Free Higher Groups in Homotopy Type Theory*

Nicolai Kraus
University of Nottingham

Thorsten Altenkirch
University of Nottingham

Abstract

Given a type A in homotopy type theory (HoTT), we can define the *free ∞-group on A* as the loop space of the suspension of $A+1$. Equivalently, this free higher group can be defined as a higher inductive type $\mathbf{F}(A)$ with constructors $\mathsf{unit} : \mathbf{F}(A)$, $\mathsf{cons} : A \to \mathbf{F}(A) \to \mathbf{F}(A)$, and conditions saying that every $\mathsf{cons}(a)$ is an auto-equivalence on $\mathbf{F}(A)$. Assuming that A is a set (i.e. satisfies the principle of unique identity proofs), we are interested in the question whether $\mathbf{F}(A)$ is a set as well, which is very much related to an open problem in the HoTT book [22, Ex. 8.2]. We show an approximation to the question, namely that the fundamental groups of $\mathbf{F}(A)$ are trivial, i.e. that $\|\mathbf{F}(A)\|_1$ is a set.

CCS Concepts • **Theory of computation → Type theory**;

Keywords homotopy type theory, higher algebraic structures, truncation levels

ACM Reference Format:
Nicolai Kraus and Thorsten Altenkirch. 2018. Free Higher Groups in Homotopy Type Theory. In *LICS '18: 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, July 9–12, 2018, Oxford, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3209108.3209183

1 Introduction

An important feature of *Martin-Löf type theory* (MLTT) is the identity type which makes it possible to express equality inside type theory. More precisely, if A is a type (in any context), and $x, y : A$ are terms, then $\mathsf{Id}_A(x, y)$ is a type whose inhabitants represent proofs that x and y are equal. As it is common nowadays, we write $x =_A y$ or $x = y$ instead of $\mathsf{Id}_A(x, y)$, and call its elements simply *equalities*. *Homotopy type theory* (HoTT) embraces the fact that $x = y$ may come with interesting structure. This means that, in many cases, we do not only care about the question whether x and y are equal, but also *how* or *in which ways* they are equal. For example, due to Voevodsky's univalence axiom, $\mathbf{2} =_{\mathcal{U}} \mathbf{2}$ is equivalent to $\mathbf{2}$. Here, $\mathbf{2}$ is the type of booleans, $\mathcal{U}$ is a type universe, and the two equalities stem from the two ways in which $\mathbf{2}$ is equivalent to itself. Another non-trivial example is $\mathsf{base} =_{\mathbb{S}^1} \mathsf{base}$ which turns out to be equivalent to the type of integers [17], where $\mathbb{S}^1$ is the "circle" in HoTT.

*This work was supported by the Engineering and Physical Sciences Research Council (EPSRC), grant reference EP/M016994/1, and by USAF, Airforce office for scientific research, award FA9550-16-1-0029.

LICS '18, July 9–12, 2018, Oxford, United Kingdom

ACM ISBN 978-1-4503-5583-4/18/07…$15.00
https://doi.org/10.1145/3209108.3209183

A type where each such type of equalities can have at most one inhabitant is called a *set*, and further said to satisfy the principle of *unique identity proofs* (UIP). An example for a set is the type of natural numbers: $0 =_{\mathbb{N}} 1$ is equivalent to the empty type $\mathbf{0}$, while $0 =_{\mathbb{N}} 0$ is equivalent to the unit type $\mathbf{1}$, and so on. Many algebraic structures can be implemented straightforwardly if we are happy to do everything with sets; for example, [22, Def 6.11.1] defines a set-level group to be a tuple $(G, \circ, u, \cdot^{-1}, \dots)$, where G is a set, together with a multiplication operation $\circ : G \times G \to G$, a unit element u, an inversion operator $G \to G$, and equalities expressing the usual laws. The book then further shows how one can construct the free set-level group over a given set (see [22, Chp 6.11]).

More interesting and challenging is it to define higher-level structures, not restricted to sets. Since we use HoTT (rather than, say, set theory) as our foundation, it is natural to attempt this. For example, it has been known for some time that externally, every type carries the structure of an *∞-groupoid* [18, 24]. Internally, we can play a trick and use the following definition, which probably can be considered "HoTT folklore":

Definition 1. An *∞-group* is a type G which is equivalent to a loop space. More precisely, G is a group means that we have a connected type X and a point $x : X$ together with an equivalence $G \simeq (x = x)$. If G is an ∞-group represented by (X, x) and H is a second ∞-group represented by (Y, y), then a *homomorphism* $G \to_{\infty\mathrm{grp}} H$ is a pointed function $(X, x) \to_\bullet (Y, y)$.

In other words, an *∞-group* is a type that admits a *delooping*. Clearly, the unit element of this group is refl_x, and composition is given by composition of equalities. Some theory of higher groups in homotopy type theory has very recently been developed by Buchholtz, van Doorn and Rijke [10].

It is worth noting that Definition 1 makes use of the fact that a suitable notion of an (untruncated) group already naturally exists in HoTT, which is not the case for many other interesting structures. Defining untruncated algebraic structures in general and *directly* is an important open problem in HoTT. To see why this is hard, let us start from the set-level definition $(G, \circ, u, \cdot^{-1}, \dots)$ above and remove the condition that G is a set. The equalities which guarantee that the multiplication is associative, the unit is neutral, and inverses cancel, are not sufficient anymore; they would not give a well-behaved definition of a higher group. For example, one may ask oneself how one would prove that $x \circ (y \circ (z \circ w))$ equals $((x \circ y) \circ z) \circ w$: there are two canonical ways, and these should coincide ("MacLane's pentagon"), but any such rule that we add would have to satisfy its own coherences. It is currently unknown whether it is possible to complete this sort of definition in a satisfactory way. In a nutshell, the problem is that the usual definitions would, if expressed internally in type theory, amount to infinite structures of coherences. In classical homotopy theory, these conditions are often organised in the form of an *operad* [1, 21], but a representation of such structures that can be written down in type theory has not been discovered so far. This is certainly not for a lack of trying;

cf. the much-discussed open problem of defining *semisimplicial types* [23].

In this paper, we study the *free ∞-group* over a type A. It is folklore in homotopy type theory that a suitable definition of the free ∞-group over A is given by the loop space of a "wedge of A-many circles", or put differently, the suspension $\Sigma(A + \mathbf{1})$. For us, it will however be more helpful to give a more explicit construction of this free higher group which we call $\mathbf{F}(A)$. We define $\mathbf{F}(A)$ to be the *higher inductive type* (HIT) [22, Chp 6] which as constructors has a neutral element $\mathsf{unit} : \mathbf{F}(A)$ and a multiplication operation $\mathsf{cons} : A \to \mathbf{F}(A) \to \mathbf{F}(A)$, together with conditions ensuring that each $\mathsf{cons}(a) : \mathbf{F}(A) \to \mathbf{F}(A)$ is an equivalence. This definition encodes the a priori infinite tower of coherence condition suitably and it will turn out that it is equivalent to the loop space of $\Sigma(A+\mathbf{1})$.

The most basic properties that one would expect from a free higher group are easy to prove. More intriguing is the question what the free ∞-group has to do with the free set-based group. Clearly, we would want the former to be a generalisation of the latter. The most obvious way of interpreting this is to ask whether, for a *set* A, the free higher group $\mathbf{F}(A)$ coincides with the construction of the set-based higher group. It turns out that the central question is the following:

Question 2. *If A is a set, is $\mathbf{F}(A)$ a set as well?*

One reason why we believe that Question 2 is hard is that a slight generalisation of it is a known open problem in HoTT which has been recorded in the book (see [22, Ex. 8.2]). To be precise, the open problem asks whether, for a set A, the suspension $\Sigma(A)$ is a 1-type; our question is equivalent to asking whether $\Sigma(A + \mathbf{1})$ is a 1-type. A positive answer to the open problem would imply a positive answer to our Question 2, but we do not expect that our question is fundamentally easier. (Recall from the book [22] that the suspension $\Sigma(A)$ is the HIT with constructors $\mathsf{N}, \mathsf{S} : \Sigma(A)$ and $\mathsf{mer} : A \to \mathsf{N} = \mathsf{S}$, for "north", "south", "meridian".)

The core of our paper consists of a proof of a weakened version, a first approximation, of Question 2. Our main result can be phrased as follows:

Theorem 3. *If A is a set, then all fundamental groups of $\mathbf{F}(A)$ are trivial. In other words, $\|\mathbf{F}(A)\|_1$ is a set.*

Our strategy to prove this is to define a simple reduction system together with a *non-recursive approximation*, written $\mathbf{N}(A)$, to the free ∞-group. These are both based on the usual construction of the free monoid on A, that is, $\mathsf{List}(A)$. The proof of Theorem 3, with all the tools and strategies that we need to develop, constitutes the main part of the paper.

The reason why our strategies are not sufficient to provide a full answer to Question 2 is that $\mathbf{N}(A)$ is really only an approximation. Defining $\mathbf{F}(A)$ in a non-recursive way (i.e. without using some sort of induction that quantifies over elements of $\mathbf{F}(A)$ itself) seems to, as least as far as we can see, correspond exactly to expressing the infinite coherence tower "directly". We would not be surprised if it turned out that Question 2 was in fact independent of "standard HoTT" (the type theory developed in the book [22]), and if the status of Question 2 was related to the status of semisimplicial types. We will come back to this in the conclusions of the paper.

Setting The type theory that we consider in this paper is the standard homotopy type theory developed in the book [22]. This means that we have *univalent universes* $\mathcal{U}$, *function extensionality*, and *higher inductive types* (HITs). Regarding notation, we strive to stay close to [22], with the exception that we write $(a : A) \to B(a)$ instead of $\Pi(a : A).B(a)$. All performed constructions will preserve the universe level, hence there is no risk at omitting it. Uncurrying is done implicitly, allowing us to write $f(a, b)$ instead of $f(a)(b)$.

Outline We give the precise definition of the free ∞-group $\mathbf{F}(A)$ in Section 2, together with some simple observations. The statements in this section (apart from the definition of $\mathbf{F}(A)$ and its universal property) are not important for the main part of the paper, the proof of Theorem 3, which is given in Section 3. As a corollary of the constructions, we get that $\|\mathbf{F}(A)\|_1$ coincides with the set-based construction of the free group, and $\mathbf{F}(A)$ does under the assumption that Question 2 has a positive answer. In Section 4, we make some concluding remarks and discuss related open problems in homotopy type theory.

2 Free ∞-Groups

2.1 Definition and First Properties

Let us start with an explicit construction of the free higher group as a higher inductive type $\mathbf{F}(A)$, since this is the central concept of the paper. We use a point constructor $\mathsf{unit} : \mathbf{F}(A)$ for the neutral element, and a constructor $\mathsf{cons} : A \to \mathbf{F}(A) \to \mathbf{F}(A)$ which "multiplies" an element of A with any other group element. We write $\mathsf{cons}_a : \mathbf{F}(A) \to \mathbf{F}(A)$ instead of $\mathsf{cons}(a)$ since, most of the time, we regard a as a fixed variable. The trick which completes the definition in an elegant way, due to Paolo Capriotti, is to add the condition that cons_a is an equivalence for every a. This cannot be done directly (at least not according to the usual intuitive rules for presentations of higher inductive types), but it can be "unfolded" and expressed via a suitable collection of constructors. Let us show the concrete definition.

Definition 4. The *free ∞-group* over a given type A is the following higher inductive type $\mathbf{F}(A)$:

data $\mathbf{F}(A)$ **where**

$\mathsf{unit} : \mathbf{F}(A)$

$\mathsf{cons} : A \to \mathbf{F}(A) \to \mathbf{F}(A)$

$\mathsf{icons} : A \to \mathbf{F}(A) \to \mathbf{F}(A)$

$\mu_1 : (a : A) \to (x : \mathbf{F}(A)) \to \mathsf{icons}_a(\mathsf{cons}_a(x)) = x$

$\mu_2 : (a : A) \to (x : \mathbf{F}(A)) \to \mathsf{cons}_a(\mathsf{icons}_a(x)) = x$

$\mu : (a : A) \to (x : \mathbf{F}(A)) \to \mathsf{ap}_{\mathsf{cons}(a)}(\mu_1(a, x)) = \mu_2(a, \mathsf{icons}_a(x))$

At first glance, the above HIT appears complicated and rather unappealing. Due to the "unfolding", the underlying idea that we have discussed above is somewhat hidden. The constructors unit and cons are the standard ones that one would use to define the type of lists over A, $\mathsf{List}(A)$, or, in other words, the free monoid over A. The remaining four constructors ($\mathsf{icons}, \mu_1, \mu_2, \mu$) simply say that, for every $a : A$, the function $\mathsf{cons}_a : \mathbf{F}(A) \to \mathbf{F}(A)$ is a *half-adjoint equivalence* as defined in [22, Chp 4]. This is the "unfolding" mentioned before; note that we could equally well have used other definition of equivalences, such as the "bi-invertible" or "contractible fibre" constructions. In any case, this means that we can think of $\mathbf{F}(A)$ as being fully described as a triple $(\mathsf{unit}, \mathsf{cons}, \mathsf{iseq})$, with $\mathsf{iseq} : (a : A) \to \mathsf{isequiv}(\mathsf{cons}_a)$.

To make use of the type $\mathbf{F}(A)$, we need to know an elimination principle for it. This can be stated as an induction (dependent elimination) principle, which is how it is done in the book [22]. More concise, and (we would say) conceptually clearer, is the approach of phrasing it using a universal property, in other words, a recursion (non-dependent) elimination principle with a uniqueness property. The equivalence between these approaches for inductive types has been discussed by Awodey, Gambino, and Sojakova [8], for some HITs, by Sojakova [20], and a restricted version for set-truncated HITs can be found in [2]. For *concrete* HITs, such as our $\mathbf{F}(A)$, it is straightforward to derive the various elimination principles from each other. We state the universal property using the presentation as a *homotopy-initial algebras* [8]:

Principle 5. We say that an $\mathbf{F}(A)$*-algebra structure* on a type X consists of a point $u : X$, a map $f : A \to X \to X$, and a proof $p : (a : A) \to \mathsf{isequiv}(f(a))$. We say that the type of $\mathbf{F}(A)$*-algebra morphisms* between (X, u, f, p) and (Y, v, g, q) consists of triples (h, r, s), where $h : X \to Y$, $r : f(u) = v$, and $s : h \circ f = g \circ h$. Then, the induction principle of $\mathbf{F}(A)$ is equivalent to saying that $(\mathbf{F}(A), \mathsf{unit}, \mathsf{cons}, \mathsf{iseq})$ is *homotopy initial*, i.e. that for any (X, u, f, p), the type of morphisms from $(\mathbf{F}(A), \mathsf{unit}, \mathsf{cons}, \mathsf{iseq})$ to (X, u, f, p) is contractible.

We will come back to $\mathbf{F}(A)$-algebras later.

An obvious question is whether $\mathbf{F}(A)$ really deserves to be called the *free* ∞*-group* on A. There are two points: first, we need to check that it is a higher group in the sense of Definition 1, and second, we have to justify the attribute *free*.

For the first point, note that the suspension $\Sigma(A + \mathbf{1})$ has an equivalent description which can be obtained by essentially collapsing the point S with the path given by the unit type: it is the HIT $\mathsf{W}(A)$ with a single point constructor $\mathsf{N} : \mathsf{W}(A)$ and a family of loops indexed over A, as in $\mathsf{loops} : A \to \mathsf{N} = \mathsf{N}$, a *wedge of A-many circles*. Note that $\mathsf{W}(A)$ is automatically connected. A further side remark is that $\mathsf{W}(\mathbf{1})$ is canonically equivalent to the circle $\mathbb{S}^1$. We then observe:

Lemma 6. *The free* ∞*-group* $\mathbf{F}(A)$ *is an* ∞*-group, with* $\mathsf{W}(A)$ *as its delooping. The canonical equivalence* $e : \mathbf{F}(A) \to (\mathsf{N} =_{\mathsf{W}(A)} \mathsf{N})$ *maps the structure as one would expect, i.e. we have* $e(\mathsf{unit}) = \mathsf{refl}$ *and* $e(\mathsf{cons}_a(x)) = \mathsf{loops}(a) \cdot e(x)$.

Note that this statement is completely independent from the rest of the paper.

Proof. This is a relatively straightforward generalisation of the proof that the loop space of $\mathbb{S}^1$ is equivalent to the type of integers. The proof is an application of [22, Lem 8.9.1] and does not provide much insight, which is why we choose to omit it. For a detailed argument, one can easily adapt the proof given by Brunerie (for an only slightly different statement) in [9, Sec 6]. □

From the above lemma, we can in particular observe that $\mathbf{F}(\mathbf{1})$ is a presentation of the type of integers.

Next, we need to justify why we call $\mathbf{F}(A)$ the *free* higher group. The following presentation of the argument was suggested by Paolo Capriotti. Let us consider the following diagram:

$$\mathcal{U} \underset{\mathsf{p}_1}{\overset{+\mathbf{1}}{\rightleftarrows}} \mathcal{U}_\bullet \underset{\Omega}{\overset{\Sigma}{\rightleftarrows}} \mathcal{U}_\bullet \qquad (1)$$

Here, $\mathcal{U}_\bullet$ is the universe of pointed types. The function $(+\mathbf{1}) : \mathcal{U} \to \mathcal{U}_\bullet$ maps a type X to $(X + \mathbf{1}, \mathsf{inr}(\star))$, while projection p_1 simply forgets the point. As in [22, Chp 6.5], we regard the suspension as a function $\Sigma : \mathcal{U}_\bullet \to \mathcal{U}_\bullet$, mapping (X, x) to $(\Sigma(X), \mathsf{N})$, and Ω is the loop space. For $X : \mathcal{U}$ and $Y, Z : \mathcal{U}_\bullet$, it is easy to see that there is a canonical equivalence

$$(X \to \mathsf{p}_1 Y) \simeq ((X + \mathbf{1}) \to_\bullet Y)\,, \qquad (2)$$

and by [22, Lem 6.5.4], we have

$$(Y \to_\bullet \Omega(Z)) \simeq (\Sigma(Y) \to_\bullet Z)\,. \qquad (3)$$

The above diagram (1) should for our purpose only be regarded as an illustration of these two equivalences. Talking about the adjunctions more precisely is difficult since the correct notions would be ∞-categorical. This leads into a territory that is vastly unexplored in homotopy type theory [12], although higher adjunctions can be represented using only a finite amount of data [19]; here, we do not go further into this.

Let G be a given ∞-group, represented by (Z, z). This means that we have $G \simeq (z = z)$. We can then calculate:

$$\begin{aligned} & & \mathbf{F}(A) &\to_{\infty\mathsf{grp}} G \\ &\text{by Def 1 and Lem 6} & &\simeq (\mathsf{W}(A), \mathsf{N}) \to_\bullet (Z, z) \\ & & &\simeq \Sigma(A + \mathbf{1}) \to_\bullet (Z, z) \\ &\text{by (3)} & &\simeq (A + \mathbf{1}) \to_\bullet \Omega(Z, z) \\ &\text{by (2)} & &\simeq A \to (z = z) \\ & & &\simeq A \to G. \end{aligned} \qquad (4)$$

Thus, $\mathbf{F} : \mathcal{U} \to \infty\mathsf{GRP}$ is "morally" left adjoint to the forgetful functor which returns the underlying type of a higher group.

2.2 On Alternative Constructions

As a preparation for the development in Section 3, and to better understand the difficulties with $\mathbf{F}(A)$, let us attempt to construct $\mathbf{F}(A)$ in a different way. Let us write $A^\pm$ for $A + A$; we call $A^\pm$ the type of *elements of A with a sign*, and we think of $\mathsf{inl}(a)$ as a and $\mathsf{inr}(a)$ as a^{-1}. For $a : A^\pm$, we write $\dot{a}$ for the element we get by changing the sign. Of course, this means that $\dot{\dot{a}} = a$.

Elements of the free group $\mathbf{F}(A)$ are, at least intuitively, lists over $A^\pm$. The difficulty is that different lists may represent the same group element. This happens, for example, for $[\mathsf{inl}(a), \mathsf{inr}(a)]$ (i.e. $a \cdot a^{-1}$) and the empty list, both of which represent the unit of the group. We can avoid this problem by quotienting to identify the list $[x_0, \ldots, x_k, a, \dot{a}, x_{k+1}, \ldots, x_n]$ with the list $[x_0, \ldots, x_n]$. This quotient will be a set by definition of the quotient (set quotient) operation. If we are happy to work only with sets, and to set-truncate everything, then this is entirely possible, and in fact, it is a construction of the set-based free group given in [22, Thm 6.11.7]. If, like in this paper, we do not want to restrict ourselves to sets, we might think of taking a HIT which has path constructors for each such pair of lists, without set-truncating. The problem is that we need coherences: if we use a path constructor to reduce one redex and then a second, we should get the same equality as if we reduce the second redex and then the first. When looking at *three* redexes, we need to express that these equalities "fit together", and so on. This is an instance of the problem of infinite coherences which seem to be hard and possibly impossible to express in HoTT. In Section 3, we will perform a finite approximation of this construction in order

to show Theorem 3, although we will see that a couple of additional arguments are required to complete the proof.

Alternatively, we could think to only consider lists over $A^\pm$ in *normal form*, i.e. lists which come together with a proof that they do not contain a redex. The type of lists over A in normal form is a set (assuming that A is a set), and the presentation is indeed fully coherent. The trouble is that we are in general unable to define a suitable binary operation on this set, i.e. we are lacking a group operation. If we have two lists in normal form, their concatenation might not be in normal form, and for arbitrary types, we have no way of calculating a normal form or even checking whether we already have a normal form.

Unsurprisingly, the approach with normal forms works if A has decidable equality:

Proposition 7. If A has decidable equality, in the sense that

$$(a_1, a_2 : A) \to (a_1 = a_2) + ((a_1 = a_2) \to \mathbf{0}), \tag{5}$$

then $\mathbf{F}(A)$ has decidable equality as well. Moreover, $\mathbf{F}(A)$ is in this case canonically equivalent to the set-truncated construction of the free group as given in [22, Chp 6.11].

Proof sketch. Thanks to [22, Thm 6.11.7], we can take set-quotiented lists (as described above) as the definition of the set-truncated free group. Using decidable equality of A, it is easy to see that this quotient is equivalent to the type of lists in normal form; let us write $\mathsf{LNF}(A)$ for the latter type. An element of $\mathsf{LNF}(A)$ is a list together with a propositional property, and we have an embedding $\mathsf{LNF}(A) \to \mathsf{List}(A^\pm)$. What is left to do is to compare $\mathsf{LNF}(A)$ with $\mathbf{F}(A)$. Note that $\mathsf{LNF}(A)$ is a set without being explicitly set-truncated. There is a canonical $\mathbf{F}(A)$-algebra structure on $\mathsf{LNF}(A)$, giving rise to a map $\mathbf{F}(A) \to \mathsf{LNF}(A)$. Further, one can construct a function $\mathsf{List}(A^\pm) \to \mathbf{F}(A)$, by induction on the list. The empty list is mapped to unit, $\mathsf{inl}(a)$ translates into an application of cons_a, and $\mathsf{inr}(a)$ becomes icons_a. These functions give rise to an equivalence $\mathsf{LNF}(A) \simeq \mathbf{F}(A)$, and since $\mathsf{LNF}(A)$ has decidable equality, $\mathbf{F}(A)$ enjoys the same property. □

3 The fundamental group of the free ∞-group

In this section, the core of the paper, we develop a couple of techniques that, when combined, allow us to prove Theorem 3. For the whole section, let us assume that A is a given set. Given lists $x, y : \mathsf{List}(A^\pm)$, we write xy for their *concatenation*, i.e. the list we get by simply joining the two lists as in $[a_1, a_2][a_3, a_4] = [a_1, a_2, a_3, a_4]$. Since this operation is associative (up to a canonical and fully coherent equality), we omit brackets and write xyz for both $(xy)z$ and $x(yz)$. Given $a : A^\pm$, we regard a as a one-element list and allow ourselves to write e.g. $xayz$ or $a\dot{a}y$.

3.1 A simple reduction system in type theory

As discussed in Section 2.2 above, we can think of elements of $\mathbf{F}(A)$ as lists over $A^\pm$, and the main problem is that different lists represent the same group element. This motivates the development of a system of reductions.

Definition 8. The type family $\mathsf{Red} : \mathsf{List}(A^\pm) \to \mathcal{U}$, which expresses that a list represents the same group element as the empty list (i.e. the neutral element of the group $\mathbf{F}(A)$), is defined as follows. We first define an auxiliary family $\mathsf{R} : \mathbb{N} \to \mathsf{List}(A^\pm) \to \mathcal{U}$ by induction on the natural numbers:

$$\begin{aligned}
\mathsf{R}_0(x) &:\equiv \mathsf{length}(x) = 0 \\
\mathsf{R}_{2+n}(x) &:\equiv \Sigma(a : A^\pm), (y, z : \mathsf{List}(A^\pm)), \\
&\qquad (\mathsf{length}(y) + \mathsf{length}(z) = n)\,, \\
&\qquad (x = ya\dot{a}z)\,, \\
&\qquad \mathsf{R}_n(yz)
\end{aligned}$$

Using this, we set $\mathsf{Red}(x) :\equiv \mathsf{R}_{\mathsf{length}(x)}(x)$.

If we have indexed inductive families in the theory, we can alternatively define Red directly as such a family generated by

$\mathsf{zero} : \mathsf{Red}(\mathsf{nil})$

$\mathsf{step} : (y, z : \mathsf{List}(A^\pm)) \to (a : A^\pm) \to \mathsf{Red}(yz) \to \mathsf{Red}(ya\dot{a}z)$.

The two definitions are essentially the same, only represented in different ways. In both cases, given $r : \mathsf{Red}(x)$, we say that r witnesses that x can be *reduced* to the empty list and we call r a *reduction sequence*. We view it as a sequence consisting of *steps*, each of which removes a single *redex* $a\dot{a}$. An example of a reduction sequence $r : \mathsf{Red}(a\dot{a}bc\dot{c}\dot{b})$ could be pictured as follows, where each step is represented by an arrow $\rightsquigarrow$ annotated with the redex it reduces:

$$a\dot{a}bc\dot{c}\dot{b} \overset{c\dot{c}}{\rightsquigarrow} a\dot{a}b\dot{b} \overset{a\dot{a}}{\rightsquigarrow} b\dot{b} \overset{b\dot{b}}{\rightsquigarrow} \mathsf{nil}. \tag{6}$$

Remark 9. There are a couple of points that we want to point out explicitly.

1. In the above example and in the discussions to come, $a : A^\pm$ is already positive or negative, which means that every redex is of the form $a\dot{a}$; the possibility $\dot{a}a$ is already covered.
2. The number of steps of $r : \mathsf{Red}(x)$ is simply half of the length of the list x, which means that all elements of $\mathsf{Red}(x)$ have the same number of steps. In particular, it is easy to prove that $\mathsf{Red}(x)$ is empty if $\mathsf{length}(x)$ is odd.
3. For a given list x, there is no way to *compute* a reduction sequence, since we do not know whether an occurring pair bc forms a redex. A reduction $r : \mathsf{Red}(x)$ encodes equalities which guarantee that all redexes that it reduces are really redexes. Deciding whether bc is a redex would require decidable equality on A (but of course, we can always check whether an element of $A^\pm$ is positive or negative, and this analysis might give us that bc is definitely not a redex).
4. For a given x, equality on $\mathsf{Red}(x)$ *is* decidable. This is because a sequence encodes the positions of the redexes that it reduces, and positions are decidable, while the (in general undecidable) equalities on $A^\pm$ are propositions. Similarly, if we have $r : \mathsf{Red}(xby)$, we can say in which step b is reduced, since this is encoded by a position.

Let us remind ourselves that the goal of the paper is to show that $\mathbf{F}(A)$ has trivial fundamental groups. This is a statement about equalities between equalities. If we think of a reduction sequence as a proof that a list represents the neutral group element, i.e. as something giving rise to an equality proof, it is hopefully intuitive that we now want to discuss the relationship between different reduction sequences. In a nutshell, we want to give a criterion which guarantees that two reduction sequences give rise to equal equalities. To do so, we consider *transformations*:

Definition 10. Let $w : \mathsf{List}(A^{\pm})$ be a list and $r : \mathsf{Red}(w)$ a reduction sequence. We consider the following two operations, each of which allows us to create a new reduction sequence in $\mathsf{Red}(w)$ from r:

1. Swap two consecutive independent steps in r. More precisely, if r is a sequence of the form
$$\ldots \rightsquigarrow xa\dot{a}yb\dot{b}z \overset{b\dot{b}}{\rightsquigarrow} xa\dot{a}yz \overset{a\dot{a}}{\rightsquigarrow} xyz \rightsquigarrow \ldots, \tag{7}$$
we can change it to
$$\ldots \rightsquigarrow xa\dot{a}yb\dot{b}z \overset{a\dot{a}}{\rightsquigarrow} xyb\dot{b}z \overset{b\dot{b}}{\rightsquigarrow} xyz \rightsquigarrow \ldots. \tag{8}$$
Analogously, we can change (8) into (7).
2. If a step reduces a redex $a\dot{a}$ in a list of the form $xa\dot{a}ay$, we can change this step to remove the redex $\dot{a}a$ instead, or vice versa. This means that
$$\ldots \rightsquigarrow xa\dot{a}ay \overset{a\dot{a}}{\rightsquigarrow} xay \rightsquigarrow \ldots \tag{9}$$
can be changed to
$$\ldots \rightsquigarrow xa\dot{a}ay \overset{\dot{a}a}{\rightsquigarrow} xay \rightsquigarrow \ldots, \tag{10}$$
or vice versa.

We say that $r : \mathsf{Red}(w)$ can be *transformed* into $s : \mathsf{Red}(w)$ if there is a finite chain of these operations that changes r into s.

After what we said in the paragraph before Definition 10, the best we could hope for is that *any reduction sequence can be transformed into any other reduction sequence* (of the same list w). Indeed, this is what we will show. We start with a technical lemma which will not only help us to prove what we just said (Corollary 12), but also another useful consequence (Corollary 13).

Lemma 11. *Assume we are given a list of the form $xa\dot{a}y$, i.e. a list in $A^{\pm}$ with an explicitly given redex $a\dot{a}$. Assume further that we have a reduction sequence $s : \mathsf{Red}(xa\dot{a}y)$. It is possible to transform s into a reduction sequence which reduces the redex $a\dot{a}$ in the first step, i.e. starts with $xa\dot{a}y \overset{a\dot{a}}{\rightsquigarrow} xy \rightsquigarrow \ldots$.*

Proof. Let x, a, y, and $r : \mathsf{Red}(xa\dot{a}y)$ be given. Let us write m for the number of the step in which a is reduced, and n for the number of the step in which $\dot{a}$ is reduced. There are three cases:

- If $m = n$, then the redex $a\dot{a}$ is reduced in step n. If $n = 0$, there is nothing to do. Otherwise, we can swap this step with step $(n-1)$, since the two steps will be independent of each other. Swapping a further $(n-1)$ times, we can move the step reducing $a\dot{a}$ to the beginning of the sequence.
- If $m > n$, then $a\dot{a}$ are not reduced together, but $\dot{a}$ is reduced with some $\dot{\dot{a}}$ to its right instead. Note that $\dot{\dot{a}} = a$. Before step n, the list thus has to be of the form $ua\dot{a}av$, and step n consists of reducing $\dot{a}a$. We define r_1 to be the reduction sequence which is identical to r in every step expect in step n where it reduces $a\dot{a}$; this is the second of the two possible operations in Definition 10. We are now in case one ($m = n$).
- The case $m < n$ is analogous to the case $m > n$. □

Corollary 12. *Any reduction sequence can be transformed into any other reduction sequence. More precisely, for $w : \mathsf{List}(A^{\pm})$ and $r, s : \mathsf{Red}(w)$, we can transform r into s.*

Proof. A reduction sequence is given by a chain of reduction steps, and the number of steps in r and s are equal (both are $\mathsf{length}(w)/2$). Thus, it is sufficient to transform r into a sequence which consists of the same steps as s. By the above lemma, we can transform r into a sequence r' which in the first step reduces whichever redex s reduces in the first step. Applying the same argument to the "tail" of the sequences (note that r' and s, each with the first step removed, still reduce the same list), we get a transformation into a sequence which in every step mirrors the reduction of s and is thus equal to s. □

A second easy consequence is that, if a list is reducible, then we cannot "get stuck" while reducing: we can start reducing at an arbitrary position without risking of ending up with an unreducible list. Note that we write $B \leftrightarrow C$ for $(B \to C) \times (C \to B)$.

Corollary 13. *For any lists y, z and $a : A^{\pm}$, we have*
$$\mathsf{Red}(ya\dot{a}z) \leftrightarrow \mathsf{Red}(yz). \tag{11}$$

Proof. The direction $\leftarrow$ is immediate, by adding a single reduction step reducing $a\dot{a}$. The direction $\rightarrow$ is an application of Lemma 11. □

Remark 14. Note that Corollary 12 subtly but crucially depends on the assumption that A is a set, while Lemma 11, as formulated, would work for arbitrary types A. It is true independently of A that a reduction sequence is given by a chain of reduction steps. A reduction step encodes the *position* at which the reduction is taking place (say, the length of the list y in Definition 8), together with a proof that the reduction is possible (i.e. a proof that the pair at the position is actually a redex). The second part amounts to an equality in $A^{\pm}$ (since "ab being a redex" means $a = \dot{b}$); thus, it is a proposition if A is a set. In this case, a reduction step is determined by the position, and a reduction sequence is determined by the chain of positions which it encodes. The proof of Corollary 12 relies on this.

Lemma 11 holds even without the requirement of A being a set. However, note that the proof of Lemma 11, when it uses the second operation in Definition 10, has to construct a new equality (this is hidden in the sentence "Before step n, the list thus has to be of the form $ua\dot{a}av$"). Therefore, the new sequence constructed in Lemma 11 *will* reduce $a\dot{a}$ in the first step, but the proof that $a\dot{a}$ is indeed a redex could be a nontrivial one.

3.2 A non-recursive approximation to the free ∞-group

We are ready to define a *non-recursive approximation* to the free group $\mathbf{F}(A)$, a HIT that we call $\mathbf{N}(A)$. By *non-recursive*, we mean that constructors of $\mathbf{N}(A)$ do not use points or paths of $\mathbf{N}(A)$ in their arguments.

Definition 15. We define $\mathbf{N}(A)$ to be the HIT with the following constructors:

$$\begin{aligned}
\eta :&\ \ \mathsf{List}(A^{\pm}) \to \mathbf{N}(A)\\
\tau :&\ \ (x : \mathsf{List}(A^{\pm})) \to (a : A^{\pm}) \to (y : \mathsf{List}(A^{\pm}))\\
&\quad \to \eta(xa\dot{a}y) = \eta(xy)\\
\mathsf{sw} :&\ (x : \mathsf{List}(A^{\pm})) \to (a : A^{\pm}) \to (y : \mathsf{List}(A^{\pm}))\\
&\quad \to (b : A^{\pm}) \to (z : \mathsf{List}(A^{\pm}))\\
&\quad \to \tau(x, a, yb\dot{b}z) \cdot \tau(xy, b, z) = \tau(xa\dot{a}y, b, z) \cdot \tau(x, a, yz)\\
\mathsf{ov} :&\ (x : \mathsf{List}(A^{\pm})) \to (a : A^{\pm}) \to (y : \mathsf{List}(A^{\pm}))\\
&\quad \to \tau(x, a, ay) = \tau(xa, \dot{a}, y)\\
\mathsf{tr} :&\ \ \mathsf{is\text{-}1\text{-}type}(\mathbf{N}(A))
\end{aligned}$$

We can think of $\mathbf{N}(A)$ (without the last constructor) as a "wild" quotient of $\mathsf{List}(A^{\pm})$. Recall that we said that lists over $A^{\pm}$ correspond to very intentional representations of group elements. The HIT with constructors η and τ can be thought of as a "level 0 approximation" to a fully coherent non-recursive quotient of $\mathsf{List}(A^{\pm})$: we identify some lists which represent the same group element, but the equalities are incoherent. This is partially remedied by the constructors sw ("swap") and ov ("overlap"), ensuring that the equalities generated by τ satisfy basic coherence. They can be pictured as follows:

$$\begin{array}{ccc}
\eta(xa\dot{a}yb\dot{b}z) & \xrightarrow{\tau(x,a,yb\dot{b}z)} & \eta(xyb\dot{b}z)\\
{\scriptstyle \tau(xa\dot{a}y,b,z)}\downarrow & \mathsf{sw}(x,a,y,b,z) & \downarrow{\scriptstyle \tau(x,a,yb\dot{b}z)}\\
\eta(xa\dot{a}yz) & \xrightarrow[\tau(xa\dot{a}y,b,z)]{} & \eta(xyz)
\end{array} \tag{12}$$

$$\begin{array}{ccc}
 & \overset{\tau(x,a,ay)}{\frown} & \\
\eta(xa\dot{a}ay) & \mathsf{ov}(x,a,y) & \eta(xay)\\
 & \underset{\tau(xa,\dot{a},y)}{\smile} &
\end{array} \tag{13}$$

sw and ov themselves are not directly guaranteed to be coherent; if we omit the constructor tr, we can think of $\mathbf{N}(A)$ as a "level 1 approximation". tr ensures that all higher equalities hold, by forcing $\mathbf{N}(A)$ to be 1-truncated. The statement that $\mathbf{N}(A)$ is an approximation to the free higher group can then be made by drawing a connection to $\|\mathbf{F}(A)\|_1$, which we will do later.

If a list can be reduced, then in $\mathbf{N}(A)$, it is indistinguishable from the empty list:

Lemma 16. *We have a function*

$$\mathsf{red\text{-}is\text{-}neutral} : (z : \mathsf{List}(A^{\pm})) \to \mathsf{Red}(z) \to \eta(z) = \eta(\mathsf{nil}). \tag{14}$$

Proof. We need to analyse the element $r : \mathsf{Red}(z)$. It encodes a finite number of reduction steps. The first reduction step shows that z is of the form $z = xa\dot{a}y$, thus the constructor $\tau(x, a, y)$ (transported along the equality $z = xa\dot{a}y$) provides us with the equality $\eta(z) = \eta(xy)$. Similarly, each of the remaining reduction steps encoded in r shows how τ can be applied, and the concatenation of all these equalities yields $\eta(z) = \eta(\mathsf{nil})$.

If Red is defined as an indexed inductive family, we can construct $\mathsf{red\text{-}is\text{-}neutral}(x)(r)$ by induction on r, and the induction step is given by the constructor τ. □

Not only can we show that reducible lists are equal to nil in $\mathbf{N}(A)$, it is also the case that the concrete witness of reducibility does not matter:

Lemma 17. *For any given x, the function*

$$\mathsf{red\text{-}is\text{-}neutral}(x) : \mathsf{Red}(x) \to \eta(x) = \eta(\mathsf{nil}) \tag{15}$$

is weakly constant, in the following sense:

$$(r, s : \mathsf{Red}(x)) \to \mathsf{red\text{-}is\text{-}neutral}(x)(r) = \mathsf{red\text{-}is\text{-}neutral}(x)(s). \tag{16}$$

Proof. The constructors sw and ov ensure that, if two reduction sequences can be transformed into each other, then they lead to equal proofs of $\eta(x) = \eta(\mathsf{nil})$. More precisely, the first operation in Definition 10 is exactly covered by the constructor sw, while the second operation is covered by ov. The statement thus follows from Corollary 12. □

The point of $\mathbf{N}(A)$ is that it is easier to reason about $\mathbf{N}(A)$ than about $\mathbf{F}(A)$, thanks to the absence of recursive constructors; one can say that $\mathbf{N}(A)$ attempts to bridge the gap between $\mathsf{List}(A^{\pm})$ and $\mathbf{F}(A)$. We first define a property stating that an element of $\mathbf{N}(A)$ can be reduced. We write hProp for $\Sigma\,(X : \mathcal{U})\,.\,\mathsf{is\text{-}prop}(X)$ as usual.

Lemma 18. *The family $\|-\| \circ \mathsf{Red} : \mathsf{List}(A^{\pm}) \to \mathsf{hProp}$ extends to a family $\mathsf{red} : \mathbf{N}(A) \to \mathsf{hProp}$ as in the following commuting triangle:*

$$\begin{array}{ccccc}
\mathsf{List}(A^{\pm}) & \xrightarrow{\mathsf{Red}} & \mathcal{U} & \xrightarrow{\|-\|} & \mathsf{hProp}\\
{\scriptstyle \eta}\downarrow & & & \nearrow_{\mathsf{red}} & \\
\mathbf{N}(A) & & & &
\end{array} \tag{17}$$

Proof. We do induction on $\mathbf{N}(A)$. Clearly, we have to set $\mathsf{red}(\eta(x)) :\equiv \|\mathsf{Red}(x)\|$. The proof obligation of the constructor τ is met by Corollary 13. The remaining two constructors are trivial, since they ask for equalities between elements of propositions. □

To avoid confusion with elements of $\mathsf{List}(A^{\pm})$, which we call $x, y, z, \ldots$, we use Greek letters for elements of $\mathbf{N}(A)$. If $\gamma : \mathbf{N}(A)$ is reducible, it is equal to the neutral element:

Lemma 19. *There is a function of type*

$$(\gamma : \mathbf{N}(A)) \to \mathsf{red}(\gamma) \to \gamma = \eta(\mathsf{nil}). \tag{18}$$

Proof. We do induction on γ. First, we consider the case $\gamma \equiv \eta(x)$, and we want to find $f_x : \mathsf{red}(\eta(x)) \to \eta(x) = \eta(\mathsf{nil})$. Recall that a weakly constant function into a set (which the codomain here is) factors through the propositional truncation [16], hence since $\mathsf{red}(\eta(x)) \equiv \|\mathsf{Red}(x)\|$ by definition, Lemma 17 gives us a function

$$f_x : \mathsf{red}(\eta(x)) \to \eta(x) = \eta(\mathsf{nil}) \tag{19}$$

such that $f_x(|r|) = \mathsf{red\text{-}is\text{-}neutral}(x)(r)$. We want to extend this function to $\mathbf{N}(A)$. Induction on γ requires us to provide constructions corresponding to τ, sw, and ov. The latter two are contractible, and we do not need to worry about them. The proof obligation

for τ says that, for any y, a, z, and witnesses $s : \mathsf{red}(\eta(yz))$, $s' : \mathsf{red}(\eta(ya\dot{a}z))$, the triangle

$$\begin{array}{ccc} \eta(ya\dot{a}z) & \xrightarrow{\tau(y,a,z)} & \eta(yz) \\ & \searrow^{f_{ya\dot{a}z}(s')} & \downarrow{\scriptstyle f_{yz}(s)} \\ & & \eta(\mathsf{nil}) \end{array} \tag{20}$$

commutes. This is a proposition, thus we can assume that s, s' come from actual reduction sequences, i.e. we have $r : \mathsf{Red}(yz)$ with $|r| = s$ and $r' : \mathsf{Red}(ya\dot{a}z)$ with $|r'| = s'$. This simplifies the triangle to:

$$\begin{array}{ccc} \eta(ya\dot{a}z) & \xrightarrow{\tau(y,a,z)} & \eta(yz) \\ & \searrow^{\mathsf{red\text{-}is\text{-}neutral}(ya\dot{a}z)(r')} & \downarrow{\scriptstyle \mathsf{red\text{-}is\text{-}neutral}(yz)(r)} \\ & & \eta(\mathsf{nil}) \end{array} \tag{21}$$

If we write $r'' : \mathsf{Red}(ya\dot{a}z)$ for the sequence r, extended by the single step reducing $a\dot{a}$ in the beginning, we see that composition of the horizontal and vertical arrow give $\mathsf{red\text{-}is\text{-}neutral}(ya\dot{a}z)(r'')$. Thus, Lemma 17 yields the required commutativity of the triangle. □

This allows us to conclude:

Lemma 20. *$\mathbf{N}(A)$ is a set locally at $\eta(\mathsf{nil})$, in the sense that the type $\eta(\mathsf{nil}) =_{\mathbf{N}(A)} \eta(\mathsf{nil})$ is contractible.*

Proof. If equality is implied by a "reflexive mere relation", then the type is a set ([22, Thm 7.2.2.], sometimes called "Rijke's theorem"). Here, we need the local formulation of this statement as given in [15], together with Lemma 19. □

Next, we want to extend this observation and show that $\mathbf{N}(A)$ is a set. In general, if a type X is a set locally at $x_0 : X$ and we have an equivalence $e : X \to X$, then X is also a set locally at $e(x_0)$, using that ap_e will be an equivalence. Therefore, if for every $y : X$ there is an equivalence mapping x_0 to y (we can say that such an X is "homogeneous"), then X is a set; and in fact, it is enough if for a given y the equivalence merely exists (i.e. hidden with a truncation). This is one motivation for the following technical lemma, where we construct equivalences $\mathbf{N}(A) \to \mathbf{N}(A)$. Another motivation is that these equivalences are the main part of an $\mathbf{F}(A)$-algebra structure on $\mathbf{N}(A)$, but we will come back to this later.

Lemma 21. *There is a function $f : A^{\pm} \to \mathbf{N}(A) \to \mathbf{N}(A)$ such that, for every $c : A^{\pm}$, the map $f_c : \mathbf{N}(A) \to \mathbf{N}(A)$ is an equivalence with $f_{\dot{c}}$ as its inverse. Further, the construction can be done such that, for every $x : \mathsf{List}(A^{\pm})$, we have $f_c(\eta(x)) \equiv \eta(cx)$.*

Proof. Let $c : A$ be given. We need to define $f_c : \mathbf{N}(A) \to \mathbf{N}(A)$, i.e. for a given $\alpha : \mathbf{N}(A)$, we need $f_c(\alpha) : \mathbf{N}(A)$. This can be done by recursion on α in the obvious way:

- We set $f_c(\eta(x)) :\equiv \eta(cx)$.
- Next, we need a witness of $f_c(\eta(xa\dot{a}y)) = f_c(\eta(xy))$. Slightly abusing notation, we write $f_c(\tau(x,a,y))$ for this[1], and we set $f_c(\tau(x,a,y) :\equiv \tau(cx,a,y)$.
- Similarly, we set $f_c(\mathsf{sw}(x,a,y,z)) :\equiv \mathsf{sw}(cx,a,y,z)$;
- and $f_c(\mathsf{ov}(x,a,y)) :\equiv \mathsf{ov}(cx,a,y)$;
- and finally, we have $f_c(\mathsf{tr}) :\equiv \mathsf{tr}$.

We need to show that f_c is an inverse of $f_{\dot{c}}$. It is sufficient to show that, for $\alpha : \mathbf{N}(A)$, we have $f_c(f_{\dot{c}}(\alpha)) = \alpha$ and $f_{\dot{c}}(f_c(\alpha)) = \alpha$. Let us concentrate on the first of these, as the second is no more than a copy which switches the sign of c. Note that the goal is an equality in the 1-type $\mathbf{N}(A)$ and thus a set. Thus, when we do induction on α, in order to construct a function $h : (\alpha : \mathbf{N}(A)) \to f_c(f_{\dot{c}}(\alpha)) = \alpha$, the proof obligations for sw, ov, and tr are trivial. For η and τ, the constructions work as follows:

- For η, we need $h(\eta(x))$ of type $f_c(f_{\dot{c}}(\eta(x))) = \eta(x)$, which reduces to $\eta(c\dot{c}x) = \eta(x)$. Therefore, we can set $h(\eta(x)) :\equiv \tau(\mathsf{nil}, c, x)$.
- For τ, we need to construct $h(\tau(x,a,y))$ which shows that $h(\eta(xa\dot{a}y))$ and $h(\eta(xy))$ are equal as paths over $\tau(x,a,y)$. After unfolding what this means, we see that the type of $h(\tau(x,a,y))$ is:

$$\mathsf{ap}_{f_{c\dot{c}}}(\tau(x,a,y)) \cdot \tau(\mathsf{nil}, c, xy) = \tau(\mathsf{nil}, c, xa\dot{a}y) \cdot \tau(x,a,y). \tag{22}$$

By the given construction of f_c above, this simplifies to

$$\tau(c\dot{c}x,a,y) \cdot \tau(\mathsf{nil}, c, xy) = \tau(\mathsf{nil}, c, xa\dot{a}y) \cdot \tau(x,a,y), \tag{23}$$

which is given by $\mathsf{sw}(\mathsf{nil}, c, x, a, y)$. □

From Lemma 21, it is very easy to derive an $\mathbf{F}(A)$-algebra structure. We will record this later in Corollary 23. Before going there, we draw another immediate conclusion:

Lemma 22. *The type $\mathbf{N}(A)$ is a set.*

Proof. It suffices to show the that, for any given $\alpha : \mathbf{N}(A)$, the type $\alpha =_{\mathbf{N}(A)} \alpha$ is contractible. We do induction on α. Since the goal is a proposition which becomes trivial for all higher constructors, we only need to show the statement for the point constructor η. Thus, assuming $x : \mathsf{List}(A^{\pm})$, we need to show that $\eta(x) =_{\mathbf{N}(A)} \eta(x)$ is contractible. We do induction again, this time on the list x. If x is the empty list nil, then the statement is given by Lemma 20. Otherwise, x is ay with $a : A^{\pm}$. Consider the equivalence $f(a) : \mathbf{N}(A) \to \mathbf{N}(A)$ from Lemma 21. It gives us an equivalence $\mathsf{ap}_{f(a)} : \eta(y) = \eta(y) \to \eta(ay) = \eta(ay)$, the domain of which is contractible by the induction hypothesis. □

3.3 Connection between approximations of the free group

In order to make use of $\mathbf{N}(A)$ and the results we have found so far, we show in this section that $\|\mathbf{F}(A)\|_1$ is equivalent to $\mathbf{N}(A)$. A direct proof via "maps in both directions which are inverse to each other" would in principle be possible. Our calculations however led to a very messy argument, which did not provide much insight. In this paper, we therefore proceed a bit differently: after constructing $\mathbf{F}(A)$- and $\mathbf{N}(A)$-algebra structures on both $\mathbf{N}(A)$ and $\|\mathbf{F}(A)\|_1$ (which corresponds to constructing the two functions), we show that the structures are "compatible", i.e. that a certain $\mathbf{N}(A)$-algebra map is also an $\mathbf{F}(A)$-algebra morphism. We will later explain in detail what this means.

Recall from the statement of Principle 5 that an $\mathbf{F}(A)$-*algebra structure* on a type X consists of a point $u : X$ and a family $f : A \to X \to X$ such that each f_a is an equivalence on X, witnessed by some $p : (a : A) \to \mathsf{isequiv}(f_a)$. An $\mathbf{F}(A)$-algebra is a type X with such a structure, i.e. a tuple (X, u, f, p). Also recall that an $\mathbf{F}(A)$-algebra morphism between (X, u, f, p) and (Y, v, g, q) is a triple

[1] The more accurate notation might be $\mathsf{ap}_{f_c}(\tau(x,a,y))$.

(h, r, s), where $h : X \to Y$, $r : f(u) = v$, and $s : h \circ f = g \circ h$. Similarly, we say that a type Y carries an $\mathbf{N}(A)$-algebra structure if we have a tuple (e, t, s, o, h) mirroring the constructors of $\mathbf{N}(A)$, with $\mathbf{N}(A)$-algebra morphisms defined in the obvious way. Then, $(\mathbf{N}(A), \eta, \tau, \mathsf{sw}, \mathsf{ov}, \mathsf{tr})$ is homotopy initial among all $\mathbf{N}(A)$-algebras.

From Lemma 21, we immediately get a canonical $\mathbf{F}(A)$-algebra. Note that here and later we write _ (blank) for a "nameless" component which should be clear from the context.

Corollary 23. *We have an $\mathbf{F}(A)$-algebra $(\mathbf{N}(A), \eta(\mathsf{nil}), \overline{f}, \_)$, where $\overline{f}$ is given by the function Lemma 21 composed with the embedding $\mathsf{inl} : A \to A^{\pm}$ of a into "positively signed a". Since $\mathbf{F}(A)$ carries the initial such structure, we get a canonical map $\mathbf{F}(A) \to \mathbf{N}(A)$.*

It does not seem to be the case in general that truncations preserve algebra structure, since this seems to require a choice principle; see e.g. the infinitary branching trees in [4, 5]. Fortunately, it is very simple in our case:

Lemma 24. *The type $\|\mathbf{F}(A)\|_1$ carries an $\mathbf{F}(A)$-algebra structure, and $|-| : \mathbf{F}(A) \to \|\mathbf{F}(A)\|_1$ is an $\mathbf{F}(A)$-algebra morphism.*

Proof. This follows easily from the fact that $|-|$ preserves equivalences. □

Corollary 23 can be reversed if we add a truncation:

Lemma 25. *The type $\|\mathbf{F}(A)\|_1$ carries an $\mathbf{N}(A)$-algebra structure.*

Proof. Doing this in detail is tedious, but there is no hidden difficulty. The components corresponding to the constructors $(\eta, \tau, \mathsf{sw}, \mathsf{ov})$ could all be constructed using $\mathbf{F}(A)$ directly, we simply need to throw in $|-| : \mathbf{F}(A) \to \|\mathbf{F}(A)\|_1$ at the right places. The component corresponding to η, which has type

$$e : \mathsf{List}(A^{\pm}) \to \|\mathbf{F}(A)\|_1, \tag{24}$$

is simply given by composing instances of $|\mathsf{cons}|$ or $|\mathsf{icons}|$ with each other, one for each element of the list x (we use $|\mathsf{cons}|$ for positive list elements and $|\mathsf{icons}|$ for negative ones), and applying them on the unit element $|\mathsf{unit}|$. We write $e(x) :\equiv |\vec{\mathsf{cons}}_x|$ for this. For example, if x is the list $\dot{abc}$ (where a, b, c are now all assumed to be positive), then $e(x) \equiv |\vec{\mathsf{cons}}_x| \equiv |\mathsf{cons}_a|(|\mathsf{icons}_b|(|\mathsf{cons}_c|(|\mathsf{unit}|)))$.

The component corresponding to τ, which has type

$$t : (x : \mathsf{List}(A^{\pm})) \to (a : A^{\pm}) \to (y : \mathsf{List}(A^{\pm})) \to e(x\dot{a}a\dot{y}) = e(xy), \tag{25}$$

is then given by "whiskering" as in (let us for simplicity assume that a is positive):

$$t(x, a, y) :\equiv \mathsf{ap}_{|\vec{\mathsf{cons}}(x)|}(|\mu_2|(a, y)) \tag{26}$$

The components for sw and ov are essentially naturality of whiskering and μ, respectively, while the fact that we have 1-truncated $\mathbf{F}(A)$ gives us the component for the constructor tr. □

Using that $\mathbf{F}(A)$ carries the (homotopy) initial $\mathbf{F}(A)$-algebra structure, and $\mathbf{N}(A)$ the (homotopy) initial $\mathbf{N}(A)$-algebra structure, the statements of Corollary 23 and Lemma 25 give us maps h and k as follows:

$$\mathbf{F}(A) \xrightarrow[h]{\text{map of } \mathbf{F}(A)\text{-algs}} \mathbf{N}(A) \xrightarrow[k]{\text{map of } \mathbf{N}(A)\text{-algs}} \|\mathbf{F}(A)\|_1 \tag{27}$$

In the next lemma, we show that *both* these functions are maps of $\mathbf{F}(A)$-algebras. This will be sufficient to show that $\mathbf{N}(A)$ is a retract of $\|\mathbf{F}(A)\|_1$. It was a suggestion by Paolo Capriotti that this lemma might lead to a cleaner proof of the property we ultimately want, which, we think, is indeed the case.

Lemma 26. *The map k in (27) is a map of $\mathbf{F}(A)$-algebras, with respect to the $\mathbf{F}(A)$-algebra structures constructed in Corollary 23 and Lemma 24.*

Proof. We need to show that the points and the equivalences are preserved, independently of each other. The point in $\mathbf{N}(A)$ is $\eta(\mathsf{nil})$, which is mapped to the $|\mathsf{unit}|$ as required. For the equivalence, we only need to check that the underlying functions match accordingly. This corresponds to showing commutativity of the following square, for any given $c : A$:

$$\begin{array}{ccc} \mathbf{N}(A) & \xrightarrow{f_c \text{ (Lem 21)}} & \mathbf{N}(A) \\ {\scriptstyle k}\downarrow & & \downarrow{\scriptstyle k} \\ \|\mathbf{F}(A)\|_1 & \xrightarrow[|\mathsf{cons}_c|]{} & \|\mathbf{F}(A)\|_1 \end{array} \tag{28}$$

We do induction on $\alpha : \mathbf{N}(A)$. The goal is an equality in a 1-type, i.e. a set, which means that we only have to check the constructors η and τ. Tracing the explicit construction in Lemma 25 through the square, we can check directly that the square commutes in both cases (strictly speaking, in the case for τ, it is a cube):

$$\begin{array}{ccc} \eta(x) & \longmapsto & \eta(cx) \\ \downarrow & & \downarrow \\ |\vec{\mathsf{cons}}_x|(|\mathsf{unit}|) & \longmapsto & |\mathsf{cons}_c|(|\vec{\mathsf{cons}}_x|(|\mathsf{unit}|)) \end{array} \tag{29}$$

and:

$$\begin{array}{ccc} \tau(x, a, y) & \longmapsto & \tau(cx, a, y) \\ \downarrow & & \downarrow \\ \mathsf{ap}_{|\vec{\mathsf{cons}}(x)|}(|\mu_2|(a, y)) & \longmapsto & \mathsf{ap}_{|\mathsf{cons}_c|(|\vec{\mathsf{cons}}(x)|)}(|\mu_2|(a, y)) \end{array} \tag{30}$$

The commutativity is judgmental in the first square, and the second square only uses the usual equality $\mathsf{ap}_g \circ \mathsf{ap}_f = \mathsf{ap}_{g \circ f}$. □

This finally allows us to show:

Theorem 3. *If A is a set, then all fundamental groups of $\mathbf{F}(A)$ are trivial. In other words, $\|\mathbf{F}(A)\|_1$ is a set.*

Proof. By the previous lemma, the composition of the maps in (27) is an $\mathbf{F}(A)$-algebra map. But so is the map $|-| : \mathbf{F}(A) \to \|\mathbf{F}(A)\|_1$ by Lemma 24. Since $\mathbf{F}(A)$ is the *initial* such algebra, these two functions must coincide, which means that $|-| : \mathbf{F}(A) \to \|\mathbf{F}(A)\|_1$ factors through $\mathbf{N}(A)$. We know from Lemma 22 that $\mathbf{N}(A)$ is a set. This implies that $\|\mathbf{F}(A)\|_1$ is a set, which is the second part of the theorem.

To see the first part, take $q : \mathbf{F}(A)$. $\mathbf{F}(A)$ having trivial fundamental groups means that $\left\| q =_{\mathbf{F}(A)} q \right\|_0$ is contractible. By [22], we have

$$\left\| q =_{\mathbf{F}(A)} q \right\|_0 \;\simeq\; \left(|q| =_{\|\mathbf{F}(A)\|_1} |q| \right). \tag{31}$$

The second type is contractible since $\|\mathbf{F}(A)\|_1$ is a set. □

Having proved the main result, we add two results that now have become very easy:

Lemma 27. *For a set A, the two approximations of the free higher group which we have considered are equivalent, i.e. $\|\mathbf{F}(A)\|_1 \simeq \mathbf{N}(A)$.*

Proof. From the argument in the proof of the previous theorem, we can follow that $\|\mathbf{F}(A)\|_1$ is a retract of $\mathbf{N}(A)$. Thus, we still need to show that the composition $\mathbf{N}(A) \to \|\mathbf{F}(A)\|_1 \to \mathbf{N}(A)$ is the identity. But now that we know that everything is a set, it is easy to do this by induction on $\mathbf{N}(A)$. □

Theorem 28. *The type $\|\mathbf{F}(A)\|_1$ is equivalent to the purely set-based free group over A as constructed in [22, Chp 6.11]. If Question 2 can be answered positively, then our free group does indeed generalise the free group construction of [22] from only sets to arbitrary types.*

Proof. Since $\mathbf{N}(A)$ is a set by Lemma 22, it is easy to see that it is equivalent to the set-quotient of $\mathsf{List}(A^{\pm})$ by the relation that identifies a list with the list then one gets after reducing; this is essentially because, when we know that $\mathbf{N}(A)$ is a set, the constructors sw and ov become obsolete, and what remains is just this set-quotient. But this set-quotient is exactly the purely set-based free group of [22] by [22, Thm 6.11.7].

If Question 2 turns out to have a positive answer, then $\mathbf{F}(A)$ and $\|\mathbf{F}(A)\|_1$ are equivalent, and everything that holds for the latter is also true for the former. □

4 Conclusions

The central and guiding question of this paper was the problem of showing that the free ∞-group $\mathbf{F}(A)$ over a set is a set as well. We have proved a first approximation of this, namely that $\mathbf{F}(A)$ has trivial fundamental groups. This is done entirely in "book HoTT", the type theory developed in [22]. It would be very interesting to formalise the complete argument in a proof assistant, and we expect that this would be challenging. For example, the use of list concatenation in the constructors of the higher inductive type $\mathbf{N}(A)$ would lead to many application of *transport* (*substitution*). It is likely that a different representation of $\mathbf{N}(A)$ and the reduction relation would enable a more elegant formalisation. For a human reader, the presentation in terms of lists is the most intuitive and understandable one that we could think of.

Brunerie has discussed the James construction in homotopy type theory [9]. In this context, a type A with a point $\star_A : A$ is given, and the higher inductive type JA is defined to be the *free monoid* over A where $\star_A$ plays the role of the neutral element. Brunerie then constructs a non-recursive version of JA. Of special interest for him is the case that A is connected (i.e. $\|A\|_0$ is contractible), and in this case, JA becomes very similar to our free group. However, connectedness would be a very unnatural assumption in the present paper; in fact, since we are interested in the case that A is a set, our case of interest is orthogonal to Brunerie's. If A is not known to be connected, then, compared to our $\mathbf{F}(A)$, JA is lacking the condition that every cons_a is invertible, which is the main source of difficulty in our work.

Related to the current paper is also previous work by Capriotti, Vezzosi, and the current first author [13]. That work gives a necessary and sufficient condition for a function $X \to Y$ to factor through $\|X\|_n$, assuming that Y is $(n+1)$-truncated. In the current paper, we have been particularly interested in the situation that n is 0, X is the "level 0 approximation" of the free group (see the description after Definition 15), and Y is $\|\mathbf{F}(A)\|_1$. The reason why we have not directly applied the result of [13] is that, in our case of interest, showing the mentioned condition is tricky. This difficulty corresponds to what in our presentation has made the more refined approximation with the constructors sw and ov necessary. We do not know whether there is an alternative proof of our main result which uses [13] directly.

Let us further analyse the methods we have used in the paper. In principle, the strategy which we have developed should be applicable to more general results than the one we have proved; for example, with some more effort, we expect that it should be possible to show that $\|\mathbf{F}(A)\|_2$ and $\|\mathbf{F}(A)\|_3$ (which are better approximations to $\mathbf{F}(A)$) are sets. The obvious attempt to do this is to work with a "better" non-recursive approximation, i.e. a refined version of $\mathbf{N}(A)$ which would use higher path constructors to guarantee the coherence of sw and ov. One would then include a 2- or 3-truncation constructor instead of the 1-truncation constructor tr. It seems plausible that this could work; for example, instead of constructing a weakly constant function

$$\mathsf{Red}(x) \to \eta(x) = \eta(\mathsf{nil}) \tag{32}$$

as in Lemma 17, we would have to construct a constant function satisfying one or more coherence conditions [14], and the new constructors of $\mathbf{N}(A)$ would be chosen in such a way that this would be possible.

The additional value that such a generalisation would give us is unclear. Of course, what we want is to show that $\mathbf{F}(A)$ is a set, not just a finite truncation of it. If we try to use our approach, it seems we would need to find a way to encode the *whole infinite* tower of coherences in a non-recursive type, and it looks suspiciously similar to the long-standing open problem of defining semisimplicial types in HoTT [23]. (To clarify, we would not need a single HIT with infinitely many constructors, since we could take a sequential colimit; and the absence of a general version of Whitehead's principle does not seem to be a problem as long as we can show that (32) satisfies the coherence conditions given in [14], which does not rely on hypercompleteness either.)

Our problem of showing that $\mathbf{F}(A)$ is a set is not much different from the open problem of HoTT which asks whether the suspension of a set is a 1-type; as we have already discussed, what we are asking is essentially whether a the suspension of a set with a distinguished isolated point is a 1-type. Thus, our question is slightly weaker and, as far as we can see, an answer to the weaker question would not be sufficient to answer the more general question.[2] However, it seems plausible that an approach similar to ours is applicable to the more general question as well. In this case, being able to encode infinite towers of coherences could potentially be key to both open problems, although of course there would still be a lot of work to do (which might or might not even be impossible).

Theories such as Voevodsky's *homotopy type system* (HTS) [25], *two-level type theory* (2LTT) [3, 7] or *computational higher type theory* [6] are variations of standard HoTT in which such infinite structures can be constructed. We believe it would be worth investigating whether the "suspension of a set" problem can be resolved in such systems. Our preliminary investigations hint that it is at

[2] Related is the discussion *Does "adding a path" preserve truncation levels?* at https://groups.google.com/forum/#!topic/homotopytypetheory/gVmcvaOeD5c.

least be possible to define a "completely non-recursive" version of $\mathbf{F}(A)$, which would be a starting point. However, actually *using* this construction to mimic the proof that we have given in this paper is, of course, a completely different story.

If (we are now in the realm of complete speculation) it turns out that HTS can prove that the suspension of a set is a 1-type, it would be even more interesting whether "standard HoTT" can do it as well. This is because one would need to come up with a completely different argument in standard HoTT, and if it turns out that the open problem is independent of standard HoTT, hope for a conservativity result would be lost for all theories that are powerful enough to encode semisimplicial types. Recall that we have a conservativity result for 2LTT, due to Capriotti [11], which says that a fibrant type in 2LTT can only be inhabited if the corresponding type in HoTT (assuming it exists) is inhabited as well. This however only works for a version of 2LTT where we do *not* have semisimplicial types in the usual formulation (we only have semisimplicial types indexed over the pretype of strict natural numbers, but not over the type fibrant natural numbers). Thus, in this version of 2LTT, the sketched approach to solve the open problem regarding the suspension of a set would not work. This might be more than a coincidence.

Acknowledgments

We are very grateful to Paolo Capriotti for many discussions on the topic, and for several remarks which have influenced this paper. Most importantly, the construction of a free group using the constructors of the free monoid and conditions ensuring that cons_a is an equivalence is due to Paolo, as well as the decomposition shown in (1). It was also him who suggested using the statement of Lemma 26 to complete the main proof of the paper, which has led to a cleaner proof than if we had done it with a more direct argument. We further thank Rafaël Bocquet for discussions on free groups, and the anonymous reviewers whose comments have helped us to improve the presentation and readability of the paper.

References

[1] John Frank Adams. 1978. Infinite loop spaces. *Annals of Mathematics Studies* 90 (1978).

[2] Thorsten Altenkirch, Paolo Capriotti, Gabe Dijkstra, Nicolai Kraus, and Fredrik Nordvall Forsberg. 2018. Quotient inductive-inductive types. In *Foundations of Software Science and Computation Structures (FoSSaCS 2018)*, Christel Baier and Ugo Dal Lago (Eds.). Springer International Publishing, 293–310. https://doi.org/10.1007/978-3-319-89366-2_16

[3] Thorsten Altenkirch, Paolo Capriotti, and Nicolai Kraus. 2016. Extending Homotopy Type Theory with Strict Equality. In *25th EACSL Annual Conference on Computer Science Logic (CSL 2016)*, Jean-Marc Talbot and Laurent Regnier (Eds.), Vol. 62. Dagstuhl, Germany, 21:1–21:17. https://doi.org/10.4230/LIPIcs.CSL.2016.21

[4] Thorsten Altenkirch, Nils Anders Danielsson, and Nicolai Kraus. 2017. Partiality, Revisited. In *Foundations of Software Science and Computation Structures (FoSSaCS 2017)*, Javier Esparza and Andrzej S. Murawski (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 534–549. https://doi.org/10.1007/978-3-662-54458-7_31

[5] Thorsten Altenkirch and Ambrus Kaposi. 2016. Type Theory in Type Theory Using Quotient Inductive Types. *SIGPLAN Not.* 51, 1 (Jan. 2016), 18–29. https://doi.org/10.1145/2914770.2837638

[6] C. Angiuli, K.-B. Hou, and R. Harper. 2017. Computational Higher Type Theory III: Univalent Universes and Exact Equality. *ArXiv e-prints* (Dec. 2017). arXiv:cs.LO/1712.01800

[7] Danil Annenkov, Paolo Capriotti, and Nicolai Kraus. 2017. Two-Level Type Theory and Applications. *Arxiv e-prints* (2017). http://arxiv.org/abs/1705.03307

[8] Steve Awodey, Nicola Gambino, and Kristina Sojakova. 2012. Inductive Types in Homotopy Type Theory. In *Logic in Computer Science (LICS)*. IEEE Computer Society, Washington, DC, USA, 95–104. https://doi.org/10.1109/LICS.2012.21

[9] Guillaume Brunerie. 2017. The James construction and $\pi_4(\mathrm{S}^3)$ in homotopy type theory. *CoRR* (2017). http://arxiv.org/abs/1710.10307

[10] Ulrik Buchholtz, Floris van Doorn, and Egbert Rijke. 2018. Higher Groups in Homotopy Type Theory. (2018). https://arxiv.org/abs/1802.04315

[11] Paolo Capriotti. 2016. *Models of Type Theory with Strict Equality*. Ph.D. Dissertation. School of Computer Science, University of Nottingham, Nottingham, UK. Available online at https://arxiv.org/abs/1702.04912.

[12] Paolo Capriotti and Nicolai Kraus. 2017. Univalent Higher Categories via Complete Semi-Segal Types. *Proc. ACM Program. Lang.* 2, POPL, Article 44 (Dec. 2017), 29 pages. https://doi.org/10.1145/3158132

[13] Paolo Capriotti, Nicolai Kraus, and Andrea Vezzosi. 2015. Functions out of Higher Truncations. In *24th EACSL Annual Conference on Computer Science Logic (CSL) 2015) (Leibniz International Proceedings in Informatics (LIPIcs))*, Stephan Kreutzer (Ed.), Vol. 41. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 359–373. https://doi.org/10.4230/LIPIcs.CSL.2015.359

[14] Nicolai Kraus. 2015. The General Universal Property of the Propositional Truncation. In *20th International Conference on Types for Proofs and Programs (TYPES 2014) (Leibniz International Proceedings in Informatics (LIPIcs))*, Hugo Herbelin, Pierre Letouzey, and Matthieu Sozeau (Eds.), Vol. 39. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 111–145. https://doi.org/10.4230/LIPIcs.TYPES.2014.111

[15] Nicolai Kraus. 2015. *Truncation Levels in Homotopy Type Theory*. Ph.D. Dissertation. School of Computer Science, University of Nottingham, Nottingham, UK.

[16] Nicolai Kraus, Martín H. Escardó, Thierry Coquand, and Thorsten Altenkirch. 2017. Notions of Anonymous Existence in Martin-Löf Type Theory. *Logical Methods in Computer Science* Volume 13, Issue 1 (mar 2017). https://doi.org/10.23638/LMCS-13(1:15)2017 In the special issue of TLCA'13.

[17] Daniel R. Licata and Michael Shulman. 2013. Calculating the Fundamental Group of the Circle in Homotopy Type Theory. In *Proceedings of the 2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '13)*. IEEE Computer Society, Washington, DC, USA, 223–232. https://doi.org/10.1109/LICS.2013.28

[18] Peter LeFanu Lumsdaine. 2009. Weak omega-Categories from Intensional Type Theory. In *Typed Lambda Calculi and Applications (TLCA)*. Springer-Verlag, 172–187. https://doi.org/10.1007/978-3-642-02273-9_14

[19] Emily Riehl and Dominic Verity. 2016. Homotopy coherent adjunctions and the formal theory of monads. *Advances in Mathematics* 286 (2016), 802 – 888. https://doi.org/10.1016/j.aim.2015.09.011

[20] Kristina Sojakova. 2015. Higher Inductive Types As Homotopy-Initial Algebras. In *Principles of Programming Languages (POPL)*. ACM, New York, NY, USA, 31–42. https://doi.org/10.1145/2676726.2676983

[21] Jim Stasheff. 1963. Homotopy associativity of H-spaces. *Trans. Amer. Math. Soc.* 108 (1963), 275–292, 293–312. https://doi.org/10.2307/1993609

[22] The Univalent Foundations Program. 2013. *Homotopy Type Theory: Univalent Foundations of Mathematics*. http://homotopytypetheory.org/book/, Institute for Advanced Study.

[23] The Univalent Foundations Program. 2013. Semi-simplicial types. (2013). Wiki page of the Univalent Foundations project at the Institute for Advanced Studies, https://uf-ias-2012.wikispaces.com/Semi-simplicial+types.

[24] Benno van den Berg and Richard Garner. 2011. Types are Weak ω-Groupoids. *Proceedings of the London Mathematical Society* 102, 2 (2011), 370–394. https://doi.org/10.1112/plms/pdq026

[25] Vladimir Voevodsky. 2013. A simple type system with two identity types. (2013). Unpublished note.