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

A Higher Inductive Type in 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 [6], however, given the constructors, there is no general rule generating the associated induction principle. Indeed one might wonder why the torus in [5] has the described induction principle, and why it is sensible. Some work towards a syntactic formulation of HITs has been done, for example [3, 4]; 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 [2], 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.

The case of HITs with only level 0 and level 1 constructors seems to be rather clear. The idea is that constructors of a HIT T define its structure, and to perform elimination we have to recognise 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 induction on $\mathbb{S}^1$ using $C : \mathbb{S}^1 \to \mathcal{U}$ we need a "proof" for $\mathsf{base} : \mathbb{S}^1$, which is a point $c_b : C(\mathsf{base})$, and a "proof" for $\mathsf{loop} : \mathsf{base} =_{\mathbb{S}^1} \mathsf{base}$, which is a "pathover" $c_l : c_b =^{C}_{\mathsf{loop}} c_b$.

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.

In general, if $\chi : \alpha = \beta$ is an n-path, $\mathsf{trsp}^C_n(\chi)$ maps an $(n-1)$-pathover over α to a pathover over β. n-pathovers are then defined in the natural way, as well as pathovers concatenation and inversion. With these ingredients we are able to describe the inference rules to define a general nonrecursive HIT T. Following the style in [1] most rules are introductions of constants.

Defining higher transport and pathover We give now the definition of transport along a level n path. Let $A : \mathcal{U}$ be a type and let $C : A \to \mathcal{U}$. From now on we will generally use p, q to refer to $(n-2)$-paths in A, α, β to refer to $(n-1)$-paths and χ, ξ to refer to n-paths. Moreover, trsp denotes the usual transport function defined in [6].

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* $\mathsf{trsp}_n^C(\chi, c_\alpha) :\equiv \mathsf{trsp}^{\lambda\gamma:(p=q).c_p =_\gamma^{n-1,C} c_q}(\chi, c_\alpha)$.

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 pathover operations correspondig 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 $g^{-1} : v =_{g^{-1}}^{n,C} u$. We use different symbols to distinguish these operations from the usual on paths.

Formation rule The type T we are defining may have formation parameters of type $F_1, \ldots, F_w$, to allow parametric types like suspensions (section 6.5 of [6]). The formation rule for a HIT T introduces a constant $T : \prod_{(f:F)_w} \mathcal{U}_i$, where $\prod_{(f:F)_w}$ abbreviates $\prod_{(f_1:F_1)} \prod_{(f_2:F_2)} \cdots \prod_{(f_w:F_w)}$. For the sake of simplicity in the rest of this abstract we assume T has no formation parameters.

Introduction rule 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 introduces the constant $\mathbb{K}_i : \prod_{(x:R)_l} H_i$, where H_i is some n-path space in T (defined below) and each R_k is a type not involving T in any way.

If $n \geq 1$, an n-path space in T 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 some previous level $(n-1)$ constructor $\mathbb{K}_h$ (with $h < i$) instantiated to a proper sequence of terms $\bar{t}_h$. ϵ_w can be (-1) denoting path inversion, or nothing. If a or b are zero, then we have a suitable refl path. If $n = 0$ the only path space is T istelf.

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" (which we will call *inductive clauses*) of some property $C : T \to \mathcal{U}$ on the constructors of T. More precisely, for each constructor $\mathbb{K}_i : \prod_{(x:R)_l} H_i$ we have to provide a term $c_i : \prod_{(x:R)_l} \mathcal{H}_i$ where $\mathcal{H}_i$ is the *lifted path space* of $(\mathbb{K}_i(\bar{x}) : H_i)$, defined as:

- if H_i is a level $n \geq 1$ path space $\mathbb{P}^{\epsilon_{i_1}}_{i_1} \blacksquare \cdots \blacksquare \mathbb{P}^{\epsilon_{i_a}}_{i_a} = \mathbb{P}^{\epsilon_{j_1}}_{j_1} \blacksquare \cdots \blacksquare \mathbb{P}^{\epsilon_{j_b}}_{j_b}$, then $\mathcal{H}_i$ is $\mathcal{P}^{\varepsilon_{i_1}}_{i_1} * \cdots * \mathcal{P}^{\varepsilon_{i_a}}_{i_a} =^{n,C}_{\mathbb{K}_i(\bar{x})} \mathcal{P}^{\varepsilon_{j_1}}_{j_1} * \cdots * \mathcal{P}^{\varepsilon_{j_b}}_{j_b}$, where if $\mathbb{P}_w$ is $\mathbb{K}_h(\bar{t}_h)$, then $\mathcal{P}_w$ is $c_h(\bar{t}_h)$. 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 inductive clauses again. Note that the lifted path space is a pathover over $\mathbb{K}_i(\bar{x})$. Moreover $*$ e $-\mathbb{1}$ denote concatenation and inversion of pathovers.
- if H_i is a level 0 path space, then $\mathcal{H}_i$ is $C(\mathbb{K}_i(\bar{x}))$

Computation rule It is possible to generalize the notion of apd in [6] to higher paths: if $\chi : \alpha = \beta$ is an n-path and $g : \prod_{x:A} C(x)$ is a dependent map, then $\mathsf{apd}^n_g(\chi) : \mathsf{apd}^{n-1}_g(\alpha) =^{n,C}_{\chi} \mathsf{apd}^{n-1}_g(\beta)$.

For the rest of the section let $g :\equiv \mathsf{ind}_T(C, c_1, \ldots, c_\kappa) : \prod_{x:T} C(x)$. Let us now consider constructor $\mathbb{K}_i : \prod_{(x:R)_l} H_i$ 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 $\mathbb{C}_i : \prod_{(x:R)_l} \mathsf{apd}^1_g(\mathbb{K}_i(\bar{x})) = c_i(\bar{x})$ If $n > 1$ we would like to give the same rule, but there is a type mismatch: in fact, if H 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

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

However, these types can be proved equivalent exploiting the computation rules of the previous constructors. Let $\Phi_{\mathbb{K}_i(\bar{x})}$ be this equivalence. Then we can introduce a constant

$$\mathbb{C}_i : \prod_{(x:R)_l} \Phi_{\mathbb{K}_i(\bar{x})}\Big(\mathsf{apd}^n_g(\mathbb{K}_i(\bar{x}))\Big) = c_i(\bar{x})$$

Conclusion In this extended abstract 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 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, as the contractibility of the k-dimensional disk.

Moreover, it seems possible to extend this approach to HITs with recursive constructors, which we leave for future work. 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. Dybjer, P., Moeneclaey, H.: Finitary higher inductive types in the groupoid model. Electronic Notes in Theoretical Computer Science **336**, 119–134 (2018)
3. Kaposi, A., Kovács, A.: Signatures and induction principles for higher inductive-inductive types. arXiv preprint arXiv:1902.00297 (2019)
4. 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)
5. 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)
6. The Univalent Foundations Program: Homotopy Type Theory: Univalent Foundations of Mathematics. https://homotopytypetheory.org/book, Institute for Advanced Study (2013)

Appendix A Additional definitions

In this section we provide more details on the notions that were sketched in the extended abstract.

Definition 3 (Higher transport). *Let $n > 1$. Let $A : \mathcal{U}$ and $C : A \to \mathcal{U}$. We define*

$$\mathsf{trsp}^C_n : \prod_{(x,y:A)} \cdots \prod_{(p,q:\_\_)} \prod_{(\alpha,\beta:p=q)} \prod_{(\chi:\alpha=\beta)} \prod_{(c_x:C(x))} \prod_{(c_y:C(y))} \cdots \prod_{(c_p:\_\_)} \prod_{(c_q:\_\_)} (c_\alpha : c_p =^{n-1,C}_{\alpha} c_q) \to (c_p =^{n-1,C}_{\beta} c_q)$$

by $\mathsf{trsp}^C_n(\chi, c_\alpha) :\equiv \mathsf{trsp}^{\lambda\gamma:(p=q).c_p =^{n-1,C}_{\gamma} c_q}(\chi, c_\alpha)$.

The first set of arguments passed to the transport funcion is the sequence of paths (from level 0 to n), then the second set of arguments is the sequence of the associated pathovers (from level 0 to $n-1$). Note that in our notation we omit the first part of each sequence, as it can be inferred by the last term.

Operations on pathovers can be defined as follows:

Lemma 1 (Concatenation of pathovers). *Let $\alpha, \beta, \gamma : p = q$ be $(n-1)$-paths in A. Let $\chi : \alpha = \beta$ and $\xi : \beta = \gamma$ be n-paths in A. Let $u : c_p =^{n-1,C}_{\alpha} c_q$, $v : c_p =^{n-1,C}_{\beta} c_q$ and $w : c_p =^{n-1,C}_{\gamma} c_q$ be $(n-1)$-pathovers (so c_p and c_q are $(n-2)$-pathovers of an appropriate type).*

Finally let $g : u =^{n,C}_{\chi} v$ and $h : v =^{n,C}_{\xi} w$. Then there is a pathover

$$g{*}h : u =^{n,C}_{\chi \blacksquare \xi} w$$

Proof. By path induction on χ and ξ. In that case, $*$ reduces to the standard $\blacksquare$.

Notice that $*$ is not associative (but only associative up to some equivalence). We assume it associates on the right, although it does not make any difference in this paper.

Lemma 2 (Inversion of pathovers). *Let $\alpha, \beta : p = q$ be $(n-1)$-paths in A. Let $\chi : \alpha = \beta$ be an n-path in A. Let $u : c_p =^{n-1,C}_{\alpha} c_q$, $v : c_p =^{n-1,C}_{\beta} c_q$.*

Let $g : u =^{n,C}_{\chi} v$ be an n-pathover over χ. Then there is a pathover

$$g^{-\mathbb{1}} : (v =^{n,C}_{\chi^{-1}} u)$$

Proof. By path induction on χ. In that case, $(-)^{-\mathbb{1}}$ reduces to $(-)^{-1}$.

Finally, higher dependent application is defined as follows:

Definition 4 (Higher apd). *Let $A : \mathcal{U}$, $C : A \to \mathcal{U}$ and let $g : \prod_{x:A} C(x)$. Then we have*

$$\mathsf{apd}_g^n : \prod_{(x,y:A)} \cdots \prod_{(p,q:\_\_)} \prod_{(\alpha,\beta:p=q)} \prod_{(\chi:\alpha=\beta)} \mathsf{apd}_g^{n-1}(\alpha) =_\chi^{n,C} \mathsf{apd}_g^{n-1}(\beta)$$

Defined by path induction on χ as $\mathsf{apd}_g^n(\mathsf{refl}_\alpha) :\equiv \mathsf{refl}_{\mathsf{apd}_g^{n-1}(\alpha)}$. For $n = 0$ we say that apd_g^0 is exactly g.

Note that for $n = 1$ this definition becomes exactly the one given in [6]. Again, we just write $\mathsf{apd}_g^n(\chi)$ omitting most of the sequence of paths.

Appendix B Rules of HoTT with nonrecursive HITs

We provide here the rules describing nonrecursive HITs in HoTT. Here we also let the HIT T we are defining to have formation parameters.

The formation rule for a HIT T introduces T as a constant:

$$\frac{\Gamma \vdash \prod_{(f:F)_w} \mathcal{U}_i : \mathcal{U}_{i+1}}{\Gamma \vdash T : \prod_{(f:F)_w} \mathcal{U}_i}\, T\text{−form}$$

The introduction rule for the i-th constructor of T is:

$$\frac{\Gamma \vdash \prod_{(f:F)'_w} \prod_{(x:R)_l} H_i : \mathcal{U}_i}{\Gamma \vdash \mathbb{K}_i : \prod_{(f:F)'_w} \prod_{(x:R)_l} H_i}\, T\text{−intro}_i$$

where H_i is an n-path space, as stated in the extended abstract.

The elimination rule is:

$$\frac{\begin{array}{l}\Gamma \vdash \prod\limits_{(f:F)_w} \\ \quad \prod C : \prod\limits_{(f:F)_w} T\bar{f} \to \mathcal{U} \\ \quad \left(\prod c_i : \prod\limits_{(f:F)'_w} \prod\limits_{(x:R)_l} \mathcal{H}_i\right)_{i=1}^{\kappa} \\ \quad \prod\limits_{x:T\bar{f}} C(\bar{f}, x) \qquad\qquad : \mathcal{U}_j\end{array}}{\begin{array}{l}\Gamma \vdash \mathsf{ind}_T : \prod\limits_{(f:F)_w} \\ \quad \prod C : \prod\limits_{(f:F)_w} T\bar{f} \to \mathcal{U} \\ \quad \left(\prod c_i : \prod\limits_{(f:F)'_w} \prod\limits_{(x:R)_l} \mathcal{H}_i\right)_{i=1}^{\kappa} \\ \quad \prod\limits_{x:T\bar{f}} C(\bar{f}, x)\end{array}} T-\mathsf{elim}$$

where κ is the number of constructors of T, and $\mathcal{H}_i$ is the lifted path space of $\mathbb{K}_i(\hat{f}', \bar{x})$ defined as in the extended abstract (although with some minor technicalities regarding the sequence of formation parameters).

Finally, the computation rules are the following:

– If $\mathbb{K}_i$ is a level 0 constructor we use the same rule as in [1]. As in the extended abstract, let $g :\equiv \mathsf{ind}_T(\bar{f}, C, c_1, \ldots, c_\kappa) : \prod_{x:T\bar{f}} C(\bar{f}, x)$.

$$\frac{\begin{array}{c}\Gamma \vdash C : \prod\limits_{(f:F)_w} T\bar{f} \to \mathcal{U} \\ \left(\Gamma \vdash c_i : \prod\limits_{(f:F)'_w} \prod\limits_{(x:R)_l} \mathcal{H}_i\right)_{i=1}^{\kappa} \\ \Gamma \vdash \mathbb{K}_i(\hat{f}', \bar{x}) : T\bar{f}'\end{array}}{\Gamma \vdash g(\mathbb{K}_i(\hat{f}', \bar{x})) \equiv c_i(\hat{f}', \bar{x}) : C(\bar{f}', \mathbb{K}_i(\hat{f}', \bar{x}))} T-\mathsf{comp}_i^0$$

– If $\mathbb{K}_i$ is a level 1 constructor:

$$\frac{\Gamma \vdash \prod\limits_{(f:F)'_w} \prod\limits_{(x:R)_l} \mathsf{apd}_g^1(\mathbb{K}_i(\hat{f}', \bar{x})) = c_i(\hat{f}', \bar{x}) : \mathcal{U}_j}{\Gamma \vdash \mathbb{C}_i : \prod\limits_{(f:F)'_w} \prod\limits_{(x:R)_l} \mathsf{apd}_g^1(\mathbb{K}_i(\hat{f}', \bar{x})) = c_i(\hat{f}', \bar{x})} T-\mathsf{comp}_i^1$$

– If $\mathbb{K}_i$ is a level n constructor for $n > 1$:

$$\frac{\Gamma \vdash \prod_{(f:F)'_w} \prod_{(x:R)_l} \Phi_{\mathbb{K}_i(\hat{f}',\bar{x})}\Big(\mathsf{apd}_g^n(\mathbb{K}_i(\hat{f}',\bar{x}))\Big) = c_i(\hat{f}',\bar{x}) : \mathcal{U}_j}{\Gamma \vdash \mathbb{C}_i : \prod_{(f:F)'_w} \prod_{(x:R)_l} \Phi_{\mathbb{K}_i(\hat{f}',\bar{x})}\Big(\mathsf{apd}_g^n(\mathbb{K}_i(\hat{f}',\bar{x}))\Big) = c_i(\hat{f}',\bar{x})}\ T-\mathsf{comp}_i^n$$

where $\Phi_{\mathbb{K}_i(\hat{f}',\bar{x})}$ is the equivalence between the type of $\mathsf{apd}_g^n(\mathbb{K}_i(\hat{f}',\bar{x}))$ and the type of $c_i(\hat{f}',\bar{x})$ hinted in the extended abstract. Proving this equivalence relies heavily on the computation rules for the constructors preceding $\mathbb{K}_i$.

Notice that all the rules but the level-0 computation are introductions of constants in the calculus. This is done following the style in [1].

Appendix C An example

Consider the HIT T without any formation parameter which has the following constructors:

– Three 0-constructors $N, S, P : T$
– Three 1-constructors: $(W : N = S)$, $(E_1 : N = P)$, $(E_2 : P = S)$
– A 2-constructor $(\mathsf{fill} : W = E_1 \blacksquare E_2)$

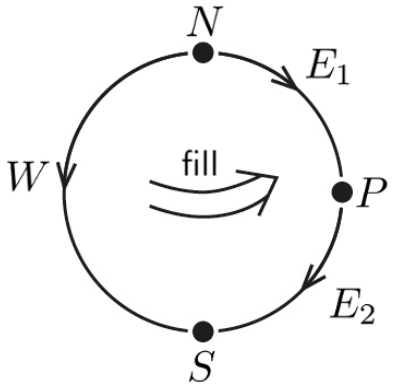

Fig. 1.

Intuitively this is the HIT given in fig. 1, that is a variant of the 2-disk.
Given $C : T \to \mathcal{U}$, the corresponding inductive clauses for the eliminator are:

– Three 0-pathovers $c_N : C(N),\ c_S : C(S),\ c_P : C(P)$

- Three 1-pathovers: $(c_W : c_N =^{1,C}_{W} c_S)$, $(c_{E_1} : c_N =^{1,C}_{E_1} c_P)$, $(c_{E_2} : c_P =^{1,C}_{E_2} c_S)$
- A 2-pathover $(c_{\mathsf{fill}} : c_W =^{2,C}_{\mathsf{fill}} c_{E_1} * c_{E_2})$

In practice, what happens is that the paths given by the constructors are replaced by their inductive clauses in the eliminator.

As for the computation rules, let $g :\equiv \mathsf{ind}_T(C, c_N, c_S, c_P, c_W, c_{E_1}, c_{E_2}, c_{\mathsf{fill}})$:

- At level 0 we have the usual $(g(N) \equiv c_N)$, $(g(S) \equiv c_S)$ and $(g(P) \equiv c_P)$.
- At level 1 we have three propositional equalities: $(\mathbb{C}_W : \mathsf{apd}^1_g(W) = c_W)$, $(\mathbb{C}_{E_1} : \mathsf{apd}^1_g(E_1) = c_{E_1})$ and $(\mathbb{C}_{E_2} : \mathsf{apd}^1_g(E_2) = c_{E_2})$.
- At level 2 let $\Phi_{\mathsf{fill}} : \Big(\big(\mathsf{apd}^1_g(W) =^{2,C}_{\mathsf{fill}} \mathsf{apd}^1_g(E_1 \blacksquare E_2)\big) \simeq \big(c_W =^{2,C}_{\mathsf{fill}} c_{E_1} * c_{E_2}\big)\Big)$, be the map that given $\gamma : \mathsf{apd}^1_g(W) =^{2,C}_{\mathsf{fill}} \mathsf{apd}^1_g(E_1 \blacksquare E_2)$ performs the following concatenation of paths:

  $$c_W \overset{\mathbb{C}_W^{-1}}{=\!=} \mathsf{apd}^1_g(W) \overset{\gamma}{=\!=}^{2,C}_{\mathsf{fill}} \mathsf{apd}^1_g(E_1 \blacksquare E_2) = \mathsf{apd}^1_g(E_1) * \mathsf{apd}^1_g(E_2) \overset{\mathbb{C}_{E_1}, \mathbb{C}_{E_2}}{=\!=\!=} c_{E_1} * c_{E_2}$$

  The path on the left is actually $\mathsf{ap}_{\mathsf{trsp}^C_2(\mathsf{fill})}(\mathbb{C}_W^{-1})$ in order to typecheck, but it is convenient to omit that in the notation: whenever a path is composed on the left of a pathover, there is a suitable implicit application of trsp.
  So the computation rule for fill states that there is a propositional equality $\big(\mathbb{C}_{\mathsf{fill}} : \Phi_{\mathsf{fill}}(\mathsf{apd}^2_g(\mathsf{fill})) = c_{\mathsf{fill}}\big)$. It can be shown that this computation rule states the commutativity of the following square:

$$\begin{array}{ccc}
\mathsf{apd}^1_g(W) & \overset{\mathsf{apd}^2_g(\mathsf{fill})}{=\!=\!=\!=\!=\!=\!=\!=}^{2\,C}_{\mathsf{fill}} & \mathsf{apd}^1_g(E_1 \blacksquare E_2) \\
\Big\| \mathbb{C}_W & & \Big\| \\
c_W & \overset{c_{\mathsf{fill}}}{=\!=\!=\!=\!=\!=\!=\!=}^{2\,C}_{\mathsf{fill}} & \mathsf{apd}^1_g(E_1) * \mathsf{apd}^1_g(E_2) \\
 & & \Big\| \mathbb{C}_{E_1}\, \mathbb{C}_{E_2} \\
 & & c_{E_1} * c_{E_2}
\end{array}$$

  Also, the unlabeled equality above is given by the following lemma:

Lemma 3. *Let $A : \mathcal{U}$ and $C : A \to \mathcal{U}$. Let $\chi : \alpha = \beta$ and $\xi : \beta = \gamma$ be n-paths in A. Let $g : \prod_{x:A} C(x)$ be a dependent map. Then*

$$\mathsf{apd}^n_g(\chi \blacksquare \xi) = \mathsf{apd}^n_g(\chi) * \mathsf{apd}^n_g(\xi)$$

Proof. By path induction on χ and ξ, the goal reduces to prove that

$$\mathsf{apd}^n_g(\mathsf{refl}_\alpha) = \mathsf{apd}^n_g(\mathsf{refl}_\alpha) * \mathsf{apd}^n_g(\mathsf{refl}_\alpha)$$

but by unfolding the definitions it is easy to see these are judgmentally equal.