $\pi((pair, y)(\mathbf{Q}, \mathbf{r}, \mathbf{q})) \mapsto \mathbf{r}$,

4. if $r_4(v)$ then $(\pi', y)(\mathbf{Q}, (pair, y)(\mathbf{Q}, \mathbf{r}, \mathbf{q})) \mapsto \mathbf{q}$,

5. if $r_5(v)$ then $(pair, y)(\mathbf{Q}, \pi(\mathbf{z}), (\pi', y)(\mathbf{Q}, \mathbf{z})) \mapsto \mathbf{z}$,

6. if $r_6(v)$ then $ev'(\mathbf{R}, \mathbf{f}, ev'(\mathbf{Q}, \mathbf{g}, \mathbf{u})) \mapsto ev'(\mathbf{Q}, (\lambda, z)(\mathbf{Q}, ev(\mathbf{f}, ev(\mathbf{g}, z)), \mathbf{u})$,

7. if $r_7(v)$ then $ev'(\mathbf{R}, \mathbf{f}, id(\mathbf{r})) \mapsto id(ev(\mathbf{f}, \mathbf{r}))$,

8. if $r_8(v)$ then $ev'(\mathbf{R}, (\lambda, y)(\mathbf{R}, y), \mathbf{u}) \mapsto \mathbf{u}$.

Again, one can write these rules in a more formal way. For example one can define $r_1(\mathbf{T}, v)$ as $\mathbf{T}(\mathbf{T}'/[v])$ where $\mathbf{T}' = br_2ch_1(v)(br_2(v)/y)$.

Definition 3.1 *[**redrel**] We say that $\mathbf{T}'$ is a reduction of $\mathbf{T}$ and write $\mathbf{T} \twoheadrightarrow \mathbf{T}'$ if there exists a sequence of expressions $\mathbf{T} \cong \mathbf{T}_0, \dots, \mathbf{T}_n \cong \mathbf{T}'$, vertices $v_i \in \mathbf{T}_i$ and numbers $n_i \in \{1, \dots, 8\}$ such that $\mathbf{T}_{i+1} \cong r_{n_i}(\mathbf{T}_i, v_i)$.*

Let us write $\mathbf{T} \sim \mathbf{T}'$ if there exists $\mathbf{T}''$ such that $\mathbf{T} \twoheadrightarrow \mathbf{T}''$ and $\mathbf{T}' \twoheadrightarrow \mathbf{T}''$. We can now formulate the "conversion rule". It is a term constructor of the following form:

- One has
$$[\mathbf{conversion}]\frac{\Gamma \vdash \mathbf{r} : \mathbf{R} \qquad \Gamma \vdash \mathbf{R}' : Type}{\Gamma \vdash \mathbf{r} : \mathbf{R}'} \tag{14}$$
if $\mathbf{R} \sim \mathbf{R}'$.

The part of $H\lambda$ which is described above is very similar to other dependent type theories. The introduction and elimination rules for the dependent sum and the dependent product as well as the related conversions are the standard ones with the elimination rule for the sums being the "strong" version (see e.g. [**?**,]). The first of the equivalence rules 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 $\mathbf{f}$, two terms $\mathbf{r}$ and $\mathbf{r}'$ in the source and an equivalence between these two terms one gets an equivalence between the images of these terms. The notation $ev'(\mathbf{f}, \mathbf{u})$ signifies that we "evaluate" $\mathbf{f}$ not on a term but on an equivalence between two terms. The three associated conversions ensure that pushing an equivalence through a composition is the same as pushing it through the first function and then through the second, that pushing an equivalence through the identity does nothing and that pushing through the identity equivalence in the source one gets the identity in the target.

We will now introduce a number of additional rules related to equivalence types. These are purely "introduction" rules i.e. they come without any additional conversions. The term constructor rules in $H\lambda_{01}$ are the ones in $H\lambda_{00}$ plus the following ones:

$$[\mathbf{smart1a}]\frac{\Gamma \vdash \mathbf{u} : eq(\mathbf{r}, \mathbf{r}') \qquad \Gamma \vdash \mathbf{v} : eq(\mathbf{r}, \mathbf{r}'')}{\Gamma \vdash \sigma(\mathbf{u}, \mathbf{v}) : eq((\sum, y)(\mathbf{R}, eq(\mathbf{r}, y)), \langle \mathbf{r}', \mathbf{u} \rangle, \langle \mathbf{r}'', \mathbf{v} \rangle)} \tag{15}$$

Set

$$s(\mathbf{u},\mathbf{v}) = ev'(\sum y : \mathbf{R}.eq(\mathbf{r},y), \lambda z : \sum y : \mathbf{R}.eq(\mathbf{r},y).\pi\, z, \sigma(\mathbf{u},\mathbf{v})).$$

It is a term of type $eq(\mathbf{R},\mathbf{r}',\mathbf{r}'')$.

$$[\mathbf{smart1b}]\frac{\Gamma \vdash \mathbf{u} : eq(\mathbf{r},\mathbf{r}')}{\Gamma \vdash \iota(\mathbf{u}) : eq(s(id(\mathbf{r}),\mathbf{u}),\mathbf{u})} \tag{16}$$

$$[\mathbf{smart1c}]\frac{\Gamma \vdash \mathbf{u} : eq(\mathbf{r},\mathbf{r}')}{\Gamma \vdash \epsilon(\mathbf{u}) : eq(s(s(\mathbf{u},id(\mathbf{r})),id(\mathbf{r}')),\mathbf{u})} \tag{17}$$

The rule (15) asserts that for two equivalences $\mathbf{u},\mathbf{v}$ in $\mathbf{R}$ starting at the same term $\mathbf{r}$ we are given an equivalence $\sigma(\mathbf{u},\mathbf{v})$ between them in the space of equivalences starting in $\mathbf{r}$. Pushing this equivalence through the projection $\pi : \sum y : \mathbf{R}.eq(\mathbf{r},y) \to \mathbf{R}$ we get $s(\mathbf{u},\mathbf{v})$ which is a member of $eq(\mathbf{r}',\mathbf{r}'')$ corresponding on the intuitive level to the composition of the inverse to the first equivalence with the second. In particular $s(id,\mathbf{u})$ should be equivalent to u as the composition of $\mathbf{u}$ with the identity and $s(s(\mathbf{u},id),id)$ should be equivalent to $\mathbf{u}$ as the inverse to the inverse to $\mathbf{u}$. This is the meaning of the rules (16) and (17). The rule (15) implies in particular that the inhabitation of the types $eq(\mathbf{R};-,-)$ defines an equivalence relation on terms of $\mathbf{R}$. We will see more sophisticated examples of how these rules are used in the section about the levels structure. Semantically, these rules are related to the extension of covering homotopy property for fibrations which plays an important role in many basic constructions. I am not completely sure at the moment that the rules (15)-(17) 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.

In some dependent type systems there is also the equality elimination rule. It ensures that if we have a type expression $\mathbf{Q}$ depending on a variable y of type $\mathbf{R}$ and if we have an equivalence $\phi : eq(\mathbf{R};y,y')$ then there is a way to produce terms of type $\mathbf{Q}(y'/y)$ from terms of type $\mathbf{Q}$ (one in fact considers the case when $\mathbf{Q}$ depends on two variables from $\mathbf{R}$). As was mentioned above the only way to create a type expression dependent on a term variable in $H\lambda$ is through the use of the eq-constructor. Since equivalences can be "pushed through" all term expressions with the help of rule (13) this implies that we only need an analog of the equality elimination rule for the expressions $\mathbf{Q} = eq(\mathbf{R};x,y)$. This is achieved by our rules (15)-(17) which therefore may be considered as a replacement for the equality elimination rule in $H\lambda$. From this point of view the rule (16) corresponds to the β-conversion for the equality. We could have introduced a conversion instead of the equivalence $\iota(-)$ but this approach allows more flexibility in the models.

To complete the description of $H\lambda_0$ we will add one more rule which in our system expresses 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. To avoid new conversions we introduce functional extensionality through the condition that a certain map is an equivalence. Let us first introduce some abbreviations which will play an important role throughout the theory.

For a type expression $\mathbf{R}$ set:

$$Contr(\mathbf{R}) = \sum y_0 : \mathbf{R}. \prod y : \mathbf{R}.eq(y_0, y)$$

For a function $\mathbf{f} : \mathbf{R} \to \mathbf{Q}$ and a term $\mathbf{q} : \mathbf{Q}$ set:

$$\mathbf{f}^{-1}(\mathbf{q}) = \sum y : \mathbf{R}.eq(\mathbf{Q}; ev(\mathbf{f}, y), \mathbf{q})$$

and

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

For $\mathbf{f}, \mathbf{f}' : \prod y : \mathbf{R}.\mathbf{Q}$ define a function

$$\chi(\mathbf{f}, \mathbf{f}') : eq(\mathbf{f}, \mathbf{f}') \to \prod y : \mathbf{R}.eq(\mathbf{Q}; ev(\mathbf{f}, y), ev(\mathbf{f}', y))$$

by the formula:

$$\chi(\mathbf{f}, \mathbf{f}') = \lambda u : eq(\mathbf{f}, \mathbf{f}').\lambda y' : \mathbf{R}.ev'(\prod y : \mathbf{R}.\mathbf{Q}, \lambda g : \prod y : \mathbf{R}.\mathbf{Q}.ev(g, y'), u).$$

The last rule of $H\lambda_0$ which completes $H\lambda_{01}$ to $H\lambda_{02}$ looks as follows:

- $$[\mathbf{fex}]\frac{\Gamma \vdash \mathbf{f} : \prod y : \mathbf{R}.\mathbf{Q} \quad \Gamma \vdash \mathbf{f}' : \prod y : \mathbf{R}.\mathbf{Q}}{\Gamma \vdash fex(\mathbf{f}, \mathbf{f}') : Eq(\chi(\mathbf{f}, \mathbf{f}'))}. \tag{18}$$

We will see in the next section that for any map $\mathbf{f}$ the type $Eq(\mathbf{f})$ is a "property" i.e. if it is inhabited then all its inhabitants are canonically equivalent to each other. Taking this into account one can say that our last constructor is actually an axiom and not a structure.

4 Parsing lemmas

In the previous section we defined the sets of sequents of $H\lambda_0$ as the subsets of the corresponding ambient sets of "sentences" generated by some operations. In this section we prove a number of technical lemmas which characterize these sets in a way which can be used by a computer to recognize when a given sentence is a sequent. This lemmas will also be used to prove the Semantics Theorem **??**. We do not get complete algorithms for recognizing sequents in this section. What remains is how to recognize when two expressions are convertible to each other i.e. when $\mathbf{T} \sim \mathbf{T}'$. This part is unnecessary for the semantic theorem and will be considered in Section **??**.

The proofs of the more basic syntactic lemmas is based on the concept of a numbered sequent. Define the set of numbered sequents as a subset in the set of pairs (s, n) where s is a sentence in the vocabulary of $H\lambda_0$ and $n \in \mathbf{N}$ which is generated by the following rules:

1. $(\vdash, 0)$ is a numbered sequent,

2. for any of the generating rules of $H\lambda_0$ of the form

$$\frac{s_1 \dots s_k}{s}$$

there is a rule for numbered sequents of the form

$$\frac{(s_1, n_1) \dots (s_n, n_k)}{(s, max(n_1, \dots, n_k) + 1)}.$$

Clearly, for any sequent s there exist numbers n such that (s, n) is a numbered sequent. Morally, (s, n) is a numbered sequent iff s can be obtained from the empty sequent by a tree of rule applications whose longest branch is of length n. Since numbered sequents come equipped with a natural counter it is easy to do inductive proofs with them. We start with the following two lemmas. We let $\mathcal{G}$ in these lemmas denote any of the possible endings of a sequent. They are proved using numbered sequents and the obvious induction. Since the proofs are very boring they are omitted. We also systematically ignore the fact that in some lemmas some expressions have to be replaced by α-equivalent ones.

Lemma 4.1 *[**substitution**] Let $\Gamma \vdash \mathbf{t} : \mathbf{T}$ and $\Gamma, z : \mathbf{T}, \Delta \vdash \mathcal{G}$ be sequents. Then $\Gamma, \Delta(\mathbf{t}/z) \vdash \mathcal{G}(\mathbf{t}/z)$ is a sequent.*

Lemma 4.2 *[**extcont**] If $\Gamma \vdash \mathcal{G}$ is a sequent and $\Gamma, \Delta \vdash$ is a sequent then $\Gamma, \Delta \vdash \mathcal{G}$ is a sequent.*

Let Exp be as in the discussion preceding the rule (14). For each sequent $\Gamma \vdash$ let $Rexp(\Gamma)$ be the subset in Exp which consists of expressions $\mathbf{S}$ such that $\Gamma \vdash \mathbf{S} : Type$ (is a sequent) and for $\mathbf{S} \in Rexp(\Gamma)$ let $lexp(\Gamma; \mathbf{S})$ be the subset in Exp which consists of all expressions $\mathbf{s}$ such that $\Gamma \vdash \mathbf{s} : \mathbf{S}$. We let $\overset{\Gamma}{\approx}$ (resp. $\overset{\Gamma}{\approx}_{\mathbf{S}}$) denote the equivalence relation on $Rexp(\Gamma)$ (resp. $lexp(\Gamma; \mathbf{S})$) generated by the relation $\sim$. Note that in view of the rule (14) we have

$$lexp(\Gamma; \mathbf{S}) = lexp(\Gamma; \mathbf{S}')$$

if $\mathbf{S} \overset{\Gamma}{\approx} \mathbf{S}'$. We will see below that $\overset{\Gamma}{\approx}$ and $\overset{\Gamma}{\approx}_{\mathbf{S}}$ coincide with $\sim$ but for now we do not know this since we do not know that $\sim$ is transitive.

Lemma 4.3 *[**eqrep**] Let $\Gamma, y : \mathbf{R} \vdash \mathcal{G}$ and $\Gamma \vdash \mathbf{R}' : Type$ be sequents such that $\mathbf{R} \overset{\Gamma}{\approx} \mathbf{R}'$. Then $\Gamma, y : \mathbf{R}' \vdash \mathcal{G}$ is a sequent.*

Proof: Follows by induction on numbered sequents using the rule (14).

Lemma 4.4 *[**eqrep2**] Let $\Gamma, y : \mathbf{R} \vdash \mathbf{Q} : Type$, $\Gamma, y : \mathbf{R} \vdash \mathbf{Q}' : Type$ and $\Gamma \vdash \mathbf{r} : \mathbf{R}$ be sequents. If $\mathbf{Q} \overset{\Delta}{\approx} \mathbf{Q}'$ where $\Delta = \Gamma, y : \mathbf{R}$ then $\mathbf{Q}(\mathbf{r}/y) \overset{\Gamma}{\approx} \mathbf{Q}'(\mathbf{r}/y)$.*

Proof: If $\mathbf{Q} \sim \mathbf{Q}'$ the statement follows from Lemma **??**. The general case $\mathbf{Q} \sim \mathbf{Q}_1 \sim \ldots \sim \mathbf{Q}_n \sim \mathbf{Q}'$ follows by obvious induction.

This lemma has an important for us particular case which we formulate separately.

Lemma 4.5 *[**eqrep3**] If* $\Gamma \vdash \mathbf{Q} : Type$, $\Gamma \vdash \mathbf{Q}' : Type$ *and* $\Gamma \vdash \mathbf{r} : \mathbf{R}$ *then* $\mathbf{Q} \overset{\Delta}{\approx} \mathbf{Q}'$ *(where* $\Delta = \Gamma, y : \mathbf{R}$*)implies* $\mathbf{Q} \overset{\Gamma}{\approx} \mathbf{Q}'$.

Let us for convenience list the rules producing sequents of different kinds. We have:

$\Gamma \vdash$ - (1), (2)

$\Gamma \vdash \mathbf{S} : Type$ - (3), (4), (5), (6)

$\Gamma \vdash \mathbf{s} : \mathbf{S}$ - (7), (8), (9), (10), (11), (12), (13), (15), (16), (17), (18) and (14).

Lemma 4.6 *[**triv**] If* $\Gamma \vdash$ *is a sequent and* Γ *is non-empty then* Γ *is of the form* $\Gamma', c : \mathbf{R}$ *such that* $\Gamma' \vdash \mathbf{R} : Type$ *is a sequent and* $c \in var - var(\Gamma') - var(\mathbf{R})$.

Proof: Any non-empty sequent of the form $\Gamma \vdash$ is generated by the rule (7). This implies the lemma.

Lemma 4.7 *[**simple11**] If* $\Gamma \vdash \mathbf{S} : Type$ *is a sequent then* $\Gamma \vdash$ *is a sequent.*

Proof: By induction using numbered sequents and the explicit form of the type constructor rules.

Lemma 4.8 *[**contextdeconstr**] A sequence*

$$[\mathbf{context2}]c_1 : \mathbf{R}_1, \ldots, c_m : \mathbf{R}_m \vdash \tag{19}$$

is a sequent iff the following conditions hold:

1. $c_1, \ldots, c_m \in var - \cup_{j=1}^{m} var(\mathbf{R}_j)$ *and* $c_i \neq c_j$ *for* $i \neq j$,
2. *for any* $j = 0, \ldots, m-1$ *the sentence*

$$c_1 : \mathbf{R}_1, \ldots, c_j : \mathbf{R}_j \vdash \mathbf{R}_j : Type$$

is a sequent.

Proof: The "if" part follows immediately from the rule (2). Then "only if" part follows by induction on m from Lemmas 4.6 and 4.7.

Lemma 4.9 *[**deconstr**] If* $\Gamma \vdash \mathbf{s} : \mathbf{S}$ *is a sequent then* $\Gamma \vdash \mathbf{S} : Type$ *is a sequent.*

Proof: By induction using numbered sequents. The inductive step is straightforward for sequents generated by all the rules except (7), (9b) and (11). For the first of these rules one has to use Lemmas 4.8 and Lemma 4.2 for the other two Lemma 4.1.

For an expression $\mathbf{T}$ let $Nd(\mathbf{T})$ be the number of occurrences among labels of $\mathbf{T}$ of the names of type constructors i.e. elements of the set $\{\sum, \prod, eq\}$.

Lemma 4.10 *[**typeind1**] A sentence* $\Gamma \vdash \mathbf{S} : Type$ *is a sequent iff one of the following mutually exclusive conditions holds:*

1. $\mathbf{S} = T$ *for some* $T \in GT$ *and one has*
   - (a) $\Gamma \vdash$ *is a sequent,*
   - (b) $Nd(\mathbf{S}) = 0$
2. $\mathbf{S} = (\sum, y)(\mathbf{R}, \mathbf{Q})$ *and one has*
   - (a) $\Gamma \vdash \mathbf{R} : Type$ *and* $\Gamma, y : \mathbf{R} \vdash \mathbf{Q} : Type$ *are sequents,*
   - (b) $Nd(\mathbf{S}) = 1 + Nd(\mathbf{R}) + Nd(\mathbf{Q})$.
3. $\mathbf{S} = (\prod, y)(\mathbf{R}, \mathbf{Q})$ *and one has*
   - (a) $\Gamma \vdash \mathbf{R} : Type$ *and* $\Gamma, y : \mathbf{R} \vdash \mathbf{Q} : Type$ *are sequents,*
   - (b) $Nd(\mathbf{S}) = 1 + Nd(\mathbf{R}) + Nd(\mathbf{Q})$.
4. $\mathbf{S} = eq(\mathbf{R}, \mathbf{r}, \mathbf{r}')$ *and one has:*
   - (a) $\Gamma \vdash \mathbf{R} : Type$, $\Gamma \vdash \mathbf{r} : \mathbf{R}$ *and* $\Gamma \vdash \mathbf{r}' : \mathbf{R}$ *are sequents*
   - (b) $Nd(\mathbf{S}) = 1 + Nd(\mathbf{R}) + Nd(\mathbf{r}) + Nd(\mathbf{r}')$

Proof: The fact that the conditions are mutually exclusive is obvious. Sequents of the form $\Gamma \vdash \mathbf{S} : Type$ are generated by the rules (3), (4), (5) and (6). They exactly correspond to the four different cases of the lemma.

Lemma 4.11 *[**typeconstreq**] Let* $\Gamma \vdash$ *and* $\mathbf{S}, \mathbf{S} \in Rexp(\Gamma)$ *be two expressions. Then* $\mathbf{S} \overset{\Gamma}{\approx} \mathbf{S}'$ *iff one of the following mutually exclusive conditions holds:*

1. $\mathbf{S} = \mathbf{S}' = T$ *for some* $T \in GT$,

2. $\mathbf{S} = (\sum, y)(\mathbf{R}, \mathbf{Q})$, $\mathbf{S}' = (\sum, y)(\mathbf{R}', \mathbf{Q}')$ *and one has* $\mathbf{R} \overset{\Gamma}{\approx} \mathbf{R}'$ *and* $\mathbf{Q} \overset{\Delta}{\approx} \mathbf{Q}'$ *where* $\Delta = \Gamma, y : \mathbf{R}$,

3. $\mathbf{S} = (\prod, y)(\mathbf{R}, \mathbf{Q})$, $\mathbf{S}' = (\prod, y)(\mathbf{R}', \mathbf{Q}')$ *and one has* $\mathbf{R} \overset{\Gamma}{\approx} \mathbf{R}'$ *and* $\mathbf{Q} \overset{\Delta}{\approx} \mathbf{Q}'$ *where* $\Delta = \Gamma, y : \mathbf{R}$,

4. $\mathbf{S} = eq(\mathbf{R}, \mathbf{r}, \mathbf{r}')$, $\mathbf{S}' = eq(\mathbf{R}', \mathbf{r}'', \mathbf{r}''')$ *and one has* $\mathbf{R} \overset{\Gamma}{\approx} \mathbf{R}'$, $\mathbf{r} \overset{\Gamma}{\approx}_{\mathbf{R}} \mathbf{r}''$ *and* $\mathbf{r}' \overset{\Gamma}{\approx}_{\mathbf{R}} \mathbf{r}'''$.

Proof: Let us prove the "only if" part first. It is clear from the definition of the relation $\sim$ that if $\mathbf{T} \sim \mathbf{T}'$ where $\mathbf{T}$ is an expression such that the root label of $\mathbf{T}$ contains one of the type constructors $\sum, \prod, eq$ then the root label of $\mathbf{T}'$ coincides with the root label of $\mathbf{T}$ and the branches of $\mathbf{T}$ and $\mathbf{T}'$ are in the relation $\sim$ with each other. Together with Lemma 4.10 this implies immediately that if $\mathbf{S} \sim \mathbf{S}'$ then one of the four conditions of the lemma holds. Suppose now that $\mathbf{S} \sim \mathbf{S}_1 \sim \ldots \sim \mathbf{S}_n \sim \mathbf{S}'$ for some $\mathbf{S}_i \in Rexp(\Gamma)$. By obvious induction it is sufficient to consider the case $n = 1$. If the first condition holds for both pairs $\mathbf{S}, \mathbf{S}_1$ and $\mathbf{S}_1, \mathbf{S}'$ there is nothing to do. Suppose that the second one does. Then $\mathbf{S} = (\sum, y)(\mathbf{R}_1, \mathbf{Q}_1)$ where $\mathbf{R} \sim \mathbf{R}_1 \sim \mathbf{R}'$ and $\mathbf{Q} \sim \mathbf{Q}_1 \sim \mathbf{Q}'$. On the other hand by Lemma 4.10 and Lemma 4.3 we know that $\mathbf{R}_1 \in Rexp(\Gamma)$ and $\mathbf{Q}_1 \in Rexp(\Delta)$ therefore $\mathbf{R} \overset{\Gamma}{\approx} \mathbf{R}'$ and $\mathbf{Q} \overset{\Delta}{\approx} \mathbf{Q}'$. The case when the third condition holds is strictly analogous. Suppose that the fourth condition holds. Then $\mathbf{S}_1 = eq(\mathbf{R}_1, \mathbf{r}_1, \mathbf{r}'_1)$ and the same reasoning works.

Consider now the "if" part. In the first case the statement is obvious. Suppose that the second possibility occurs. Obvious induction shows again that it is enough to consider the case when $\mathbf{R} \sim \mathbf{R}_1 \sim \mathbf{R}'$ and $\mathbf{Q} \sim \mathbf{Q}_1 \sim \mathbf{Q}'$ with $\mathbf{R}_1 \in Rexp(\Gamma)$ and $\mathbf{Q}_1 \in Rexp(\Delta)$. Then by the sum constructor rule we have

$$\mathbf{S}_1 = (\sum, y)(\mathbf{R}_1, \mathbf{Q}_1) \in Rexp(\Gamma)$$

and since $\mathbf{S} \sim \mathbf{S}_1 \sim \mathbf{S}'$ we conclude that $\mathbf{S} \overset{\Gamma}{\approx} \mathbf{S}'$. The case of the product is strictly analogous. Consider the *eq* case. We have $\mathbf{R} \sim \mathbf{R}_1 \sim \mathbf{R}'$, $\mathbf{r} \sim \mathbf{r}_1 \sim \mathbf{r}''$ and $\mathbf{r}' \sim \mathbf{r}'_1 \sim \mathbf{r}'''$ where $\mathbf{R}_1 \in Rexp(\Gamma)$ and $\mathbf{r}_1, \mathbf{r}'_1 \in lexp(\Gamma; \mathbf{R})$. Then

$$\mathbf{S}_1 = eq(\mathbf{R}_1, \mathbf{r}_1, \mathbf{r}'_1) \in Rexp(\Gamma)$$

by the eq-constructor rule and (14) and since $\mathbf{S} \sim \mathbf{S}_1 \sim \mathbf{S}'$ we conclude that $\mathbf{S} \overset{\Gamma}{\approx} \mathbf{S}'$.

Lemma 4.12 *[**tr110**] Let* $\Gamma \vdash \mathbf{s} : \mathbf{S}$ *be a sequent. Then there exist a sequent of the form* $\Gamma \vdash \mathbf{s} : \mathbf{S}'$ *such that one has:*

1. $\mathbf{S} \overset{\Gamma}{\approx} \mathbf{S}'$
2. *the sequent* $\Gamma \vdash \mathbf{s} : \mathbf{S}'$ *is the output of one of the rules (7), (8), (9), (10), (11), (12), (13), (15), (16), (17) and (18).*

Proof: Any sequent of the form $\Gamma \vdash \mathbf{s} : \mathbf{S}$ is the output of one of the rules in our list or (14). By the obvious induction it follows that any such sequent is obtained by one of the rules (7), (8), (9), (10), (11), (12), (13), (15), (16), (17), (18) followed by sequence of rules (14). Together with the definition of $\overset{\Gamma}{\approx}$ this implies the lemma.

For an expression (a sentence) $\mathbf{T}$ let $nd(\mathbf{T})$ be the number of occurrences in $\mathbf{T}$ of term constructor symbols i.e. elements of the set

$$\{ev, \lambda, \pi, \pi', pair, id, ev', \sigma, \iota, \epsilon, fex\}.$$

Lemma 4.13 *[**termind1**] A sentence $\Gamma \vdash \mathbf{s} : \mathbf{S}$ where $\Gamma = c_1 : \mathbf{R}_1, \ldots, c_m : \mathbf{R}_m$ is a sequent iff $\Gamma \vdash \mathbf{S} : Type$ and one of the following mutually exclusive conditions holds:*

1. $\mathbf{s} = c_j$ *for some* $j = 1, \ldots, m$ *and one has:*
   (a) $\mathbf{S} \overset{\Gamma}{\approx} \mathbf{R}_j$,
   (b) $nd(\mathbf{s}) = 0$
2. $\mathbf{s} = (pair, y)(\mathbf{Q}, \mathbf{r}, \mathbf{q})$ *and one has:*
   (a) *there are sequents of the form* $\Gamma, y : \mathbf{R} \vdash \mathbf{Q} : Type$, $\Gamma \vdash \mathbf{r} : \mathbf{R}$ *and* $\Gamma \vdash \mathbf{q} : \mathbf{Q}(\mathbf{r}/y)$,
   (b) $\mathbf{S} \overset{\Gamma}{\approx} (\sum, y)(\mathbf{R}, \mathbf{Q})$,
   (c) $nd(\mathbf{s}) = 1 + nd(\mathbf{r}) + nd(\mathbf{q}) + nd(\mathbf{Q})$.
3. $\mathbf{s} = \pi\mathbf{z}$ *and one has:*
   (a) *there is a sequent of the form* $\Gamma \vdash \mathbf{z} : (\sum, y)(\mathbf{R}, \mathbf{Q})$,
   (b) $\mathbf{S} \overset{\Gamma}{\approx} \mathbf{R}$,
   (c) $nd(\mathbf{s}) = 1 + nd(\mathbf{z})$.
4. $\mathbf{s} = (\pi', y)(\mathbf{Q}, \mathbf{z})$ *and one has:*
   (a) *there is a sequent of the form* $\Gamma \vdash \mathbf{z} : (\sum, y)(\mathbf{R}, \mathbf{Q})$,
   (b) $\mathbf{S} \overset{\Gamma}{\approx} \mathbf{Q}(\pi(\mathbf{z})/y)$,
   (c) $nd(\mathbf{s}) = 1 + nd(\mathbf{z}) + nd(\mathbf{Q})$.
5. $\mathbf{s} = (\lambda, y)(\mathbf{R}, \mathbf{q})$ *and one has:*
   (a) *there are sequents of the form* $\Gamma \vdash (\prod, y)(\mathbf{R}, \mathbf{Q})$ *and* $\Gamma, y : \mathbf{R} \vdash \mathbf{q} : \mathbf{Q}$,
   (b) $\mathbf{S} \overset{\Gamma}{\approx} (\prod, y)(\mathbf{R}, \mathbf{Q})$,
   (c) $nd(\mathbf{s}) = 1 + nd(\mathbf{q}) + nd(\mathbf{R})$.
6. $\mathbf{s} = ev(\mathbf{f}, \mathbf{r})$ *and one has:*
   (a) *there are sequents of the form* $\Gamma \vdash \mathbf{f} : (\prod, y)(\mathbf{R}, \mathbf{Q})$ *and* $\Gamma \vdash \mathbf{r} : \mathbf{R}$,
   (b) $\mathbf{S} \overset{\Gamma}{\approx} \mathbf{Q}(\mathbf{r}/y)$,
   (c) $nd(\mathbf{s}) = 1 + nd(\mathbf{f}) + nd(\mathbf{r})$.
7. $\mathbf{s} = id(\mathbf{r})$ *and one has:*
   (a) *there is a sequent of the form* $\Gamma \vdash \mathbf{r} : \mathbf{R}$,

(b) $\mathbf{S} \overset{\Gamma}{\approx} eq(\mathbf{R}, \mathbf{r}, \mathbf{r})$,

(c) $nd(\mathbf{s}) = 1 + nd(\mathbf{r})$.

8. $\mathbf{s} = ev'(\mathbf{R}, \mathbf{f}, \mathbf{u})$ *and one has:*

   (a) there are sequents of the form $\Gamma \vdash \mathbf{f} : \mathbf{R} \to \mathbf{Q}$, $\Gamma \vdash \mathbf{u} : eq(\mathbf{R}, \mathbf{r}, \mathbf{r}')$,

   (b) $\mathbf{S} \overset{\Gamma}{\approx} eq(\mathbf{Q}, ev(\mathbf{f}, \mathbf{r}), ev(\mathbf{f}, \mathbf{r}'))$,

   (c) $nd(\mathbf{s}) = 1 + nd(\mathbf{R}) + nd(\mathbf{f}) + nd(\mathbf{u})$.

9. $\mathbf{s} = \sigma(\mathbf{u}, \mathbf{v})$ *and one has:*

   (a) there are sequents of the form $\Gamma \vdash \mathbf{u} : eq(\mathbf{R}, \mathbf{r}, \mathbf{r}')$, $\Gamma \vdash \mathbf{v} : eq(\mathbf{R}, \mathbf{r}, \mathbf{r}'')$,

   (b) $\mathbf{S} \overset{\Gamma}{\approx} eq((\sum, y)(\mathbf{R}, eq(\mathbf{r}, y)), \langle \mathbf{r}', \mathbf{u} \rangle, \langle \mathbf{r}'', \mathbf{v} \rangle)$

   (c) $nd(\mathbf{s}) = 1 + nd(\mathbf{u}) + nd(\mathbf{v})$.

10. $\mathbf{s} = \iota(\mathbf{u})$ *and one has:*

    (a) there is a sequent of the form $\Gamma \vdash \mathbf{u} : eq(\mathbf{R}, \mathbf{r}, \mathbf{r}')$,

    (b) $\mathbf{S} \overset{\Gamma}{\approx} eq(s(id(\mathbf{r}), \mathbf{u}), \mathbf{u})$,

    (c) $nd(\mathbf{s}) = 1 + nd(\mathbf{u})$.

11. $\mathbf{s} = \epsilon(\mathbf{u})$ *and one has:*

    (a) there is a sequent of the form $\Gamma \vdash \mathbf{u} : eq(\mathbf{R}, \mathbf{r}, \mathbf{r}')$,

    (b) $\mathbf{S} \overset{\Gamma}{\approx} eq(s(s(\mathbf{u}, id(\mathbf{r})), id(\mathbf{r}')), \mathbf{u})$,

    (c) $nd(\mathbf{s}) = 1 + nd(\mathbf{u})$.

12. $\mathbf{s} = fex(\mathbf{f}, \mathbf{f}')$ *and one has:*

    (a) there are sequents of the form $\Gamma \vdash \mathbf{f} : (\prod, y)(\mathbf{R}, \mathbf{Q})$, $\Gamma \vdash \mathbf{f}' : (\prod, y)(\mathbf{R}, \mathbf{Q})$,

    (b) $\mathbf{S} \overset{\Gamma}{\approx} Eq(\chi(\mathbf{f}, \mathbf{f}'))$ *where the right hand side is the type expression introduced before the rule (18),*

    (c) $nd(\mathbf{s}) = 1 + nd(\mathbf{f}) + nd(\mathbf{f}')$.

Proof: Follows immediately from Lemma 4.12 and the explicit form of the term constructor rules.

Let $lexp(\Gamma)$ be the set of all valid term expressions in Γ i.e.

$$lexp(\Gamma) = \bigcup_{\mathbf{S} \in Rexp(\Gamma)} lexp(\Gamma; \mathbf{S}).$$

Lemma 4.14 *[***cantype***] For any* $\Gamma \vdash$ *there exists a unique mapping*

$$\Theta : lexp(\Gamma) \to Rexp(\Gamma)$$

such that for $\mathbf{s} \in lexp(\Gamma)$ *one has:*

1. *If* $\mathbf{s} = c_j$ *then*
$$\Theta(\mathbf{s}) = \mathbf{R}_j$$

2. *If* $\mathbf{s} = (pair, y)(\mathbf{Q}, \mathbf{r}, \mathbf{q})$ *then*
$$\Theta(\mathbf{s}) = (\sum, y)(\Theta(\mathbf{r}), \mathbf{Q}) = (\sum, y)(\Theta(br_2\mathbf{s}), br_1\mathbf{s})$$

3. *If* $\mathbf{s} = \pi\mathbf{z}$ *then*
$$\Theta(\mathbf{s}) = \mathbf{R} = br_1\Theta(br_1\mathbf{s})$$

4. *If* $\mathbf{s} = (\pi', y)(\mathbf{Q}, \mathbf{z})$ *then*
$$\Theta(\mathbf{s}) = \mathbf{Q}(\pi(\mathbf{z})/y) = br_1\mathbf{s}(\pi(br_1\mathbf{s})/y)$$

5. *If* $\mathbf{s} = (\lambda, y)(\mathbf{R}, \mathbf{q})$ *then*
$$\Theta(\mathbf{s}) = (\prod, y)(\mathbf{R}, \Theta(\mathbf{q})) = (\prod, y)(br_1\mathbf{s}, \Theta(br_2\mathbf{s}))$$

6. *If* $\mathbf{s} = ev(\mathbf{f}, \mathbf{r})$ *then*
$$\Theta(\mathbf{s}) = br_2\Theta(\mathbf{f})(\mathbf{r}/y) = br_2\Theta(br_1\mathbf{s})(br_2\mathbf{s}/y)$$

7. *If* $\mathbf{s} = id(\mathbf{r})$ *then*
$$\Theta(\mathbf{s}) = eq(\Theta(\mathbf{r}), \mathbf{r}, \mathbf{r}) = eq(\Theta(br_1\mathbf{s}), br_1\mathbf{s}, br_1\mathbf{s})$$

8. *If* $\mathbf{s} = ev'(\mathbf{R}, \mathbf{f}, \mathbf{u})$ *then*
$$\Theta(\mathbf{s}) = eq(\mathbf{Q}, ev(\mathbf{f}, \mathbf{r}), ev(\mathbf{f}, \mathbf{r}')) = eq(br_2\Theta(br_2\mathbf{s}), ev(br_2\mathbf{s}, br_2\Theta(br_3\mathbf{s})), ev(br_2\mathbf{s}, br_3\Theta(br_3\mathbf{s})))$$

9. *If* $\mathbf{s} = \sigma(\mathbf{u}, \mathbf{v})$ *then*
$$\Theta(\mathbf{s}) = eq((\sum, y)(\mathbf{R}, eq(\mathbf{r}, y)), \langle \mathbf{r}', \mathbf{u}\rangle, \langle \mathbf{r}'', \mathbf{v}\rangle)$$
In this and in the following three cases we will not write the expression for $\Theta(\mathbf{s})$ explicitly in terms of branches since it is too long and can be easily deduced from our semi-formal description. More detailed formulas are given in the proof of the lemma.

10. *If* $\mathbf{s} = \iota(\mathbf{u})$ *then*
$$\Theta(\mathbf{s}) = eq(s(id(\mathbf{r}), \mathbf{u}), \mathbf{u})$$

11. *If* $\mathbf{s} = \epsilon(\mathbf{u})$ *then*
$$\Theta(\mathbf{s}) = eq(s(s(id(\mathbf{r})), id(\mathbf{r}')), \mathbf{u})$$

12. *If* $\mathbf{s} = fex(\mathbf{f}, \mathbf{f}')$ *then*
$$\Theta(\mathbf{s}) = Eq(\chi(\mathbf{f}, \mathbf{f}'))$$

This mapping has the following two properties:

1. *one has* $\Gamma \vdash \mathbf{s} : \Theta(\mathbf{s})$,

2. *if* $\Gamma \vdash \mathbf{s} : \mathbf{S}$ *then* $\mathbf{S} \overset{\Gamma}{\approx} \Theta(\mathbf{s})$.

Proof: Let $lexp(\Gamma)_{\leq n}$ be the subset of $lexp(\Gamma)$ which consists of $\mathbf{s}$ such that $nd(\mathbf{s}) \leq n$. Let us show by induction on n that for any n there exists a unique map

$$\Theta : lexp(\Gamma)_{\leq n} \to Rexp(\Gamma)$$

satisfying conditions (1)-(12) and (1)-(2). The uniqueness is obvious from Lemma 4.13. To start the induction consider the case $nd(\mathbf{s}) = 0$. Then by Lemma 4.13 we have $\mathbf{s} = c_j$ and we set $\tau(\mathbf{s}) = \mathbf{R}_j$. Since $\Gamma \vdash c_j : \mathbf{R}_j$ we know that $\tau(\mathbf{s}) \in Rexp(\Gamma)$ by Lemma 4.9 and condition (1) holds. Condition (2) saying that if $\Gamma \vdash c_j : \mathbf{S}$ then $\mathbf{S} \overset{\Gamma}{\approx} \mathbf{R}_j$ holds by Lemma 4.13(1a).

To make the inductive step we should show that for $\mathbf{s}$ of each kind the expression $\tau(\mathbf{s})$ defined by the corresponding clause of our lemma is in $Rexp(\Gamma)$ and that the conditions (1) and (2) hold.

1. $\mathbf{s} = c_j$ only occurs in the case $nd = 0$ which we have already dealt with.

2. $\mathbf{s} = (pair, y)(\mathbf{Q}, \mathbf{r}, \mathbf{q})$ - by Lemma 4.13(2) we have

   $$\Gamma, y : \mathbf{R} \vdash \mathbf{Q} : Type \quad \Gamma \vdash \mathbf{r} : \mathbf{R} \quad \Gamma \vdash \mathbf{q} : \mathbf{Q}(\mathbf{r}/y)$$

   for some $\mathbf{R}$. By the clause (2) of our lemma we set

   $$\tau(\mathbf{s}) = (\sum, y)(\tau(\mathbf{r}), Q)$$

   to check that it is a well-formed type expression in Γ we need to know that $\Gamma, y : \tau(\mathbf{r}) \vdash \mathbf{Q}$. By the inductive assumption $\tau(\mathbf{r}) \overset{\Gamma}{\approx} \mathbf{R}$ and our claim follows from Lemma 4.3. By Lemma 4.11 we have

   $$\tau(\mathbf{s}) \overset{\Gamma}{\approx} (\sum, y)(\mathbf{R}, \mathbf{Q})$$

   and therefore $\mathbf{S} \overset{\Gamma}{\approx} \tau(\mathbf{s})$ which implies conditions (1) and (2).

3. $\mathbf{s} = \pi\mathbf{z}$ - by Lemma 4.13(3) we have

   $$\Gamma \vdash \mathbf{z} : (\sum, y)(\mathbf{R}, \mathbf{Q})$$

   for some $\mathbf{R}$ and $\mathbf{Q}$. By the inductive assumption we have

   $$\tau(\mathbf{z}) \overset{\Gamma}{\approx} (\sum, y)(\mathbf{R}, \mathbf{Q})$$

   and therefore by Lemma 4.11 we have

   $$\tau(\mathbf{z}) = (\sum, y)(\mathbf{R}', \mathbf{Q}')$$

   where $\mathbf{R} \overset{\Gamma}{\approx} \mathbf{R}'$. By the clause (3) of our lemma we set

   $$\tau(\mathbf{s}) = \mathbf{R}'.$$

   Conditions (1) and (2) follow.

4. $\mathbf{s} = (\pi', y)(\mathbf{Q}, \mathbf{z})$ - by Lemma 4.13(4) we have

$$\Gamma \vdash \mathbf{z} : (\sum, y)(\mathbf{R}, \mathbf{Q})$$

for some $\mathbf{R}$. By the inductive assumption we have

$$\tau(\mathbf{z}) \overset{\Gamma}{\approx} (\sum, y)(\mathbf{R}, \mathbf{Q})$$

By clause (4) of our lemma we set

$$\tau(\mathbf{s}) = \mathbf{Q}(\pi(\mathbf{z})/y)$$

This expression is well-formed and has the property that $\tau(\mathbf{s}) \overset{\Gamma}{\approx} \mathbf{S}$ by Lemma 4.13(4b).

5. $\mathbf{s} = (\lambda, y)(\mathbf{R}, \mathbf{q})$ - by Lemma 4.13(5) we have

$$\Gamma \vdash (\prod, y)(\mathbf{R}, \mathbf{Q}) : Type \quad \Gamma, y : \mathbf{R} \vdash \mathbf{q} : \mathbf{Q}$$

for some $\mathbf{Q}$. By clause (5) of our lemma we set

$$\tau(\mathbf{s}) = (\prod, y)(\mathbf{R}, \tau(\mathbf{q}))$$

Using again the inductive assumption and Lemma 4.3 we conclude that $\tau(\mathbf{s})$ is well-formed and $\tau(\mathbf{s}) \overset{\Gamma}{\approx} \mathbf{S}$ which implies conditions (1) and (2).

6. $\mathbf{s} = ev(\mathbf{f}, \mathbf{r})$ - by Lemma 4.13(6) we have

$$\Gamma \vdash \mathbf{f} : (\prod, y)(\mathbf{R}, \mathbf{Q}) \quad \Gamma \vdash \mathbf{r} : \mathbf{R}$$

for some $\mathbf{R}$ and $\mathbf{Q}$. By the inductive assumption and Lemma 4.3 we have

$$\tau(\mathbf{f}) = (\prod, y)(\mathbf{R}', \mathbf{Q}')$$

where $\mathbf{R}' \overset{\Gamma}{\approx} \mathbf{R}$ and $\mathbf{Q}' \overset{\Delta}{\approx} \mathbf{Q}$ for $\Delta = \Gamma, y : \mathbf{R}$. By clause (6) of our lemma we set

$$\tau(\mathbf{s}) = \mathbf{Q}'(\mathbf{r}/y)$$

By Lemma 4.4 we conclude that this expression is well-formed in Γ and $\tau(\mathbf{s}) \overset{\Gamma}{\approx} \mathbf{S}$.

7. $\mathbf{s} = id(\mathbf{r})$ - by Lemma 4.13(7) we have

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

for some $\mathbf{R}$. By clause (6) of our lemma we set

$$\tau(\mathbf{s}) = eq(\tau(\mathbf{r}), \mathbf{r}, \mathbf{r})$$

By the inductive assumption we have $\tau(\mathbf{r}) \overset{\Gamma}{\approx} \mathbf{R}$ and everything follows as above.

8. $\mathbf{s} = ev'(\mathbf{R}, \mathbf{f}, \mathbf{u})$ - by Lemma 4.13(8) we have

$$\Gamma \vdash \mathbf{Q} : Type \quad \Gamma \vdash \mathbf{f} : (\prod, y)(\mathbf{R}, \mathbf{Q}) \quad \Gamma \vdash \mathbf{u} : eq(\mathbf{R}, \mathbf{r}, \mathbf{r}')$$

for some $\mathbf{Q}$, $\mathbf{r}$ and $\mathbf{r}'$. By the inductive assumption and Lemma 4.3 we have

$$\tau(\mathbf{f}) = (\prod, y)(\mathbf{R}', \mathbf{Q}')$$

where $\mathbf{R} \overset{\Gamma}{\approx} \mathbf{R}'$ and $\mathbf{Q} \overset{\Delta}{\approx} \mathbf{Q}'$ where $\Delta = \Gamma, y : \mathbf{R}$ and

$$\tau(\mathbf{u}) = eq(\mathbf{R}'', \mathbf{r}'', \mathbf{r}''')$$

where $\mathbf{R} \overset{\Gamma}{\approx} \mathbf{R}''$, $\mathbf{r} \overset{\Gamma}{\approx}_{\mathbf{R}} \mathbf{r}''$ and $\mathbf{r}' \overset{\Gamma}{\approx}_{\mathbf{R}} \mathbf{r}'''$. By clause (8) of our lemma we set

$$\tau(\mathbf{s}) = eq(\mathbf{Q}', ev(\mathbf{f}, \mathbf{r}''), ev(\mathbf{f}, \mathbf{r}'''))$$

The inductive assumption implies easily that this expression is well-formed and that $\mathbf{S} \overset{\Delta}{\approx} \tau(\mathbf{s})$ where Δ is as above. Using Lemma 4.5 we conclude that $\mathbf{S} \overset{\Gamma}{\approx} \tau(\mathbf{s})$.

9. $\mathbf{s} = \sigma(\mathbf{u}, \mathbf{v})$ - by Lemma 4.13(9) we have

$$\Gamma \vdash \mathbf{u} : eq(\mathbf{R}, \mathbf{r}, \mathbf{r}') \quad \Gamma \vdash \mathbf{v} : eq(\mathbf{R}, \mathbf{r}, \mathbf{r}'')$$

for some $\mathbf{R}$, $\mathbf{r}$ and $\mathbf{r}'$. By the inductive assumption and Lemma 4.11 we have

$$\tau(\mathbf{u}) = eq(\mathbf{R}_1, \mathbf{r}_1, \mathbf{r}_1') \quad \tau(\mathbf{v}) = eq(\mathbf{R}_2, \mathbf{r}_2, \mathbf{r}_2'')$$

where $\mathbf{R}_2 \overset{\Gamma}{\approx} \mathbf{R} \overset{\Gamma}{\approx} \mathbf{R}_1$, $\mathbf{r}_1 \overset{\Gamma}{\approx}_{\mathbf{R}} \mathbf{r} \overset{\Gamma}{\approx}_{\mathbf{R}} \mathbf{r}_2$ and $\mathbf{r}' \overset{\Gamma}{\approx}_{\mathbf{R}} \mathbf{r}_1'$, $\mathbf{r}_2'' \overset{\Gamma}{\approx}_{\mathbf{R}} \mathbf{r}''$. By clause (9) of our lemma we set

$\tau(\mathbf{s}) = eq((\sum, y)(\mathbf{R}_1, eq(\mathbf{R}_1, \mathbf{r}_1, y)), (pair, y)(eq(\mathbf{R}_1, \mathbf{r}_1, y), \mathbf{r}_1', \mathbf{u}), (pair, y)(eq(\mathbf{R}_1, \mathbf{r}_1, y), \mathbf{r}_2'', \mathbf{v}))$

(of course we might have chosen a slightly different expression for example taking $\mathbf{R}_2$ everywhere instead of $\mathbf{R}_1$).$Usingagaintheinductiveassumptionand Lemma 4.11 we conclude that \tau(\mathbf{s})$ is well-formed in Γ and $\tau(\mathbf{s}) \overset{\Gamma}{\approx} \mathbf{S}$.

10. Similar to (9)
11. Similar to (9)
12. Similar to (9)

5 Theorems about reductions

Before we define the language of $H\lambda$ we need to define a "pre-language". The need for such a two-step process is due to the fact that the units of $H\lambda$ are naturally not sentences in a certain vocabulary but equivalence classes of such sentences. While it is possible to describe the whole structure (i.e. the generating rules and the equivalence relation) in one step it is more convenient

to do it in two steps. The formalism described in this section is a generalization of the formalism of expressions and conversions which appears in many situations in type theory. The classic example of such a formalism is the untyped λ-calculus going back to the work of A. Church in the 1930-ies. Describing syntactic constructions mathematically is never a pleasant affair. The constructions of this section are not very neat but I do not know how to do it in a more elegant way. One way or another one needs to prove things analogous to the Church-Rosser theorem and in order to proof statements about something one first need to define this something formally.

Informally speaking we consider the following situation. There is some kind of a language with the notion of free and bound variables and an unknown collection of special symbols. Unknown because we will extend the language as we move on and we want the results of this section to remain valid. This unknown collection includes the special symbols $\lambda, ev, pair, \pi, \pi', id, ev'$ which will appear in the next section as the basic term constructors of the homotopy λ-calculus. Now one considers a number of rearrangements of the expressions in this language which are called reductions. They are directional and we can write something like $\mathbf{E} \stackrel{(v,6)}{\longrightarrow} \mathbf{F}$ to say that the expression $\mathbf{E}$ can be reduced (rearranged) at place v according to the rule No.6 and the result of this reduction is $\mathbf{F}$. One says that $\mathbf{E} \cong \mathbf{E}'$ if $\mathbf{E}$ and $\mathbf{E}'$ can be reduced to a same expression $\mathbf{F}$ i.e. if one has $\mathbf{E} \to \mathbf{F} \leftarrow \mathbf{E}'$ where the arrows denote an arbitrary sequence of reductions performed in the correct direction. The main result we need is Theorem 5.2 saying that $\cong$ is an equivalence relation i.e. that it is transitive. There is as far as I know absolutely no reason why this should be true other than the fact that the particular form of reductions was chosen by trial and error to make it work. In our case we have 8 kinds of reductions. Five of them are standard in the dependent type theories and three additional ones are peculiar to the homotopy λ-calculus. In the case of the classical Church-Rosser theorem one has to consider only one kind of reduction and also does not have to worry about the "unknown" part of the language. To be fair I should say that this original kind of reduction, called β-reduction, is the most troublesome one of the eight.

We now consider the following very particular situation. Assume that Sp contains the set $Sp_0 = \{\lambda, ev, pair, \pi, \pi', id, ev'\}$. Let S be the subset in $T(Sp, var)$ which consists of T such that for any $v \in v(T)$ the following conditions hold:

1. if $l(v) = (\lambda, y)$ then $val(v) = 2$
2. if $l(v) = ev$ then $val(v) = 3$
3. if $l(v) = (pair, y)$ then $val(v) = 3$
4. if $l(v) = \pi$ then $val(v) = 1$
5. if $l(v) = (\pi', y)$ then $val(v) = 2$
6. if $l(v) = id$ then $val(v) = 1$
7. if $l(v) = ev'$ then $val(v) = 3$.

The class S has the following obvious properties:

1. S is closed under α-equivalence,

2. for $T_1, \ldots, T_3 \in S$ the trees $ev(T_1, T_2, T_3)$, $(\lambda, y)(T_1, T_2)$, $(pair, y)(T_1, T_2, T_3)$, $\pi(T_1)$, $(\pi', y)(T_1, T_2)$, $id(T_1)$ and $ev'(T_1, T_2, T_3)$ are in S,

3. for $T, T' \in S$ and $v \in v(T)$ the trees $[v]$ and $T(T'/[v])$ are in S,

4. for $T, T' \in S$ and $y \in var(T) - bnd(T)$ the tree $T(T'/y)$ is in S.

Let $T \in S$ and $v \in v(T)$.

Lemma 5.1 *[**alpha**] Let $T, T' \in S$ be α-equivalent. Then form any $v \in v(T) = v(T')$ and any $i = 1, \ldots, n$ one has $r_i(v)_T \Leftrightarrow r_i(v)_{T'}$ and if $r_i(v)$ holds then $r_i(T, v)$ is α-equivalent to $r_i(T', v)$.*

Proof: Clear.

For $n = 1, \ldots, 8$ let us write $T \stackrel{n}{\to} T'$ if there exists $v \in v(T)$ such that $r_n(v)$ and $r_n(T, v) = T'$ where the equality holds up to an α-equivalence. We will also write $T \stackrel{0}{\to} T'$ if T and T' are α-equivalent. Let us write $T \stackrel{n^k}{\to} T'$ if there exist a sequence $T = T_0 \stackrel{n}{\to} T_1 \stackrel{n}{\to} \ldots \stackrel{n}{\to} T_k = T'$. Let us write $T \stackrel{I}{\to} T'$ where $I = (n_1, \ldots, n_k)$ if there exists a sequence $T = T_0 \stackrel{n_1}{\to} \ldots \stackrel{n_k}{\to} T_k = T'$. Finally let us write $T \to T'$ if there exists I such that $T \stackrel{I}{\to} T'$.

Theorem 5.2 *[**mainthis**] Let $T, T', T'' \in S$ and $T \to T'$, $T \to T''$. Then there exists $T''' \in S$ such that*

$$\begin{array}{ccc} T & \longrightarrow & T'' \\ \downarrow & & \downarrow \\ T' & \longrightarrow & T''' \end{array}$$

One can easily see that it is sufficient to consider the case when the first reduction is an elementary one i.e. when $T \stackrel{n}{\to} T'$ for some $n = 1, \ldots, 8$. This gives us eight cases to consider. We start with the following special lemma and then proceed to consider all eight cases one after another. It turns out to be more convenient to consider the first reduction to be of the form $T \stackrel{n^k}{\to} T'$.

Lemma 5.3 *[**chro**] Let $T \stackrel{1}{\to} T'$ and $T \stackrel{1^k}{\to} T''$. Then there exists T''' and l, m such that*

$$\begin{array}{ccc} T & \stackrel{1^k}{\longrightarrow} & T'' \\ {\scriptstyle 1}\downarrow & & \downarrow{\scriptstyle 1^m} \\ T' & \stackrel{1^l}{\longrightarrow} & T''' \end{array}$$

Proof: This is basically the classical "strip lemma" from the λ-calculus in our context. Later.

Proposition 5.4 *[**case1**] Let $T \xrightarrow{1^k} T'$ and $T \xrightarrow{I} T''$ where $I = (n_1, \ldots, n_l)$. Then there exists T''', m and J such that*

$$\begin{array}{ccc} T & \xrightarrow{I} & T'' \\ {\scriptstyle 1^k}\downarrow & & \downarrow{\scriptstyle 1^m} \\ T' & \xrightarrow{J} & T''' \end{array}$$

Proof: We start with the following lemma.

Lemma 5.5 *[**case1n**] Let $T \xrightarrow{1} T'$ and $T \xrightarrow{n} T''$ where $n \neq 1$. Then there exist T''' , k and $e = 0, 1$ such that*

$$\begin{array}{ccc} T & \xrightarrow{n} & T'' \\ {\scriptstyle 1}\downarrow & & \downarrow{\scriptstyle 1^e} \\ T' & \xrightarrow{n^k} & T''' \end{array}$$

Proof: Let v be the vertex of the $T \to T'$ reduction and w be the vertex of $T \to T''$ reduction. Since $n \neq 1$ we have $l(v) \neq l(w)$ and therefore $v \neq w$. If $w \notin [v]$ and $v \notin [w]$ then it is clear that two reductions "commute" and we can take $k = 1$ and $e = 1$. We will now use the notations used in the description of $r_i(T, -)$. If $w \in [v]$ there are the following cases. If $n \neq 2$ then the form of $l(w)$ implies that we must have $w \in \mathbf{R}$, $w \in \mathbf{q}$ or $w \in \mathbf{r}$. In the first case one can take $k = 0$ and $e = 1$. In the second case the n-reduction occurs inside $\mathbf{q}$ and we can take $k = 1$ and $e = 1$. In the third case the n-reduction occurs inside $\mathbf{r}$ which gets reproduced in T' as many times as there were occurrences of y in $\mathbf{q}$. We need to take k to be this number and $e = 1$.

If $n = 2$ there is an extra possibility that $[v]$ is of the form $ev((\lambda, y)(\mathbf{R}, ev(f, y)), \mathbf{r})$. Then we have:

$$\begin{array}{ccc} ev((\lambda, y)(\mathbf{R}, ev(f, y)), \mathbf{r}) & \xrightarrow{2} & ev(f, \mathbf{r}) \\ {\scriptstyle 1}\downarrow & & \downarrow{\scriptstyle 0} \\ ev(f, \mathbf{r}) & \xrightarrow{0} & ev(f, \mathbf{r}) \end{array}$$

i.e. $T' = T''$ and we take $k = 0$ and $e = 0$.

Consider now the case $v \in [w]$. If $n \neq 2$ then v lies in one of the subtrees of $[w]$ which either disappears completely in the n-conversion or is reproduced in the resulting tree once without change. In the first case we need to take $e = 0$ and $k = 1$ and in the second case we need to take $e = 1$ and $k = 1$. If $n = 2$ there is also a possibility that $[w]$ is of the form $(\lambda, y)(\mathbf{R}, ev((\lambda, z)(\mathbf{Q}, \mathbf{s}), y)))$. Then we have:

$$\begin{array}{ccc} (\lambda, y)(\mathbf{R}, ev((\lambda, z)(\mathbf{Q}, \mathbf{s}), y))) & \xrightarrow{2} & (\lambda, z)(\mathbf{Q}, \mathbf{s}) \\ {\scriptstyle 1}\downarrow & & \downarrow{\scriptstyle 0} \\ (\lambda, y)(\mathbf{R}, \mathbf{s}(y/z)) & \xrightarrow{0} & (\lambda, y)(\mathbf{R}, \mathbf{s}(y/z)) \end{array}$$

i.e. T' is α-equivalent to T'' and we take $k = 0$ and $e = 0$.

Let us now prove Proposition 5.4. Will have to do a nested induction. First proceed by induction on a_1 where a_1 is the number of 1's in I. For $a_1 = 0$ proceed by induction on k. For $k = 0$ there is nothing to prove. The inductive step follows immediately from Lemma 5.5 since $e = 1$ or $e = 0$. To make the inductive step in a_1 one uses Lemma 5.3.

Proposition 5.6 *[**case2**] Let $T \stackrel{2^k}{\to} T'$ and $T \stackrel{I}{\to} T''$ where $I = (n_1, \ldots, n_l)$. Then there exists T''', m and J such that*

$$\begin{array}{ccc} T & \stackrel{I}{\longrightarrow} & T'' \\ {\scriptstyle 2^k}\downarrow & & \downarrow{\scriptstyle 2^m} \\ T' & \stackrel{J}{\longrightarrow} & T''' \end{array}$$

Proof: We start with the following lemma.

Lemma 5.7 *[**case2n**] Let $T \stackrel{2}{\to} T'$ and $T \stackrel{n}{\to} T''$ where $n \neq 1$. Then there exist T''' , $f = 0, 1$ and $e = 0, 1$ such that*

$$\begin{array}{ccc} T & \stackrel{n}{\longrightarrow} & T'' \\ {\scriptstyle 2}\downarrow & & \downarrow{\scriptstyle 2^e} \\ T' & \stackrel{n^f}{\longrightarrow} & T''' \end{array}$$

Proof: Let v be the vertex of $T \to T'$ reduction and w the vertex of $T \to T''$ reduction. We have $l(v) = (\lambda, y)$. We have the following four cases $v = w$, $v \in [w] - w$, $w \in [v] - v$ and the remaining case when v and w are independent. The first case may only occur when $n = 2$ and we take $e = f = 1$. In the second case the form of the reductions implies that v occurs in a subtree of $[w]$ which either disappears in the n-conversion or is passed down without change. Correspondingly we take $f = 1$ and $e = 0$ or $e = 1$. In the third case we again have the same situation i.e. w lies in $\mathbf{R}$ part of $[v]$ which disappears or in the $\mathbf{f}$ part of $[v]$ which is passed down without change. Correspondingly we take $e = 1$ and $f = 0$ or $f = 1$. Finally in the fourth case when v and w are independent we take $e = f = 1$.

To prove the proposition start with the induction by the number a_1 of 1's in I. If $a_1 = 0$ then the statement follows easily from Lemma 5.7. To make the inductive step in a_1 one uses Lemma 5.5 with $n = 2$.

Proposition 5.8 *[**case3**] Let $T \stackrel{2^k}{\to} T'$ and $T \stackrel{I}{\to} T''$ where $I = (n_1, \ldots, n_l)$. Then there exists T''', m and J such that*

$$\begin{array}{ccc} T & \stackrel{I}{\longrightarrow} & T'' \\ {\scriptstyle 3^k}\downarrow & & \downarrow{\scriptstyle 3^m} \\ T' & \stackrel{J}{\longrightarrow} & T''' \end{array}$$

Proof: We start with the following lemma.

Lemma 5.9 *[**case3n**] Let $T \stackrel{3}{\to} T'$ and $T \stackrel{n}{\to} T''$ where $n \neq 1$. Then there exist T''' , $f = 0, 1$ and $e = 0, 1$ such that*

$$\begin{array}{ccc} T & \stackrel{n}{\longrightarrow} & T'' \\ {\scriptstyle 3}\downarrow & & \downarrow{\scriptstyle 3^e} \\ T' & \stackrel{n^f}{\longrightarrow} & T''' \end{array}$$

Proof: We use similar reasoning as in the proofs of Lemmas 5.5 and 5.7. The only case which one has to look closely at is the case $n = 5$. Let v be the $T \to T'$ conversion vertex and w the $T \to T''$ conversion vertex. There are two non-trivial possibilities. If $v \in [w] - w$ lies there non-trivially we have $[w] = (pair, y)(\pi((pair, z)(\mathbf{Q}, \mathbf{r}, \mathbf{q})), \pi'((pair, z)(\mathbf{Q}, \mathbf{r}, \mathbf{q})))$ we have:

$$\begin{array}{ccc} (pair, y)(\mathbf{S}, \pi((pair, z)(\mathbf{Q}, \mathbf{r}, \mathbf{q})), \pi'((pair, z)(\mathbf{Q}, \mathbf{r}, \mathbf{q}))) & \stackrel{5}{\longrightarrow} & (pair, z)(\mathbf{Q}, \mathbf{r}, \mathbf{q}) \\ {\scriptstyle 3}\downarrow & & \downarrow \\ (pair, y)(\mathbf{S}, \mathbf{r}, \pi'((pair, z)(\mathbf{Q}, \mathbf{r}, \mathbf{q}))) & \stackrel{4}{\longrightarrow} & (pair, y)(\mathbf{S}, \mathbf{r}, \mathbf{q}) \end{array}$$

6 Semantics of $H\lambda_0$

Let us describe now the semantics of $H\lambda_0$. We will assume that we have chosen spaces $X_1, \ldots, X_n$ corresponding to the generating types $T_1, \ldots, T_n$.

As noted above we want to construct for any $\Gamma \vdash$ a tower of fibrations:

$$E(\Gamma) \to E(\Gamma_{\leq m-1}) \to \ldots \to E(\Gamma_{\leq 0}) = pt$$

for any sequent of the form $\Gamma \vdash \mathbf{R} : Type$ a fibration

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

and for any sequent of the form $\Gamma \vdash \mathbf{r} : \mathbf{R}$ a section $s(\mathbf{r})$ of $p_{\mathbf{R}}$. We also want to check that for any sequent of the form $\Gamma \vdash \mathbf{r} \equiv \mathbf{r}' : \mathbf{R}$ one has $s(\mathbf{r}) = s(\mathbf{r}')$.

We will be doing inductively the following:

1. Assuming that $E(\Gamma)$ is constructed and $\Gamma \vdash \mathbf{S} : Type$ we will construct $p_{\mathbf{S}} : E(\Gamma, \mathbf{S}) \to E(\Gamma)$ and define a fibration $p_{\mathbf{S}} : E(\Gamma, \mathbf{S}) \to E(\Gamma)$ setting $E(\Gamma, \mathbf{S}) = E(\Gamma) \times_{\mathbf{S}} \tilde{\Omega}$.
2. Assuming the same as above and in addition that $\Gamma \vdash \mathbf{s} : \mathbf{S}$ we will construct $\mathbf{s} : E(\Gamma) \to \tilde{\Omega}$ and define a section $s(\mathbf{s})$ of $p_{\mathbf{S}}$ as the map $E(\Gamma) \to E(\Gamma, \mathbf{S})$ corresponding to $\mathbf{s}$.

3. Assuming the same as above and in addition that $\Gamma \vdash \mathbf{s} \equiv \mathbf{s}' : \mathbf{S}$ we will show that the sections $s(\mathbf{s})$ and $s(\mathbf{s}')$ coincide. There is no need to consider sequents of the form $\Gamma \vdash \mathbf{S} \equiv \mathbf{S}' : Type$ separately.

For Γ of the form $T_1, \ldots, T_n : Type$ we set $E(\Gamma) = \Omega^n$ and to define $E(\Gamma_{\leq j+1})$ from $E(\Gamma_{\leq j})$ we set $E(\Gamma_{\leq j+1}) = E(\Gamma_{\leq j}, \mathbf{R}_{j+1})$.

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:

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

   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

$$\begin{array}{ccccc} E(\Gamma, \mathbf{R}_i) & \longrightarrow & E(\Gamma_{\leq i}, \mathbf{R}_i) & \longrightarrow & E(\Gamma_{<i}, \mathbf{R}_i) \\ \downarrow & & \downarrow & & \downarrow \\ E(\Gamma) & \longrightarrow & E(\Gamma_{\leq i}) & \longrightarrow & E(\Gamma_{<i}) \end{array}$$

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:

$$\begin{array}{ccc} E(\Gamma, \mathbf{R}, \mathbf{Q}, \sum y : \mathbf{R}.Q) & \longrightarrow & E(\Gamma, \sum y : \mathbf{R}.Q) \\ \downarrow & & \downarrow \\ E(\Gamma, \mathbf{R}, \mathbf{Q}) & \longrightarrow & E(\Gamma) \end{array}$$

   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

$$E(\Gamma, \sum y : \mathbf{R}.\mathbf{Q}) = E(\Gamma, \mathbf{R}, \mathbf{Q}) \stackrel{p_{\mathbf{Q}}}{\to} E(\Gamma, \mathbf{R}) \stackrel{p_{\mathbf{R}}}{\to} E(\Gamma)$$

   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

$$\begin{array}{ccc} E(\Gamma, \mathbf{Q}[\pi \mathbf{u}/y]) & \longrightarrow & E(\Gamma, \mathbf{R}, \mathbf{Q}) \\ \downarrow & & \downarrow \\ E(\Gamma) & \stackrel{s(\pi \mathbf{u})}{\longrightarrow} & E(\Gamma, \mathbf{R}) \end{array}$$

   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

$$[\mathbf{prodsem1}] E(\Gamma, \prod y : \mathbf{R}.\mathbf{Q}) \to E(\Gamma) \tag{20}$$

   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 (20) 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

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

and

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

respectively. Observe that there is a pull-back square

$$\begin{array}{ccc} E(\Gamma, \mathbf{Q}[\mathbf{r}/y]) & \longrightarrow & E(\Gamma, \mathbf{R}, \mathbf{Q}) \\ \big\downarrow & & \big\downarrow \\ E(\Gamma) & \xrightarrow{s(\mathbf{r})} & E(\Gamma, \mathbf{R}) \end{array} \qquad [\mathbf{anothersq}] \tag{21}$$

and we need to get a section of the left hand side vertical arrow.
By definition of $E(\Gamma, \prod y : \mathbf{RQ})$ 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 (21). 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

$$[\mathbf{needsec}]E(\Gamma, eq(\mathbf{Q}; \mathbf{q}_1, \mathbf{q}_2), eq(\mathbf{R}; \mathbf{r}[\mathbf{q}_1/v], \mathbf{r}[\mathbf{q}_2/v])) \to E(\Gamma, eq(\mathbf{Q}; \mathbf{q}_1, \mathbf{q}_2)) \tag{22}$$

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 (22).

(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

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

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.

Example 6.1 [**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 $(X_1, X_2; f : X_1 \to X_2)$.

7 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 $(T_1, \dots, T_n; \dots)$ instead of $(\dots)$

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 7.3 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 7.1 *[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 (13) 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 7.2 *[**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 7.3 *[**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 7.4 *[**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 7.5 *[**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 7.1 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 7.6 *[**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 7.6. 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

$$[\mathbf{univdef}]\mathbf{Un}(f) = \prod v, v' : T_2.\mathbf{Eq}(\mathbf{haeq}(f, v, v')) \tag{23}$$

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 1.5.

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 $\Gamma = (\mathcal{U}, \tilde{\mathcal{U}}; \upsilon : \tilde{\mathcal{U}} \to \mathcal{U})$ 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

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

Theorem 7.7 *[**main2**] For any* $\mathbf{R}$ *and* $\mathbf{S}$ *as above there is a term expression* $\mathbf{c}$ *such that*

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

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

Proof: Later.

Theorem 7.7 shows that for univalent υ we may treat $\mathbf{Class}(\mathbf{R}, \mathbf{S}, \upsilon)$ 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 $\mathbf{Class}(\mathbf{R}, \mathbf{S}, \upsilon)$ 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 7.7 asserts that in this case the space of all possible ways to achieve such a classification it is contractible.

Set

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

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

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

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

where π and π' are the term constructors of (9) which take $\langle u, f\rangle$ to u and f respectively.

Set in Γ:

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

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

According to Theorem 7.7 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

$$\begin{array}{ccc} \tilde{U} & \longrightarrow & \tilde{U} \\ {\scriptstyle\Delta}\downarrow & & \downarrow{\scriptstyle p} \\ \tilde{U} \times_U \tilde{U} & \xrightarrow{Eq} & U \end{array}$$

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 7.8 *[**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? ????

8 Basic layer - generalized models

For a category $\mathcal{C}$ and two morphisms $f : X \to Y$, $g : X' \to Y$ we let $Hom(f, g)$ denote the set of morphisms from X to X' over Y. We also write $Tot(f)$ for the source of a morphism f and $Base(f)$ for the target. A set-based category is an internal category in the category of sets i.e. a set with a structure.

Define an H_{00}-category as the following collection of data:

1. A set-based category $\mathcal{C}$.
2. A final object pt in $\mathcal{C}$.
3. A subset F of the set $Mor(\mathcal{C})$ of morphisms in $\mathcal{C}$ whose elements are called fibrations such that:
   (a) F contains isomorphisms,
   (b) for each $X \in \mathcal{C}$ the map $t_X : X \to pt$ is in F,
   (c) F is closed under compositions.
4. For any $f : X \to Y$ and any $p : Y' \to Y$ with $p \in F$ a pull-back square

$$\begin{array}{ccc} Tot(f^*(p)) & \xrightarrow{p^*(f)} & Y' \\ {\scriptstyle f^*(p)}\downarrow & & \downarrow{\scriptstyle p} \\ X & \xrightarrow{f} & Y \end{array}$$

   such that $f^*(p) \in F$. We will write $X \times X$ for $Top(t_X^*(t_X))$.

5. For any $X \xrightarrow{p} Y \xrightarrow{q} Z$ with $p, q \in F$ a morphism $q_*(p) : Tot(q_*(p)) \to Z$ in F and a morphism $\eta : q^*q_*(p) \to p$ over Y such that for any $r : W \to Z$ the map

$$Hom(r, q_*p) \to Hom(q^*r, q^*q_*p) \to Hom(q^*r, p)$$

is a bijection.

6. A functor $E : \mathcal{C} \to \mathcal{C}$.

7. Three natural transformations $i : Id \to E$ and $d_0, d_1 : E \to Id$ such that:

   (a) $d_0 \circ i = d_1 \circ i = id$,

   (b) for any X the map $d_1(X) \times d_2(X) : E(X) \to X \times X$ is in F.

Note that the condition on $q_*(p)$ and η means simply that $p \to q_*(p)$ is a functor from Fib/Y to Fib/Z which is right adjoint to the pull-back functor $r \mapsto q^*(r)$.

Lemma 8.1 *[**cartcl**] Any H_{00}-category is Cartesian closed i.e. it has finite products and internal hom-objects.*

Proof: Since all morphisms t_X are fibrations and $\mathcal{C}$ has fiber products for corners where at least one side is a fibration we know that any H_{00}-category has finite products. For $X, Y \in \mathcal{C}$ define $\underline{Hom}(X, Y)$ as $Tot((t_X)_* t_X^*(t_Y))$. One can easily see using the properties of t_X^* and $(t_X)^*$ that it is indeed an internal hom-object.

Example 8.2 [**etriv**] Consider the case when $Id \to E$ is a fibration. Then every morphism is a fibration. Indeed any morphism $f : X \to Y$ is the composition of the graph inclusion $X \to X \times Y$ with the projection $X \times Y \to Y$. The projection is a fibration in any H_{00}-category. The graph inclusion is the pull-back of the diagonal $Y \to Y \times Y$ and the diagonal is a fibration as a composition of two fibrations $Y \to E(Y) \to Y \times Y$.

Example 8.3 Let us say that H_{00}-category is discreet if $Id \to E$ is an isomorphisms. Such H_{00}-categories are up to an equivalence the same as categories with finite limits an such that for any $f : X \to Y$ the pull-back functor $f^* : \mathcal{C}/Y \to \mathcal{C}/X$ has a right adjoint. For example any small elementary topos is a discreet H_{00}-category.

Example 8.4 [**Kan**] Let $\mathcal{C}$ be the category of Kan simplicial sets in some universe U, F be the class of Kan fibrations, $EX = \underline{Hom}(\Delta^1, X)$ be the space of free paths on X and $X \to EX$, $EX \to X \times X$ be the obvious morphisms. Then $\mathcal{C}$ is a H_{00}-category with respect to the usual fiber products and the direct images $q_*(p)$ defined as follows. Let $X \xrightarrow{p} Y \xrightarrow{q} Z$ be a pair of fibrations. Consider the relative internal hom-object $\underline{Hom}_Z(Y, X)$ and the map $\phi : \underline{Hom}_Z(Y, X) \to \underline{Hom}_Z(Y, Y)$ define by the composition with p. Then $Tot(q_*(p)) = \phi^{-1}(Id_Y)$. It is pretty clear how to define η and verify

the main condition on $q_*(p)$. To verify that $q_*(p)$ is a fibration note that by adjunction fillings of the square

$$\begin{array}{ccc} A & \longrightarrow & q_*(p) \\ \downarrow & & \downarrow \\ B & \longrightarrow & Z \end{array}$$

are in one to one correspondence with fillings of the square

$$\begin{array}{ccc} A \times_Z Y & \longrightarrow & X \\ \downarrow & & \downarrow \\ B \times_Z Y & \longrightarrow & Y \end{array}$$

If $A \to B$ is a trivial cofibration then $A \times_Z Y \to B \times_Z Y$ is because $q : Y \to Z$ is a fibration. Hence, if $p : X \to Y$ is a fibration then so is $q_*(p)$. The fiber of this fibration over $z \in Z_0$ is the space of sections of the fibration $X_z \to Y_z$. This example clearly extends to the categories of fibrant objects in model categories with appropriate properties.

For a fibration $p : X \to Y$ we let $sec(p)$ denote the set of sections of p. For a fibration $p : X \to Y$ and a pair of sections s, s' we let $P(p; s, s')$ denote the fibration given by the pull-back square:

$$\begin{array}{ccc} Tot(P(p;s,s')) & \longrightarrow & EX \\ {\scriptstyle P(p;s,s')}\downarrow & & \downarrow \\ Y & \xrightarrow{s \times s'} & X \times X \end{array}$$

Note that in the case of $\mathcal{C}$ as in Example 8.4 we $P(p; s, s')$ is the fibration over Y whose fiber over $y \in Y_0$ is the space of paths from $s(y)$ to $s'(y)$ in $X_y = p^{-1}(y)$. ne to. need relative E.

An interpretation of $H\lambda_{00}$ in an H_{00}-category $\mathcal{C}$ is a triple of maps

$$\Phi_0 : S_0 \to ob(\mathcal{C})$$

$$\Phi_1 : S_1 \to Fin(\mathcal{C})$$

$$\Phi_2 : S_2 \to Mor(\mathcal{C})$$

which satisfy a number of conditions. To simplify the notation we will write Φ without the index since the index is clear from the form of the argument. The conditions are as follows:

1. One has:

$$Base(\Phi(\Gamma \vdash \mathbf{S} : Type)) = \Phi(\Gamma \vdash)$$

$$Tot(\Phi(\Gamma \vdash \mathbf{S} : Type)) = \Phi(\Gamma, y : \mathbf{S} \vdash)$$

$$\Phi(\Gamma \vdash \mathbf{s} : \mathbf{S}) \in sec(\Phi(\Gamma \vdash \mathbf{S} : Type))$$

   Note that these conditions make sense by Lemmas **??**,**??**.

2. One has:

$$\Phi(\Gamma \vdash (\sum, y)(\mathbf{R}, \mathbf{Q})) = \Phi(\Gamma \vdash \mathbf{R} : Type) \circ \Phi(\Gamma, y : \mathbf{R} \vdash \mathbf{Q} : Type)$$

$$\Phi(\Gamma \vdash (\prod, y)(\mathbf{R}, \mathbf{Q})) = (\Phi(\Gamma \vdash \mathbf{R} : Type))_*(\Phi(\Gamma, y : \mathbf{R} \vdash \mathbf{Q} : Type))$$

$$\Phi(\Gamma \vdash eq(\mathbf{R}, \mathbf{r}, \mathbf{r}')) =$$

We can now formulate the "semantics theorem" for $H\lambda_{00}$. Recall that GT is the set of generating types of $H\lambda_{00}$.

Theorem 8.5 *[**semantics**] Let $\mathcal{C}$ be an H_{00}-category. Then for any map $X : GT \to ob(\mathcal{C})$ there exists a unique interpreation Φ of $H\lambda_{00}$ in $\mathcal{C}$ such that $\Phi(\vdash) = pt$ and for any $T \in GT$,*

$$\Phi(\vdash T : Type) = (X(T) \to pt).$$

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

$$\Gamma = c_1 : \mathbf{R}_1, \ldots, c_m : \mathbf{R}_m$$

by induction on m. Such an induction makes sense because the rules of lambda calculus imply that for any context Γ the sequences $\Gamma_{\leq j} = c_1 : \mathbf{R}_1, \ldots, c_j : \mathbf{R}_j$ 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 $c \in var - \{c_1, \ldots, c_m\}$ 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 (**??**). 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 (**??**). 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

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

   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 (**??**). 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 8.6 *[**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

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

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
$$E(\Gamma) \stackrel{Id \times e(\mathbf{r})}{\longrightarrow} E(\Gamma) \times M(\mathbf{R}) \stackrel{e(\mathbf{q})}{\rightarrow} E(\mathbf{Q})$$
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 $\mathcal{C}(T_1, \dots, T_n : Type)$ are in one to one correspondence with contexts with the base $T_1, \dots, T_n : Type$ 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 (**??**) the category $\mathbf{C}(\Gamma)$ can be identified with the slice category $\mathcal{C}(T_1, \dots, T_n : Type)/B$ where B is the object of $\mathcal{C}(T_1, \dots, T_n : Type)$ given by $(\mathbf{R}_1, \dots, \mathbf{R}_j)$ 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.

9 Homotopy λ-calculus - logic layer

10 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. $\upsilon : \tilde{\mathcal{U}} \to \mathcal{U}$, $a : \mathbf{Un}(\upsilon)$
2. The basic closeness axioms $a_{prod} : \mathbf{Cl\_prod}(\upsilon)$, $a_{sum} : \mathbf{Cl\_sum}(\upsilon)$, $a_{eq} : \mathbf{Cl\_eq}(\upsilon)$.
3. Define the empty type $\emptyset = \emptyset(\upsilon) = \prod y : \mathcal{U}.\upsilon^{-1}(y)$. Require $\emptyset$ to be small i.e. $a_\emptyset : \mathbf{Class}(\emptyset, \upsilon)$.
4. Define $\mathcal{U}_n = \{u : \mathcal{U}, a : \mathbf{Lv}_n(\upsilon^{-1}(u))\}$ i.e. $\mathcal{U}_n$ is the part of the universe span by n-types. Require $\mathcal{U}_0$ to be small i.e. $a_{prop} : \mathbf{Class}(\mathcal{U}_0, \upsilon)$. This is an analog of "impredicativity of Prop".

5. If desired add the boolean axiom $a_{bool} : \prod u : \mathcal{U}_0.(((\upsilon^{-1}(u) \to \emptyset) \to \emptyset) \to \upsilon^{-1}(u))$.

6. Impose an analog of Proposition **??** 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 $u : \mathcal{U}$, $f : \upsilon^{-1}(u) \to \mathcal{U}$ fix

$$univ(u, f) : \mathcal{U}, \qquad \Theta(u, f) : \upsilon^{-1}(univ(u, f)) \to \mathcal{U}.$$

Set

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

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

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

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

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

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

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 $\upsilon : \tilde{\mathcal{U}} \to \mathcal{U}$ 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 (23)).

11 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

$$\mathbf{a} : eq_{\mathbf{R}}(\mathbf{a}, \mathbf{b}) \tag{24}$$

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 (24) 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 $\mathbf{a}$ and $\mathbf{b}$ 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.

12 The leftovers

Remark 12.1 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:

$$[\mathbf{eqdef1}]\{f : T_1 \to T_2, b : T_2 \to T_1, \phi : eq_{T_1 \to T_1}(Id, bf), \psi : eq_{T_2 \to T_2}(Id, fb), \alpha : eq_{eq_{T_1 \to T_2}(f,fbf)}(\psi * f, f * \phi)\} \tag{25}$$

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 (25) 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 (13). 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

$$[\mathbf{inv1}]\frac{\Gamma \vdash \phi : eq_Q(x, y)}{\Gamma \vdash \phi^{-1} : eq_Q(y, x), \Theta : eq_{eq_Q(x,x)}(\phi^{-1}\phi, Id)} \tag{26}$$

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 (26) 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 $\sum_{y:Q} S$.

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 12.2 *[**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* $X \to Hom(Hom(X, \{0,1\}), \{0,1\})$.

The lemmas formulated below are simple corollaries of the (sem-)formal description of the syntax which is given in the appendix. The formal description in terms of rules is needed only to ensure that a language satisfying these lemmas exists. In practice the lemmas provide a more convenient description than the rules. When we formulate a lemma for $H\lambda$ without an index it means that it holds in all three systems $H\lambda_i$. The equality sign everywhere below is used in the notational sense i.e. to indicate that both sides are syntactically equal.

Lemma 12.3 *[**context0**] A sentence of the form $c_1 : \mathbf{R}_1, \dots, c_m : \mathbf{R}_m \vdash$ is a sequent in $H\lambda$ iff one of the following conditions holds:*

1. $m = n = 0$ *i.e. the sentence is of the form* $\vdash$,

2. $m = 0$, $n \geq 1$, $T_1, \ldots, T_n \in GT$ *and* $T_i \neq T_j$ *for* $i \neq j$,

3. $m > 0$, $c_1 : \mathbf{R}_1, \ldots, c_{m-1} : \mathbf{R}_{m-1} \vdash \mathbf{R}_m : Type$ *is a sequent and* $c_m \in var - \{c_1, \ldots, c_{m-1}\}$.

Lemma 12.4 *[***context1***] Let* $\Gamma = c_1 : \mathbf{R}_1, \ldots, c_m : \mathbf{R}_m$ *be such that* $\Gamma \vdash$ *is a sequent in* $H\lambda$. *Then one has:*

1. $T_i \neq T_j$ *for* $i \neq j$

2. $c_i \neq c_j$ *for* $i \neq j$

3. *let* $j < m$ *and* $\Gamma_{\leq j} = c_1 : \mathbf{R}_1, \ldots, c_j : \mathbf{R}_j$ *then* $\Gamma_{\leq j} \vdash \mathbf{R}_{j+1} : Type$ *is a sequent.*

Lemma 12.5 *[***type0***] Let* Γ *be a sentence of the standard form (***??***). A sentence of the form* $\Gamma \vdash \mathbf{S} : Type$ *is a sequent in* $H\lambda_0$ *iff* $\Gamma \vdash$ *is a sequent and one of the following conditions holds:*

1. $\mathbf{S} \in \{T_1, \ldots, T_n\}$

2. $\mathbf{S} = \sum y : \mathbf{R}.\mathbf{Q}$ *and* $\Gamma, y : \mathbf{R} \vdash \mathbf{Q} : Type$ *is a sequent,*

3. $\mathbf{S} = \prod y : \mathbf{R}.\mathbf{Q}$ *and* $\Gamma, y : \mathbf{R} \vdash \mathbf{Q} : Type$ *is a sequent,*

4. $\mathbf{S} = eq(\mathbf{R}; \mathbf{r}, \mathbf{r}')$ *and* $\Gamma \vdash \mathbf{r} : \mathbf{R}$ *and* $\Gamma \vdash \mathbf{r}' : \mathbf{R}$ *are sequents.*

Lemma 12.6 *[***term0***] Let* Γ *be a sentence of the standard form (***??***). A sentence of the form* $\Gamma \vdash \mathbf{s} : \mathbf{S}$ *is a sequent in* $H\lambda_0$ *iff* $\Gamma \vdash \mathbf{S} : Type$ *is a sequent and one of the following conditions holds:*

1. $\mathbf{s} : \mathbf{S} = c_j : \mathbf{R}_j$ *for some* $j = 1, \ldots, m$

2.

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 $T_1 \ldots T_n$ 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 $\Gamma = (T_1, \ldots, T_n : Type)$. A model M of Γ is a collection of topological spaces $X_i = M(T_i)$ for $i = 1, \ldots, n$. 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 $(T : Type)$ a type expression **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 $End(X) \to End(X')$.

Lemma 12.7 *[**typeeq**] A sentence $\Gamma \vdash \mathbf{S} \equiv \mathbf{S}' : Type$ is a sequent iff one of the following mutually exclusive conditions holds:*

1. *Condition (1) of Lemma 4.10 holds for $\mathbf{S}$ and $\mathbf{S} = \mathbf{S}'$*
2. *Condition (2) of Lemma 4.10 holds for $\mathbf{S}$ and one has:*
   (a) $\mathbf{S}' = \sum y' : \mathbf{R}'.\mathbf{Q}'$
   (b) $\Gamma \vdash \mathbf{R} \equiv \mathbf{R}' : Type$ *and* $\Gamma \vdash \mathbf{Q} \equiv \mathbf{Q}'(y/y') : Type$ *are sequents.*
3. *Condition (3) of Lemma 4.10 holds for $\mathbf{S}$ and one has:*
   (a) $\mathbf{S}' = \prod y' : \mathbf{R}'.\mathbf{Q}'$
   (b) $\Gamma \vdash \mathbf{R} \equiv \mathbf{R}' : Type$ *and* $\Gamma \vdash \mathbf{Q} \equiv \mathbf{Q}'(y/y') : Type$ *are sequents.*
4. *Condition (4) of Lemma 4.10 holds for $\mathbf{S}$ and one has*
   (a) $\mathbf{S}' = eq(\mathbf{R}'; \mathbf{r}'', \mathbf{r}''')$
   (b) $\Gamma \vdash \mathbf{R} \equiv \mathbf{R}'$, $\Gamma \vdash \mathbf{r} \equiv \mathbf{r}'' : \mathbf{R}$ *and* $\Gamma \vdash \mathbf{r}' \equiv \mathbf{r}'' : \mathbf{R}$ *are sequents.*

Proof: The fact that the conditions are mutually exclusive follows from the fact that the conditions in Lemma 4.10 are mutually exclusive. Let us show that for each $\Gamma \vdash \mathbf{S} \equiv \mathbf{S}' : Type$ one of the four conditions holds.

Sequents of the form $\Gamma \vdash \mathbf{S} \equiv \mathbf{S}' : Type$ are generated by the rules (**??**), (**??**) and (**??**). Assume that $\mathbf{S}$ satisfies condition (1) of Lemma 4.10.