Finitary Higher Inductive Types in the Groupoid Model

Citation for the original published paper (version of record):
Dybjer, P., Moeneclaey, H. (2018). Finitary Higher Inductive Types in the Groupoid Model. Electronic Notes in Theoretical Computer Science, 336: 119-134.
http://dx.doi.org/10.1016/j.entcs.2018.03.019

Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science

Electronic Notes in Theoretical Computer Science 336 (2018) 119–134

www.elsevier.com/locate/entcs

Finitary Higher Inductive Types in the Groupoid Model

Peter Dybjer[1,2]

Department of Computer Science and Engineering
Chalmers University of Technology
Gothenburg, Sweden

Hugo Moeneclaey[3]

Ecole normale supérieure de Paris-Saclay
Paris, France

Abstract

A higher inductive type of level 1 (a 1-hit) has constructors for points and paths only, whereas a higher inductive type of level 2 (a 2-hit) has constructors for surfaces too. We restrict attention to finitary higher inductive types and present general schemata for the types of their point, path, and surface constructors. We also derive the elimination and equality rules from the types of constructors for 1-hits and 2-hits. Moreover, we construct a groupoid model for dependent type theory with 2-hits and point out that we obtain a setoid model for dependent type theory with 1-hits by truncating the groupoid model.

Keywords: intuitionistic type theory, identity types, homotopy type theory, higher inductive types, setoids, groupoids

1 Introduction

Martin-Löf [14] introduced the general identity type former $\mathrm{I}(A, a, a')$, the elements of which are proofs that a and a' are equal elements of A. Since A can be any type, even an identity type, we can iterate this type former and obtain an infinite tower of higher identity types $A, \mathrm{I}(A, a, a'), \mathrm{I}(\mathrm{I}(A, a, a'), p, p'), \mathrm{I}(\mathrm{I}(\mathrm{I}(A, a, a'), p, p'), \theta, \theta')$, etc. In extensional type theory [15,16] this hierarchy collapses, since it has a rule forcing $\mathrm{I}(A, a, a')$ to have at most one element. However, this is not so in intensional type theory [14,17]. As Hofmann and Streicher [12] showed, it has a model, the

[1] The first author was supported in part by the Swedish Research Council (Vetenskapsrådet) grant "Types for Proofs and Programs".

[2] Email: peterd@chalmers.se

[3] Email: hmoenecl@ens-paris-saclay.fr

https://doi.org/10.1016/j.entcs.2018.03.019

groupoid model, where $\mathrm{I}(A, a, a')$ may have more than one element. In fact, it was later shown that intensional type theory has infinite-dimensional models in Kan simplicial sets [25] and Kan cubical sets [2] which support interesting new axioms such as Voevodsky's univalence axiom.

A higher inductive type (hit) in the sense of Bauer, Lumsdaine, Schulman, and Warren is a type where all the iterated identity types are inductively generated. Higher inductive types occupy a central role in homotopy type theory [22], where the tower of identity types is given a topological interpretation: a and a' above are points in a space, p and p' are paths from a to a', θ and θ' are homotopies (or surfaces) between p and p', etc. In this way hits are used as abstract models of certain spaces, such as the interval, the circle, the sphere, the torus, etc.

Although the idea behind higher inductive types is simple, a comprehensive account of their syntax and semantics is still lacking. There is work on their categorical semantics [13,24] and on the semantics of some examples in cubical sets [3]. However, neither of these works answers the question of what a general notion of higher inductive type is. We quote the HoTT book [22, p 179]:

> In this book we do not attempt to give a general formulation of what constitutes a "higher inductive definition" and how to extract the elimination rule from such a definition – indeed, this is a subtle question and the subject of current research. Instead we will rely on some general informal discussion and numerous examples.

So we would like to know what the types of constructors for hits look like in general and what the associated elimination and equality rules are. We would also like to provide a model which shows the consistency of a general theory of higher inductive types. Actually, what is their foundational status and their relation to Martin-Löf's meaning explanations [15,16]? Can type theory with higher inductive types be modelled in type theory without higher inductive types, by reducing the meaning of higher inductive types to the standard inductive or inductive-recursive types?

Our goal is thus to formulate a general theory of higher inductive types similar to the theory of inductive families [5,4] and the theory of inductive-recursive definitions [8,9,10]. In this paper we take a step towards this goal by limiting ourselves to the simplest kinds of hits, the 1-hits with only point and path constructors and the 2-hits which may have surface constructors too. Moreover, we limit ourselves to *finitary* such constructors in a sense to be made precise below. We propose general schemata for introduction rules, and a method for deriving the elimination and equality rules from the introduction rules. The schema for 1-hits can be interpreted in the setoid model, where types are interpreted as setoids (sets with equivalence relations). Similarly, we interpret our schema for 2-hits in the groupoid model, where types are groupoids (categories where all arrows are isomorphisms).

Plan of the paper

In Section 2 we present an example of a 1-hit where the points are combinatory terms and the paths are proofs that two combinatory terms are convertible. We also sketch the interpretation of this 1-hit as a setoid. The aim here is to highlight

the close connection between 1-hits and equational theories. In Section 3 we first recall the general schema for inductive types and inductively defined binary relations. Although these schemata admit *generalized inductive definitions* of infinitely branching trees (such as the elements of W-types) we explain why we here restrict attention to 1-hits given by *finitary inductive definitions*. We propose a schema for the introduction rules for such types and show how to derive their elimination and equality rules. In Section 4 we recall how a 2-hit can represent the torus. In Section 5 we propose an extended schema for 2-hits. We also discuss which of the hits in the HoTT-book are covered by our schema and which are not. In Section 6 we show how to model 2-hits as groupoids. For reasons of space we do not present the setoid model of 1-hits separately. Instead the groupoid model is presented in such a way that the setoid model can be read off as a truncated version of the groupoid model. In Section 7 we mention some related work and suggestions for further research.

2 Combinatory Logic as a 1-Hit

We start by presenting a small fragment of dependent type theory with dependent functions types written $(x : A) \to B(x)$ and non-dependent ones written $A \to B$. Identity types are written $a =_A a'$ rather than $\mathrm{I}(A, a, a')$. To this theory we add the rules for the 1-hit CL with its formation rule, introduction rules for points and paths, and elimination and equality rules.

We use "mathematical" notation $f(x)$ for function application, $f(x, y)$ for $f(x)(y)$, etc. Furthermore, $x_1 : A_1, \ldots, x_n : A_n \vdash A$ means that A is a type in the context $x_1 : A_1, \ldots, x_n : A_n$. If $x : A \vdash C$ is a type (or term) depending on a variable x we often (but not always) use the notation $C(x)$ to emphasize this dependence, and also write $C(a)$ for the result of substituting a for x in C.

The *introduction rule* for identity types is

$$\mathrm{refl} : (x : A) \to x =_A x$$

for any type A. The *elimination rule* and *equality rule* (Paulin [21]) are

$$\mathrm{J}_C \ : \ (x : A) \to C(x, \mathrm{refl}(x)) \to (y : A) \to (z : x =_A y) \to C(y, z)$$
$$\mathrm{J}_C(x, d, x, \mathrm{refl}(x)) = d$$

where $x : A, y : A, z : x =_A y \vdash C(y, z)$. The usual rules of equality reasoning (transitivity, symmetry, replacement of equals for equals) can be derived from these rules. We shall also derive rules for a *heterogeneous identity type* $a =_p^B a'$ which lets us compare elements $a : B(x)$ and $a' : B(x')$ where $p : x =_A x'$, but $B(x)$ and $B(x')$ are not necessarily *definitionally (judgmentally)* equal. Another derivable rule is that a function $f : (x : A) \to B(x)$ must preserve identity: $\mathbf{apd}_f : (p : x =_A x') \to f(x) =_p^B f(x')$.

2.1 Rules for combinatory logic as a 1-hit

The formation rule states that CL is a type. The introduction rules are divided into two parts: the types of the point constructors $\mathrm{K}, \mathrm{S} : \mathrm{CL}$ and $\mathrm{app} : \mathrm{CL} \to \mathrm{CL} \to \mathrm{CL}$ and the types of the path constructors (for combinatory conversion)

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

The other rules of combinatory conversion (reflexivity, transitivity, symmetry, application preserves equality) follow from the fact that $=_{\mathrm{CL}}$ is an identity type.

The *elimination rule* for CL expresses how to define a function $f : (x : \mathrm{CL}) \to C(x)$ by structural induction on the point and path constructors (showing that it preserves identity). This rule has the assumptions $\tilde{\mathrm{K}} : C(\mathrm{K}), \tilde{\mathrm{S}} : C(\mathrm{S}), \tilde{\mathrm{app}} : (x : \mathrm{CL}) \to C(x) \to (y : \mathrm{CL}) \to C(y) \to C(\mathrm{app}(x, y))$, and also

$$\begin{aligned}
\tilde{\mathrm{K}}_{conv} &: (x, y : \mathrm{CL}) \to (\tilde{x} : C(x)) \to (\tilde{y} : C(y)) \\
&\to \tilde{\mathrm{app}}(\mathrm{app}(\mathrm{K}, x), \tilde{\mathrm{app}}(\mathrm{K}, \tilde{\mathrm{K}}, x, \tilde{x}), y, \tilde{y}) =^{C}_{\mathrm{K}_{conv}(x,y)} \tilde{x} \\
\tilde{\mathrm{S}}_{conv} &: (x, y, z : \mathrm{CL}) \to (\tilde{x} : C(x)) \to (\tilde{y} : C(y)) \to (\tilde{z} : C(z)) \\
&\to \tilde{\mathrm{app}}(\mathrm{app}(\mathrm{app}(\mathrm{S}, x), y), \tilde{\mathrm{app}}(\mathrm{app}(\mathrm{S}, x), \tilde{\mathrm{app}}(\mathrm{S}, \tilde{\mathrm{S}}, x, \tilde{x}), y, \tilde{y}), z, \tilde{z}) \\
&=^{C}_{\mathrm{S}_{conv}(x,y,z)} \tilde{\mathrm{app}}(\mathrm{app}(x, z), \tilde{\mathrm{app}}(x, \tilde{x}, z, \tilde{z}), \mathrm{app}(y, z), \tilde{\mathrm{app}}(y, \tilde{y}, z, \tilde{z})))
\end{aligned}$$

The *equality rules* are

$$\begin{aligned}
f(\mathrm{K}) &= \tilde{\mathrm{K}} \\
f(\mathrm{S}) &= \tilde{\mathrm{S}} \\
f(\mathrm{app}(x, y)) &= \tilde{\mathrm{app}}(x, f(x), y, f(y)) \\
\mathbf{apd}_f(\mathrm{K}_{conv}(x, y)) &= \tilde{\mathrm{K}}_{conv}(x, y, f(x), f(y)) \\
\mathbf{apd}_f(\mathrm{S}_{conv}(x, y, z)) &= \tilde{\mathrm{S}}_{conv}(x, y, z, f(x), f(y), f(z))
\end{aligned}$$

2.2 Setoid model

The above theory, dependent type theory with $(x : A) \to B(x)$, $a =_A a'$, and CL, has a setoid model [11]. In this model, a type is interpreted as a *setoid* A consisting of a set A_0 together with an equivalence relation R. Here we represent an equivalence relation as a binary family of sets $(A_1(x, x'))_{x,x' \in A_0}$ such that $A_1(x, x')$ is inhabited iff $R(x, x')$ holds and empty otherwise. Moreover, a *setoid map* between two setoids $A = (A_0, A_1)$ and $B = (B_0, B_1)$ is a function $f_0 : A_0 \to B_0$ together with a proof that it preserves the equivalence relation: $f_1 : (A_1(x, x')) \to B_1(f(x), f(x'))$.

There are two reasons for this representation. Although we officially use set theory as the metalanguage for our model constructions, we conjecture that the construction can be carried out in extensional type theory, where an equivalence relation will be implemented by a family of types. (Cf Hofmann and Streicher's similar claim [12] about their groupoid model.) However, there is another advantage: the setoid model can then be viewed as a truncated version of the groupoid model. The set of objects A_0 together with the hom-sets $A_1(x, x')$ of a groupoid form a setoid. In the sequel we will not distinguish between the family A_1 and the equivalence relation R which it represents. (We also remark that extensional type theory has a direct interpretation in classical set theory [4], so that type-theoretic notation can be read off as official set theory.)

See Moeneclaey [18] for the details of the construction of a category with families

[6] of setoids and setoid maps which supports dependent function types, intensional identity types, and CL. Suffice it here to say that we interpret the type CL as the setoid $(\mathrm{CL}_0, \mathrm{CL}_1)$ where CL_0 is an inductive type generated by K, S, and app and CL_1 is an inductive family [5,4] generated by K_{conv} and S_{conv} and the constructors for transitivity, reflexivity, symmetry, and preservation of equality by app.

3 A Schema for 1-Hits

We now ask ourselves what introduction rules for points and paths look like in general. We also construct a setoid model for such a general notion of 1-hit. As mentioned above we are looking for a schema for hits in the style of the schema for inductive families [5,4]. The obvious first try is to stay as close as possible to that schema and stipulate that the type of a point constructor for a hit can have the same form as the type of a constructor for an inductive type; and the type of a path constructor for a hit can have the same form as the type of a constructor for a binary inductive family.

The general form of the type of a constructor for an inductive type H is

$$(x_1 : A_1) \to \cdots \to (x_m : A_m(x_1, \ldots, x_{m-1}))$$
$$\to (B_1(x_1, \ldots, x_m) \to \mathrm{H}) \to \cdots \to (B_n(x_1, \ldots, x_m) \to \mathrm{H}) \to \mathrm{H}$$

where A_1 is a type, $\ldots, A_m(x_1, \ldots, x_{m-1})$ is a type if $(x_1 : A_1, \ldots, x_{m-1} : A_{m-1}(x_1, \ldots, x_{m-2}))$, and $B_1(x_1, \ldots, x_m), \ldots, B_n(x_1, \ldots, x_m)$ are types if $x_1 : A_1, \ldots, x_m : A_m(x_1, \ldots, x_{m-1})$, and all those judgments are true in the theory without the rules for H. So A_i and B_j do not depend on H.

This general form is obtained by specializing the schema for inductive families [5,4] to the case of inductive types. We remark that B_j can in fact be any sequence of types. We call $x_i : A_i$ a *non-inductive* premise (or *side condition*) and $y_j : B_j(x_1, \ldots, x_n) \to \mathrm{H}$ an *inductive premise*. If B_j is the empty sequence for all j, then we have an *ordinary* or *finitary* inductive definition, otherwise we have a *generalized* inductive definition.

Note that this is only a mild generalization of the type of the constructor of the W-type [15], which is obtained by setting $m = n = 1$. In fact, in extensional type theory any inductive type with constructors of the above form is equivalent to a W-type. This is a special case of a more general theorem about the encoding of strictly positive inductive types as W-types [7].

However, a complication arises in the setoid interpretation of generalized inductive definitions of a hit H. Assume for example that H has a point constructor $\mathrm{c}_0 : (B \to \mathrm{H}) \to \mathrm{H}$ and that it is interpreted by the set of points H_0 and the family of sets of paths $\mathrm{H}_1(x, y)$ for $x, y \in \mathrm{H}_0$. Since functions must preserve the equivalence relation, we expect the type of the interpreted constructor for H_0 to be

$$\mathrm{c}_{00} : (f \in B_0 \to \mathrm{H}_0) \to ((x, x' \in B_0) \to B_1(x, x') \to \mathrm{H}_1(f(x), f(x'))) \to \mathrm{H}_0$$

(Note that this is a typing in the metalanguage and that we use "$\in$" rather than ":" for set/type membership.) Since H_1 will be interpreted as an inductive family, we conclude that H_0 and H_1 are simultaneously defined by an *inductive-inductive*

definition. However, we prefer not to complicate the metatheory - after all the theory of inductive-inductive definitions is complex [20,19]. For this reason, we choose here to restrict to the simpler case of finitary hits which still cover most hits in the HoTT-book [22], see Section 5.3. As the reader will see, already the theory of finitary 2-hits is quite complex, and we think that it is preferable to leave the additional complexity of generalized inductive definitions to future work.

3.1 A schema for point constructors

The general form of a type of a point constructor for a finitary hit is

$$\mathrm{c}_0 : (x_1 : A_1) \to \cdots \to (x_m : A_m(x_1, \ldots, x_{m-1})) \to \mathrm{H} \to \cdots \to \mathrm{H} \to \mathrm{H}$$

where A_1 is a type, ..., and $A_m(x_1, \ldots, x_{m-1})$ is a type if $x_1 : A_1, \ldots, x_{m-1} : A_{m-1}(x_1, \ldots, x_{m-2})$, all those judgments being true in the theory without the rules for H.

3.2 A schema for path constructors

The general form of the type of a path constructor for a finitary hit is:

$$\begin{aligned}
&\mathrm{c}_1 : (x_1 : B_1) \to \cdots \to (x_n : B_n(x_1, \ldots, x_n)) \\
&\quad \to (y_1 : \mathrm{H}) \to \cdots \to (y_{n'} : \mathrm{H}) \\
&\quad \to p_1(x_1, \ldots, x_n, y_1, \ldots, y_{n'}) =_{\mathrm{H}} q_1(x_1, \ldots, x_n, y_1, \ldots, y_{n'}) \\
&\quad \vdots \\
&\quad \to p_{n''}(x_1, \ldots, x_n, y_1, \ldots, y_{n'}) =_{\mathrm{H}} q_{n''}(x_1, \ldots, x_n, y_1, \ldots, y_{n'}) \\
&\quad \to p'(x_1, \ldots, x_n, y_1, \ldots, y_{n'}) =_{\mathrm{H}} q'(x_1, \ldots, x_n, y_1, \ldots, y_{n'})
\end{aligned}$$

where neither H nor $=_{\mathrm{H}}$ may appear in B_i. Moreover, the terms

$$\begin{aligned}
&p_1(x_1, \ldots, x_n, y_1, \ldots, y_{n'}), q_1(x_1, \ldots, x_n, y_1, \ldots, y_{n'}), \\
&\vdots, \\
&p_{n''}(x_1, \ldots, x_n, y_1, \ldots, y_{n'}), q_{n''}(x_1, \ldots, x_n, y_1, \ldots, y_{n'}), \\
&p'(x_1, \ldots, x_n, y_1, \ldots, y_{n'}), q'(x_1, \ldots, x_n, y_1, \ldots, y_{n'})
\end{aligned}$$

are *point constructor patterns* built up according to the following syntax

$$p ::= y \mid \mathrm{c}_0(a_1, \ldots, a_m, p_1, \ldots, p_k)$$

where $y : \mathrm{H}$ is a variable among $y_1, \ldots, y_{n'}$, where $p_j : \mathrm{H}$ are point constructor patterns, and where $x_1 : B_1, \ldots, x_n : B_n(x_1, \ldots, x_n) \vdash a_i : A_i(a_1, \ldots, a_{i-1})$ are terms built without using any rule for H.

Note that if we delete the second line of the type of the path constructor c_1, the schema looks exactly like the schema for constructors for binary finitary inductive families [5] (except that the present schema allows only point constructor patterns and not general terms for the p_j). However, we now ask ourselves whether H can appear in a side-condition B_j. Unfortunately, it cannot appear in an arbitrary way,

since a negative occurrence of H could lead to a contradiction. Therefore we simply forbid H to appear in B_j and have a separate list of premises of the form $y_k : \mathrm{H}$.

3.3 A simplified form for point and path constructors

In order to simplify the presentation of the elimination and equality rules, we only spell out the special case with one point constructor with $m = 1$ (one side condition) and one inductive premise

$$\mathrm{c}_0 : A \to \mathrm{H} \to \mathrm{H}$$

and one path constructor with $n = n' = n'' = 1$:

$$\mathrm{c}_1 : (x : B) \to (y : \mathrm{H}) \to p(x, y) =_{\mathrm{H}} q(x, y) \to p'(x, y) =_{\mathrm{H}} q'(x, y)$$

We emphasize that this simplification is only a matter of presentation. It is routine but verbose to generalize this simplified form to the general form for constructors for ordinary hits. (The general form is of course necessary to get interesting hits.)

3.4 Elimination and equality rules

The elimination rule expresses how to define a function $f : (x : \mathrm{H}) \to C(x)$ by structural induction on the points and paths (showing that the function preserves identity). More specifically, given

$$\tilde{\mathrm{c}_0} : (x : A) \to (y : \mathrm{H}) \to C(y) \to C(\mathrm{c}_0(x, y))$$
$$\tilde{\mathrm{c}_1} : (x : B) \to (y : \mathrm{H}) \to (\tilde{y} : C(y))$$
$$\to (z : p =_{\mathrm{H}} q) \to \mathrm{T}_0(p) =^{C}_{z} \mathrm{T}_0(q) \to \mathrm{T}_0(p') =^{C}_{\mathrm{c}_1(x,y,z)} \mathrm{T}_0(q')$$

where $\mathrm{T}_0(p) : C(p)$ is the *lifting* of $p : \mathrm{H}$ (to be defined below), we can define f by

$$f(\mathrm{c}_0(x, y)) = \tilde{\mathrm{c}_0}(x, y, f(y))$$
$$\mathbf{apd}_f(\mathrm{c}_1(x, y, z)) = \tilde{\mathrm{c}_1}(x, y, f(y), x, \mathbf{apd}_f(z))$$

Note that the second equality is a *definitional* equality, rather than only a *propositional* one, as in the HoTT-book [22]. This definitional equality will be validated by the groupoid model below.

3.5 The lifting function

We denote the "lifting" of a point constructor pattern $p : \mathrm{H}$ by $\mathrm{T}_0(p) : C(p)$. The idea is that $\mathrm{T}_0(p)$ will be equal to $f(p)$, but the latter is not defined yet. For example the liftings of the two point constructor patterns in the equation for $\tilde{\mathrm{K}}_{conv}$ are

$$\mathrm{T}_0(\mathrm{app}(\mathrm{app}(\mathrm{K}, x), y)) = \tilde{\mathrm{app}}(\mathrm{app}(\mathrm{K}, x), \tilde{\mathrm{app}}(\mathrm{K}, \tilde{\mathrm{K}}, x, \tilde{x}), y, \tilde{y})$$
$$\mathrm{T}_0(x) = \tilde{x}$$

This lifting function for CL is defined by $\mathrm{T}_0(x) = \tilde{x}, \mathrm{T}_0(y) = \tilde{y}, \mathrm{T}_0(\mathrm{K}) = \tilde{\mathrm{K}}, \mathrm{T}_0(\mathrm{S}) = \tilde{\mathrm{S}}$, and $\mathrm{T}_0(\mathrm{app}(t, t')) = \tilde{\mathrm{app}}(t, \mathrm{T}_0(t), t', \mathrm{T}_0(t'))$

In the general (simplified) schema the definition of $\mathrm{T}_0(p(x, y)) : C(p(x, y))$ is by induction on the form of the point constructor patterns $p(x, y)$:

$$\begin{aligned} \mathrm{T}_0(y) &= \tilde{y} \\ \mathrm{T}_0(\mathrm{c}_0(a,p)) &= \tilde{\mathrm{c}_0}(a,p,\mathrm{T}_0(p)) \end{aligned}$$

Hence, $x : B, y : \mathrm{H}, \tilde{y} : C(y) \vdash \mathrm{T}_0(p(x,y)) : C(p(x,y))$ and $\mathrm{T}_0(p)(x,y,f(y)) = f(p(x,y))$. Therefore the equation for $\mathbf{apd}_f(\mathrm{c}_1(x,y,z))$ above is well-typed.

We refer the reader to Moeneclaey [18] for the setoid model of the schema for 1-hits and again point out that it arises by truncating the groupoid model.

4 The Torus as a 2-Hit

We shall now consider 2-hits with surface constructors in addition to point and path constructors. An example is the following hit which represents the torus T^2 as a CW-complex [22]. It has four constructors:

$$\begin{aligned} \mathrm{base} &: \mathrm{T}^2 \\ \mathrm{path}_1 &: \mathrm{base} =_{\mathrm{T}^2} \mathrm{base} \\ \mathrm{path}_2 &: \mathrm{base} =_{\mathrm{T}^2} \mathrm{base} \\ \mathrm{surf} &: \mathrm{path}_1 \circ \mathrm{path}_2 =_{\mathrm{base}=_{\mathrm{T}^2}\mathrm{base}} \mathrm{path}_2 \circ \mathrm{path}_1 \end{aligned}$$

In order to state its elimination principle we will make use of a heterogeneous identity type of level 2. Let $a, a' : A$, $p, p' : a =_A a'$, $\theta : p =_{a=_A a'} p'$, $b : B(a)$, $b' : B(a')$, $q : b =^B_p b'$, $q' : b =^B_{p'} b'$. We write

$$q =^{b=^B_- b'}_{\theta} q'$$

for the heterogeneous identity of the paths q, q'.

We can now prove that functions preserve level 2 identities by identity elimination. If $f : (x : A) \to C(x)$ then

$$\mathbf{apd}^2_f : (\theta : p =_{x=_A x'} p') \to \mathbf{apd}_f(p) =^{f(x)=^C_- f(x')}_{\theta} \mathbf{apd}_f(p')$$

Now we can formulate the elimination rule for T^2. Assume $x : \mathrm{T}^2 \vdash C(x)$ and $\tilde{b} : C(\mathrm{base}), \tilde{p}_1 : \tilde{b} =^C_{\mathrm{path}_1} \tilde{b}, \tilde{p}_2 : \tilde{b} =^C_{\mathrm{path}_2} \tilde{b}$, and also

$$\tilde{s} : \tilde{p}_1 \circ' \tilde{p}_2 =^{\tilde{b}=^C\tilde{b}}_{\mathrm{surf}} \tilde{p}_2 \circ' \tilde{p}_1$$

Note the composition $\circ'$ of heterogeneous paths, which can be derived from the J-eliminator. Then there exists a function $f : (x : \mathrm{T}^2) \to C(x)$ such that

$$\begin{aligned} f(\mathrm{base}) &= \tilde{b} \\ \mathbf{apd}_f(\mathrm{path}_1) &= \tilde{p}_1 \\ \mathbf{apd}_f(\mathrm{path}_2) &= \tilde{p}_2 \\ \mathbf{apd}^2_f(\mathrm{surf}) &= \tilde{s} \end{aligned}$$

The last equation is well typed using the definitional equality $\mathbf{apd}_f(p \circ q) = \mathbf{apd}_f(p) \circ' \mathbf{apd}_f(q)$ which is true in the groupoid model. These equalities (and similar ones for heterogeneous inverses) are needed for our schema to be well-typed.

5 A Schema for Finitary 2-Hits

5.1 A schema for surface constructors

We shall now present a general schema for 2-hits. The form of point and path constructors are as for 1-hits. The simplified form of a surface constructor is

$$\begin{aligned}&\mathrm{c}_2 : (x : D) \to (y : \mathrm{H}) \to (z : p_3(x,y) =_{\mathrm{H}} q_3(x,y))\\ &\quad \to g_1(x,y,z) =_{p_4(x,y)=_{\mathrm{H}} q_4(x,y)} h_1(x,y,z)\\ &\quad \to g_2(x,y,z) =_{p_5(x,y)=_{\mathrm{H}} q_5(x,y)} h_2(x,y,z)\end{aligned}$$

(In the general form the four premises $x : D, y : \mathrm{H}, z : p_3 =_{\mathrm{H}} q_3, z' : g_1 =_{p_4 =_{\mathrm{H}} q_4} h_1$ become four finite (possibly empty) *sequences* of premises. Cf the general form of point constructors in 3.1 and path constructors in 3.2.) Here D is a correct type in a theory without the rules for H, so neither $\mathrm{H}, =_{\mathrm{H}}$ nor $=_{\_=_{\mathrm{H}}\_}$ may appear in it. Moreover, in the theory extended with H-formation and H-introduction for the point constructor c_0, all of $x : D, y : \mathrm{H} \vdash p_3, q_3, p_4, q_4, p_5, q_5 : \mathrm{H}$ are point constructor patterns (built up from variables by c_0) and g_1, h_1, g_2, h_2 are *path constructor patterns* built from the following grammar:

$$g ::= z \mid \mathrm{c}_1(a, p, g) \mid g \circ g \mid \mathrm{id} \mid g^{-1}$$

where $z : p_3 =_H q_3$ is a path variable, $x : D \vdash a : B$ is a term built without using rules for H, and $p : \mathrm{H}$ is a point constructor pattern under the assumption $x : D, y : \mathrm{H}$.

Each path constructor pattern comes with some definitional equalities which are valid in the groupoid model. It would also be natural to add a fourth "default constructor" $\mathbf{ap}_{\mathrm{c}_0}$ (corresponding to the arrow part of the point constructor in the model) to the grammar for path constructor patterns. However, checking the interpretation of $\mathbf{ap}_{\mathrm{c}_0}$ in the groupoid model requires lengthy calculations, which we have not yet completed.

5.2 Elimination and equality rules

The elimination rule expresses how to define a function $f : (x : \mathrm{H}) \to C(x)$ by induction with one case for each of the point, path, and surface constructors. To this end we assume we have step functions $\tilde{\mathrm{c}_0}$ and $\tilde{\mathrm{c}_1}$ as in the elimination rule for 1-hits and moreover a step function for the surface constructor

$$\begin{aligned}&\tilde{\mathrm{c}_2} : (x : D) \to (y : \mathrm{H}) \to (\tilde{y} : C(y)) \to (z : p_3 =_{\mathrm{H}} q_3)\\ &\quad \to (\tilde{z} : \mathrm{T}_0(p_3) =^{C}_{z} \mathrm{T}_0(q_3)) \to (t : g_1 =_{p_4 =_{\mathrm{H}} q_4} h_1)\\ &\quad \to \mathrm{T}_1(g_1) =^{\mathrm{T}_0(p_4)=^{\mathrm{H}}_{\_}\mathrm{T}_0(q_4)}_{t} \mathrm{T}_1(h_1) \to \mathrm{T}_1(g_2) =^{\mathrm{T}_0(p_5)=^{\mathrm{H}}_{\_}\mathrm{T}_0(q_5)}_{\mathrm{c}_2(x,y,z,t)} \mathrm{T}_1(h_2)\end{aligned}$$

The lifting $\mathrm{T}_1(g) : \mathrm{T}_0(p) =^{C}_{g} \mathrm{T}_0(q)$ of a path constructor pattern $g : p =_{\mathrm{H}} q$ is defined in an analogous way to the lifting $\mathrm{T}_0(p)$ of a point constructor pattern:

$$\begin{aligned}\mathrm{T}_1(z) &= \tilde{z}\\ \mathrm{T}_1(\mathrm{c}_1(x,y,g)) &= \tilde{\mathrm{c}_1}(x, y, \mathrm{T}_0(y), g, \mathrm{T}_1(g))\end{aligned}$$

$$\begin{aligned}\mathrm{T}_1(g \circ g') &= \mathrm{T}_1(g) \circ' \mathrm{T}_1(g')\\ \mathrm{T}_1(\mathrm{id}) &= \mathrm{id}\\ \mathrm{T}_1(g^{-1}) &= \mathrm{T}_1(g)^{-1'}\end{aligned}$$

where $p^{-1'}$ denotes the inverse of a path p.

The equality rules for f are:

$$\begin{aligned}f(\mathrm{c}_0(x,y)) &= \tilde{c}_0(x,y,f(y))\\ \mathbf{apd}_f(\mathrm{c}_1(x,y,z)) &= \tilde{c}_1(x,y,f(y),z,\mathbf{apd}_f(z))\\ \mathbf{apd}^2_f(\mathrm{c}_2(x,y,z,t)) &= \tilde{c}_2(x,y,f(y),z,\mathbf{apd}_f(z),t,\mathbf{apd}^2_f(t))\end{aligned}$$

Note that all three equality rules are *definitional* equalities which are valid in the groupoid model. The last equation type-checks for similar reasons as the second equation (see Section 3.5). We prove by induction on path constructor patterns that $\mathrm{T}_1(g)(x,y,f(y),z,\mathbf{apd}_f(z)) = \mathbf{apd}_f(g(x,y,z))$. This proof uses the definitional equalities $\mathbf{apd}_f(p \circ q) = \mathbf{apd}_f(p) \circ' \mathbf{apd}_f(q)$ and $\mathbf{apd}_f(p^{-1}) = \mathbf{apd}_f(p)^{-1'}$ which are valid in the groupoid model.

Even without these equalities the types of the two sides of the equalities are isomorphic. So it would be possible to add the suitable isomorphisms everywhere, but some coherence problems would arise.

5.3 Hits in the HoTT-book

Many of the hits in the HoTT-book are instances of our general *syntactic* schema. For example the interval, the circle, propositional truncation, the suspension, and the pushout are instances of our schema for 1-hits. The 2-sphere, the torus, the 0-truncation, and the set-quotient are instances of our schema for 2-hits. However, the alternative definition of the torus using hubs and spokes [22, p192] is not an instance of our schema for 1-hits, since it breaks the requirement that only point constructor patterns are allowed in the indices (endpoints) of the types of path constructors. Moreover, the alternative definition of 0-truncation [22, p199] uses generalized induction, and is not covered by our schema either.

6 Groupoid Model of the Schema for 2-Hits

We build on Hofmann and Streicher's [12] groupoid model of intensional type theory and have also made use of Ruch's path model [23]. Just as Hofmann and Streicher we work in set-theoretic meta-language, but conjecture that the model can also be carried out in extensional type theory. In order to suggest why this is the case we will write our definitions in a type-theoretic style, and use the fact that extensional type theory has a set-theoretic model, see e g [4].

A groupoid H (in type-theoretic style, cf the discussion of the representation of setoids) is a tuple $(\mathrm{H}_0, \mathrm{H}_1, \mathrm{H}_2, \circ, \mathrm{id}, (-)^1, \mathrm{tran}, \mathrm{refl}, \mathrm{sym}, \mathrm{w}_0, \mathrm{w}_1, \alpha, \lambda, \rho, \iota_0, \iota_1)$ where H_0 is the set of objects, $\mathrm{H}_1(a,a')_{a,a' \in \mathrm{H}_0}$ is the set of arrows from a to a' and $\mathrm{H}_2(f,f')_{f,f' \in \mathrm{H}_1(a,a')}$ is the set of proofs of equality of the arrows f and f'. Moreover, $\circ, \mathrm{id}, (-)^1$ denote respectively the composition, identity, and inverse opera-

tions. The remaining components witness that equality of arrows is a congruence relation $(\mathrm{w}_0, \mathrm{w}_1)$ which satisfies associativity (α), identity (λ, ρ), and inverse laws (ι_0, ι_1). When refering to a groupoid we typically omit all components except the three first: $\mathrm{H} = (\mathrm{H}_0, \mathrm{H}_1, \mathrm{H}_2)$.

The usual set-theoretic notion of groupoid is recovered by defining hom-sets as quotients $\mathrm{H}_1(a, a')/R(a, a')$, where $R(a, a')$ is the equivalence relation for arrows between a and a' generated by H_2. Furthermore, when we use set-theoretic metalanguage we shall for simplicity identify all proofs of equality of arrows so that $\theta, \theta' \in \mathrm{H}_2(f, f')$ implies $\theta = \theta'$. We call this unique element $*$. This representation does not only pave the way for a type-theoretic implementation of the groupoid model, but also for extending our work to the interpretation of 3-hits in a weak 2-groupoid model.

In the groupoid model, a family of types $x : A \vdash C(x)$ is interpreted as a functor from the groupoid A into the category of groupoids. We let C_0 denote the object part of that functor, so that $C_0(x)$ is a groupoid for $x \in A_0$, and C_1 denote the arrow part, so that $C_1(f) : C_0(x) \to C_0(x')$ is a *transport functor* for $f \in A_1(x, x')$. Furthermore, we use the notation $C_1'(x, x', f, y, y')$ for $f \in A_1(x, x'), y \in C_0(x), y' \in C_0(x')$ for the set of heterogeneous paths between points in different fibres (mediated by the transport). Moreover, we write $C_2'(f, f', \theta, z, z', g, g')$ where $f, f' \in A_1(x, x'), \theta \in A_2(f, f'), z \in C(x), z' \in C(x'), g \in C_1'(z, z', f), g' \in C_1'(z, z', f')$ for the heterogeneous equality (mediated by the transport) of the heterogeneous paths g, g'. A function $f : A \to B$ is interpreted as a functor F between groupoids. It is represented type-theoretically by a triple (F_0, F_1, F_2) representing the object, arrow, and proof of preservation of equality of arrows parts. Similarly, a function $x : A \vdash f(x) : C(x)$ is interpreted as a "dependent functor" between the source groupoid and the family of fibre groupoids. We refer to Hofmann and Streicher [12] for details of the interpretation.

Note that we justify definitional computation rules also for $\mathbf{ap}_f$ and $\mathbf{ap}^2_f$, whereas in the HoTT-book there is an informal discussion about models leading to those computation rules being propositional equalities only.

The groupoid model below captures the structure of paths up to homotopy. For example, one can show that in this model the hit T^2 is interpreted by its fundamental groupoid $\Pi_1(\mathrm{T}^2)$ (i.e. isomorphic to $\mathbb{Z} \oplus \mathbb{Z}$). However, the groupoid model does not capture the structure in the higher dimensions. For example, although the 2-sphere is a 2-hit, the groupoid model does not capture its non-trivial structure in dimension 2. Similarly, although we can show that the circle S^1 is interpreted by $\Pi_1(\mathrm{S}^1) \cong \mathbb{Z}$ in the groupoid model, it is trivial in the setoid model.

6.1 Formation rule

Recall the types of the point and path constructors of the schematic 2-hit above:

$$\mathrm{c}_0 : A \to \mathrm{H} \to \mathrm{H}$$
$$\mathrm{c}_1 : (x : B) \to (y : \mathrm{H}) \to p =_{\mathrm{H}} q \to p' =_{\mathrm{H}} q'$$
$$\mathrm{c}_2 : (x : D) \to (y : \mathrm{H}) \to (z : p_3 =_{\mathrm{H}} q_3) \to g_1 =_{p_4 =_{\mathrm{H}} q_4} h_1 \to g_2 =_{p_5 =_{\mathrm{H}} q_5} h_2$$

Let the type A be interpreted by the groupoid (A_0, A_1, A_2), B be interpreted by the groupoid (B_0, B_1, B_2), and D be interpreted by the groupoid (D_0, D_1, D_2). Moreover, p is interpreted as the dependent functor (p_0, p_1, p_2) (cf the groupoid model of terms-in-context, see Hofmann-Streicher or Ruch) consisting of an object, arrow, and preservation of equality part. And similarly for $p', q', p_3, q_3, p_4, q_4, p_5, q_5$. The interpretations of g_1, h_1, g_2, h_2 are also dependent functors.

We interpret the schematic hit H as the inductively generated groupoid $(\mathrm{H}_0, \mathrm{H}_1, \mathrm{H}_2)$ the constructors of which will ensure that the types of the point, path and surface constructors in the theory of 2-hits are interpreted as appropriate dependent functors between groupoids. The key observation is that all the constructors thus obtained have types which are instances of the general form of a type for a constructor of a finitary inductive family [5]. Hence we conjecture that the groupoid model of 2-hits can be developed inside a core extensional type theory extended with a schema for these finitary inductive families.

- H_0 is inductively generated by a constructor for the object part of the point constructor

  $$\mathrm{c}_{00} \in A_0 \to \mathrm{H}_0 \to \mathrm{H}_0$$

  Using c_{00}, a term $p_0(x, y) \in \mathrm{H}_0$, where $x \in B_0$ and $y \in \mathrm{H}_0$, can be defined by induction on the structure of a point constructor pattern p:

  $$\begin{aligned} y_0(x, y) &= y \\ (\mathrm{c}_0(s, t))_0(x, y) &= \mathrm{c}_{00}(s_0(x), t_0(x, y)) \end{aligned}$$

  where s_0 is provided by the hypothesis that s is a term of type A where H does not occur, whereas t_0 is provided by the induction hypothesis.

- H_1 is inductively generated by:
  · a constructor for the object part of the path constructor

  $$\mathrm{c}_{10} \in (x \in B_0) \to (y \in \mathrm{H}_0) \to \mathrm{H}_1(p_0(x, y), q_0(x, y)) \to \mathrm{H}_1(p_0'(x, y), q_0'(x, y))$$

  · a constructor for the arrow part of the point constructor:

  $$\begin{aligned} \mathrm{c}_{01} \in (x, x' \in A_0) &\to A_1(x, x') \to (y, y' \in \mathrm{H}_0) \\ &\to \mathrm{H}_1(y, y') \to \mathrm{H}_1(\mathrm{c}_{00}(x, y), \mathrm{c}_{00}(x', y')) \end{aligned}$$

  · constructors for composition, identity, and inverse of paths

  $$\begin{aligned} \circ &\in (x, y, z \in \mathrm{H}_0) \to \mathrm{H}_1(x, y) \to \mathrm{H}_1(y, z) \to \mathrm{H}_1(x, z) \\ \mathrm{id} &\in (x \in \mathrm{H}_0) \to \mathrm{H}_1(x, x) \\ (-)^{-1} &\in (x, y \in \mathrm{H}_0) \to \mathrm{H}_1(x, y) \to \mathrm{H}_1(y, x) \end{aligned}$$

  For a point constructor pattern p and $x, x' \in B_0$, $y, y' \in \mathrm{H}_0$, $ex \in B_1(x, x')$ and $ey \in \mathrm{H}_1(y, y')$ we define $p_1(ex, ey) \in \mathrm{H}_1(p_0(x, y), p_0(x', y'))$ by

  $$\begin{aligned} y_1(ex, ey) &= ey \\ (\mathrm{c}_0(s, t))_1(ex, ey) &= \mathrm{c}_{01}(s_0(x), s_0(x'), s_1(ex), t_0(x, y), t_0(x', y'), t_1(ex, ey)) \end{aligned}$$

  where s_1 comes from the fact that s is a groupoid map and t_1 from the induction hypothesis.

Similarly for g a path constructor pattern from p to q and $x \in D_0, y \in \mathrm{H}_0, z \in \mathrm{H}_1((p_3)_0(x,y),(q_3)_0(x,y))$, we define $g_0(x,y,z) \in \mathrm{H}_1(p_0(x,y),q_0(x,y))$ by

$$z_0(x,y,z) = z$$
$$\mathrm{c}_1(s,t,g')_0(x,y,z) = \mathrm{c}_{10}(s_0(x),t_0(x,y),g'_0(x,y,z))$$

where s_0 comes from the fact that s is a term, t_0 because t is a point constructor pattern and g'_0 by induction. The equations for identity, composition and inverse are omitted.

- H_2 (representing equality of paths) is inductively generated by
  - the object part of the surface constructor

$$\mathrm{c}_{20} \in (x \in D_0) \to (y \in \mathrm{H}_0) \to (z \in \mathrm{H}_1((p_3)_0,(q_3)_0)) \to \mathrm{H}_2((p_4)_0,(q_4)_0,(g_1)_0,(h_1)_0) \to \mathrm{H}_2((p_5)_0,(q_5)_0,(g_2)_0,(h_2)_0)$$

  - the arrow part of the path constructor:

$$\begin{aligned}\mathrm{c}_{11} \in (x,x' \in B_0) &\to (ex \in B_1(x,x')) \to (y,y' \in \mathrm{H}_0) \to (ey \in \mathrm{H}_1(y,y'))\\ &\to (z \in \mathrm{H}_1(p_0(x,y),q_0(x,y))) \to (z' \in \mathrm{H}_1(p_0(x',y'),q_0(x',y')))\\ &\to \mathrm{H}_2(p_0(x,y),q_0(x',y'),z \circ q_1(ex,ey),p_1(ex,ey) \circ z')\\ &\to \mathrm{H}_2(p'_0(x,y),q'_0(x',y'),\mathrm{c}_{10}(x,y,z) \circ q'_1(ex,ey),p'_1(ex,ey) \circ \mathrm{c}_{10}(x',y',z'))\end{aligned}$$

  - the surface (preservation of equality of arrows) part of the point constructor:

$$\begin{aligned}\mathrm{c}_{02} \in (x,x' \in A_0) &\to (ex,ex' \in A_1(x,x')) \to A_2(x,x',g,g')\\ &\to (y,y' \in \mathrm{H}_0) \to (ey,ey' \in \mathrm{H}_1(y,y')) \to \mathrm{H}_2(y,y',ey,ey')\\ &\to \mathrm{H}_2(\mathrm{c}_{00}(x,y),\mathrm{c}_{00}(x',y'),\mathrm{c}_{01}(x,x',ex,y,y',ey),\mathrm{c}_{01}(x,x',ex',y,y',ey'))\end{aligned}$$

  - the functor laws for the point constructor:

$$\mathrm{c}_0^{\mathrm{id}} \in (x \in A_0) \to (y \in \mathrm{H}_0) \to \mathrm{H}_2(\mathrm{c}_{00}(x,y),\mathrm{c}_{00}(x,y),\mathrm{c}_{01}(x,x,\mathrm{id}_x,y,y,\mathrm{id}_y),\mathrm{id}_{\mathrm{c}_{00}(x,y)})$$

$$\begin{aligned}\mathrm{c}_0^{\circ} \in (x,x',x'' \in A_0) &\to (ex \in A_1(x,x')) \to (ex' \in A_1(x',x''))\\ &\to (y,y',y'' \in \mathrm{H}_0) \to (ey \in \mathrm{H}_1(y,y')) \to (ey' \in \mathrm{H}_1(y',y''))\\ &\to \mathrm{H}_2(\mathrm{c}_{00}(x,y),\mathrm{c}_{00}(x'',y''),\\ &\qquad \mathrm{c}_{01}(x,x',ex,y,y',ey) \circ \mathrm{c}_{01}(x',x'',ex',y',y'',ey'),\\ &\qquad \mathrm{c}_{01}(x,x'',ex \circ ex',y,y'',ey \circ ey'))\end{aligned}$$

  - witnesses that H_2 is a family of equivalence relations:

$$\begin{aligned}\mathrm{tran} \in (x,y : \mathrm{H}_0) &\to (u,v,w : \mathrm{H}_1(x,y))\\ &\to \mathrm{H}_2(u,v) \to \mathrm{H}_2(v,w) \to \mathrm{H}_2(u,w)\\ \mathrm{refl} \in (x,y \in \mathrm{H}_0) &\to (u \in \mathrm{H}_1(x,y)) \to \mathrm{H}_2(u,u)\\ \mathrm{sym} \in (x,y \in \mathrm{H}_0) &\to (u,v \in \mathrm{H}_1(x,y)) \to \mathrm{H}_2(u,v) \to \mathrm{H}_2(v,u)\end{aligned}$$

  - witnesses (whiskerings) that composition preserves equality:

$$\begin{aligned}\mathrm{w}_0 \in (x,y,z \in \mathrm{H}_0) &\to (u,v \in \mathrm{H}_1(x,y))\\ &\to (w \in \mathrm{H}_1(y,z)) \to \mathrm{H}_2(u,v) \to \mathrm{H}_2(u \circ w, v \circ w)\\ \mathrm{w}_1 \in (x,y,z \in \mathrm{H}_0) &\to (u,v \in \mathrm{H}_1(y,z))\\ &\to (w \in \mathrm{H}_1(x,y)) \to \mathrm{H}_2(u,v) \to \mathrm{H}_2(w \circ u, w \circ v)\end{aligned}$$

  - witnesses for the groupoid laws:

$$
\begin{aligned}
A &\in (x_0, x_1, x_2, x_3 \in \mathrm{H}_0) \to (u \in \mathrm{H}_1(x_0, x_1)) \\
&\quad \to (v \in \mathrm{H}_1(x_1, x_2)) \to (w \in \mathrm{H}_1(x_2, x_3)) \\
&\quad \to \mathrm{H}_2((u \circ v) \circ w, u \circ (v \circ w)) \\
\lambda &\in (x, y \in \mathrm{H}_0) \to (u \in \mathrm{H}_1(x, y)) \to \mathrm{H}_2(u, \mathrm{id}_x \circ u) \\
\rho &\in (x, y \in \mathrm{H}_0) \to (u \in \mathrm{H}_1(x, y)) \to \mathrm{H}_2(u, u \circ \mathrm{id}_y) \\
\iota_0 &\in (x, y \in \mathrm{H}_0) \to (u \in \mathrm{H}_1(x, y)) \to \mathrm{H}_2(u \circ u^{-1}, \mathrm{id}_x) \\
\iota_1 &\in (x, y \in \mathrm{H}_0) \to (u \in \mathrm{H}_1(x, y)) \to \mathrm{H}_2(u^{-1} \circ u, \mathrm{id}_y)
\end{aligned}
$$

It follows immediately that $(\mathrm{H}_0, \mathrm{H}_1, \mathrm{H}_2)$ is a groupoid.

6.2 Introduction rules

The point constructor $\mathrm{c}_0 : A \to \mathrm{H} \to \mathrm{H}$ is interpreted by the functor on groupoids with object part c_{00}, arrow part c_{01} and preservation of equality part c_{02}. The functor laws are witnessed by the constructors $\mathrm{c}_0^{\mathrm{id}}$ and c_0°.

A groupoid interpreting $x =_{\mathrm{H}} y$ is a setoid and hence functors on such groupoids degenerate to setoid-maps. Hence, the path constructor

$$\mathrm{c}_1 : (x : B) \to (y : \mathrm{H}) \to p(x, y) =_{\mathrm{H}} q(x, y) \to p'(x, y) =_{\mathrm{H}} q'(x, y)$$

is interpreted by the setoid map with underlying function c_{10} and preservation of equality part c_{11}.

A groupoid interpreting $f =_{x =_{\mathrm{H}} x'} f'$ has only one object and one arrow (up to equality). Hence it suffices to state that the constructor for surfaces c_2 is interpretreted by the constructor c_{20}.

6.3 Elimination and equality rules

To justify the elimination and equality rules we need to construct a dependent groupoid functor $f : (x : \mathrm{H}) \to C(x)$ such that

$$
\begin{aligned}
f(\mathrm{c}_0(x, y)) &= \tilde{\mathrm{c}_0}(x, y, f(y)) \\
\mathbf{apd}_f(\mathrm{c}_1(x, y, z)) &= \tilde{\mathrm{c}_1}(x, y, f(y), z, \mathbf{apd}_f(z)) \\
\mathbf{apd}^2_f(\mathrm{c}_2(x, y, z, w)) &= \tilde{\mathrm{c}_2}(x, y, f(y), z, \mathbf{apd}_f(z), w, \mathbf{apd}^2_f(w))
\end{aligned}
$$

First we define the object part $f_0 : (x \in \mathrm{H}_0) \to C_0(x)$ by H_0-elimination:

$$f_0(\mathrm{c}_{00}(x, y)) = (\tilde{\mathrm{c}_0})_0(x, y, f_0(y)))$$

Then we define the arrow part

$$f_1 \in (x, x' \in \mathrm{H}_0) \to (g \in \mathrm{H}_1(x, x')) \to C'_1(g, f_0(x), f_0(x'))$$

where C'_1 is the heterogeneous equality (between elements of different fibres). This is done by H_1-elimination:

$$
\begin{aligned}
f_1(\mathrm{c}_{10}(x, y, z)) &= (\tilde{\mathrm{c}_1})_0(x, y, f_0(y), z, f_1(p_0(x, y), q_0(x, y), z)) \\
f_1(\mathrm{c}_{01}(x, x', e, y, y', d)) &= (\tilde{\mathrm{c}_0})_1(x, x', e, y, y', d, f_0(y), f_0(y'), f_1(y, y', d))
\end{aligned}
$$

where we have omitted the clauses which say that f_1 maps an identity on H to an identity, a composition to a composition, and an inverse to an inverse.

Finally, we prove by H_2-elimination that f_1 preserves equality of arrows:

$$f_2 : (x, x' \in \mathrm{H}_0) \to (ex, ex' \in \mathrm{H}_1(x, x')) \\ \to (* \in \mathrm{H}_2(x, x', ex, ex')) \to C_2'(*, f_1(x, x', ex), f_1(x, x', ex'))$$

Here C_2' is a heterogeneous notion of equality between heterogeneous paths. (Note that there is only one proof of this notion of equality, so all the equations are into a singleton.)

The calculations required for the verification of the elimination and equality rules in the groupoid model are lengthy, but routine, and are left out because of space restrictions. They are included in the full version of the paper which can be found at http://www.cse.chalmers.se/~peterd/papers/hits.html.

7 Related and further research

Some of the material in this paper (the schema for 1-hits and its setoid model) can be found in the internship report by Moeneclaey [18]. There is also recent work by Basold, Geuvers, and Van der Weide [1]. They formulate a schema for 1-hits and derive elimination rules from the types of the constructors. Their schema is more restricted than ours (for 1-hits) since their path constructors are witnesses of equations, whereas our path constructors can also be witnesses of conditional equations. Moreover, they do not discuss semantics.

The present paper is only a step towards a general theory of hits. There are several possible generalizations, for example, to 3-hits and their interpretation as weak 2-groupoids, to hits with generalized induction, and to hits with a more general class of constructor point and path expressions. In order to get a general theory of hits in cubical type theory [3], a useful step might be to reformulate our theory of 1-hits and 2-hits using degeneracies and restriction maps.

References

[1] Henning Basold, Herman Geuvers, and Niels van der Weide. Higher inductive types in programming. *Journal of Universal Computer Science*, 23(1):63–88, 2017.

[2] Marc Bezem, Thierry Coquand, and Simon Huber. A model of type theory in cubical sets. In *19th International Conference on Types for Proofs and Programs, TYPES 2013, April 22-26, 2013, Toulouse, France*, pages 107–128, 2013.

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

[4] Peter Dybjer. Inductive sets and families in Martin-Löf's type theory and their set-theoretic semantics. In Gerard Huet and Gordon Plotkin, editors, *Logical Frameworks*, pages 280–306. Cambridge University Press, 1991.

[5] Peter Dybjer. Inductive families. *Formal Aspects of Computing*, 6:440–465, 1994.

[6] Peter Dybjer. Internal type theory. In *Types for Proofs and Programs, International Workshop TYPES'95, Torino, Italy, June 5-8, 1995, Selected Papers*, pages 120–134, 1995.

[7] Peter Dybjer. Representing inductively defined sets by wellorderings in Martin-Löf's type theory. *Theor. Comput. Sci.*, 176(1-2):329–335, 1997.

[8] Peter Dybjer. A general formulation of simultaneous inductive-recursive definitions in type theory. *Journal of Symbolic Logic*, 65(2), June 2000.

[9] Peter Dybjer and Anton Setzer. A finite axiomatization of inductive-recursive definitions. In Jean-Yves Girard, editor, *Typed Lambda Calculi and Applications*, volume 1581 of *Lecture Notes in Computer Science*, pages 129–146. Springer, April 1999.

[10] Peter Dybjer and Anton Setzer. Indexed induction-recursion. *J. Log. Algebr. Program.*, 66(1):1–49, 2006.

[11] Martin Hofmann. Elimination of extensionality in Martin-Löf type theory. In *Types for Proofs and Programs, International Workshop TYPES'93, Nijmegen, The Netherlands, May 24-28, 1993, Selected Papers*, pages 166–190, 1993.

[12] Martin Hofmann and Thomas Streicher. The groupoid model refutes uniqueness of identity proofs. In *Proceedings of the Ninth Annual Symposium on Logic in Computer Science (LICS '94), Paris, France, July 4-7, 1994*, pages 208–212, 1994.

[13] Peter LeFanu Lumsdaine and Michael Shulman. Semantics of higher inductive types. 2012.

[14] Per Martin-Löf. An intuitionistic theory of types: Predicative part. In H. E. Rose and J. C. Shepherdson, editors, *Logic Colloquium '73*, pages 73–118. North Holland, 1975.

[15] Per Martin-Löf. Constructive mathematics and computer programming. In *Logic, Methodology and Philosophy of Science, VI, 1979*, pages 153–175. North-Holland, 1982.

[16] Per Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, 1984.

[17] Per Martin-Löf. Amendment to intuitionistic type theory. Notes from a lecture given in Göteborg, March 1986.

[18] Hugo Moeneclaey. Higher inductive types of level 1. http://www.cse.chalmers.se/~peterd/papers/moeneclaey.pdf, August 2016. Internship report, ENS Cachan.

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

[20] Fredrik Nordvall Forsberg and Anton Setzer. Inductive-inductive definitions. In *Computer Science Logic, 24th International Workshop, CSL 2010, 19th Annual Conference of the EACSL, Brno, Czech Republic, August 23-27, 2010. Proceedings*, pages 454–468, 2010.

[21] Christine Paulin-Mohring. Inductive definitions in the system Coq - rules and properties. In *Proceedings Typed λ-Calculus and Applications*, pages 328–345. Springer-Verlag, LNCS, March 1993.

[22] Univalent Foundations Project. *Homotopy Type Theory – Univalent Foundations of Mathematics*. 2013.

[23] Fabian Ruch. The path model of intensional type theory. Master of Science Thesis in Computer Science, Chalmers University of Technology, 2015.

[24] Kristina Sojakova. Higher inductive types as homotopy-initial algebras. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, January 15-17, 2015*, pages 31–42, 2015.

[25] Vladimir Voevodsky. The equivalence axiom and univalent models of type theory. *arXiv 1402.5556*, page 1–11, 2010.