A Schema for Higher Inductive Types of Level One and Its Interpretation

Hugo Moeneclaey
hmoenecl@ens-cachan.fr

Under the supervision of :

Peter Dybjer
Department of Computing Science and Engineering,
Chalmers University of Technology.
peterd@chalmers.se

August 31, 2016

Abstract

Here we present a general schema for (non-dependent) higher inductive types built using only point and path constructors (HITs of level 1). Moreover we only allow ordinary inductive premisses for the constructors. We prove that the setoid model supports this schema. This model uses set-theory as a meta-language.

Summary

General context

This report is about dependent type theory, which is a typed λ-calculus allowing types to depend on terms. In this type system, virtually any specification of programs is expressible as a type, e.g. there is a type of programs sorting lists of integers. This type theory was intended as a framework to do constructive mathematics, and is the basis of the well-known proof assistant COQ.

Two crucial features of this type system are inductive types and identity types. Inductive types are defined by a list of constructors, and an induction principle can be extracted from this list. As an example $\mathbb{N}$ is defined by two constructors $0 : \mathbb{N}$ and $s : \mathbb{N} \to \mathbb{N}$, and have the usual induction principle. Identity types represent the proposition stating that two elements of a type are equal, and are called propositional equalities, as opposed to judgmental equalities (which are sequents stating that two terms are equal). Judgmental

equality should imply the propositional one. One of the main questions of dependent type theory is whether these two equalities should collapsed.

Homotopy type theory is an extension of dependent type theory in which types are seen as spaces (up to homotopy equivalence). Indeed in this system propositional and judgmental equalities are distinct, and any type comes with a family of (propositional) equalities, equalities of proofs of equality, and so on. The proofs of equality can be seen as paths, and the proofs of equality of proofs of equality can be seen as homotopies between paths, and so on.

The problem studied

One of the new features of homotopy type theory is higher inductive types (HITs), that is inductive types defined with constructors for points, paths, and so on. Several examples of HITs are well-known, but for now there is no definitive general schema for them.

Such a general schema for HITs is of course necessary for homotopy type theory, which is intended as a computer-friendly foundation of mathematics, in what is called the univalent foundation program. HITs are often needed in very general forms in order to build some usual mathematical objects.

The main problem is to determine what is a correct constructor for a HIT and how to extract the induction principle from a list of (correct) constructors.

Personal contribution

Here we take a small step toward this goal by providing a general schema for HITs of level 1 (i.e. defined using constructors for points and paths). The main idea is to express them as usual inductive types in some well-chosen model. This was the guideline of our research work.

We will artificially restrict the number of HITs cover by the schema, because we would otherwise need a generalization of usual inductive definitions in the model called inductive-inductive definitions. We nevertheless manage to give a (restricted) schema and interpret it in a model.

The value of the contribution

Our solution is valuable because it makes use of elementary tools. It has a syntactical flavor which is original. Moreover it seems to be adaptable to more general situations.

Conclusion and further work

One possible next step would be to provide a fully rigorous proof, using a proof assistant. Then some generalizations could be explored, e.g. dependent HITs of level 1, simultaneous HITs of level 1, HITs of level 2 and extensions using inductive-inductive definitions.

Of course, the most important next step would be to extend the schema to arbitrary dimension and interpret it in a well-chosen model.

Contents

1 Introduction

Dependent type theory [16, 18] is a typed λ-calculus which is useful for doing constructive mathematics [1, 17]. It has also been used for classical mathematics, an early example being the AUTOMATH project by De Bruijn [4]. The well-known proof assistant COQ is also based on dependent type theory.

This system makes crucial use of the Curry-Howard correspondence. The types are seen as propositions, and the terms of a given type are seen as proofs of the corresponding proposition, and the other way around. There is a crucial distinction between propositions and judgements in the theory. Propositions are types, and proving a proposition A is the same as constructing an object of type A. On the other hand, judgments have one of the following forms

- $\Gamma \vdash$, meaning that Γ is a well formed context.
- $\Delta = \Gamma \vdash$
- $\Gamma \vdash A$, meaning that A is a well formed type in the context Γ
- $\Gamma \vdash A = A'$
- $\Gamma \vdash a : A$, meaning that a is a term of type A in the context Γ
- $\Gamma \vdash a = a' : A$

and the inference rules of the system lays down how to derive new valid judgments from old ones.

One of the most important features of dependent type theory is inductive definitions. As an example of such a definition, the type of natural numbers is the "free" type $\mathbb{N}$ such that $0 : \mathbb{N}$ and that if $n : \mathbb{N}$ then $s(n) : \mathbb{N}$. We say that $0 : \mathbb{N}$ and $s : \mathbb{N} \to \mathbb{N}$ are the constructors of $\mathbb{N}$. There exists a general schema [6] which describes the form of the type of a constructor in general, and which extracts an induction principle from such a list of constructors. This principle gives a precise definition of a "free" structure having such constructors. In the case of natural numbers, it is the usual induction principle.

But we cannot use induction to derive judgmental equalities. As an example, we are not able to derive the judgmental equality $x : \mathbb{N}, y : \mathbb{N} \vdash x+y = y+x : \mathbb{N}$. So we need a propositional equality, that is if $\Gamma \vdash a : A$ and $\Gamma \vdash a' : A$, then we need a new type $\Gamma \vdash a =_A a'$ representing the proposition stating that a and a' are equal.

In extensional type theory [17] it is assumed that the judgmental and propositional equalities are inter-derivable. Moreover it is assumed that any two inhabitants of an identity type are equal. This is called the unicity of identity proofs. But these assumptions introduce the undecidability of judgments [2].

In intensional type theory, it is only assumed that the judgmental equality implies the propositional equality, and all judgements are decidable. But a question arises naturally : the question whether the unicity of identity proofs is also valid in Intensional Type Theory. The groupoid model of Hofmann and Streicher [13] shows that it is not.

Moreover the simplicial model by Voevodsky [14] shows that there should not be unicity of proofs of equality of proofs of equality, and so on. And the family of equalities, equalities of equalities, ..., of a type forms a weak-∞-groupoid. This structure plays an important role in homotopy theory, and so the resulting type system is called homotopy type theory. Roughly, we have the following correspondence

Type theory	Homotopy theory
Types	Spaces (up to homotopy equivalence)
Terms	Points
Proofs of equality	Paths between points
Proofs of equality of proofs of equality	Homotopies between paths
...	...

The Hott Book [20] describes this theory.
There are two prominent new features in homotopy type theory.

- The univalence axiom, roughly stating that homotopy equivalent types are equal and vice-versa, that is $(A =_{\mathcal{U}} B) \cong (A \cong B)$ where $\mathcal{U}$ is the universe i.e. a type of small types.
- Higher inductive types (HITs), which are types defined by induction with constructors not only for points, but for paths, homotopies between paths,..., as well.

The univalence axiom is one of the main ingredients of Voevodsky's univalent foundations program, which aims to provide a new foundation of mathematics based on Homotopy Type Theory. It gives a precise meaning to an informal but often used "property" : isomorphic structures are equal. However we will not be concerned with it here.

Instead we will only be concerned with HITs. The aim is to provide a schema to define HITs by a list of constructors and extract the inductive principle from this list. Moreover we would like to interpret them as ordinary inductive types and families in some model. Inductive families are inductive types which can depend on terms of previously defined types.

There is a paper in preparation by PL Lumsdaine and Michael Shulman [15] on the categorical semantics of HITs and the interpretation of some HITs in the cubical set model is know [3]. But there lack a systematic treatment of them.

We take a small step toward this goal by defining HITs of level 1 (built using point and path constructors). We will restrict to ordinary inductive premisses, as explained later. So what we would like is a treatment of higher inductive types of level 1 analogous to the treatment of inductive families in [6] and of inductive-recursive definitions in [9, 8]. This means that we would like to

(i) Specify the general form of the type of point and path constructors.

(ii) Derive the elimination rule for an arbitrary HIT specified by a list of point and path constructors.

(iii) Build a set-theoretic model which shows the consistency of intensional type theory with HITs of level 1 as specified by (i) and (ii)

(iv) Show that this set-theoretic model can be formalized inside extensional type theory with the schema for inductive families following [6].

In this paper we carry out steps (i) - (iii) and pave the way for carrying out (iv). This last step means that we need to show the stronger property that the model used at step (iii) (called the setoid model) can be constructed in extensional type theory with inductive families. We believe this to be the case.

We also sketch in appendix a similar treatment of HITs of level 2. We use the groupoid model as defined by Fabian Ruch [19] at step (iii). Note Hoffmann and Streicher conjecture that even the more complex groupoid model can be

defined in extensional type theory, so step (iv) might be possible to carry out at level 2 as well.

Several examples of HITs are presented in the Hott book [20], and all of them which do not use constructors of dimension greater than one fit in our schema. For example the propositional truncation of a type and the circle fit in our schema. On the other hand the 0-truncation of a type and the torus do not.

2 An example of a HIT of level 1 : Combinatory logic

2.1 Notations

Because we are trying to express HITs as usual inductive types in some model, we need two different formal systems. One is the intensional type theory with HITs of level 1 defined in this paper. The other is the (set-theoretic version of) extensional type theory with usual inductive families used for the model. We present the notations for both of them here.

We describe here the syntax for intensional type theory with HITs of level 1. Types A and terms a are given by

$$A ::= (x : A) \twoheadrightarrow A(x) \mid a =_A a \mid H$$

$$a ::= x \mid a(a) \mid \lambda(x : A).a \mid \mathbf{refl}_a \mid J \mid r \mid s \mid \mathrm{Rec}$$

where x is a variable, H is a HIT of level 1, r and s its constructors and Rec its elimination principle.

To state that A is a type in the context Γ we write $\Gamma \vdash A$. To state that a term a is of type A in the context Γ, we write $\Gamma \vdash a : A$, or simply $a : A$ if Γ is empty.

We present briefly those types, without displaying the full typing system.

The Π-types $(x : A) \twoheadrightarrow A(x)$ are the types of dependent functions. Note that we use the set-theoretic notation $f(x)$ for application instead of the standard juxtaposition of λ-calculus. The typing rules are as expected.

For the identity types $x =_A y$ we have that if $\Gamma \vdash a : A$ then $\Gamma \vdash \mathbf{refl}_a : a =_A a$. Moreover J is the elimination principal, that is if $\Gamma, x, y : A, r : x =_A y \vdash C$ and $\Gamma, x : A \vdash d : C[y := x, r := \mathbf{refl}_x]$ then $\Gamma, x, y : A, r : x =_A y \vdash J(C, d) : C$.

Now, some of the syntax for the (set-theoretic) extensional counterpart in the model. Here A is a set and a is a member of a set.

$$A ::= (x \in A) \to A(x) \mid \mathrm{Id}_A(a, a) \mid P$$

$$a ::= x \mid a(a) \mid \lambda(x : A).a \mid \mathbf{refl}_a \mid J \mid r \mid \mathrm{Rec}$$

where P is an inductively generated type or family, r its constructor and Rec its elimination principle.

Note that the terms follow the same syntax, but there is no ambiguity arising anyway. We use the usual notation $a \in A$.

The correspondence between the two notations is clear. We actually did not use directly this extensional type theory, but rather its counterpart in set-theory. So types are sets, and as an example $f \in (x \in A) \to B(x)$ will simply indicate that f is a family $(f_x)_{x \in A}$ such that for all $x \in A$, we have $f_x \in B(x)$. Since our solution is expressing HITs as usual inductive types in the model, we will use inductively defined sets to represent them.

However, even if we are working with set theory as meta-language in this report for simplicity (and lack of time), we believe it could be done in extensional type theory with inductive types.

In both case we will use some standard abbreviations and conventions, as examples, $A \twoheadrightarrow B$ stands for $(x : A) \twoheadrightarrow B(x)$ if B does not depend on x, $A \twoheadrightarrow A \twoheadrightarrow A$ stands for $A \twoheadrightarrow (A \twoheadrightarrow A)$, $f(x, y)$ stands for $f(x)(y)$, $(x, y : A) \twoheadrightarrow B$ stands for $(x : A) \twoheadrightarrow ((y : A) \twoheadrightarrow B)$. We will omit the type for λ-abstraction and write $\lambda x.a$ if the type of x can be inferred from the context.

We use the same notations for the set-theoretic counterpart.

2.2 Some informal homotopy type theory

Here we give some intuition about the intensional type theory with HITs.

Each type comes with a family of equalities, equalities of equalities, and so on. Again, assuming $\Gamma \vdash A$ and $\Gamma \vdash p, q : A$, then $\Gamma \vdash p =_A q$ is a new type. We can moreover iterate this construction, for example if $\Gamma \vdash r, s : p =_A q$, then $\Gamma \vdash r =_{p =_A q} s$ is a type.

Any (non-dependent) function, $f : A \twoheadrightarrow B$ should preserve equality, that is we have a term

$$\mathbf{ap}_f : (x, y : A) \twoheadrightarrow x =_A y \twoheadrightarrow f(x) =_B f(y)$$

We will use standard abuse of notation for $\mathbf{ap}_f$ by omitting x and y, which can be inferred form the context. So if $r : x =_A y$ we abbreviate $\mathbf{ap}_f(x, y, r)$ as $\mathbf{ap}_f(r)$. This term is built using the eliminator for equality.

Let $f : (x : A) \twoheadrightarrow B(x)$ be a dependent function, where $B(x)$ is a type depending on $x : A$. We need to be able to state that f preserves equality, but since $f(x) : B(x)$ and $f(y) : B(y)$, the equality $f(x) =_B f(y)$ does not type-check.

Hopefully there is a way around, indeed using the elimination principle of identity we can define a type $u =_r^B v$ from $x, y : A$, $r : x =_A y$, $u : B(x)$ and $v : B(y)$, such that if $x = y$, then $u =_{\mathbf{refl}_x}^B v$ is $u =_B v$. These types are called heterogeneous equalities, or equalities over a path.

These heterogeneous equalities allow us to formulate a "dependent functoriality" of f : there exists a term

$$\mathbf{apd}_f : (x, y : A) \to (r : x =_A y) \to f(x) =^B_r f(y)$$

We will use the same notation as for $\mathbf{ap}_f$, that is we will abbreviate $\mathbf{apd}_f(x, y, r)$ as $\mathbf{apd}_f(r)$.

2.3 Combinatory logic as a HIT of level 1

We would like to represent combinatory logic quotiented by conversion as a HIT of level 1, in order to give an intuition about HITs. This will be our running example. We will denote this type CL. To define it we use three point constructors

$$\mathrm{K} : \mathrm{CL}$$

$$\mathrm{S} : \mathrm{CL}$$

$$\mathrm{app} : \mathrm{CL} \to \mathrm{CL} \to \mathrm{CL}$$

so we have the set of terms inductively generated by these constructors. Then we want to build the equivalence relation corresponding to conversion. For this we add the two path constructors

$$\mathrm{K}_{conv} : (x, y : \mathrm{CL}) \to \mathrm{app}(\mathrm{app}(\mathrm{K}, x), y) =_{\mathrm{CL}} x$$

$$\mathrm{S}_{conv} : (x, y, z : \mathrm{CL}) \to \mathrm{app}(\mathrm{app}(\mathrm{app}(\mathrm{S}, x), y), z) =_{\mathrm{CL}} \mathrm{app}(\mathrm{app}(x, z), \mathrm{app}(y, z))$$

Please note that the rules of reflexivity, symmetry and transitivity of $=_{\mathrm{CL}}$, as well as the fact that it is preserved through an application, follow from the elimination for equality, so there is no need to add them explicitly.

We now have a type CL together with some terms having the type of the specified constructors. What is the induction principle ?

Intuitively, the induction principle of a type states that any element of the type (or of its equalities, and so on) is build from a constructor, and that there is as little equality constraint as possible.

More rigorously, if $x : \mathrm{CL} \vdash B(x)$, then in order to inductively generates $f : (x : \mathrm{CL}) \to B(x)$ it is enough to define f on the constructors of CL. Assume that for each constructor r of CL we have a term $\tilde{r}$ allowing to interpret f on r, i.e. assume we have :

- The terms $\tilde{\mathrm{K}} : B(\mathrm{K})$ and $\tilde{\mathrm{S}} : B(\mathrm{S})$
- A term $\tilde{\mathrm{app}}$ of type $(x : \mathrm{CL}) \to B(x) \to (y : \mathrm{CL}) \to B(y) \to B(\mathrm{app}(x, y)))$
- A term $\tilde{\mathrm{K}}_{conv}$ of type

  $(x, y : \mathrm{CL}) \to (\tilde{x} : B(x)) \to (\tilde{y} : B(y)) \to \tilde{\mathrm{app}}(\mathrm{app}(\mathrm{K}, x), \tilde{\mathrm{app}}(\mathrm{K}, \tilde{\mathrm{K}}, x, \tilde{x}), y, \tilde{y}) =^B_{\mathrm{K}_{conv}(x,y)} \tilde{x}$

- A term $\tilde{\mathrm{S}}_{conv}$ of type

$$(x, y, z : \mathrm{CL}) \twoheadrightarrow (\tilde{x} : B(x)) \twoheadrightarrow (\tilde{y} : B(y)) \twoheadrightarrow (\tilde{z} : B(z))$$

$$\twoheadrightarrow \tilde{\mathrm{ap}}(\mathrm{app}(\mathrm{app}(\mathrm{S}, x), y), \tilde{\mathrm{ap}}(\mathrm{app}(\mathrm{S}, x), \tilde{\mathrm{ap}}(\mathrm{S}, \tilde{\mathrm{S}}, x, \tilde{x}), y, \tilde{y}), z, \tilde{z})$$

$$=^{B}_{\mathrm{S}_{conv}(x,y,z)} \tilde{\mathrm{ap}}(\mathrm{app}(x, z), \tilde{\mathrm{ap}}(x, \tilde{x}, z, \tilde{z}), \mathrm{app}(y, z), \tilde{\mathrm{ap}}(y, \tilde{y}, z, \tilde{z})))$$

Then there exists a function $f : (x : CL) \twoheadrightarrow B(x)$ such that

- $f(\mathrm{K}) = \tilde{\mathrm{K}},\ f(\mathrm{S}) = \tilde{\mathrm{S}}$
- $f(\mathrm{app}(x, y)) = \tilde{\mathrm{ap}}(x, f(x), y, f(y))$
- $\mathbf{apd}_f(\mathrm{K}_{conv}(x, y)) = \tilde{\mathrm{K}}_{conv}(x, y, f(x), f(y))$
- $\mathbf{apd}_f(\mathrm{S}_{conv}(x, y, z)) = \tilde{\mathrm{S}}_{conv}(x, y, z, f(x), f(y), f(z))$

The reader is invited to check that the judgmental equations on f are well-typed, and that the judgmental equations on $\mathbf{apd}_f$ are well-typed admitting the equations on f are true.

This example suggests that HITs of level 1 are inductively generated sets with inductively generated equivalence relations. So we want a model suitable to represent equivalence relations, that is a model in which types in the empty context are sets with an equivalence relation.

Since a set with an equivalence relation is called a setoid, this model will be called the setoid model.

2.4 Combinatory logic as a setoid

How is CL interpreted in the setoid model ?

Since it is a type in the empty context, we only need a setoid, i.e. a set $\mathrm{Ob}_{\mathrm{CL}}$ together with an equivalence relation $\sim_{\mathrm{CL}}$.

$\mathrm{Ob}_{\mathrm{CL}}$ is simply the set of terms inductively generated by the corresponding signature, that is terms t generated by the syntax

$$t ::= \mathrm{K} \mid \mathrm{S} \mid \mathrm{app}(t, t)$$

Note we a have set-theoretic function app of the right type.

We should have $\sim_{\mathrm{CL}}$ the smallest equivalence relation such that :

- If $x, y \in \mathrm{Ob}_{\mathrm{CL}}$, then $\mathrm{app}(\mathrm{app}(\mathrm{K}, x), y) \sim_{\mathrm{CL}} x$
- If $x, y, z \in \mathrm{Ob}_{\mathrm{CL}}$, then $\mathrm{app}(\mathrm{app}(\mathrm{app}(\mathrm{S}, x), y), z) \sim_{\mathrm{CL}} \mathrm{app}(\mathrm{app}(x, z), \mathrm{app}(y, z))$
- If $x, x', y, y' \in \mathrm{Ob}_{\mathrm{CL}}$, $x \sim_{\mathrm{CL}} x'$ and $y \sim_{\mathrm{CL}} y'$ then $\mathrm{app}(x, y) \sim_{\mathrm{CL}} \mathrm{app}(x', y')$

In order to interpret these intuitions, we need dependent functions, equalities and heterogeneous equalities in the setoid model.

3 The setoid model

Here we present the setoid model informally. A rigorous presentation of it as a category with families (CWF) can be found in appendix B. CWFs provide a standardized framework to present models of dependent type theory. It was presented firstly in [7], and it is explored more comprehensively in [12].

3.1 Contexts, types and terms

3.1.1 Contexts and non-dependent types

A context is interpreted as a pair $(\Gamma, \sim_\Gamma)$ such that Γ is a set and $\sim_\Gamma$ an equivalence relation on Γ. So this is just a setoid.

As explained before a type A in the empty context is also a setoid. We simply interpret the context $x : A$ by the setoid corresponding to A.

From now on we denote the underlying set of A by A as well, and its equivalence relation by $\sim_A$. We will write $a \in A$ when we refer to the underlying set, and $a : A$ when we refer to the type.

3.1.2 Types

A type $B(x)$ over $x : A$, is interpreted as

1) A family of sets $(B_a)_{a \in A}$

2) A family of relations $\sim^B_{a,a'}$ between B_a and $B_{a'}$ for $a, a' \in A$ such that $a \sim_A a'$, satisfying the following properties :
If $b \in B_a$ then $b \sim^B_{a,a} b$.
If $b \in B_a$, $b' \in B_{a'}$ and $b \sim^B_{a,a'} b'$ then $b' \sim^B_{a',a} b$.
If $b \in B_a$, $b' \in B_{a'}$, $b'' \in B_{a''}$, $b \sim^B_{a,a'} b'$ and $b' \sim^B_{a',a''} b''$ then $b \sim^B_{a,a''} b''$.

3) A transport operation which is described fully in appendix B.

Note that this is well defined because $\sim_A$ is an equivalence relation.

This can be extended in a straightforward way to define a type B over any context Γ. We use $\Gamma \vdash B$ as a notation to say that B is a type over Γ.

3.1.3 Terms

A term of a non-dependent type A is simply interpreted as a point in A

A term b of a dependent type B such that $x : A \vdash B(x)$ is a family of elements $(b_a \in B_a)_{a \in A}$ such that $b_a \sim^B_{a,a'} b_{a'}$ whenever $a \sim_A a'$.

This can be extended in a straightforward way to define a term of a type B over any context Γ. We use $\Gamma \vdash b : B$ as a notation to say that b is a term of type B over Γ.

3.2 Some type constructors

Here we interpret some common type.

3.2.1 Non-dependent Π-types

If A and B are non-dependent types, the type $A \twoheadrightarrow B$ is interpreted as the set of functions from A to B preserving the equivalence relations, together with the relation $f \sim_{A \twoheadrightarrow B} f'$ if and only if for all $a, a' \in A$ such that $a \sim_A a'$, we have $f(a) \sim_B f'(a')$.

3.2.2 Dependent Π-types

If $x : A \vdash B(x)$, then the type $(x : A) \twoheadrightarrow B(x)$ is interpreted as the underlying set

$$\{ f : (x \in A) \to B_x \mid \text{if } a \sim_A a' \text{ then } f(a) \sim^B_{a,a'} f(a') \}$$

The equivalence relation is defined as follow : $f \sim_{(x:A) \twoheadrightarrow B(x)} f'$ if and only if

$$\forall a, a' \in A, \text{ if } a \sim_A a' \text{ then } f(a) \sim^B_{a,a'} f'(a')$$

3.2.3 Intensional identity types

If A is a non-dependent type, and $p, q : A$, then we interpret a type $p =_A q$ by the underlying set :

$$\begin{cases} \{*\} & \text{if } p \sim_A q \\ \emptyset & \text{otherwise} \end{cases}$$

There is only one possible equivalence relation.

We see that if $f : A \twoheadrightarrow B$ is non-dependent function, then there is a term $\mathbf{ap}_f$ of type $(x, y : A) \twoheadrightarrow x =_A y \twoheadrightarrow f(x) =_B f(y)$, by definition of f.

3.2.4 Heterogeneous identity types

Assume $x : A \vdash B(x)$, and that we have in the empty context $p, q : A$, $r : p =_A q$, $u : B(p)$ and $v : B(q)$. Then we have the type $u =^B_r v$ interpreted as

$$(u =^B_r v) = \begin{cases} \{*\} & \text{if } u \sim^B_{p,q} v \\ \emptyset & \text{otherwise} \end{cases}$$

This makes sense because since $r : p =_A q$, we have $p \sim_A q$

It can be seen that there exists a term corresponding to $\mathbf{apd}_f$

4 The general schema for HITs of level 1

We now build the schema for HITs of level 1.

4.1 Usual inductive families

We recall the schema for inductive families in [6]. We use our set-theoretic notation because we will use them in the model, which is set-theoretic.

So we want to define an inductive family $P : (x :: \sigma) \to \text{Set}$ where $x :: \sigma$ is an abbreviation for a sequence of argument $x_i \in \sigma_i$ where σ_i is a type defined before P, so that

$$P : (x_1 \in \sigma_1) \to ... \to (x_n \in \sigma_n) \to \text{Set}$$

Such an inductive type is built from a list of constructor r_i of the form

$$r_i : (b :: \beta) \to (u :: \gamma) \to P(q_i)$$

where

- $b :: \beta$ is a sequence of types defined before P. These are non-inductive premisses.
- γ_k is of the form $(x :: \xi_k) \to P(p_k)$ where :

  - ξ_k is a sequence of types defined before P, under the assumption $b :: \beta$

  - $p_k :: \sigma$ under the assumption $b :: \beta$ and $x :: \xi_k$.

  They are called ordinary inductive premisses if ξ_k is empty, and generalized inductive premisses otherwise.
- $q_i :: \sigma$ under the assumption $b :: \beta$.

What is the induction principle for such a type ?

Assume given $C : (x :: \sigma) \to P(x) \to Set$. We want to define a function $f : (x :: \sigma) \to (y \in P(x)) \to C(x, y)$. Then the induction principle should state that in order to do so it is enough to have the value of f on each constructor. So we want that given for each constructor r_i a term

$$\tilde{r_i} : (b :: \beta) \to (u :: \gamma) \to (v :: \delta) \to C(p, r(b, u))$$

where δ (called the induction hypothesis) has the same length as γ and δ_k is $(x :: \xi_k) \to C(p_k, u_k(x))$, there exists a function $f : (x :: \sigma) \to (y \in P(x)) \to C(x, y)$ such that

$$f(q_i, r_i(b, u)) = \tilde{r_i}(b, u, v)$$

where $v_k = \lambda x.f(p_k, u_k(x))$.

We now state precisely what it means. For the sake of simplicity, we assume there is only one constructor and we replace lists of arguments by a unique argument.

Definition 1. *Assume given σ a type without P and a point constructor*

$$r : (b \in \beta) \to (u \in (x : \xi) \to P(p)) \to P(q)$$

with the same constraint as before.

Then we ask that $P : (x \in \sigma) \twoheadrightarrow \text{Set}$ representing this inductive family is such that

- *There is a term of the type of r.*
- *There is a term*

$$\mathrm{Rec} : (C : (x :: \sigma) \to P(x) \to Set)$$
$$\to (\tilde{r} : (b \in \beta) \to (u \in (x \in \xi) \to P(p)) \to (\tilde{u} : (x \in \xi) \to C(p, u(x))) \to C(q, r(b, u)))$$
$$\to (x :: \sigma) \to (y \in P(x)) \to C(x, y)$$

such that for $b \in \beta$, $u \in (x \in \xi) \to P(p)$ and C, $\tilde{r}$ of the right type

$$\mathrm{Rec}(C, \tilde{r}, q, r(b, u)) = \tilde{r}(b, u, \lambda(x : \xi(b)).\mathrm{Rec}(C, \tilde{r}, p, u(x)))$$

Ideally, we want to interpret HITs of level 1 as inductive families in the setoid model. Since our meta-language is set-theory we need to interpret inductive families as families of sets. Such an interpretation is provided by [5].

From now on we will use these inductively defined families of sets. Their constructors are set-theoretic functions, possibly dependent. Note that there is unicity (in the set-theoretic sense) of a function defined by set-theoretic induction.

4.2 Back to combinatory logic

We can rephrase the definitions of $\mathrm{Ob}_{\mathrm{CL}}$ and $\sim_{\mathrm{CL}}$ as instances of this schema.

We generate the set $\mathrm{Ob}_{\mathrm{CL}}$ inductively using three (set-theoretic) constructors

$$\mathrm{K} : \mathrm{Ob}_{\mathrm{CL}}$$

$$\mathrm{S} : \mathrm{Ob}_{\mathrm{CL}}$$

$$\mathrm{app} : \mathrm{Ob}_{\mathrm{CL}} \to \mathrm{Ob}_{\mathrm{CL}} \to \mathrm{Ob}_{\mathrm{CL}}$$

We now generate an inductive family of sets $\mathrm{Hom}_{\mathrm{CL}} : \mathrm{Ob}_{\mathrm{CL}} \to \mathrm{Ob}_{\mathrm{CL}} \to \mathrm{Set}$. We will then define CL as the setoid $(\mathrm{Ob}_{\mathrm{CL}}, \sim_{\mathrm{CL}})$ with for $x, x' \in \mathrm{Ob}_{\mathrm{CL}}$, $x \sim_{\mathrm{CL}} x'$ if and only if $\mathrm{Hom}_{\mathrm{CL}}(x, x')$ is inhabited.

We use several constructors for $\mathrm{Hom}_{\mathrm{CL}}$:

- the path constructors

$$\mathrm{K}_{conv} : (x, y \in \mathrm{Ob}_{\mathrm{CL}}) \to \mathrm{Hom}_{\mathrm{CL}}(\mathrm{app}(\mathrm{app}(\mathrm{K}, x), y), x)$$
$$\mathrm{S}_{conv} : (x, y, z \in \mathrm{Ob}_{\mathrm{CL}}) \to \mathrm{Hom}_{\mathrm{CL}}(\mathrm{app}(\mathrm{app}(\mathrm{app}(\mathrm{S}, x), y), z), \mathrm{app}(\mathrm{app}(x, z), \mathrm{app}(y, z)))$$

- the constructors stating that $\sim_{\mathrm{CL}}$ is an equivalence relation

$$\mathrm{Id} : (a \in \mathrm{Ob}_{\mathrm{CL}}) \to \mathrm{Hom}_{\mathrm{CL}}(a, a)$$
$$\_^{-1} : (a, b \in \mathrm{Ob}_{\mathrm{CL}}) \to \mathrm{Hom}_{\mathrm{CL}}(a, b) \to \mathrm{Hom}_{\mathrm{CL}}(b, a)$$
$$\_ \cdot \_ : (a, b, c \in \mathrm{Ob}_{\mathrm{CL}}) \to \mathrm{Hom}_{\mathrm{CL}}(a, b) \to \mathrm{Hom}_{\mathrm{CL}}(b, c) \to \mathrm{Hom}_{\mathrm{CL}}(a, c)$$

- the constructor stating that the set-theoretic function app is a term

$$\mathbf{ap}_{\mathrm{app}} : (x, x', y, y' \in \mathrm{CL}) \to \mathrm{Hom}_{\mathrm{CL}}(x, x') \to \mathrm{Hom}_{\mathrm{CL}}(y, y') \to \mathrm{Hom}_{\mathrm{CL}}(\mathrm{app}(x, y), \mathrm{app}(x', y'))$$

Note that all these constructors are instances of the schema in section 4.1. Moreover it is an ordinary inductive definition since ξ is always empty.

4.3 Building the schema

4.3.1 A first try

Now we want to define the form of constructors for H a non-dependent HIT of level 1.

A straightforward generalization of the usual notion of inductive type would be to use point constructors of the same form as the usual type constructors, and path constructors of the form

$$s_i : (b :: \beta) \twoheadrightarrow (u :: \gamma) \twoheadrightarrow (v :: \delta) \twoheadrightarrow q_1 =_H q_2$$

where

- $b :: \beta$ is a sequence of types defined before H,
- γ_i is of the form $(x :: \xi_i) \twoheadrightarrow H$ where ξ_i is a sequence of types defined before H under the assumption $b :: \beta$.
- δ_i is of the form $(x :: \psi_i) \to p_1 =_H p_2$ where
  - ψ_i is a sequence of types defined before H under the assumption $b :: \beta$
  - p_i is of type H under the assumptions $b :: \beta$, $u :: \gamma$ and $x :: \psi_i$.
- q_i are terms of type H under the assumption $b :: \beta$ and $u :: \gamma$

But there is a problem with generalized inductive premisses.

As an example, we try to define in the setoid model H a HIT having a constructor $r : (\sigma \twoheadrightarrow H) \twoheadrightarrow H$. We need to generate a set Ob_H, intended as the underlying set of H. We do not want to generate Ob_H using the set-theoretic constructor (where σ_{set} is the underlying set of σ)

$$r : (\sigma_{set} \to \mathrm{Ob}_H) \to \mathrm{Ob}_H$$

because by doing so, we will generates a new point in Ob_H for every function from σ_{set} to Ob_H. But some of these functions might not be terms, because they might not preserve equivalence. And so we will generate too many points in Ob_H.

So we would like to rather use a constructor (where $\sigma_{eq}(x, x')$ is a singleton if $x \sim_\sigma x'$ and empty otherwise)

$$r' : (f : \sigma_{set} \to \mathrm{Ob}_H) \to ((x, x' : \sigma_{set}) \to \sigma_{eq}(x, x') \to \mathrm{Hom}_H(f(x), f(x'))) \to \mathrm{Ob}_H$$

But then we cannot generate Ob_H and Hom_H one after the other, we have to generate them at once. A close inspection of the general case shows that we are dealing with some kind of inductive-inductive definition [11] of $\mathrm{Ob}_H : \mathrm{Set}$ and $\mathrm{Hom}_H : \mathrm{Ob}_H \to \mathrm{Ob}_H \to \mathrm{Set}$.

Inductive-inductive definitions are a (recent) generalization of usual inductive definitions allowing simultaneous inductive definitions of $A : \mathrm{Set}$ and $B : A \to \mathrm{Set}$. We restrict the schema to ordinary inductive premisses, and leave the extension to generalized induction with inductive-inductive definitions as a possible future extension. A key problem is to make sure that the inductive-inductive definitions we need are valid, e.g. by constructing a set-theoretic model of them. One would need to check whether the thesis by Fredrik Nordvall Forsberg [10] covers this case, and otherwise try to extend his work, if possible.

4.3.2 Syntax of the constructors

We now define the form of the constructors. For the sake of simplicity we will assume there is only one constructor on each level. Moreover we will restrict lists of arguments to one argument.

A (non-dependent) higher inductive type H is given by a point constructor r of the form

$$r : \alpha \twoheadrightarrow H \twoheadrightarrow H$$

where there is no occurrences of H in α. This means that α has been defined before H.

We also have a path constructor s of the form

$$s : (a : \beta) \twoheadrightarrow (b : H) \twoheadrightarrow p =_H q \twoheadrightarrow p' =_H q'$$

where there is no occurrences of H in β. Moreover we ask for p, q, p', q' to be built using only variables a and b, λ-abstraction, application and the point constructor. We will call such a term a r-term.

Note that, as an example, r could have any finite number of arguments of type without H, or of type H, and similarly for s. The next definitions and properties could be extended in a straightforward way. Moreover we could have used any (finite) number of point and path constructors with the right form. These restrictions are just here to clarify notation.

Remark 1. *Note we are only able to use constructors of strictly lesser dimension in the definition of a constructor. All examples in the Hott book [20] satisfies this restriction.*

4.3.3 The induction principle : why we need a translation

Assume $x : H \vdash B(x)$. We want to know how to define a term $f : (x : H) \to B(x)$ using the induction principle. Intuitively the value of f on each constructor is enough to define f on all H. We try to make this statement precise.

For point constructor the situation is the same as for usual induction, we need a term

$$\tilde{r} : (x : \alpha) \to (y : H) \to B(y) \to B(r(x, y))$$

and f will be such that $f(r(x, y)) = \tilde{r}(x, y, f(y))$.

But for path constructor the situation is a bit trickier. As an example if we have a path constructor $s : (a : \beta) \to p' =_H q'$. Morally we should provide something like $\tilde{s} : (a : \beta) \to f(p) =^B_{s(a)} f(q')$, to define the action of f on equalities. But f is not defined yet !

Hopefully there is a way around : we are able to define, using $\tilde{r}$, a translation T_t predicting the value of f at any given r-term.

Since we need to extend T_t to any r-term in order to do a proper induction on r-term, T_t is rather complicated. But the core idea is simply that any term should have a duplicate which is more-or-less the corresponding induction hypothesis. The key case is $T_t(r) = \tilde{r}$.

Remark 2. *As an example, this translation provides the terms for the induction principle of* CL *in section 2.3.*

To sum up strictly what we need,

- We have a translation T_t such that if p is a r-term and $x : \beta, y : H \vdash p : H$ then
$$x : \beta, y : H, \tilde{y} : B(y) \vdash T_t(p) : B(p)$$
- If $f : (x : H) \to B(x)$ is such that $f(r(x, y)) = \tilde{r}(x, y, f(y))$ then
$$x : \beta, y : H \vdash f(p) = T_t(p)[\tilde{y} := f(y)] : B(p)$$

So we are indeed predicting the value of f.

We refer the reader to the appendix A for a description of this translation.

Now we are able to define completely the schema.

4.4 The schema

Definition 2. *Assume given any point constructor*

$$r : \alpha \to H \to H$$

where there is no occurrence of H in α, and any path constructor

$$s : (a : \beta) \to (b : H) \to p =_H q \to p' =_H q'$$

where there is no occurrences of H in β and p, q, p', q' are r-terms of type H under the assumptions $a : \beta$ and $b : H$.

A model supports H if there exists a type H in the empty context such that :

- *There exist terms having the type of the constructors in the empty context.*
- *There exist a term* Rec *of type*

$$\mathrm{Rec} : (B : H \to Set) \to (\tilde{r} : (x : \alpha) \to (y : H) \to B(y) \to B(r(x,y)))$$

$$\to (\tilde{s} : (x : \beta) \to (y : H) \to (\tilde{y} : B(y)) \to (z : p =_H q) \to T_t(p) =^B_z T_t(q) \to T_t(p') =^B_{s(x,y,z)} T_t(q'))$$

$$\to (x : H) \to B(x)$$

such that, assuming B, $\tilde{r}$ and $\tilde{s}$ of the right type and denoting f the function $\lambda x.\mathrm{Rec}(B, \tilde{r}, \tilde{s}, x) : (x : H) \to B(x)$, we have

$$x : \beta, y : H \vdash f(r(x,y)) = \tilde{r}(x, y, f(y)) : B(r(x,y))$$

$$x : \beta, y : H, z : p =_H q \vdash \boldsymbol{apd}_f(s(x,y,z)) = \tilde{s}(x, y, f(y), z, \boldsymbol{apd}_f(z)) : f(p') =^B_{s(x,y,z)} f(q')$$

A model support HITs of level 1 if it supports every such H.

Proof. We check that these equations are well typed. The only part which is not easy is

$$x : \beta, y : H, z : p =_H q \vdash \tilde{s}(x, y, f(y), z, \mathbf{apd}_f(z)) : f(p') =^B_{s(x,y,z)} f(q')$$

We need to show that $x : \beta, y : H, z : p =_H q \vdash \mathbf{apd}_f(z) : T_t(p)[\tilde{y} := f(y)] =^B_z T_t(q)[\tilde{y} := f(y)]$

This is true because $x : \beta, y : H \vdash f(p) = T_t(p)[\tilde{y} := f(y)] : B(p)$ by corollary 1 in appendix A, assuming $f(r(x,y)) = \tilde{r}(x, y, f(y))$.

Then we know that $x : \beta, y : H, z : p =_H q \vdash \tilde{s}(x, y, f(y), z, \mathbf{apd}_f(z)) : T_t(p')[\tilde{y} := f(y)] =^B_{s(x,y,z)} T_t(q')[\tilde{y} := f(y)]$, and so we can conclude by using corollary 1. □

Remark 3. *We will see that in the setoid model there is a unique term obeying the two equations above.*

We have a proof in ordinary λ-calculus notation that the setoid model supports HITs of level 1 which is presented in appendix C.

5 Conclusion

So we have presented here a general syntactic schema for non-dependent HITs of level 1, with ordinary inductive premisses.

The first problem we face when trying to extend it to higher inductive types of higher dimensions is of combinatorial nature, basically to determine what constructors are to be added in higher dimensions. This difficulty is inherent in higher category theory.

We present suggestions of possible further works :

- Formalize what happens on level 1 using some proof assistant.
- Present a schema for level 2. Appendix D is a sketch of this..
- Extend to dependent HITs and mutual HITs.
- Extend to generalized inductive premisses using inductive-inductive definitions. Note that inductive-inductive definitions would allow to define constructors not necessarily by order of dimensions.
- And of course, extend the schema to arbitrary dimensions !

Ackowledgement

I would like to thank everyone working on dependent type theory at Chalmers this summer for making this internship so pleasant. More particularly, I thank Simon Huber and Fabian Ruch for taking the time to answer to my questions.

And last but not least, I thank Peter Dybjer for countless interesting discussions and valuable suggestions. My first contact with dependent type theory has been highly enjoyable thanks to him.

References

[1] Peter Aczel. The type theoretic interpretation of constructive set theory. *Studies in Logic and the Foundations of Mathematics*, 96:55–66, 1978.

[2] Simon Castellan, Pierre Clairambault, and Peter Dybjer. Undecidability of equality in the free locally cartesian closed category (extended version).

[3] Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. Cubical type theory: a constructive interpretation of the univalence axiom. *Preprint, December*, 2015.

[4] Nicolaas Govert De Bruijn. A survey of the project automath. 1980.

[5] Peter Dybjer. Inductive sets and families in martin-löf's type theory and their set-theoretic semantics. *Logical frameworks*, 2:6, 1991.

[6] Peter Dybjer. Inductive families. *Formal aspects of computing*, 6(4):440–465, 1994.

[7] Peter Dybjer. Internal type theory. In *International Workshop on Types for Proofs and Programs*, pages 120–134. Springer, 1995.

[8] Peter Dybjer. A general formulation of simultaneous inductive-recursive definitions in type theory. *The Journal of Symbolic Logic*, 65(02):525–549, 2000.

[9] Peter Dybjer and Anton Setzer. Indexed induction–recursion. *The Journal of Logic and Algebraic Programming*, 66(1):1–49, 2006.

[10] Fredrik Nordvall Forsberg. *Inductive-inductive definitions*. PhD thesis, Swansea University, 2013.

[11] Fredrik Nordvall Forsberg and Anton Setzer. Inductive-inductive definitions. In *International Workshop on Computer Science Logic*, pages 454–468. Springer, 2010.

[12] Martin Hofmann. Syntax and semantics of dependent types.

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

[14] Chris Kapulkin, Peter LeFanu Lumsdaine, and Vladimir Voevodsky. The simplicial model of univalent foundations. *arXiv preprint arXiv:1211.2851*, 2012.

[15] PL Lumsdaine and Michael Shulman. Higher inductive types. *preparation (cit. on pp. 3, 6, 26)*, 2013.

[16] Per Martin-Löf. An intuitionistic theory of types: Predicative part. *Studies in Logic and the Foundations of Mathematics*, 80:73–118, 1975.

[17] Per Martin-Löf. Constructive mathematics and computer programming. *Studies in Logic and the Foundations of Mathematics*, 104:153–175, 1982.

[18] Per Martin-Löf. *Intuitionistic Type Theory: Notes by Giovanni Sambin of a series of lectures given in Padova, June 1980*. 1984.

[19] Fabian Ruch. The path model of intensional type theory. 2015.

[20] Vladimir Voevodsky et al. Homotopy type theory: Univalent foundations of mathematics. *Institute for Advanced Study (Princeton), The Univalent Foundations Program*, 2013.

A The translations

Here we present the translations necessary to state the induction principle for a HIT called H.

A.1 The translation T_t

Here we assume $x : H \vdash B(x)$. We call a simple type a type built using H, Π-types and types in which H does not occur. We say that a term is simply-typed if it is typed by a simple type.

Definition 3. *We define a translation T on pairs (p, β) such that $\Gamma \vdash p : \beta$ and β is a simple type. By induction on β :*

- $T(p, \beta) = \beta$ *if H does not occur in β.*
- $T(p, H) = B(p)$.
- $T(p, (x : P) \twoheadrightarrow Q(x)) = (x : P) \twoheadrightarrow T(p(x), Q(x))$ *if H does not occur in P.*
- $T(p, (x : P) \twoheadrightarrow Q(x)) = (x : P) \twoheadrightarrow (\tilde{x} : T(x, P)) \twoheadrightarrow T(p(x), Q(x))$ *otherwise.*

Remark 4. *If we have a point constructor $r : \alpha \twoheadrightarrow H \twoheadrightarrow H$ where H does not occurs in α, then $T(r, \alpha \twoheadrightarrow H \twoheadrightarrow H) = (x : \alpha) \twoheadrightarrow (y : H) \twoheadrightarrow B(y) \twoheadrightarrow B(r(x, y))$. This is precisely the type of the term $\tilde{r}$ required for a definition by induction.*

Lemma 1. *If $\Gamma \vdash p : \beta$ then $\Gamma \vdash T(p, \beta)$.*

Definition 4. *We now define a translation T from context made of simple type to context. By induction on the context :*

- $T(*) = *$.
- $T(\Gamma, x : P) = T(\Gamma), x : P$ *if H does not occur in P.*
- $T(\Gamma, x : P) = T(\Gamma), x : P, \tilde{x} : T(x, P)$ *otherwise.*

Proposition 1. *Assume p is a r-term built simply-typed using variables, and that we have $\tilde{r} : T(r, \alpha \twoheadrightarrow H \twoheadrightarrow H)$. Moreover β is a simple type.*
If $\Gamma \vdash p : \beta$ then there is a term $T_t(p)$ such that $T(\Gamma) \vdash T_t(p) : T(p, \beta)$.

Proof. By induction on the term

If the rule is an axiom on a variable x then we take the variable $\tilde{x}$ or x depending on whether H does occur in the type of x.

If it is the constructor r we required a term $\tilde{r}$ suitable as $T_t(r)$.

If the rule is a λ-introduction $\Gamma \vdash \lambda x.p : (x : P) \twoheadrightarrow Q(x)$ then we have by induction hypothesis :
If H does not occurs in P

$$T(\Gamma), x : P \vdash T_t(p) : T(p, Q(x))$$

So

$$T(\Gamma) \vdash \lambda x.T_t(p) : (x : P) \to T(p, Q(x))$$

but $(\lambda x.p)(x) = p$ so it is the right type.

If H does occurs in P

$$T(\Gamma), x : P, \tilde{x} : T(x, P) \vdash T_t(p) : T(p, Q(x))$$

So

$$T(\Gamma) \vdash \lambda x \lambda \tilde{x}.T_t(p) : (x : P) \to (\tilde{x} : T(x, P)) \to T(p, Q(x))$$

but $(\lambda x.p)(x) = p$ so it it is the right type.

if the rule is an application : $\Gamma \vdash p(q) : Q(q)$ with $\Gamma \vdash p : (x : P) \twoheadrightarrow Q(x)$ and $\Gamma \vdash q : P$.

By induction hypothesis if H does not occurs in P

$$T(\Gamma) \vdash T_t(p) : (x : P) \to T(p(x), Q(x))$$

so

$$T(\Gamma) \vdash T_t(p)(q) : T(p(q), Q(q))$$

Because of lemma 2, see below.

If H does occurs in P

$$T(\Gamma) \vdash T_t(p) : (x : P) \to (\tilde{x} : T(x, P)) \to T(p(x), Q(x))$$

and $T(\Gamma) \vdash T_t(q) : T(q, P)$ so $T(\Gamma) \vdash T_t(p)(q, T_t(q)) : T(p(q), Q(q))$, again by lemma 2, see below.

□

Now we display the auxiliary lemma used before.

Lemma 2. *Assume $\Gamma \vdash p : \beta$ where β is a simple type, Γ is a context made of simple types, $x : P$ is in Γ and $\Gamma \vdash q : P$, then $T(p, \beta)[x := q] = T(p[x := q], \beta[x := q])$.*

Proof. This is a straightforward induction on β. □

A.2 The translation T_f

In this section we assume a point constructor $r : \alpha \twoheadrightarrow H \twoheadrightarrow H$ together with $\tilde{r} : T(r, \alpha \twoheadrightarrow H \twoheadrightarrow H)$.

Moreover we assume a term $f : (x : H) \twoheadrightarrow B(x)$ such that $x : \alpha, y : H \vdash f(r(x, y)) = \tilde{r}(x, y, f(y)) : B(r(x, y))$. We want to show that T_t is related to f.

We recall

$T_t(x) = \tilde{x}$ if x is a variable of a type where H occurs.

$T_t(p) = p$ if p is of a type where H does not occur.

$T_t(r) = \tilde{r}$ if r is the point constructor.

$T_t(p(q)) = T_t(p)(q)$ if H does not occur in the type of q.
$T_t(p(q)) = T_t(p)(q, T_t(q))$ if H does occur in the type of q.
$T_t(\lambda x.p) = \lambda x.T_t(p)$ if H does not occur in the type of x.
$T_t(\lambda x.p) = \lambda x.\lambda \tilde{x}.T_t(p)$ if H does occur in the type of x.

Definition 5. *We define a translation T_f on pairs (p, β) such that $\Gamma \vdash p : \beta$ and β is a simple type. By induction on β :*

- $T_f(p, \beta) = p$ *if H does not occur in β.*
- $T_f(p, H) = f(p)$.
- $T_f(p, (x : P) \twoheadrightarrow Q(x)) = \lambda(x : P).T_f(p(x))$ *if H does not occur in P.*
- $T_f(p, (x : P) \twoheadrightarrow Q(x)) = \lambda(x : P).\lambda(\tilde{x} : T(x, P)).T_f(p(x))$ *otherwise.*

Lemma 3. *If $\Gamma \vdash p : P$ then $\Gamma \vdash T_f(p) : T(p, P)$.*

Lemma 4. *If $p =_{\beta\eta} p'$ then $T_f(p) =_{\beta\eta} T_f(p')$ and $T_t(p) =_{\beta\eta} T_t(p')$*

Proof. The property for T_f is proved by a straightforward induction on the type of p.

For T_t, we proceed by induction on the term. All case (including the η-reduction) are straightforward, except $T_t((\lambda x.p)\ q) = T_t(p[x := q])$. When H does not occur in the type of x we have
$T_t((\lambda x.p)(q)) = T_t(\lambda x.p)(q) = (\lambda x.T_t(p))(q) = T_t(p)[x := q]$
and otherwise
$T_t((\lambda x.p)(q)) = T_t(\lambda x.p)(q, T_t(q)) = (\lambda x.\lambda \tilde{x}.T_t(p))(q, T_t(q)) = T_t(p)[x := q, \tilde{x} := T_t(q)]$
We just have to prove that $T_t(p[x := q])$ is $T_t(p)[x := q]$, or $T_t(p)[x := q, \tilde{x} := T_t(q)]$ depending on the type of x. This is done by induction on the term. □

So we may assume that p is in normal form to prove property of $T_t(p)$.

Let us call π-type a type which is either H or where H does not occur.

Lemma 5. *Assume p is a r-term. If*

$$x_1 : P_1, ..., x_n : P_n \vdash p : P$$

with $P = (y_1 : Q_1) \to ... \to (y_m : Q_m) \to Q$ where the Q_i, P_i and Q are π-types, then

$$x_1 : P_1, ..., x_n : P_n, y_1 : Q_1, ..., y_m : Q_m \vdash T_f(p)(y_1, "T_f(y_1)", ..., y_m, "T_f(y_m)") =$$

$$T_t(p)[\tilde{x}_1 := T_f(x_1), ..., \tilde{x}_n := T_f(x_n)](y_1, "T_f(y_1)", ..., y_m, "T_f(y_m)") : T(p(y_1, ..., y_m), Q)$$

where "$T_f(y_i)$" appears if and only if H occurs in the type of y_i.

Proof. Let us denote $\Gamma = x_1 : P_1, ..., x_n : P_n, y_1 : Q_1, ..., y_m : Q_m$
By induction on the term :

If it is a variable x_i, either it has a type without H, then
$\Gamma \vdash T_t(x_i)[\tilde{x}_i := T_f(x_i)] = x_i = T_f(x_i)$,
otherwise it is of type H, then
$\Gamma \vdash T_t(x_i)[\tilde{x}_i := T_f(x_i)] = \tilde{x_i}[\tilde{x}_i := T_f(x_i)] = T_f(x_i)$

If it is a point constructor it is precisely our hypothesis $f(r(x,y)) = \tilde{r}(x,y,f(y))$

If it is a λ-abstraction $\lambda y_1.p$, then by induction hypothesis we have
$\Gamma \vdash T_t(p)[\tilde{x}_1 := T_f(x_1), ..., \tilde{y}_1 := T_f(y_1)](y_2, "T_f(y_2)", ..., y_m, "T_f(y_m)") =$
$T_f(p)(y_2, "T_f(y_2)", ..., y_m, "T_f(y_m)")$

So if H does not occur in the type of y_1 then
$\Gamma \vdash T_f(\lambda y_1.p)(y_1, y_2, "T_f(y_2)", ..., y_m, "T_f(y_m)") =$
$T_f(p)(y_2, "T_f(y_2)", ..., y_m, "T_f(y_m)") =$
$T_t(p)[\tilde{x}_1 := T_f(x_1), ..., \tilde{y}_1 := T_f(y_1)](y_2, "T_f(y_2)", ..., y_m, "T_f(y_m)") =$
$T_t(\lambda y_1.p)[\tilde{x}_1 := T_f(x_1), ..., \tilde{x}_n := T_f(x_n)](y_1, y_2, "T_f(y_2)", ..., y_m, "T_f(y_m)")$

Otherwise
$\Gamma \vdash T_f(\lambda y_1.p)(y_1, T_f(y_1), y_2, "T_f(y_2)", ..., y_m, "T_f(y_m)") =$
$T_f(p)[\tilde{y}_1 := T_f(y_1)](y_2, "T_f(y_2)", ..., y_m, "T_f(y_m)") =$
(since $\tilde{y}_1$ is not in $T_f(p)$)
$T_f(p)(y_2, "T_f(y_2)", ..., y_m, "T_f(y_m)") =$
$T_t(p)[\tilde{x}_1 := T_f(x_1), ..., \tilde{y}_1 := T_f(y_1)](y_2, "T_f(y_2)", ..., y_m, "T_f(y_m)") =$
$T_t(\lambda y_1.p)[\tilde{x}_1 := T_f(x_1), ..., \tilde{x}_n := T_f(x_n)](y_1, T_f(y_1), y_2, "T_f(y_2)", ..., y_m, "T_f(y_m)")$

If it is an application $p(q)$, then we assume that it is in normal form. It is possible to do so because of the last lemma. We first show that q is of type H or of a type in which H does not occur. It is true because p can only have arguments of the type H or of a type in which H does not occurs. Indeed, by induction on p :

- it is not a λ-abstraction because the term is in normal form.

- if it is a variable or the point constructor it is clear.

- otherwise it is an application $p = p'(q')$ and is true for p' by induction so it is true for p.

So we have $\Gamma \vdash T_t(q)[\tilde{x}_1 := T_f(x_1), ..., \tilde{x}_n := T_f(x_n)] = T_f(q)$.

Then we know by induction hypothesis
$\Gamma \vdash T_f(p)(x, "T_f(x)" y_1, T_f(y_1), ..., y_m, T_f(y_m)) =$
$T_t(p)[\tilde{x}_1 := T_f(x_1), ..., \tilde{x}_n := T_f(x_n)](x, "T_f(x)", y_1, "T_f(y_1)", ..., y_m, "T_f(y_m)")$
Where x is a new variable of the same type as q.

So if H does not occur in the type of q then
$\Gamma \vdash T_t(p(q))[\tilde{x}_1 := T_f(x_1), ..., \tilde{x}_n := T_f(x_n)](y_1, "T_f(y_1)", ..., y_m, "T_f(y_m)") =$
$(T_t(p)(q))[\tilde{x}_1 := T_f(x_1), ..., \tilde{x}_n := T_f(x_n)](y_1, "T_f(y_1)", ..., y_m, "T_f(y_m)") =$
(since $\tilde{x}_i$ does not occur in q)
$T_t(p)[\tilde{x}_1 := T_f(x_1), ..., \tilde{x}_n := T_f(x_n)](q, y_1, "T_f(y_1)", ..., y_m, "T_f(y_m)") =$
(by induction hypothesis with $x := q$)
$T_f(p)(q, y_1, "T_f(y_1)", ..., y_m, "T_f(y_m)") =$
$\lambda x.T_f(p(x))(q, y_1, "T_f(y_1)", ..., y_m, "T_f(y_m)") =$
$T_f(p(q))(y_1, "T_f(y_1)", ..., y_m, "T_f(y_m)")$

And if H does occur in the type of q then
$\Gamma \vdash T_t(p(q))[\tilde{x}_1 := T_f(x_1), ..., \tilde{x}_n := T_f(x_n)](y_1, "T_f(y_1)", ..., y_m, "T_f(y_m)") =$
$T_t(p)(q, T_t(q))[\tilde{x}_1 := T_f(x_1), ..., \tilde{x}_n := T_f(x_n)](y_1, "T_f(y_1)", ..., y_m, "T_f(y_m)") =$
since $T_t(q)[\tilde{x}_1 := T_f(x_1), ..., \tilde{x}_n := T_f(x_n)] = T_f(q)$
$T_t(p)[\tilde{x}_1 := T_f(x_1), ..., \tilde{x}_n := T_f(x_n)](q, T_f(q), y_1, "T_f(y_1)", ..., y_m, "T_f(y_m)") =$
$T_f(p)(q, T_f(q), y_1, "T_f(y_1)", ..., y_m, "T_f(y_m)") =$
$\lambda x.\lambda \tilde{x}.T_f(p\ x)(q, T_f(q), y_1, "T_f(y_1)", ..., y_m, "T_f(y_m)") =$
$T_f(p(q))[\tilde{x} := T_f(q)](y_1, "T_f(y_1)", ..., y_m, "T_f(y_m)") =$
since $\tilde{x}$ does not occur in $T_f(p(q))$
$T_f(p(q))(y_1, "T_f(y_1)", ..., y_m, "T_f(y_m)")$

□

Here we state a corollary which is useful for us.

Corollary 1. *Assume $x : \beta, y : H \vdash p : H$ where p is a r-term and H does not occur in β, then we have $x : \beta, y : H \vdash T_t(p)[\tilde{y} := f(y)] = f(p) : B(p)$.*

B The setoid model as a CWF

Here we present the (set-theoretic) setoid model as a CWF. This section is largely based on the presentation of the groupoid model by Fabian Ruch [19].

B.1 A category with family

B.1.1 Contexts

A context is interpreted as a pair $(\Gamma, \sim_\Gamma)$ such that Γ is a set and $\sim_\Gamma$ an equivalence relation on Γ.

A substitution morphism from $(\Gamma, \sim_\Gamma)$ to $(\Delta, \sim_\Delta)$ is a function f from Γ to Δ such that if $\gamma \sim_\Gamma \gamma'$ then $f(\gamma) \sim_\Delta f(\gamma')$.

The empty context is a singleton with the only possible equivalence relation. It is denoted $*$.

From now on we denote a context by its underlying set Γ.

B.1.2 Types

A type A over Γ, is interpreted as :

1) A family of sets $(A_\gamma)_{\gamma \in \Gamma}$

2) A family of relations $\sim^A_{\gamma,\gamma'}$ between A_γ and $A_{\gamma'}$ for $\gamma, \gamma' \in \Gamma$ such that $\gamma \sim_\Gamma \gamma'$, satisfying the following properties :
If $a \in A_\gamma$ then $a \sim^A_{\gamma,\gamma} a$.
If $a \in A_\gamma$, $a' \in A_{\gamma'}$ and $a \sim^A_{\gamma,\gamma'} a'$ then $a' \sim^A_{\gamma',\gamma} a$.
If $a \in A_\gamma$, $a' \in A_{\gamma'}$, $a'' \in A_{\gamma''}$, $a \sim^A_{\gamma,\gamma'} a'$ and $a' \sim^A_{\gamma',\gamma''} a''$ then $a \sim^A_{\gamma,\gamma'} a''$.

3) A transport operation which for $\gamma \sim_\Gamma \gamma'$ maps $a \in A_\gamma$ to $a^+_{\gamma'} \in A_{\gamma'}$, such that $a \sim^A_{\gamma,\gamma'} a^+_{\gamma'}$. Furthermore we ask that $(a^+_{\gamma'})^+_{\gamma''} = a^+_{\gamma''}$ if $\gamma \sim_\Gamma \gamma' \sim_\Gamma \gamma''$ (associativity), and that $a^+_\gamma = a$ (identity). This implies the symmetry, that is $(a^+_{\gamma'})^+_\gamma = a^+_\gamma = a$.

Note that this is well defined because $\sim_\Gamma$ is an equivalence relation.

$\sim_{\gamma,\gamma}$ is an equivalence relation over A_γ.

We use $\Gamma \vdash A$ as a notation to say that A is a type over Γ.

B.1.3 Terms

If $\Gamma \vdash A$, a term a of type A over Γ is interpreted as a family of elements $(a_\gamma \in A_\gamma)_{\gamma\in\Gamma}$ such that $a_\gamma \sim^A_{\gamma,\gamma'} a_{\gamma'}$ whenever $\gamma \sim_\Gamma \gamma'$.

We use $\Gamma \vdash a : A$ as a notation to say that a is a term of type A over Γ.

B.1.4 Substitution

Let $f : \Gamma \to \Delta$ be a substitution morphism. Assume $\Delta \vdash A$ and $\Delta \vdash t : A$.

Then we define a type Af by the sets $(Af)_\gamma = A_{f(\gamma)}$, together with the relations such that if $a \in A_{f(\gamma)}$ and $a' \in A_{f(\gamma')}$, then $a \sim^{Af}_{\gamma,\gamma'} a'$ if and only if $a \sim^A_{f(\gamma),f(\gamma')} a'$.

We define a term tf of type Af by $(tf)_\gamma = t_{f(\gamma)}$.

So we have $\Gamma \vdash Af$ and $\Gamma \vdash tf : Af$.

B.1.5 Context comprehension

Let Γ be a context such that $\Gamma \vdash A$, then we define the context $\Gamma.A$ as the set $\{(\gamma, a)|\gamma \in \Gamma, a \in A_\gamma\}$ with the equivalence relation $(\gamma, a) \sim_{\Gamma.A} (\gamma', a')$ if and only if $\gamma \sim_\Gamma \gamma'$ and $a \sim^A_{\gamma,\gamma'} a'$.

We define $\mathbf{p} : \Gamma.A \to \Gamma$ as the first projection and $\mathbf{q}$ of type $A\mathbf{p}$ over $\Gamma.A$ by the second.

It can be checked that $\Gamma \vdash \mathbf{q} : A\mathbf{p}$.

B.1.6 A first theorem

Theorem 1. *The setoid model is a category with family.*

This can be proven in a straightforward fashion, or by using the groupoid model by Fabian Ruch [19], which is a CWF.

Indeed if we define the set of morphisms of groupoid between γ and γ' as a singleton if $\gamma \sim_\Gamma \gamma'$ and the empty set otherwise, and similarly for types, then we have contexts, types and terms in the groupoid model, because every groupoid equations are trivially verified in singletons.

So the setoid model is the "subpart" of the groupoid model in which all sets of morphisms are supposed of cardinal at most one. In order to prove theorem 1,

we only need to check that all the involved operations on types in the groupoid model preserve this property.

However we will not prove it here.

B.2 Some type constructors

Here we present the interpretation of types useful for our purpose.

B.2.1 Π-types

Assume $\Gamma \vdash A$ and $\Gamma.A \vdash B$, then $\Gamma \vdash \Pi(A,B)$ with

$$\Pi(A,B)_\gamma = \{ f : (a \in A_\gamma) \to B_{(\gamma,a)} \mid \text{if } a \sim^A_{\gamma,\gamma} a' \text{ then } f(a) \sim^B_{(\gamma,a),(\gamma,a')} f(a') \}$$

Let $\gamma \sim_\Gamma \gamma'$, $f \in \Pi(A,B)_\gamma$ and $f' \in \Pi(A,B)_{\gamma'}$. Then $f \sim^{\Pi(A,B)}_{\gamma,\gamma'} f'$ if and only if

$$\forall a \in A_\gamma,\ \forall a' \in A_{\gamma'},\ \text{if } a \sim^A_{\gamma,\gamma'} a' \text{ then } f(a) \sim^B_{(\gamma,a),(\gamma',a')} f(a')$$

Remark 5. *We will use* $\Pi(P_1, ..., P_n)$ *as an abbreviation for* $\Pi(P_1, \Pi(P_2, ..., \Pi(P_{n-1}, P_n)...))$

B.2.2 Intensional Identity types

Assume $\Gamma \vdash A$ then we define $\Gamma.A.A\mathbf{p} \vdash \mathrm{Id}^A$ by

$$\mathrm{Id}^A_{(\gamma,a,a')} = \begin{cases} \{*\} & \text{if } a \sim^A_{\gamma,\gamma} a' \\ \emptyset & \text{otherwise} \end{cases}$$

The relation simply connects everything.

B.2.3 Heterogeneous identity types

Assume $\Gamma \vdash A$ and $\Gamma.A \vdash B$. We can then define $\Gamma.A.A\mathbf{p}.\mathrm{Id}^A.B\mathbf{pp}.B < \mathbf{pp}, \mathbf{q} > \mathbf{pp} \vdash \mathrm{HId}^B$ by

$$\mathrm{HId}^B_{(\gamma,a,a',r,b,b')} = \begin{cases} \{*\} & \text{if } b \sim^B_{(\gamma,a),(\gamma,a')} b' \\ \emptyset & \text{otherwise} \end{cases}$$

This makes sense because $r \in \mathrm{Id}^A_{(\gamma,a,a')}$, so $a \sim^A_{\gamma,\gamma} a'$ and $(\gamma, a) \sim_{\Gamma.A} (\gamma, a')$.

The relation relates any two elements.

It can be checked that these are indeed heterogeneous identity type.

B.2.4 Existence of $\mathbf{apd}_f$

Assuming $\Gamma \vdash f : \Pi(A, B)$, then we have a term $\mathbf{apd}_f$ such that

$$\Gamma \vdash \mathbf{apd}_f : \Pi(A, A\mathbf{p}, \mathrm{Id}^A, \mathrm{HId}^B < id, app(f\mathbf{p}, \mathbf{q})\mathbf{pp}, app(f\mathbf{pp}, \mathbf{q})\mathbf{p} >)$$

To provide such a term, for each $\gamma \in \Gamma$ we need a set-theoretic function of type

$$(a \in A_\gamma) \to (a' \in (A\mathbf{p})_{(\gamma,a)}) \to (r \in \mathrm{Id}^A_{(\gamma,a,a')}) \to (\mathrm{HId}^B < id, app(f\mathbf{p}, \mathbf{q})\mathbf{pp}, app(f\mathbf{pp}, \mathbf{q})\mathbf{p} >)_{(\gamma,a,a',r)}$$

that is

$$(a \in A_\gamma) \to (a' \in A_\gamma) \to (r \in \mathrm{Id}^A_{(\gamma,a,a')}) \to \mathrm{HId}^B_{(\gamma,a,a',r,f_\gamma(a),f_\gamma(a'))}$$

That is assuming $a, a' \in A_\gamma$ and $a \sim^A_{\gamma,\gamma} a'$, we need $f_\gamma(a) \sim^B_{(\gamma,a),(\gamma,a')} f_\gamma(a')$. This is true by definition of f_γ. We omit the proof that this function is a term.

C The setoid model supports HIT of level 1

In the section we use ordinary λ-calculus notation.

Assume given a point and a path constructors.

$$r : \alpha \twoheadrightarrow H \twoheadrightarrow H$$

$$s : (a : \beta) \twoheadrightarrow (b : H) \twoheadrightarrow p =_H q \twoheadrightarrow p' =_H q'$$

We want to define a type in the empty context, so we need a set Ob_H with an equivalence relation on it. For this we will inductively generates $\mathrm{Ob}_H : \mathrm{Set}$ and $\mathrm{Hom}_H : \mathrm{Ob}_H \to \mathrm{Ob}_H \to \mathrm{Set}$. Then we will define, for $a, a' : \mathrm{Ob}_H$, that $a \sim_H a'$ if and only if $\mathrm{Hom}_H(a, a')$ is inhabited.

The notation $\to$ is reserved for (dependent) set-theoretic function. If α is a type in the empty context we denote by α_{set} its underlying set. If $a, a' \in \alpha_{set}$ we denote

$$\alpha_{eq}(a, a') = \begin{cases} \{*\} & \text{if } a \sim_\alpha a' \\ \emptyset & \text{otherwise} \end{cases}$$

Moreover if $\Gamma \vdash p : P$ is a r-terms and we have a set-theoretic interpretation of r and of the variables in Γ, then we have a set-theoretic interpretation of p. We will denote p both the r-term and its underlying set-theoretic interpretation, because the context prevents ambiguity.

C.1 Definition

C.1.1 Definition of Ob_H

We generate Ob_H by the constructor

$$R : \alpha_{set} \to \mathrm{Ob}_H \to \mathrm{Ob}_H$$

So we have a set-theoretic function R.

C.1.2 Definition of Hom_H

We generate Hom_H by the following list of constructors:

- Reflexivity, symmetry and transitivity

$$\mathrm{Id} : (a \in \mathrm{Ob}_H) \to \mathrm{Hom}_H(a, a)$$

$$\_^{-1} : (a, b \in \mathrm{Ob}_H) \to \mathrm{Hom}_H(a, b) \to \mathrm{Hom}_H(b, a)$$

$$\_ \cdot \_ : (a, b, c :\in \mathrm{Ob}_H) \to \mathrm{Hom}_H(a, b) \to \mathrm{Hom}_H(b, c) \to \mathrm{Hom}_H(a, c)$$

- The fact that R preserves equivalence

$$\mathbf{ap}_R : (a, a' \in \alpha_{set}) \to \alpha_{eq}(a, a') \to (b, b' \in \mathrm{Ob}_H)$$

$$\to \mathrm{Hom}_H(b, b') \to \mathrm{Hom}_H(R(a, b), R(a', b'))$$

- And the path constructor

$$S : (a \in \beta) \to (b \in \mathrm{Ob}_H) \to \mathrm{Hom}_H(p, q) \to \mathrm{Hom}_H(p', q')$$

C.1.3 The constructors

We want to check that there are terms r and s in the model.

A term of type $\alpha \twoheadrightarrow H \twoheadrightarrow H$ is a function $f : \alpha_{set} \to \mathrm{Ob}_H \to \mathrm{Ob}_H$ such that if $a \sim_\alpha a'$ and $b \sim_H b'$ (that is, $\mathrm{Hom}_H(b, b')$ inhabited) then $f(a, b) \sim_H f(a', b')$. So the set-theoretic function R induces such a term, because of $\mathbf{ap}_R$.

The set-theoretic function S induces a term because it takes value in an identity type, where the relation connects everything.

C.2 Sketch of proof for the induction principle

Assume that $x : H \vdash B(x)$, that is we have a family of sets $(B_x)_{x \in \mathrm{Ob}_H}$. Moreover assume that we have constructors :

$$\tilde{r} : (x : \alpha) \twoheadrightarrow (y : H) \twoheadrightarrow B(y) \twoheadrightarrow B(r(x, y))$$

and

$$\tilde{s} : (x : \beta) \twoheadrightarrow (y : A) \twoheadrightarrow B(y) \twoheadrightarrow (z : p =_A q) \twoheadrightarrow T_t(p) =^B_z T_t(q) \twoheadrightarrow T_t(p') =^B_{s(x,y,z)} T_t(q'))$$

We want to show there exists a term $f : (x : H) \twoheadrightarrow B(x)$ such that $f(r(x, y)) = \tilde{r}(x, y, f(y))$. Indeed the equality on $\mathbf{apd}_f$ is trivial here since it takes value in singletons.

Firstly we want a set-theoretic function $f_0 : (x : \mathrm{Ob}_H) \to B_x$. The fact that we have such a set-theoretic function is true because of the inductive definition of the set Ob_H. Indeed we have the underlying function of $\tilde{r}$, denoted by $\tilde{R}$, is of type $(x : \alpha_{set}) \to (y : \mathrm{Ob}_H) \to B_y \to B_{R(x,y)}$. So we define f_0 by $f_0(R(x,y)) = \tilde{R}(x, y, f_0(y))$.

Remark 6. *In the model, we have the judgmental unicity of terms defined by induction. Indeed the underlying function of a term such that $f(r(x,y)) = \tilde{r}(x, y, f(y))$ have to check the equation above, which determine uniquely such a function.*

We now have to check that if $\mathrm{Hom}_H(a, a')$ is inhabited, then $f_0(a) \sim^B_{a,a'} f_0(a')$. We proceed by induction on Hom_H.

- If the relation is generated by symmetry, transitivity or reflexivity, it easily follows from the symmetry, transitivity or reflexivity of the relation over B.
- If the relation is generated by $\mathbf{ap}_R$, then we have $x, x' \in \alpha_{set}$ such that $x \sim_\alpha x'$ and $y, y' \in \mathrm{Ob}_H$, such that $\mathrm{Hom}_H(y, y')$ is inhabited. So by induction hypothesis we have $f_0(y) \sim^B_{y,y'} f_0(y')$. We need to show that $f_0(R(x,y)) \sim^B_{R(x,y),R(x',y')} f_0(R(x',y'))$, that is $\tilde{R}(x, y, f_0(y))) \sim^B_{R(x,y),R(x',y')} \tilde{R}(x', y', f_0(y')))$. This is true by definition of $\tilde{r}$ being a term, that is $\mathbf{apd}_{\tilde{r}}$ applied to $x, y, f_0(y)$ and $x', y', f_0(y')$.
- If the relation is generated by S, we have $x \in \beta_{set}$, $y \in \mathrm{Ob}_H$ and $\mathrm{Hom}_H(p, q)$ is inhabited, and by induction hypothesis $f_0(p) \sim^B_{p,q} f_0(q)$. We need to show that $f_0(p') \sim^B_{p',q'} f_0(q')$. We evaluate $\tilde{S}$ at x, y, and $f_0(y)$, and we know that $\mathrm{Hom}_H(p, q)$ is inhabited, so we have that if $T_t(p)[\tilde{y} := f_0(y)] \sim^B_{p,q} T_t(q)[\tilde{y} := f_0(b)]$ then $T_t(p')[\tilde{y} := f_0(y)] \sim^B_{p',q'} T_t(q')[\tilde{y} := f_0(b)]$. But a set-theoretic version of lemma 5 gives e.g. $T_t(q')[\tilde{y} := f_0(y)] = f_0(q')$. So we can conclude.

We do not check that the function associating f to such $\tilde{r}$ and $\tilde{s}$ is indeed a term Rec.

D Sketch of level 2

We present here a very sketchy step toward level 2. The main point is that things seem to be working similarly to level 1. There is no actual proof, and some statements might be wrong.

D.1 A schema for level 2

D.1.1 Assumption on the model

Assume we have a model of type theory supporting Π-types, intensional identity types, heterogenous identity paths.

Moreover we need heterogenous identity types of level 2, that is given $a, b : A$, $x : B(a)$, $y : B(b)$, $p, q : a =_A b$, $u : x =^B_p y$, $v : x =^B_q y$ and $i : p =_{a=_A b} q$, we have a type $u =^{x=^B_{\_} y}_i v$, such that if i is reflexivity then we have a the regular equality $u =_{x=^B_p y} v$.

If $f : (x : A) \to B(x)$, and we have $a, b : A$, $p, q : a =_A b$ and $i : p =_{a=_A b} q$, then there exists $\mathbf{apd}^2_f(i) : \mathbf{apd}_f(p) =^{f(a)=^B_{\_} f(b)}_i \mathbf{apd}_f(q)$.

D.1.2 Syntax of the constructors

We want to define a HIT of level 2 called H.

We use a point, a path and a surface constructors of the following forms :

$$r : \alpha_1 \to H \to H$$

$$s : (a_2 : \alpha_2) \to (b_2 : H) \to p_1 =_H q_1 \to p_2 =_H q_2$$

$$t : (a_3 : \alpha_3) \to (b_3 : H) \to (c_3 : p_3 =_H q_3)$$
$$\to g_1 =_{p_4 =_H q_4} h_1 \to g_2 =_{p_5 =_H q_5} h_2$$

Where

- The α_i are without occurrences of H
- The p_i and q_i are built using variables previously declared of type H or α_i, λ-abstractions, applications and the point constructor. (We recall that such a term is called a r-term)
- The h_i and g_i are built using :
  - variables previously declared of type H, $p =_H q$, or α_i
  - λ-abstractions, applications.
  - compositions, inverses, identities of paths.
  - point and path constructors r and s.

  Such a term is called a r, s-path.

Remark 7. *Should we authorize e.g. p_4 to use path constructors, compositions, and so on ? The given form is restrictive on the dependencies.*

Such a type H should have terms of the type of the constructors in the empty context. Moreover we want to define its induction principle.

D.1.3 Extending the translations

Assume $x : H \vdash B(x)$. We want to define $f : (x : H) \to B(x)$ using the induction principle for H. In order to be able to state it correctly we need to extend the translations.

We extend the translation T on types with identities between r-terms by defining (for $i : p =_H q$)

$$T(i, p =_H q) = T_t(p) =_i^B T_t(q)$$

Lemma 6. *If $\Gamma \vdash p : \beta$ then $T(\Gamma) \vdash T(p, \beta)$.*

Here we need $T(\Gamma)$ instead of Γ, because we need to interpret some translations of r-terms $T_t(p)$.

The translation on contexts made of simple types is extended to contexts built using equalities of r-terms as well.

Proposition 2. *Let p and q be r-terms, and c be a r, s-path.*

Assuming we have $\Gamma \vdash c : p =_H q$ then there is a term $T_t^2(c)$ such that $T(\Gamma) \vdash T_t^2(c) : T_t(p) =_c^B T_t(q)$.

The only new cases are compositions, inverses, identities of path. These cases ought to be axioms on heterogeneous equality.

D.1.4 The induction principle

We present the desired induction principle.

Assume given terms interpreting f on constructors, that is $\tilde{r}$ and $\tilde{s}$ of the same type as on level 1 and $\tilde{t}$ such that

$$\tilde{t} : (a_3 : \alpha_3) \to (b_3 : H) \to (\tilde{b}_3 : B(b_3)) \to (c_3 : p_3 =_H q_3) \to (\tilde{c}_3 : T_t(p_3) =_{c_3}^B T_t(q_3))$$

$$\to (d_3 : g_1 =_{p_4 =_H q_4} h_1) \to T_t^2(g_1) =_{d_3}^{T_t(p_4) =_{\_}^H T_t(q_4)} T_t^2(h_1) \to T_t^2(g_2) =_{t(a_3,b_3,c_3,d_3)}^{T_t(p_5) =_{\_}^H T_t(q_5)} T_t^2(h_2)$$

there is a function such that :

$f(r(a_1, b_1)) = \tilde{r}(a_1, b_1, f(b_1))$
$\mathbf{apd}_f(s(a_2, b_2, c_2)) = \tilde{s}(a_2, b_2, f(b_2), c_2, \mathbf{apd}_f(c_2))$
$\mathbf{apd}_f^2(t(a_3, b_3, c_3, d_3)) = \tilde{t}_i(a_3, b_3, f(b_3), c_3, \mathbf{apd}_f(c_3), d_3, \mathbf{apd}_f^2(d_3))$

The second equation is well-typed (admitting the first equation) because $T_t(p)[\tilde{b}_2 := f(b_2)] = f(p)$ for p a r-term with variables $b_2 : H$ and $a_2 : \alpha_2$.

To check that the third one is well-typed (admitting the other two) we need to show that for g a r, s-path build using variables $a_3 : \alpha_3$, $b_3 : H$ and $c_3 : p_3 =_H q_3$, we have to check that $T_t^2(g)[\tilde{b}_3 := f(b_3), \tilde{c}_3 := \mathbf{apd}_f(c_3)] = \mathbf{apd}_f(g)$.

Remark 8. *Morally, the terms $\tilde{r}$, $\tilde{s}$ and $\tilde{t}$ are simply translations for r, s and t.*

D.2 Towards the interpretation in the groupoid model

Here we present some hints toward a proof that the groupoid model (as described by Fabian Ruch [19]) supports HITs of level 2. Since now there are interesting things happening in higher dimensions, we want a type in the empty context to be a groupoid instead of a setoid. A non-dependent function between two such type is a functor.

D.2.1 Presentation

Assume given some point, path and surface constructors for a non-dependent HIT of level 2 called H.

We want to interpret H in the groupoid model, so we want a groupoid. For this we build :

- $\mathrm{Ob}_H : \mathrm{Set}$
- $\mathrm{Hom}_H : \mathrm{Ob}_H \to \mathrm{Ob}_H \to \mathrm{Set}$
- $\mathrm{Sur}_H : (a, b : \mathrm{Ob}_H) \to \mathrm{Hom}_H(a, b) \to \mathrm{Hom}_H(a, b) \to \mathrm{Set}$

An then we will define the objects of H as Ob_H and $H(a, a')$ (the morphism between a and a' in H) as $\mathrm{Hom}_H(a, a')$ quotiented by $\mathrm{Sur}_H(a, a'; \_, \_)$ taken as an equivalence relation on $\mathrm{Hom}_H(a, a')$).

Be careful, the notation Hom_H is not well chosen here : it does not denote sets of morphisms in H.

D.2.2 Construction of Ob_H and Hom_H

Ob_H is simply the inductive set generated by a constructor R (corresponding to r)

$$R : \alpha_1^{set} \to \mathrm{Ob}_H \to \mathrm{Ob}_H$$

where α_1^{set} is the underlying set of α_1.

Hom_H is the inductive family generated by

- reflexivity, symmetry and transitivity

$$\mathrm{Id} : (a \in \mathrm{Ob}_H) \to \mathrm{Hom}_H(a, a)$$

$$\_^{-1} : (a, b \in \mathrm{Ob}_H) \to \mathrm{Hom}_H(a, b) \to \mathrm{Hom}_H(b, a)$$

$$\_ \cdot \_ : (a, b, c \in \mathrm{Ob}_H) \to \mathrm{Hom}_A(a, b) \to \mathrm{Hom}_A(b, c) \to \mathrm{Hom}_A(a, c)$$

- the fact that R is acting on arrows (here $\alpha_1^{eq}(a_1, a_1')$ is the set of morphism in α_1 between a_1 and a_1') :

$$\mathbf{ap}_R : (a_1, a_1' \in \alpha_1^{set}) \to \alpha_1^{eq}(a_1, a_1') \to (b_1, b_1' \in \mathrm{Ob}_H)$$

$$\to \mathrm{Hom}_H(b_1, b_1') \to \mathrm{Hom}_H(R(a_1, b_1), R(a_1', b_1'))$$

- and the path constructor, that is

$$S : (a_2 \in \alpha_2^{set}) \to (b_2 \in \mathrm{Ob}_H) \to \mathrm{Hom}_H(p_1, q_1) \to \mathrm{Hom}_H(p_2, q_2)$$

Note we will omit the beginning and endpoint in $\mathbf{ap}_R$, abbreviating $\mathbf{ap}_R(a_1, a_1', x, b_1, b_1', y)$ as $\mathbf{ap}_R(x, y)$

D.3 Construction of Sur_H

The construction of Sur_H is more complicated, as expected.

D.3.1 Groupoid and equivalence

These rules are the 2-dimensional part of a groupoid.

- $\mathrm{Sur}_H(a, b, \_, \_)$ is an equivalence

$$\mathrm{Id} : (a, b \in \mathrm{Ob}_H) \to (p \in \mathrm{Hom}_H(a, b)) \to \mathrm{Sur}_H(a, b, p, p)$$

$$\_^{-1} : (a, b \in \mathrm{Ob}_H) \to (p, q : \mathrm{Hom}_H(a, b)) \to \mathrm{Sur}_H(a, b, p, q) \to \mathrm{Sur}_H(a, b, q, p)$$

$$\_\cdot\_ : (a, b : \mathrm{Ob}_H) \to (p, q, r : \mathrm{Hom}_H(a, b)) \to \mathrm{Sur}_H(a, b, p, q) \to \mathrm{Sur}_H(a, b, q, r) \to \mathrm{Sur}_H(a, b, p, r)$$

- Ob_H and Hom_H form a groupoid

$$neutr_1 : (a, b \in \mathrm{Ob}_H) \to (p \in \mathrm{Hom}_H(a, b)) \to \mathrm{Sur}_H(a, b, p, Id_a \cdot p)$$

$$neutr_2 : (a, b \in \mathrm{Ob}_H) \to (p \in \mathrm{Hom}_H(a, b)) \to \mathrm{Sur}_H(a, b, p, p \cdot Id_b)$$

$$assoc : (a, b, c, d \in \mathrm{Ob}_H) \to (p \in \mathrm{Hom}_H(a, b)) \to (q \in \mathrm{Hom}_H(b, c)) \to (r \in \mathrm{Hom}_H(c, d)) \to \mathrm{Sur}_H(a, d, (p \cdot q) \cdot r, p \cdot (q \cdot r))$$

$$inv_1 : (a, b \in \mathrm{Ob}_H) \to (p \in \mathrm{Hom}_H(a, b)) \to \mathrm{Sur}_H(a, a, p \cdot p^{-1}, Id_a)$$

$$inv_2 : (a, b \in \mathrm{Ob}_H) \to (p \in \mathrm{Hom}_H(a, b)) \to \mathrm{Sur}_H(b, b, p^{-1} \cdot p, Id_b)$$

- There are whiskerings

$$wisk_1 : (a, b, c \in \mathrm{Ob}_H) \to (p, q : \mathrm{Hom}_H(a, b)) \to (r : \mathrm{Hom}_H(b, c)) \to \mathrm{Sur}_H(a, b, p, q) \to \mathrm{Sur}_H(a, c, p \cdot r, q \cdot r)$$

$$wisk_2 : (a, b, c \in \mathrm{Ob}_H) \to (p, q \in \mathrm{Hom}_H(b, c)) \to (r \in \mathrm{Hom}_A(a, b)) \to \mathrm{Sur}_H(b, c, p, q) \to \mathrm{Sur}_H(a, c, r \cdot p, r \cdot q)$$

D.3.2 R is a term

- $\mathbf{ap}_R$ is well defined

$$cong_R : (a_1, a_1' \in \alpha_1^{set}) \to (x \in \alpha_1^{eq}(a_1, a_1')) \to (b_1, b_1' \in \mathrm{Ob}_H) \to (y, y' \in \mathrm{Hom}_H(b_1, b_1'))$$
$$\to \mathrm{Sur}_H(b_1, b_1', y, y') \to \mathrm{Sur}_H(R(a_1, b_1), R(a_1', b_1'), \mathbf{ap}_R(x, y), \mathbf{ap}_R(x, y'))$$

- R preserve identity

$$id_R : (a_1 \in \alpha_1^{set}) \to (b_1 \in \mathrm{Ob}_H) \to \mathrm{Sur}_H(R(a_1, b_1), R(a_1, b_1), \mathbf{ap}_R(\mathrm{Id}_{a_1}, \mathrm{Id}_{b_1}), \mathrm{Id}_{R(a_1,b_1)})$$

- R preserve composition

$$comp_R : (a_1, a_1', a_1'' \in \alpha_1^{set}) \to (p \in \alpha_1^{eq}(a_1, a_1')) \to (p' \in \alpha_1^{eq}(a_1', a_1''))$$
$$\to (b_1, b_1', b_1'' \in \mathrm{Ob}_H) \to (q \in \mathrm{Hom}_H(b_1, b_1')) \to (q' \in \mathrm{Hom}_H(b_1', b_1''))$$
$$\to \mathrm{Sur}_H(R(a_1, b_1), R(a_1'', b_1''), \mathbf{ap}_R(p \cdot p', q \cdot q'), \mathbf{ap}_R(p, q) \cdot \mathbf{ap}_R(p', q'))$$

Please note this imply the symmetry relation.

D.3.3 S is a term

We need a lemma which we have not proven yet but which sounds reasonable.

Lemma 7. *Let $p(a_2, b_2)$ be a r-term of type H build on variables $a_2 : \alpha_2$ and $b_2 : H$.*
Assume $a, a' \in \alpha_2^{set}$ and $b, b' \in \mathrm{Ob}_H$.
From $x \in \alpha_2^{eq}(a, a')$ and $y \in \mathrm{Hom}_H(b, b')$, and we can build $p(x, y) : \mathrm{Hom}_H(p(a, b), p(a', b'))$ (where $p(a, b)$ is the underlying point of the term p in a, b).

Here are the constructors stating that S is a term.

- S is well-defined

$$cong_S : (a_2 \in \alpha_2^{set}) \to (b_2 \in \mathrm{Ob}_H) \to (y, y' \in \mathrm{Hom}_H(p_1, q_1)) \to \mathrm{Sur}_H(p_1, q_1, y, y')$$
$$\to \mathrm{Sur}_H(p_2, q_2, S(a_2, b_2, y), S(a_2, b_2, y'))$$

- S acts on arrows

$$\mathbf{ap}_S : (a_2, a_2' \in \alpha_2^{set}) \to (x \in \alpha_2^{eq}(a_2, a_2')) \to (b_2, b_2' \in \mathrm{Ob}_H) \to (y \in \mathrm{Hom}_H(b_2, b_2'))$$
$$\to (c_2 \in \mathrm{Hom}_H(p_1(a_2, b_2), q_1(a_2, b_2)) \to (c_2' \in \mathrm{Hom}_H(p_2(a_2', b_2'), q_2(a_2', b_2'))$$
$$\to \mathrm{Sur}_H(p_1(a_2, b_2), q_1(a_2', b_2'), c_2 \cdot q_1(x, y), p_1(x, y) \cdot c_2')$$
$$\to \mathrm{Sur}_H(p_2(a_2, b_2), q_2(a_2', b_2'), S(a_2, b_2, c_2) \cdot q_2(x, y), p_2(x, y) \cdot S(a_2', b_2', c_2'))$$

Note that will S will be a functor.

D.3.4 The surface constructor

$T : (a_3 \in \alpha_3^{set}) \to (b_3 \in \mathrm{Ob}_H) \to (c_3 \in \mathrm{Hom}_H(p_3, q_3)) \to \mathrm{Sur}_H(p_4, q_4, g_1, h_1) \to \mathrm{Sur}_H(p_5, q_5, g_2, h_2)$

D.4 Sketch of proof of the induction principle

Assume $x : H \vdash B(x)$, and that we have translations of the point, path and surface constructors. Then we have a set B_x for each $x \in \mathrm{Ob}_H$, and a set $B_p(b, b')$ for each $p \in \mathrm{Hom}_H(a, a')$, $b \in B_a$ and $b' \in B_{a'}$. Moreover we have a composition, inverse and identities of such elements.

We want to define a term $f : (x : H) \twoheadrightarrow B(x)$ satisfying the equations

$f(r(a_1, b_1)) = \tilde{r}(a_1, b_1, f(b_1))$
$\mathbf{apd}_f(s(a_2, b_2, c_2)) = \tilde{s}(a_2, b_2, f(b_2), c_2, \mathbf{apd}_f(c_2))$
$\mathbf{apd}_f^2(t(a_3, b_3, c_3, d_3)) = \tilde{t}(a_3, b_3, f(b_3), c_3, \mathbf{apd}_f(c_3), d_3, \mathbf{apd}_f^2(d_3))$

We need to define the function on sets, and then on arrows and finally check it respect the equivalence relation induced by Sur_H.

D.4.1 Definition on objects and arrows

We simply define $f_0 : (x \in \mathrm{Ob}_H) \to B_x$ using the underlying function of $\tilde{r}$ (noted $\tilde{R}$) on R. So we have a function which maps $x \in \mathrm{Ob}_H$ to an element of B_x, with $f_0(R(a_1, b_1)) = \tilde{R}(a_1, b_1, f_0(b_1))$

Then we need its action on arrow. We would like to define $f_1 : (a, b \in \mathrm{Ob}_H) \to (p \in \mathrm{Hom}_H(a, b)) \to B_p(f_0(a), f_0(b))$. We proceed by induction on Hom_H.

- For identity, inverse and composition we simply use the identity, inverse and composition of B. So we guarantee that it will be a functor.
- In the case of an $\mathbf{ap}_R$, we assume $a_1, a_1' \in \alpha_1^{set}$, $x \in \alpha_1^{eq}(a_1, a_1')$, $b_1, b_1' \in \mathrm{Ob}_H$, and $y \in \mathrm{Hom}_H(b_1, b_1')$. By induction hypothesis we have $f_1(y) \in B_y(f_0(b_1), f_0(b_1'))$. We want something in $B_{\mathbf{ap}_R(x,y)}(f_0(R(a_1, b_1)), f_0(R(a_1', b_1')))$, that is $B_{\mathbf{ap}_R(x,y)}(\tilde{R}(a_1, b_1, f_0(b_1)), \tilde{R}(a_1', b_1', f_0(b_1')))$. So, since $\tilde{r}$ is a term we can use $\mathbf{apd}_{\tilde{r}}(x, y, f_1(y))$.
- For path constructor we simply define $f_1(S(a_2, b_2, c_2)) = \tilde{S}(a_2, b_2, f_0(b_2), c_2, f_1(c_2))$ where $\tilde{S}$ is the underlying function of $\tilde{s}$.

D.4.2 A sketch of verification that f_1 respects Sur_H

We want to show that if $\mathrm{Sur}_H(a, b, p, q)$ is inhabited, then $f_1(p) = f_1(q)$ (the equality is set-theoretic). It is even more sketchy than what comes before.

We proceed by induction on Sur_H.

- The cases of identity, inverse and composition follow from reflexivity, symmetry and transitivity of equality

- The cases of the groupoid rules and whiskerings follows from the fact that f_1 is a functor.
- If the rules is certifying that f is a term then it follows from the fact that $f_1(\mathbf{ap}_R(x, y)) = \mathbf{ap}_{\tilde{r}}(x, y, f_1(y))$, and that f_1 and $\mathbf{ap}_{\tilde{r}}$ are functors.
- The case of $cong_S$ follows from the fact that $f_1(S(a_2, b_2, y)) = \tilde{s}(a_2, b_2, f_0(b_2), y, f_1(y))$. The case of $comp_S$ is harder to see, we would need to go into the details of lemma 7.
- The case of the surface constructor should follow from the underlying function of $\tilde{t}$, using equalities relating f_1 and T_t^2.