A General Syntax for Nonrecursive Higher Inductive Types*

Marco Girardi[1] Roberto Zunino[1] and Marco Benini[2]

[1] University of Trento, Italy
[2] University of Insubria, Italy

Abstract. Higher Inductive Types are one of the most interesting features of HoTT, as they let us define geometrical objects into the theory. However, unlike inductive types, there is not yet a general schema telling us what exacly an HIT is, or what its corresponding rules in the calculus are. In fact, HITs are often given via "ad hoc" definitions, as in [7, 8]. In this paper we propose a general syntax schema that encapsulates a broad family of nonrecursive HITs. We generalize the concepts of transport and dependent application to higher paths, which we use to give a procedure to extract the elimination rule and the related computation rules.

Keywords: Homotopy Type Theory · Higher Inductive Types.

1 Introduction

A Higher Inductive Type on HoTT is defined by giving constructors not only for points (level 0 constructors), but also for (higher) paths in the type (higher level constructors). Many examples of HITs can be found in the current literature [8], however, given the constructors, there is no general rule telling us what the associated induction principle should be. Indeed one might wonder why the torus in [7] has the described induction principle, and why it is sensible. In general how to obtain the induction principle of a HIT with level 3 constructors is still a craft. The reason why these questions are hard to address at this moment is that we do not have a formal description of what an HIT is and what its corresponding induction principle should be.

HITs have been formalized in Cubical Type Theory (for example [2, 3]), but our goal is to formalize them inside HoTT so to study its proof theory. Some work towards a syntactic formulation of HITs in HoTT has been done, for example [6, 5]; these approaches involve either categorical models of the theory or some other external type theory. It is clear that using these tools brings out of the syntax of HoTT, which prevents the use of the existing knowledge of the related proof theory. In this work we propose, at least in the nonrecursive case, a more direct approach, which is similar to the one used in [4], but extends to higher

level constructors and accounts for propositional computation rules, with the aim of extending the proof techniques used in [1] to HoTT with HITs.

These questions seem to be solved in the case of HITs with only level 0 and level 1 constructors. The idea is that constructors of a HIT T define its structure, and to eliminate we have to recognize the same structure in some fibration $C : T \to \mathcal{U}$. In the case of 0 and 1-HITs, this is done by looking for "points" in the appropriate fibers, and for "paths" in path spaces that "lie over" the appropriate paths in the type T. For example, to perform $\mathbb{S}^1$ induction on $C : \mathbb{S}^1 \to \mathcal{U}$:

- we need a "proof" for $\mathsf{base} : \mathbb{S}^1$, which is a point $c_b : C(\mathsf{base})$.
- we need a "proof" for $\mathsf{loop} : \mathsf{base} =_{\mathbb{S}^1} \mathsf{base}$, which is a path that "lies over" loop whose endpoints are the proofs of base. Following the notation of [8], this is a "pathover" $c_l : c_b =^C_{\mathsf{loop}} c_b$.

Recall from [8] that the type of the pathover above abbreviates $\mathsf{trsp}^C(\mathsf{loop}, c_b) = c_b$. This notion is exactly the counterpart to paths in the fibration C we want to perform induction on. We will generalize this idea to get a reasonable concept of "higher pathover".

Suppose we have a HIT T that has the same constructors as $\mathbb{S}^1$ and an additional level 2 constructor $h : \mathsf{loop} = \mathsf{loop}$. The induction principle would require now an additional "proof" corresponding to h, that is a "2-pathover" between the proof for the starting point of h and the proof of its endpoint. Mimicking the notation for 1-pathovers, we could write this as $c_h : c_l =^{2,C}_h c_l$. This suggests that a 2-pathover space $c_l =^{2,C}_h c_l$ should be an abbreviation for $\mathsf{trsp}^C_2(h, c_l) = c_l$, where trsp^C_2 is some "higher transport along a 2-path" function to be defined. This further suggests that if we were to define a general n-transport function, its "natural" argument, besides the n-path, would be a $(n-1)$-pathover.

Once we generalize pathovers, we illustrate how to extract the induction principle of a HIT from the constructors. The computation rules associated turn out to be slightly trickier, as equalities at higher levels are only propositional. However, there are some canonical equivalences to solve this problem.

2 Preliminary definitions

We will adopt the notation used in [8]. For the rest of the section, let $A : \mathcal{U}$. When dealing with paths at level n we have to give two 0-paths $x, y : A$, two 1-paths in $x = y$, and so on until we get to two $(n-1)$-paths α, β and finally a n-path in $\alpha = \beta$. We will generally omit most of the declarations, and denote by p, q $(n-2)$-paths, by α, β $(n-1)$-paths, and by χ, ξ n-paths.

Let $C : A \to \mathcal{U}$. Recall the definition of $\mathsf{trsp} : \prod_{x,y:A} \prod_{p:x=y} C(x) \to C(y)$ given in [8] of the "transport function along a 1-path". It is given by path induction on p, stating that $\mathsf{trsp}^C(x, x, \mathsf{refl}_x) \equiv \mathsf{id}_C(x)$. When using trsp, in the sequence of arguments we usually only write the path p, leaving the endpoints x, y to be inferred. From this, we define the type of *pathovers* over p with endopints $u : C(x)$ and $v : C(y)$ as $(u =^C_p v) :\equiv (\mathsf{trsp}^C(p, u) = v)$.

We want to generalize these two notions to higher paths. The idea is that level n transport takes as input an n-path $\chi : \alpha = \beta$, an $(n-1)$-pathover that lies over α and outputs a $(n-1)$-pathover lying over β.

Definition 1. *Let $n > 1$. Given an n-path $\chi : \alpha = \beta$ and a $(n-1)$-pathover $c_\alpha : c_p =_\alpha^{n-1,C} c_q$ lying over α, we define the higher transport function by $\mathsf{trsp}_n^C(\chi, c_\alpha) :\equiv \mathsf{trsp}^{\lambda\gamma:(p=q).c_p =_\gamma^{n-1,C} c_q}(\chi, c_\alpha)$.*

Notice again that we left most of the sequence of arguments implicit, as it can be inferred by the last terms. Similarly:

Definition 2. *For $n \geq 1$, we define the type of n-*pathovers over $\chi : \alpha = \beta$ with endpoints $c_\alpha : c_p =_\alpha^{n-1,C} c_q$ and $c_\beta : c_p =_\beta^{n-1,C} c_q$ *as the following abbreviation:* $(c_\alpha =_\chi^{n,C} c_\beta) :\equiv (\mathsf{trsp}_n^C(\chi, c_\alpha) = c_\beta)$.

This is a definition by mutual induction on $n : \mathbb{N}$: in fact n-pathovers are defined in terms of trsp_n^C, and trsp_n^C is defined in terms of $(n-1)$-pathovers. To have a well founded definition, we state that trsp_1^C is the usual trsp^C and the type of 0-pathovers lying over $x : A$ is simply $C(x)$.

It is possible to define (by path induction) pathover operations corresponding to the usual operations on paths. If χ, ξ are n-paths that can be concatenated, $g : u =_\chi^{n,C} v$ and $h : v =_\xi^{n,C} w$ are n-pathovers over χ and ξ respectively, then we can concatenate them to get a pathover $g * h : u =_{\chi \blacksquare \xi}^{n,C} w$. Similarly we can invert g to get a pathover $g^{-\mathbb{1}} : v =_{g^{-1}}^{n,C} u$. We use different symbols to distinguish these operations from the usual on paths.

3 Rules for a general nonrecursive higher inductive type

In this section we describe the inference rules to define a general nonrecursive HIT T. The rules will mimic the style of [1], so most of the rules will be introduce a constant in the calculus, so that they are easier to handle when tackling proof theoretic results.

Formation rule The the type T we are defining may have formation parameters of type $F_1, \ldots, F_w$, to allow for parametric types like suspensions (section 6.5 of [8]). The formation rule for a HIT T introduces a constant $T : \prod_{(f:F)_w} \mathcal{U}_i$, were $\prod_{(f:F)_w}$ abbreviates $\prod_{(f_1:F_1)} \prod_{(f_2:F_2)} \cdots \prod_{(f_w:F_w)}$. This sequence can possibly be empty. Similar abbreviations will be used in the rest of the paper.

Introduction rules Introduction rules are given by a list of constructors. We allow each constructor to only refer to previous ones. If the i-th constructor $\mathbb{K}_i$ is a level n constructor, its introduction rule is a rule introducing the constant $\mathbb{K}_i : \prod_{(f:F)'_w} \prod_{(x:R)_l} H_i$, where H_i is some n-path space in $T\bar{f}'$ (defined below) and each R_k is a type not involving T in any way.

$(f : F)'_w$ may be a subsequence of $(f : F)_w$, as in the constructor refl for the type $=$. Also, $\bar{f}'$ is a sequence of formation parameters taken from $(f : F)'_w$ (hence, with possibly duplicates).

If $n \geq 1$, an n-path space in $T\bar{f}'$ is a type of the form

$$\mathbb{P}_{i_1}^{\epsilon_{i_1}} \blacksquare \cdots \blacksquare \mathbb{P}_{i_a}^{\epsilon_{i_a}} = \mathbb{P}_{j_1}^{\epsilon_{j_1}} \blacksquare \cdots \blacksquare \mathbb{P}_{j_b}^{\epsilon_{j_b}}$$

where each of the $\mathbb{P}_w$ is $\mathbb{K}_h(\bar{t}_h)$, with $\mathbb{K}_h$ some previous level $(n-1)$ constructor. ϵ_w can be (-1) denoting path inversion, or nothing. If n is 1, then we have no concatenation operator for 0-paths (i.e. points), hence, both a and b are equal to 1. If a (resp b) is 0, then the path on the left- (resp. right-)handside is an appropriate reflexivity path. If $n = 0$ the only path space is $T\bar{f}'$ istelf. In other words, only reflexivities and suitably instantiated previous constructors can appear in the output type of a constructor.

Elimination rule The elimination rule states the existence of an inductor (again via a constant introduction rule). The idea is that the inductor recieves as input the "proofs" (*induction methods*) of some property $C : \prod_{(f:F)_w} T\bar{f} \to \mathcal{U}$ on the constructors of T. More precisely, for each constructor $\mathbb{K}_i : \prod_{(f:F)'_w} \prod_{(x:R)_l} H_i$ we have to provide a term $c_i : \prod_{(f:F)'_w} \prod_{(x:R)_l} \mathcal{H}_i$ where $\mathcal{H}_i$ is the *lifted path space* of $(\mathbb{K}_i(\hat{f}', \bar{x}) : H_i)$, defined as:

- if H_i is a level $n \geq 1$ path space $\mathbb{P}_{i_1}^{\epsilon_{i_1}} \blacksquare \cdots \blacksquare \mathbb{P}_{i_a}^{\epsilon_{i_a}} = \mathbb{P}_{j_1}^{\epsilon_{j_1}} \blacksquare \cdots \blacksquare \mathbb{P}_{j_b}^{\epsilon_{j_b}}$, then $\mathcal{H}_i$ is $\mathcal{P}_{i_1}^{\varepsilon_{i_1}} * \cdots * \mathcal{P}_{i_a}^{\varepsilon_{i_a}} =^{n, C\bar{f}'}_{\mathbb{K}_i(\hat{f}'\bar{x})} \mathcal{P}_{j_1}^{\varepsilon_{j_1}} * \cdots * \mathcal{P}_{j_b}^{\varepsilon_{j_b}}$, where if $\mathbb{P}_w$ is $\mathbb{K}_h(\bar{t}_h)$, then $\mathcal{P}_w$ is $c_h(\bar{t}_h)$. In other words, constructors are replaced by their induction methods. Moreover, ε is $-\mathbb{1}$ whenever ϵ is -1. If either the LHS or the RHS in H are refl_r for some $(n-1)$-path r, then this gets lifted to $\mathsf{refl}_{r'}$, where r' is r with the constructors replaced by their induction methods again.
- if H_i is a level 0 path space (i.e. $\mathbb{K}_i$ is a point constructor), then $\mathcal{H}_i$ is $C(\bar{f}', \mathbb{K}_i(\hat{f}'\bar{x}))$

Above, $\hat{f}'$ denotes the sequence of formation parameters of $\mathbb{K}_i$. Note that the induction method for $\mathbb{K}_i$ can depend on any of the preceding induction methods. In conclusion, the elimination rule introduces the following constant in the calculus:

$$\mathsf{ind}_T : \prod_{(f:F)_w} \prod_{(C:\prod_{(f:F)_w} T\bar{f} \to \mathcal{U})} \prod_{(c_i:\prod_{(f:F)'_w} \prod_{(x:R)_l} \mathcal{H}_i)_{i=1}^{\kappa}} \prod_{(x:T\bar{f})} C(\bar{f}, x)$$

Computation rule It is possible to generalize the notion of apd in [8] to higher paths: if $\chi : \alpha = \beta$ is an n-path and $g : \prod_{x:A} C(x)$ is a dependent map, then $\mathsf{apd}_g^n(\chi) : \mathsf{apd}_g^{n-1}(\alpha) =^{n,C}_{\chi} \mathsf{apd}_g^{n-1}(\beta)$ is defined by path induction on χ.

For the rest of the section let $g :\equiv \mathsf{ind}_T(\bar{f}, C, c_1, \ldots, c_\kappa) : \prod_{x:T\bar{f}} C(\bar{f}, x)$. Let us now consider constructor $\mathbb{K}_i : \prod_{(f:F)'_w} \prod_{(x:R)_l} H_i$ at level n.

– if $n = 0$ we get the usual judgmental computation rule, as in [1].
– if $n = 1$ we have a rule introducing a constant

$$\mathbb{C}_i : \prod_{(f:F)'_w} \prod_{(x:R)_l} \mathsf{apd}^1_g(\mathbb{K}_i(\hat{f}', \bar{x})) = c_i(\hat{f}', \bar{x})$$

– if $n > 1$ we would like to give the same rule, however the equality does not typecheck: in fact, if H_i is $\mathbb{P}^{\epsilon_{i_1}}_{i_1} \blacksquare \cdots \blacksquare \mathbb{P}^{\epsilon_{i_a}}_{i_a} = \mathbb{P}^{\epsilon_{j_1}}_{j_1} \blacksquare \cdots \blacksquare \mathbb{P}^{\epsilon_{j_b}}_{j_b}$ we have

$$\mathsf{apd}^n_g(\mathbb{K}_i(\bar{f}', \bar{x})) : \ \mathsf{apd}^{n-1}_g(\mathbb{P}^{\epsilon_{i_1}}_{i_1} \blacksquare \cdots \blacksquare \mathbb{P}^{\epsilon_{i_a}}_{i_a}) =^{n,C\bar{f}'}_{\mathbb{K}_i(\hat{f}',\bar{x})} \mathsf{apd}^{n-1}_g(\mathbb{P}^{\epsilon_{j_1}}_{j_1} \blacksquare \cdots \blacksquare \mathbb{P}^{\epsilon_{j_b}}_{j_b})$$
$$c_i(\bar{f}', \bar{x}) : \ \mathcal{P}^{\varepsilon_{i_1}}_{i_1} * \cdots * \mathcal{P}^{\varepsilon_{i_a}}_{i_a} =^{n,C\bar{f}'}_{\mathbb{K}_i(\hat{f}',\bar{x})} \mathcal{P}^{\varepsilon_{j_1}}_{j_1} * \cdots * \mathcal{P}^{\varepsilon_{j_b}}_{j_b}$$

These types are not judgmentally equal. However, thanks to the computation rules of the previous constructors, it is possible to prove:

Theorem 1. *In the setting above, we have*

$$\Phi_{\mathbb{K}_i(\hat{f}',\bar{x})} : \left(\mathsf{apd}^{n-1}_g(\mathbb{P}^{\epsilon_{i_1}}_{i_1} \blacksquare \cdots \blacksquare \mathbb{P}^{\epsilon_{i_a}}_{i_a}) =^{n,C\bar{f}'}_{\mathbb{K}_i(\hat{f}',\bar{x})} \mathsf{apd}^{n-1}_g(\mathbb{P}^{\epsilon_{j_1}}_{j_1} \blacksquare \cdots \blacksquare \mathbb{P}^{\epsilon_{j_b}}_{j_b})\right) \simeq$$
$$\left(\mathcal{P}^{\varepsilon_{i_1}}_{i_1} * \cdots * \mathcal{P}^{\varepsilon_{i_a}}_{i_a} =^{n,C\bar{f}'}_{\mathbb{K}_i(\hat{f}',\bar{x})} \mathcal{P}^{\varepsilon_{j_1}}_{j_1} * \cdots * \mathcal{P}^{\varepsilon_{j_b}}_{j_b}\right)$$

Moreover, this equivalence does not depend on the particular path $\mathbb{K}_i(\hat{f}', \bar{x})$.

Thanks to this equivalence, we can now write the computation rule for constructors of level greater than 1 as a rule introducing the constant

$$\mathbb{C}_i : \prod_{(f:F)'_w} \prod_{(x:R)_l} \Phi_{\mathbb{K}_i(\hat{f}',\bar{x})}\left(\mathsf{apd}^n_g(\mathbb{K}_i(\hat{f}', \bar{x}))\right) = c_i(\hat{f}', \bar{x})$$

4 An example: the torus

Consider the torus T^2 as defined in [8], which is a HIT without any formation parameter with the following constructors:

– A 0-constructor $b : T^2$
– Two 1-constructors: $(p : b = b)$ and $(q : b = b)$
– A 2-constructor $(\mathsf{surf} : p \blacksquare q = q \blacksquare p)$

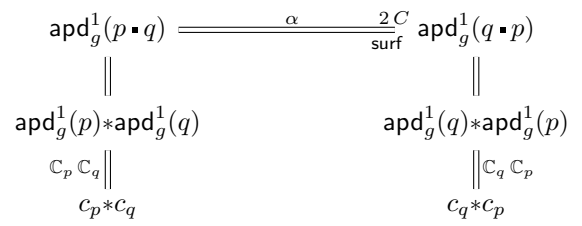
Given $C : T^2 \to \mathcal{U}$, the corresponding induction methods for the eliminator are:

– A 0-pathover $c_b : C(b)$
– Two 1-pathovers: $(c_p : c_b =^{1,C}_p c_b)$, and $(c_q : c_b =^{1,C}_q c_b)$
– A 2-pathover $(c_{\mathsf{surf}} : c_p * c_q =^{2,C}_{\mathsf{surf}} c_q * c_p)$

In practice, what happens is that the paths given by the constructors are replaced by their induction method in the eliminator. Note how this elimination principle better reflects the intuition of recognizing the structure of T^2 in the fibration C when compared to the one used in [7]. This is because the latter one states explicitly every transport involved, while our schema hides them behind the scene. It can be proven that in fact the two induction principles are equivalent.

As for the computation rules, let $g :\equiv \mathsf{ind}_{T^2}(C, c_b, c_p, c_q, c_{\mathsf{surf}}) : \prod_{x:T^2} C(x)$.

- At level 0 we have as usual the judgmental equality $(g(b) \equiv c_b)$.
- At level 1 we have two propositional equalities: $(\mathbb{C}_p : \mathsf{apd}^1_g(p) = c_p)$ and $(\mathbb{C}_q : \mathsf{apd}^1_g(q) = c_q)$.
- At level 2 let $\Phi_{\mathsf{surf}} : \Big(\big(\mathsf{apd}^1_g(p \blacksquare q) =^{2,C}_{\mathsf{surf}} \mathsf{apd}^1_g(q \blacksquare p)\big) \simeq \big(c_p * c_q =^{2,C}_{\mathsf{surf}} c_q * c_p\big)\Big)$ be the equivalence given by theorem 1, which maps α to the following path:



There is an hidden application of $\mathsf{trsp}^C_2(\mathsf{surf})$ on the paths on the left to make everything typecheck. So the computation rule for surf states that there is a propositional equality $\big(\mathbb{C}_{\mathsf{surf}} : \Phi_{\mathsf{surf}}(\mathsf{apd}^2_g(\mathsf{surf})) = c_{\mathsf{surf}}\big)$, that amounts to the commutativity of a square as in [7].

5 Conclusion

In this paper we have proposed a syntax for a broad class of HITs. This allows us in the first place to tackle some proof theoretic questions about HoTT with HITs: for example, it should be easy to prove normalization of the calculus (meaning that every reduction sequence of a judgment terminates) by extending the proof in [1]. Moreover it is also possible to use the elimination principle proposed here to prove in the theory some properties we would expect, such as the contractibility of the k-dimensional disk.

Moreover, it seems possible to extend the eliminator to HITs with recursive constructors, so to include e.g. truncations, which we are currently working on. Dealing with computation rules is harder in the recursive case.

References

1. Bonacina, R.: Semantics for Homotopy Type Theory. Ph.D. thesis, Università degli Studi dell'Insubria (2019)
2. Cavallo, E., Harper, R.: Higher inductive types in cubical computational type theory. Proceedings of the ACM on Programming Languages **3**(POPL), 1–27 (2019)
3. Coquand, T., Huber, S., Mörtberg, A.: On higher inductive types in cubical type theory. In: Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science. pp. 255–264 (2018)
4. Dybjer, P., Moeneclaey, H.: Finitary higher inductive types in the groupoid model. Electronic Notes in Theoretical Computer Science **336**, 119–134 (2018)
5. Kaposi, A., Kovács, A.: Signatures and induction principles for higher inductive-inductive types. arXiv preprint arXiv:1902.00297 (2019)
6. Lumsdaine, P.L., Shulman, M.: Semantics of higher inductive types. In: Mathematical Proceedings of the Cambridge Philosophical Society. pp. 1–50. Cambridge University Press (2019)
7. Sojakova, K.: The equivalence of the torus and the product of two circles in homotopy type theory. ACM Transactions on Computational Logic **17**(4), 1–19 (2016)
8. The Univalent Foundations Program: Homotopy Type Theory: Univalent Foundations of Mathematics. https://homotopytypetheory.org/book, Institute for Advanced Study (2013)