Delooping generated groups in homotopy type theory

Camil Champin[1], Samuel Mimram[2], and Émile Oleon[2]

[1]École Normale Supérieure de Lyon
[2]LIX, CNRS, École polytechnique, Institut Polytechnique de Paris, 91120 Palaiseau, France

Homotopy type theory is a logical paradigm based on Martin-Löf type theory which allows performing geometric constructions and proofs in a synthetic way. Namely, in this setting, types can be interpreted as spaces (up to continuous deformation) and proofs as homotopy invariant constructions. In this context, the loop spaces of types with a distinguished element (more precisely, pointed connected groupoids), provide a natural representation of groups, what we call here *internal groups*. The construction which internalizes a given group is called *delooping*, because it provides a formal inverse to the loop space operator. As we recall in the article, this delooping operation has a concrete definition for any group G given by the type of *G-torsors*. Those are particular sets together with an action of G, which means that they come equipped with an endomorphism for every element of G. We show that, when a generating set is known for the group, we can construct a smaller representation of the type of *G-torsors*, using the fact that we only need automorphisms for the elements of the generating set. We thus obtain a more concise and practical definition of (internal) groups in homotopy type theory. The developments performed in the article have been formalized using the cubical version of the Agda proof assistant.

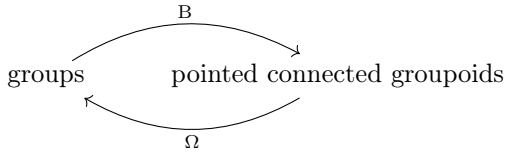
Introduction

Homotopy type theory was introduced around 2010 [14]. It is based on Martin-Löf type theory [9], starting from the idea that types in logic should not only be interpreted as sets, as traditionally done in semantics of logic, but rather as *spaces* (in the sense of topological spaces, for instance) considered up to deformation, or homotopy. Namely, the identities between two elements of a type can be thought of as paths between points corresponding to the elements, identities on identities as homotopies between paths, and so on. Moreover, this correspondence can be made to work precisely, by postulating the *univalence axiom* [7], which states that identities between types coincide with equivalences. This opens the way to implementing constructions of geometric nature in a synthetic way, by performing operations on types, which will semantically correspond to the desired operations on spaces. In this way, one can often perform fully mechanized proofs of properties of geometric nature, which generally provides new insights on those, and are invariant under homotopy by construction.

In this setting, we are interested in providing ways to construct models of groups which allow working with them in a synthetic way, and are concise in order to allow for proofs which are as small and simple as possible. Moreover, in order to show the practical applicability of our constructions, the results presented here have been formalized using the cubical variant of the Agda proof assistant [16] using the "standard library" which has been developed for it [13]: the corresponding developments are freely available [4].

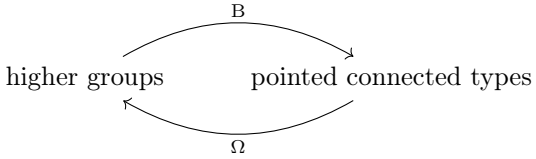
Delooping groups. Following a well-known construction from algebraic topology due to Poincaré at the end of the 19th century [11], to any type A which is pointed, i.e. equipped with a distinguished element $\star$, we can canonically associate a group $\pi_0(A) := \|\star = \star\|_0$, called the *fundamental group*, whose elements are homotopy classes of paths from $\star$ to itself, with composition given by concatenation and identities by constant paths. Moreover, when the type A is a groupoid, in the sense that any two homotopies between paths are homotopic, this fundamental group coincides with the *loop space* $\Omega\, A := (\star = \star)$, defined similarly to the fundamental group, but without quotienting paths up to homotopy. Once this observation made, it is natural to wonder whether every group G arises as the loop space of some groupoid. It turns out that this is the case, and it can be shown (again) using some classical notions of algebraic topology. Namely, one can consider *G-sets* which are sets equipped with an action of G. Among those, there is a canonical one, called the *principal G-torsor* P_G, which arises from the action of the group G on itself by left multiplication. It can be shown that the loop space of the type of G-sets, pointed on the principal G-torsor P_G, is the group G. Moreover, if one restricts the type of G-sets to the connected component of the principal G-torsor, one obtains the type of *G-torsors*, which is connected, pointed (by the principal torsor) and whose loop space is still G: such a type is called a *delooping* of G and often noted $\mathrm{B}\, G$.

Even if the above constructions might be hard to follow for now, and will hopefully be more clear when detailed in the article, the main point is that they induce an equivalence between the type of groups and the type of pointed connected groupoids (theorem 2.7.1):



This thus provides us with two alternative descriptions of groups in homotopy type theory. The one as (loop spaces of) pointed connected groupoids can be thought of as an *internal* one, since the structure is deduced from the types without imposing further axioms; by opposition, the traditional one, as groups, i.e. sets equipped with multiplication and unit operations satisfying suitable axioms, is rather an *external* one.

We should also say here that pointed connected types (which are not necessarily groupoids) can be thought of as higher versions of groups, where the axioms only hold up to higher identities which are themselves coherent, and so on. The idea is thus that the above correspondence should be the restriction of a more general correspondence



This can be shown by using meta-theoretic arguments [15], but not internally for now: no type-theoretic definition of higher groupoids is known for now (this is closely related to the open problem of defining semi-simplical types in homotopy type theory [6]).

Smaller deloopings of groups. In the view of the above correspondence, when we one is willing to work with a particular group G in a synthetic way, it is generally convenient to consider instead the associated pointed connected space $\mathrm{B}\, G$. However, the description obtained by directly using the above definition (the connected component of the principal G-torsor) is sometimes difficult to work with, and simpler descriptions of $\mathrm{B}\, G$ can often be obtained.

As an illuminating example, consider the case $G = \mathbb{Z}$, whose delooping is known to be the circle $\mathrm{B}\,\mathbb{Z} = \mathrm{S}^1$. The type $\mathcal{U}^{\circlearrowleft}$ of all endomorphisms, on any type, contains, as a particular element, the successor function $s : \mathbb{Z} \to \mathbb{Z}$. It can be shown that the connected component of s in $\mathcal{U}^{\circlearrowleft}$ is a delooping of $\mathbb{Z}$. And, this description is arguably simpler than the one of $\mathbb{Z}$-torsors. A slight variant of this observation (by considering the type of automorphisms instead of the type of endomorphisms) was for instance used to construct the circle in UniMath [2]. The reason why the above construction is simpler than the general one can be explained as follows: morphisms of $\mathbb{Z}$-sets are required to preserve the action of every element of $\mathbb{Z}$, while morphisms in $\mathcal{U}^{\circlearrowleft}$ are only required to preserve the action of the successor (which corresponds to the element 1 in the principal $\mathbb{Z}$-torsor). This suggests that, given a group G together with a set X of generators, one can restrict the construction of $\mathrm{B}\,G$ from G-sets to X-sets (i.e. we preserve the action of elements of X instead of every element of G) and still obtain a delooping of G. This is in fact the case and constitutes the main result of this article (theorem 5.2.2).

Formalization. The results presented in this article have been formalized in Agda [4]. Our formalization has the advantage of providing a short and self-contained proof of the classical constructions and results: G-sets and their equivalences, the principal G-torsor P_G, the main equivalence $\Omega\,\mathrm{B}\,G = G$. It also formalizes the above mentioned construction of a smaller delooping for groups equipped with a set of generators, which is new to our knowledge.

Related work. The construction of G-torsors is also developed in homotopy type theory in [1] and formalized in the `agda-unimath` library, see [12, `group-theory.torsors`]. The construction of delooping of groups can also be performed by using higher inductive types (HITs). In fact, those can be obtained as special cases of the construction of Eilenberg-Mac Lane spaces, which was done by Licata and Finster with HITs [8]: we namely have $\mathrm{B}\,G = \mathrm{K}(G, 1)$. Delooping pointed types (as opposed to groups), along with delooping associated morphisms (which shows that the delooping operation B is functorial) is discussed in [17].

Plan of the paper. We begin by briefly recalling the fundamental notions of homotopy type theory which will be used throughout the paper in section 1. We then introduce the notion of delooping of a group (section 2), of generating set for a group (section 3), and of group action (section 4), in the context of homotopy type theory. It is well-known that these notions allow constructing a delooping of any group G as the connected component of the principal torsor associated to G. We show in section 5 that we can obtain a delooping which is smaller, and thus easier to manipulate, when a generating set is known for G: the main result of this article is theorem 5.2.2. Finally, we conclude and present some future directions for this work in section 6.

1 Homotopy type theory

We unfortunately do not have enough space here to provide an introduction from scratch to dependent type theory and homotopy type theory, and refer the reader to the reference book for an in depth presentation [14]. The main purpose of this section is to fix some terminology and notations for classical notions.

1.1 Universe. We write $\mathcal{U}$ for the *universe*, i.e. the large type of all small types (this is often denoted `Type` in proof assistants), which we suppose to be closed under Σ-types (dependent sums) and Π-types (dependent products). We often write $(x : A) \to B$ for $\Pi(x : A).B$, and $A \to B$ for non-dependent Π-types (where x does not occur freely in B).

The two projections from a Σ-type are respectively written fst and snd, and we write $A \times B$ for non-dependent Σ-types.

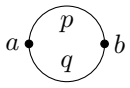
1.2 Paths. Given a type $A : \mathcal{U}$ and two elements $a, b : A$, we write $a =_A b$ (or $a = b$) for the type of *identities* (or *paths*) between a and b: its elements are proofs of equality between a and b. In particular, for every $a : A$, the type $a = a$ contains a particular term refl_a witnessing for reflexivity of equality. The induction principle for equality types states that, given a $a : A$ and a property $P : (x : A) \to (a = x) \to \mathcal{U}$, the canonical map

$$((x : A) \to (p : a = x) \to P\, x\, p) \to P\ \mathrm{refl}_a$$

given by evaluation at refl_a is an equivalence (see section 1.4 below). In particular, it admits an inverse, which encodes the induction principle for paths (aka *path induction*) and is commonly noted J: this induction principle means that, in order to show a property P depending on a path p whose target endpoint is free, it is enough to show it when the path is refl. By path induction, the following can be shown. Any path $p : A = B$ between two types $A, B : \mathcal{U}$ induces a function $p^{\rightarrow} : A \to B$, called the *transport* along p, as well as an inverse function $p^{\leftarrow} : B \to A$. Given a type A and a type family $B : A \to \mathcal{U}$, a path $p : x = y$ in A induces a function $p_B^{\rightarrow} : B(x) \to B(y)$ witnessing for the fact that equality is *substitutive*. Finally, given a function $f : A \to B$, any path $p : x = y$ in A induces a path $f^{=}(p) : f(x) = f(y)$ witnessing for the fact that equality is a *congruence*.

1.3 Higher inductive types. Many functional programming languages allow the definition of inductive types, which are freely generated by constructors. For instance, the type S^0 of booleans is generated by two elements (respectively corresponding to true and false). In the context of homotopy type theory, languages such as cubical Agda feature a useful generalization of such types, called *higher inductive types*, which allow, in addition to traditional constructors for elements of the type, constructors for equalities between elements of the type. For instance, the type corresponding to the circle S^1 can be defined as generated by two points a and b and two equalities between those points:

```
data S¹ : Type where
  a b : S¹
  p q : a ≡ b
```



Higher-dimensional spheres S^n can be defined in a similar way.

1.4 Univalence. A map $f : A \to B$ is an *equivalence* when it admits left and right inverses. In particular, every isomorphism is an equivalence. We write $A \simeq B$ the type of equivalences from A to B. The identity is clearly an equivalence and we thus have, by path induction, a canonical map $(A = B) \to (A \simeq B)$ for every types A and B. The *univalence axiom* states that this map is an equivalence. In particular, every equivalence $A \simeq B$ induces a path $A = B$. It is known, and non-trivial, that univalence implies the *function extensionality* principle [14, Section 2.9]: given functions $f, g : A \to B$, if $f(x) = g(x)$ for any $x : A$ then $f = g$ (and the expected generalization to dependent function types is also valid).

1.5 Homotopy levels. A type A is *contractible* when the type $\Sigma(x : A).(y : A) \to (x = y)$ is inhabited: this means that we have a "contraction point" $a_0 : A$, and a continuous family of paths from a_0 to every other point in A. A type A is a *proposition* (resp. a *set*, resp. a *groupoid*) when $(x = y)$ is contractible (resp. a proposition, resp. a set) for every $x, y : A$. Intuitively, a contractible type is a point (up to homotopy), a proposition is a point or is empty, a set is a collection of points and a groupoid is a space which bears no non-trivial 2-dimensional structure. For instance, consider the following three types corresponding to

spheres of respective dimensions -1, 0, 1 and 2:

They are respectively contractible, a set, a groupoid and not a groupoid (because there is a 2-dimensional "hole" in it). We write Set for the type of sets. Given a type A, we write $\mathrm{isSet}(A)$ for the predicate indicating that A is a set.

1.6 Truncation. Given a type A, its *propositional truncation* turns it into a proposition in a universal way. It consists of a type $\|A\|_{-1}$, which is a proposition, equipped with a map $\tau : A \to \|A\|_{-1}$ such that, for any proposition B, the canonical map

$$(\|A\|_{-1} \to B) \to (A \to B)$$

given by precomposition by τ, is an equivalence. Intuitively, the type $\|A\|_{-1}$ behaves like A, excepting that we do not have access to its individual elements: the elimination principle for propositional truncation states that, in order to construct an element of B from an element of $\|A\|_{-1}$, we can only assume that we have an element of A if B itself is a proposition. In practice, given $x : A$, we often write $|x|_{-1}$ instead of $\tau(x)$. The *set truncation* $\|A\|_0$ of a type A is defined similarly, as the universal way of turning A into a set.

1.7 Connected types. A type A is *connected* when the type $\|A\|_0$ is contractible: this expresses the fact that A has one connected component. For instance, S^{-1}, S^1 and S^2 are connected, but not S^0.

2 Delooping of groups

2.1 The external point of view. It is common knowledge that a *group* consists of a set A, together with an operation $m : A \to A \to A$ (the *multiplication*), an element $e : A$ (the *unit*), and an operation $i : A \to A$ (the *inverse*) such that multiplication is associative, admits e as unit, and $i(x)$ is the two-sided inverse of any element $x : A$. We write Group for the type of all groups:

$$\begin{aligned}\mathrm{Group} := \Sigma(A : \mathrm{Set}).(m : A \to A \to A) \times (e : A) \times (i : A \to A \to A) \times \\ ((x, y, z : A) \to m(m(x, y), z) = m(x, m(y, z)) \times \\ ((x : A) \to m(e, x) = m(x, e) = x) \times \\ ((x : A) \to m(i(x), x) = m(x, i(x)) = e)\end{aligned}$$

In the following, we use the traditional notations for groups: given two elements $x, y : G$, we simply write xy instead of $m(x, y)$, 1 instead of e, and x^{-1} instead of $i(x)$.

2.2 The internal point of view. A *pointed* type consists of a type A together with a distinguished element, often written $\star$ and sometimes left implicit. Given a pointed type $(A, \star)$, we define its *loop space* as the type of paths from $\star$ to itself:

$$\Omega\, A \quad := \quad (\star = \star)$$

The elements of this types are sometimes called *loops*. By path induction one can construct, for every two paths $p : a = b$ and $q : b = c$, a path in $a = c$ called their *concatenation* and written $p \cdot q$. Similarly, every path $p : a = b$, admits an *inverse* path $p^- : b = a$. When A is a pointed groupoid, $\Omega\, A$ is a set, and these operations canonically equip this set with a

structure of group, as detailed in [14, Section 2.1]. Writing a for the distinguished element of A, we sometimes write $\mathrm{Aut}\, a$ for this group, whose elements are the *automorphisms* of a. In particular, since type Set of sets is a groupoid [14, Theorem 7.1.11], any set A admits a group $\mathrm{Aut}\, A$ of automorphisms.

2.3 Delooping groups. A *delooping* of a group G is a pointed connected groupoid $\mathrm{B}\, G$ together with an identification $\mathrm{d}_G : \Omega\, \mathrm{B}\, G = G$. The notation is justified by the fact that deloopings are unique, see below. For instance, it is known that the fundamental group of the circle S^1 is $\mathbb{Z}$ [14, Section 8.1], and that S^1 is a groupoid, hence we have $\Omega\, \mathrm{S}^1 = \mathbb{Z}$, and thus S^1 is a delooping of $\mathbb{Z}$.

2.4 Delooping loop spaces. As a first but useful example, consider a pointed type A: we are going to construct a delooping of the loop space $\Omega\, A$. Note that A is "almost" a delooping of $\Omega\, A$, since we can take reflexivity as identity $\mathrm{d}_{\Omega\, A} : \Omega\, A = \Omega\, A$, excepting that A is (in general) not connected. In order to fix this, we can restrict to the connected component of A (which contains the distinguished element) in the following way.

We define the *connected component* of the pointed type A as the type of points which are merely connected to the distinguished point of A. This type is noted $\mathrm{Comp}\, A$ (or $\mathrm{Comp}(A, \star)$ when we want to specify the distinguished element $\star$). Formally,

$$\mathrm{Comp}\, A \quad := \quad \Sigma(x : A).\|\star = x\|_{-1}$$

This type is canonically pointed by $(\star, |\,\mathrm{refl}\,|_{-1})$. This construction deserves its name because it produces a connected space, whose geometry is the same as the original space around the distinguished point, as shown in the following two lemmas.

Lemma 2.4.1. The type $\mathrm{Comp}\, A$ is connected.

Proof. It can be shown that a type X is connected precisely when both $\|X\|_{-1}$ and $(x, y : X) \to \|x = y\|_{-1}$ are inhabited, i.e. when X merely has a point and any two points are merely equal [14, Exercise 7.6]. In our case, the type $\mathrm{Comp}\, A$ is pointed and thus $\|\mathrm{Comp}\, A\|_{-1}$ holds. Moreover, suppose that there are two points (x, p) and (y, q) in $\mathrm{Comp}\, A$ with $x, y : A$, $p : \|\star = x\|_{-1}$ and $q : \|\star = y\|_{-1}$. Our goal is to show that $\|(x, p) = (y, q)\|_{-1}$ holds, which is a proposition, so by elimination of propositional truncation, we can therefore assume that p (resp. q) has type $\star = x$ (resp. $\star = y$). Hence, we can construct a path $p^- \cdot q$ of type $x = y$, and therefore $(x, p) = (y, q)$ because the second components belong to a proposition by propositional truncation. We conclude that $\|(x, p) = (y, q)\|_{-1}$ and finally that $\mathrm{Comp}\, A$ is connected. □

Lemma 2.4.2. We have $\Omega\, \mathrm{Comp}\, A = \Omega\, A$.

Proof. We begin by showing that the type

$$\Sigma((x, t) : \mathrm{Comp}\, A).(\star = x) \tag{1}$$

is contractible. In order to do so, observe that we have the following equivalence of types:

$$\begin{aligned}\Sigma((x, t) : \mathrm{Comp}\, A).(\star = x) &\simeq \Sigma((x, t) : \Sigma(x : A).\|\star = x\|_{-1}).(\star = x)\\ &\simeq \Sigma(x : A).(\|\star = x\|_{-1}) \times (\star = x)\\ &\simeq \Sigma((x, p) : \Sigma(x : A).(\star = x)).\|\star = x\|_{-1}\end{aligned}$$

using classical associativity and commutativity properties of Σ-types. Moreover, the type $\Sigma(x : A).(\star = x)$ is contractible [14, Lemma 3.11.8], therefore the whole type on the last line is a proposition (as a sum of propositions over a proposition), and therefore also the original type (1). We write $\star'$ for the element $(\star, \|\,\mathrm{refl}_\star\,\|_{-1})$ of $\mathrm{Comp}\, A$. The type (1) is pointed by the canonical element $(\star', \mathrm{refl})$ and thus contractible as a pointed proposition.

We have a morphism

$$\begin{array}{rcccc} F : ((x,t) : \operatorname{Comp} A) & \to & (\star' = (x,t)) & \to & (\star = x) \\ (x,t) & & p & \mapsto & \operatorname{fst}^{=} p \end{array}$$

sending a path p to the path obtained by applying first projection. It canonically induces a morphism

$$\begin{array}{c} \Sigma((x,t) : \operatorname{Comp} A).(\star' = (x,t)) \to \Sigma((x,t) : \operatorname{Comp} A).(\star = x) \\ ((x,t),p) \mapsto ((x,t), \operatorname{fst}^{=} p \end{array}$$

between the corresponding total spaces. Since the left member is contractible (by [14, Lemma 3.11.8] again) and the right member is contractible (as shown above), this is an equivalence. By [14, Theorem 4.7.7], for every $x : \operatorname{Comp} A$, the fiber morphism Fx is also an equivalence. In particular, with x being $\star'$, we obtain $\Omega \operatorname{Comp} A \simeq \Omega A$ (as a type) and we can conclude by univalence. Note that the equivalence preserves the group structure so that the equality also holds in groups. □

As a direct corollary of the two above lemmas, we have:

Proposition 2.4.3. Given a pointed type A, $\operatorname{Comp} A$ is a delooping of ΩA.

Remark 1. Some people write $\operatorname{Aut} A$ for ΩA and the above proposition states that we have $\operatorname{B}\operatorname{Aut} A = \operatorname{Comp} A$. For this reason, the (confusing) notation $\operatorname{BAut} A$ is also found in the literature for $\operatorname{Comp} A$.

2.5 Formalization. Let us briefly present the formalization we preformed of the above notions [4], based on the cubical Agda library [13]. Given a type `A` with structure (e.g. a pointed type is a type with a distinguished element, a group is a type with the structure of a group, etc), we write `⟨ A ⟩` for the underlying type. For a pointed type `A`, the distinguished element is denoted `pt A`. We write `∥ A ∥₁` for the propositional truncation of a type `A` (the indices start at 0 instead of -2 in the library, thus the index `1` instead of `-1`). For simplicity, we omit mentioning universe levels.

Taking the connected component can be expressed as a function operating on pointed types as follows:

```
Comp : Pointed → Pointed
Comp A = (Σ ⟨ A ⟩ (λ x  → ∥ pt A ≡ x ∥₁) , (pt A , ∣ refl ∣₁))
```

We can then give a formal proof of lemma 2.4.2:

```
loopCompIsLoop : {A : Pointed} → Ω (Comp A) ≃∙ Ω A
loopCompIsLoop {A} = isoToEquiv e , refl
  where
  e : Iso (fst (Ω (Comp A))) (fst (Ω A))
  Iso.fun      e p = cong fst p
  Iso.inv      e p = ΣPathP (p , toPathP (isPropPropTrunc _ _))
  Iso.rightInv e p = refl
  Iso.leftInv  e p = isoFunInjective (
    equivToIso (invEquiv (Σ≡PropEquiv (λ _ → isPropPropTrunc)))) _ _ refl
```

Notice how we take an approach different from the above proof, explicitly exhibiting the functions and invertibility.

2.6 Functoriality of delooping. One of the main properties of this operation is that it is a local inverse to taking loop spaces in the following sense:

Proposition 2.6.1. Given pointed connected groupoids A and B and a group morphism $f : \Omega\, A \to \Omega\, B$, there is a unique pointed morphism $g : A \to B$ such that $\Omega\, g = f$.

Proof. By [17, Corollary 12], with $n = 0$, we have that the type $\Sigma(g : A \to B).\, \Omega\, g = f$ is equivalent to $(p, q : \star_A = \star_A) \to f(p \cdot q) = f(p) \cdot f(q)$. This type is a proposition because $\Omega\, B$ is a set (because B is a groupoid), and inhabited (because f is a morphism of groups), and thus contractible. □

As an immediate consequence of the above lemma, deloopings are unique:

Proposition 2.6.2. Given two deloopings $\mathrm{B}\, G$ and $\mathrm{B}'\, G$ of a group G, we have $\mathrm{B}\, G = \mathrm{B}'\, G$.

Given a group morphism $f : G \to H$ such that both G and H admit deloopings (and we will see in theorem 4.3.2 below that this is in fact the case for all groups), the *delooping* of f is the morphism

$$\mathrm{B}\, f : \mathrm{B}\, G \to \mathrm{B}\, H$$

associated, by proposition 2.6.1, to the morphism $\mathrm{d}^{\leftarrow}_H \circ f \circ \mathrm{d}^{\rightarrow}_G : \Omega\, \mathrm{B}\, G \to \Omega\, \mathrm{B}\, H$. By proposition 2.6.1, this operation is functorial in the sense that that it preserves identities and composition.

2.7 Equivalence between the two points of view. Although this will not be central in this article, we shall mention here the fundamental equivalence provided by the above constructions. Details can be found in [1]. We write IntGroup for the type of internal groups, i.e. pointed connected groupoids.

Theorem 2.7.1. The maps Ω : IntGroup $\to$ Group and B : Group $\to$ IntGroup form an equivalence of types.

Proof. Given a group G, we have $\Omega\, \mathrm{B}\, G = G$ by definition of $\mathrm{B}\, G$. Given an internal group A, we have $\mathrm{B}\, \Omega\, A \simeq A$ by proposition 2.6.2. □

That is, looping and delooping operators allow us to go back and forth between the external and the internal point of view of group theory in homotopy type theory.

3 Generating groups

3.1 Free groups. Given a set X, we write X^* for the *free group* over X: its elements are words of the form $x_1^{\epsilon_1} x_2^{\epsilon_2} \dots x_n^{\epsilon_n}$ with $x_i \in X$ and $\epsilon_i \in \{-, +\}$ considered up to the congruence identifying $x^- x^+$ and $x^+ x^-$ with the empty word for every $x \in X$. Of course, the element x^+ (resp. x^-), corresponds to an element of X (resp. its inverse) in the freely generated group, and we write $\eta : X \to X^*$ for the canonical inclusion sending an element x of X to x^+. In the context of homotopy type theory, the classical universal property of this group can be reformulated as follows:

Proposition 3.1.1. Given a group G, precomposition by η induces a function from morphisms of groups $X^* \to G$ to morphisms of sets $X \to G$ (sending $g : X^* \to G$ to $g \circ \eta$). This function is an equivalence.

The function of the above proposition being an equivalence, it admits an inverse: given a function $f : X \to G$, we write $f^* : X^* \to G$ for the function it induces. This function is determined by the fact that it satisfies

$$f^*(x^+) = f(x) \qquad f^*(x^-) = f(x)^{-1} \qquad f^*(xy) = f^*(x) f^*(y) \qquad f^*(1) = 1$$

3.2 Generation. Traditionally, a subset $X \subseteq G$ of a group is said to generate the group G when every element $u \in G$ admits a decomposition as a product of generators $u = x_1^{\epsilon_1} x_2^{\epsilon_2} \dots x_n^{\epsilon_n}$. This amounts to say that, writing $f : X \to G$ for the inclusion, the map $f^* : X^* \to G$ is surjective. Here, a function $f : A \to B$ is said to be *surjective* when every element of B merely admits a preimage, i.e. the type

$$(b : B) \to \|\Sigma(a : A).f(a) = b\|_{-1}$$

is inhabited.

Remark 3.2.1. Another possible way of formulating the fact that X generates G could be to require that the function $f : X \to G$ admits a section, i.e. that there exists a function $g : B \to A$ such that $f \circ g = \mathrm{id}_G$. This notion is stronger (in the sense that is always implies the former) and thus less general. We will not use it in the following because we only need the "weaker" version of generation.

3.3 Formalization. In the standard library of cubical Agda [13], the notion of free group is naturally implemented as a higher inductive type:

```
data FreeGroup (X : Type) : Type where
  η     : X → FreeGroup X
  _·_   : FreeGroup X → FreeGroup X → FreeGroup X
  ε     : FreeGroup X
  inv   : FreeGroup X → FreeGroup X
  assoc : ∀ x y z → x · (y · z) ≡ (x · y) · z
  idr   : ∀ x → x ≡ x · ε
  idl   : ∀ x → x ≡ ε · x
  invr  : ∀ x → x · (inv x) ≡ ε
  invl  : ∀ x → (inv x) · x ≡ ε
  trunc : isSet (FreeGroup X)
```

This type is then proven to be a group through a function called `freeGroupGroup`. It also has a recursor

```
rec : {X : Type} {G : Group} →
      (X → ⟨ G ⟩) → GroupHom (freeGroupGroup X) Group
```

which is a translation of proposition 3.1.1, and formalizes the existence of a universal group homomorphism extending any function. The predicate of generation can then be formalized as:

```
generates : {X : hSet} {G : Group} (ι : ⟨ X ⟩ → G ) → Type
generates ι = isSurj (rec ι)
```

the predicate isSurj above (surjectivity) being defined in the cubical library as:

```
isSurj : (f : GroupHom G H) → Type _
isSurj f = ∀ y → ∥ Σ ⟨ G ⟩ (λ x → ⟨ f ⟩ x ≡ y) ∥₁
```

where `⟨ f ⟩` is the underlying function of the morphism `f`.

4 Group actions

4.1 Group actions. Given a group G and a set A, an *action* of G on A is an (external) group morphism from G to $\mathrm{Aut}(A)$, that is a map $\alpha : G \to \mathrm{Aut}(A)$ together with proofs that

$$\alpha(xy) = \alpha(x) \circ \alpha(y) \qquad \alpha(1) = \mathrm{id}_A$$

for all $x, y : G$. Alternatively, by curryfication, an *action* of G on A can also be seen as a map $\alpha : G \times A \to A$ which is compatible with the multiplication and the unit of G, in the sense that for every $x, y : G$ and every $a : A$ we have

$$\alpha(xy, a) = \alpha(x, \alpha(y, a)) \qquad \alpha(1, a) = a$$

A *G-set* is a set equipped with an action of G, and we write Set_G for the resulting type. We often simply denote a G-set by the associated action α and write $\mathrm{dom}(\alpha)$ for the set on which G acts. Given G-sets α and β, a *homomorphism* between them consists of a function $f : \mathrm{dom}(\alpha) \to \mathrm{dom}(\beta)$ which preserves the group action, in the sense that, for every $x : G$ and $a : A$, we have

$$\beta(x, f(a)) = f(\alpha(x, a)).$$

A homomorphism which is also an equivalence is called an *isomorphism* and we write $\alpha \simeq^{\mathrm{Set}_G} \beta$ for the type of isomorphisms between α and β. We write $\mathrm{Aut}(\alpha)$ for the type of automorphisms $\alpha \simeq^{\mathrm{Set}_G} \alpha$, which is a group under composition.

4.2 Identities of G-sets. The equalities between G-sets can be conveniently characterized as follows.

Proposition 4.2.1. Given two G-sets (A, α) and (B, β) an equality between them consists of an equality $p : A = B$ such that the diagram

$$\begin{array}{ccc} A & \xrightarrow{p^{\rightarrow}} & B \\ {\scriptstyle \alpha(x)}\downarrow & & \downarrow{\scriptstyle \beta(x)} \\ A & \xrightarrow[p^{\rightarrow}]{} & B \end{array} \tag{2}$$

commutes for every $x \in G$, i.e. $\beta(x) \circ p^{\rightarrow} = p^{\rightarrow} \circ \alpha(x)$.

Proof. The characterization of equalities between Σ-types [14, Theorem 2.7.2] entails that an equality between (A, α) and (B, β) is a pair consisting of an equality $p : A = B$ and an equality $q : p^{\rightarrow}_{A \mapsto X \to \mathrm{Aut}(A)} \alpha = \beta$ (we can forget about the equality between the components expressing that the properties required for group actions are satisfied since those are propositions). By [14, Lemma 2.9.6] and function extensionality, we finally have that the type of q is equivalent to the type $\beta(x) \circ p^{\rightarrow} = p^{\rightarrow} \circ \alpha(x)$. □

In the above proposition, note that B is a set, so that the commutation of the above diagram (2) is a proposition.

It easily follows from this proposition that any equality between G-sets induces an isomorphism of G-sets, as customary for equalities between algebraic structures [14, Section 2.14]. In fact, this map from equalities to isomorphisms can itself be shown to be an equivalence: the two types are roughly the same:

Proposition 4.2.2. Given G-sets α and β, the following types are equivalent:

$$(\alpha = \beta) \simeq (\alpha \simeq^{\mathrm{Set}_G} \beta)$$

Moreover, given a G-set α, the induced equivalence

$$(\alpha = \alpha) \simeq (\alpha \simeq^{\mathrm{Set}_G} \alpha)$$

is compatible with the canonical group structures on both types.

Proof. A proof for this particular case can be found in our formalization [4]. This is actually an instance of a general fact of identities between algebraic structures, which is known under the name of *structure identity principle* and proved in [5] (it can be understood as a generalisation of univalence for types having an algebraic structure). □

4.3 Torsors. As said earlier, any group G has a canonical G-set called the *principal G-torsor* and noted P_G, where the corresponding action $G \times G \to G$ is given by the multiplication of the group. Moreover, its group of automorphisms is precisely the group G:

Proposition 4.3.1. Given a group G, we have an equality of groups $(\mathrm{P}_G \simeq^{\mathrm{Set}_G} \mathrm{P}_G) = G$.

Proof. Note that any element $f : \mathrm{Aut}\,\mathrm{P}_G$, i.e. morphism of actions $f : \mathrm{P}_G \to \mathrm{P}_G$, satisfies $xf(y) = f(xy)$ for any $x, y : G$. The two functions

$$\begin{aligned} \phi : \mathrm{Aut}\,\mathrm{P}_G &\to G & \qquad \psi : G &\to \mathrm{Aut}\,\mathrm{P}_G \\ f &\mapsto f(1) & \qquad x &\mapsto y \mapsto yx \end{aligned}$$

are group morphisms: given $f, g : \mathrm{Aut}\,\mathrm{P}_G$, we have

$$\phi(g \circ f) = g \circ f(1) = g(f(1)1) = f(1)g(1) = \phi(f)\phi(g) \qquad \phi(\mathrm{id}) = \mathrm{id}(1) = 1$$

and given $x, y : G$, we have for every $z : G$,

$$\psi(xy)(z) = z(xy) = (zx)y = \psi(y) \circ \psi(x)(z) \qquad \psi(1)(x) = x1 = \mathrm{id}(x)$$

Moreover, they are mutually inverse. Namely, given $f : \mathrm{Aut}\,\mathrm{P}_G$ and $x : G$, we have

$$\psi \circ \phi(f)(x) = xf(1) = f(x1) = f(x)$$

and, given $x : G$,

$$\phi \circ \psi(x) = 1x = x$$

We thus have $\mathrm{Aut}\,\mathrm{P}_G \simeq G$ and we conclude by univalence. □

As a direct corollary, we have a way to construct the delooping of any group:

Theorem 4.3.2. The connected component of P_G in Set_G, i.e. the type $\mathrm{Comp}(\mathrm{Set}_G, \mathrm{P}_G)$, is a delooping of G.

Proof. By proposition 2.4.3, we know that $\mathrm{Comp}(\mathrm{Set}_G, \mathrm{P}_G)$ is a delooping of $\Omega(\mathrm{Set}_G, \mathrm{P}_G)$, which is equivalent to $\mathrm{P}_G \simeq^{\mathrm{Set}_G} \mathrm{P}_G$ by proposition 4.2.2, and thus to G by proposition 4.3.1. □

The elements of the connected component of the principal G-set are usually called *G-torsors*.

4.4 Formalization. The notion of an action of a group on a set can be formalized as a record whose fields ensure that the type on which we act is a set, define the action itself and provide proofs that the required axioms hold:

```
record Action (G : Group) (A : Type) : Type where
  field
    is-set : isSet A
    _*_ : ⟨ G ⟩ → A → A
    ·Unit : (x : A) → 1 * a ≡ a
    ·Composition : (x y : ⟨ G ⟩) (a : A) → x * (y * a) ≡ (x · y) * a
```

A G-set is then a type equipped with an action:

```
GSet : Group → Type
GSet G = Σ Type (λ A → Action G A)
```

The characterization of equivalences is then a direct translation of proposition 4.2.2:

```
GSetUnivalence : {G : Group} {α β : GSet G} → (α ≡ β) ≃ (GSetEquiv α β)
```

where the type `GSetEquiv α β` of equivalences between two actions `α` and `β` is defined as a record, in a similar way as above.

We can formalize the fact that left multiplication induces an action of G on itself:

```
left-action : (G : Group) → Action G ⟨ G ⟩
left-action G = record {
  _*_ = _·_ ;
  is-set = is-set ;
  ·Unit = ·IdL ;
  ·Composition = ·Assoc
  }
```

where the fields are filled with proofs from the structure of `G`, from which we can define the principal G-set associated to a group

```
P : (G : Group) → GSet G
P G = ⟨ G ⟩ , gsetstr (left-action G)
```

as well as the delooping of a group

```
B : (G : Group) → Pointed
B G = Comp (GSet G , P G)
```

which can be shown to be a groupoid:

```
isGroupoidBG : (G : Group) → isGroupoid ⟨ B G ⟩
```

We write

```
π₁ : {A : Pointed} → isGroupoid ⟨ A ⟩ → Group
```

for the operation which computes the fundamental group of a groupoid (which coincides with its loop space). The characterization of loop spaces of G-torsors (proposition 4.3.1) can then be formalized as

```
PGloops : (G : Group) → GroupIso (π₁ (isGroupoidGSet G) (GSet G , P G)) G
```

and follows the proof given above. The formalization of theorem 4.3.2 then follows easily:

```
torsorsDeloops : (G : Group) →
  isConnected 1 (B G) × isGroupIso (π₁ (isGroupoidBG G) (B G)) G
```

4.5 Internal group actions. In a similar way that the traditional notion of group admits an internal reformulation (section 2.2), the notion of action also admits an internal counterpart which can be defined as follows. Given a group G, an *internal* action of G on a set A is a function

$$\alpha : \mathrm{B}\,G \to \mathrm{Set}$$

such that $f(\star) = A$. Since Set is a groupoid [14, Theorem 7.1.11], by theorem 2.7.1, we have equivalences of types

$$(\mathrm{B}\,G \to \mathrm{Set}) \quad \simeq \quad (\Omega\,\mathrm{B}\,G \to \Omega(\mathrm{Set}, A)) \quad \simeq \quad (G \to \mathrm{Aut}\,A)$$

which show that internal group actions correspond to external ones: the delooping operator internalizes an external group action, and the looping operator externalizes an internal group action.

4.6 Actions of sets. The preceding construction of an action of a group on a set can be generalized to one of a set on a set as follows. Given a type A, the type $\mathrm{End}(A)$ of *endomorphisms* of A is $A \to A$. An *action* of a set X on a set A is a morphism $X \to \mathrm{End}(A)$, i.e. a family of endomorphisms of A indexed by X. We write Set_X for the type

$$\mathrm{Set}_X \quad := \quad \Sigma(A : \mathrm{Set}).(X \to \mathrm{End}(A))$$

of all actions of X, an element of this type being called an *X-set*. The identities between X-sets can be characterize a similar was as for G-sets (proposition 4.2.1) excepting that (2) has to commute for every x in X (as opposed to every x in G). Proposition 4.2.2 also extends in the expected way.

Given a function $\iota : X \to G$, from a set X to a group G, which can be thought of as the inclusion of a set of generators into a group, we have a function from G-sets to X-sets induced by precomposition by ι:

$$\begin{aligned} U_\iota : \mathrm{Set}_G &\to \mathrm{Set}_X \\ (A, \alpha) &\mapsto (A, \alpha \circ \iota) \end{aligned}$$

It can be thought of as a forgetful functor from G-sets to X-sets.

5 Generated G-torsors

Given a group G, we have seen in the previous section that the connected component in G-sets of the principal G-torsor P_G is a delooping of G. When a generating set X is known for G, we show here that this construction can be simplified by taking the component of a restriction of P_G to X-sets, which often results in a much simpler construction of the delooping.

5.1 Applications. Before proving this theorem, which is formally stated as theorem 5.2.2 below, we shall first illustrate its use on concrete examples. Consider the cyclic group with n elements, $\mathbb{Z}_n$. We write $s : \mathbb{Z}_n \to \mathbb{Z}_n$ for the successor (modulo n) function, which is an isomorphism. By theorem 4.3.2, we know that the type

$$\Sigma(A : \mathrm{Set}_{\mathbb{Z}_n}).\| \mathrm{P}_{\mathbb{Z}_n} = A\|_{-1}$$

of $\mathbb{Z}_n$-torsors is a model of $B\mathbb{Z}_n$. This type is the connected component of $\mathrm{P}_{\mathbb{Z}_n}$ in the universe $\mathrm{Set}_{\mathbb{Z}_n}$ of sets with an action of the group $\mathbb{Z}_n$, i.e. sets A equipped with a morphism

$$\alpha : \mathbb{Z}_n \to \mathrm{Aut}\, A$$

Such a set A is thus equipped with one automorphism $\alpha(k)$ for every element $k : \mathbb{Z}_n$ (therefore k automorphisms in this case). However, one can observe that most of them are superfluous because $\alpha(1)$ generates all the other by composition, and the reason for this is that 1 generates all the elements of $\mathbb{Z}_n$ by addition. The useful data of a $\mathbb{Z}_n$-set thus boils down to a set A together with one automorphism $\alpha : \mathrm{Aut}\, A$ such that $\alpha^n = \mathrm{id}_A$.

Indeed, writing

$$\mathrm{Set}^{\circlearrowleft} \quad := \quad \Sigma(A : \mathrm{Set}).(A \to A)$$

for the type of all *endomorphisms* (of any set), our theorem will imply that the type

$$\Sigma((A, f) : \mathrm{Set}^{\circlearrowleft}).\|(\mathbb{Z}_n, s) = (A, f)\|_{-1} \tag{3}$$

(the connected component of the successor modulo n in the universe of set endomorphisms) is still a delooping of $\mathbb{Z}_n$. Note that we didn't assume that f is an isomorphism nor that it should verify $f^n = \mathrm{id}$. This is because both properties follow from the fact that f is in the

connected component of the successor (which verify these properties). Similarly, we do not need to explicitly assume the domain of the endomorphism is a set.

We can do this with every group for which a generating set is known (and, of course, the smaller the better). For instance, given a natural number n, the *dihedral group* D_n is the group of symmetries of a regular polygon with n sides. It is known that this group has $2n$ elements and is generated by two elements $\overline{s}$ and $\overline{r}$, respectively corresponding to an axial symmetry and a rotation by an angle $2\pi/n$. Our theorem will thus imply that the connected component of the triple $(D_n, \overline{s}, \overline{r})$ (the connected component of the symmetry and the rotation in the type of pairs of set endomorphisms), i.e.

$$\Sigma(A : \mathrm{Set}).\Sigma(f : A \to A).\Sigma(g : A \to A).\|(D, \overline{s}, \overline{r}) = (A, f, g)\|_{-1}$$

is a delooping of the dihedral group D_n.

5.2 The theorem. In this section, we suppose fixed a function $\iota : X \to G$ from a set X to a group G which is surjective; otherwise said, X generates G. We write $\mathrm{P}_X := U_\iota \mathrm{P}_G$ for the X-set which deserves the name of *principal X-torsor* as our theorem shows.

Proposition 5.2.1. We have an equivalence of groups

$$\Omega \mathrm{P}_G \simeq \Omega \mathrm{P}_X$$

Proof. From proposition 4.2.1, an element of ΩP_G consists of an equality $p : G = G$ in $\mathcal{U}$ such that

$$\mathrm{P}_G(x) \circ p^{\to} = p^{\to} \circ \mathrm{P}_G(x)$$

for every $x \in G$. By function extensionality, this is equivalent to requiring, for every $x, y \in G$ that

$$\mathrm{P}_G(x)(p^{\to}(y)) = p^{\to}(\mathrm{P}_G(x)(y))$$

i.e. by definition of the action P_G,

$$x(p^{\to}(y)) = p^{\to}(xy) \tag{4}$$

Note that the above equality is between elements of G, which is a set, and is thus a proposition. Similarly, an element of $\Omega \mathrm{P}_X$ consists of an equality $p : G = G$ in $\mathcal{U}$ satisfying

$$\iota(x)(p^{\to}(y)) = p^{\to}(\iota(x)y) \tag{5}$$

for every $x : X$ and $y : G$.

Clearly, any equality $p : G = G$ in $\Omega \mathrm{P}_G$ also belongs to $\Omega \mathrm{P}_X$ since the condition (5) is a particular case of (4). We thus have a function $\phi : \Omega \mathrm{P}_G \to \Omega \mathrm{P}_X$. Conversely, consider an element $p : G = G$ of $\Omega \mathrm{P}_X$, i.e. satisfying (5) for every $x : X$ and $y : G$. Our aim is to show that it belongs to $\Omega \mathrm{P}_G$. Given $x : X$ and $y : G$, we thus want to show that (4) holds. Since ι^* is surjective, we know that there merely exists an element u of X^* such that $\iota^*(u) = x$. Since (4) is a proposition, by the elimination principle of propositional truncation, we can actually suppose given u, and we have

$$\begin{aligned}
x(p^{\to}(y)) &= \iota^*(u)(p^{\to}(y)) && \text{since } \iota^*(u) = x \\
&= p^{\to}(\iota^*(u)y) && \text{by repeated application of (5)} \\
&= p^{\to}(xy) && \text{since } \iota^*(u) = x.
\end{aligned}$$

The second equality can be detailed as follows. Writing $u = x_1 \dots x_n$ with $x_i : X$, we have:

$$\begin{aligned}
\iota^*(u)(p^{\rightarrow}(y)) &= \iota^*(x_1 \dots x_n)(p^{\rightarrow}(y)) && \text{by definition of } u \\
&= \iota^*(x_1 \dots x_{n-1})\iota(x_n)(p^{\rightarrow}(y)) && \iota^* \text{ is a group morphism extending } \iota \\
&= \iota^*(x_1 \dots x_{n-1})(p^{\rightarrow}(\iota(x_n)y)) && \text{by (5)} \\
&= \iota^*(x_1 \dots x_{n-2})\iota(x_{n-1})(p^{\rightarrow}(\iota(x_n)y)) && \iota^* \text{ is a group morphism extending } \iota \\
&= \iota^*(x_1 \dots x_{n-2})(p^{\rightarrow}(\iota(x_{n-1})\iota(x_n)y)) && \text{by (5)} \\
&= \iota^*(x_1 \dots x_{n-2})(p^{\rightarrow}(\iota^*(x_{n-1}x_n)y)) && \iota^* \text{ is a group morphism extending } \iota \\
&\vdots \\
&= \iota^*(1)(p^{\rightarrow}(\iota^*(x_1 \dots x_n)y)) \\
&= 1(p^{\rightarrow}(\iota^*(x_1 \dots x_n)y)) && \iota^* \text{ is a group morphism} \\
&= p^{\rightarrow}(\iota^*(x_1 \dots x_n)y) && 1 \text{ is a unit} \\
&= p^{\rightarrow}(\iota^*(u)y) && \text{by definition of } u
\end{aligned}$$

This thus induces a function $\psi : \Omega\,\mathrm{P}_X \to \Omega\,\mathrm{P}_G$. The functions ϕ and ψ clearly preserve the group structure (given by concatenation of paths) and are mutually inverse of each other, hence we have the equivalence we wanted. □

Theorem 5.2.2. The type $\mathrm{Comp}\,\mathrm{P}_X$ is a delooping of G.

Proof. By proposition 2.4.3, we know that $\mathrm{Comp}\,\mathrm{P}_X$ is a delooping of $\Omega\,\mathrm{P}_X$, which is equal to $\Omega\,\mathrm{P}_G$ by proposition 5.2.1, and thus to G by theorem 4.3.2. □

The delooping of G constructed in previous theorem is the component of P_X in X-sets, i.e.

$$\Sigma(A : \mathcal{U}).\Sigma(S : \mathrm{isSet}\,A).\Sigma(f : X \to A \to A).\|\,\mathrm{P}_X = (A, S, f)\|_{-1}$$

Since for any type A, the $\mathrm{isSet}\,A$ is a proposition [14, Theorem 7.1.10], and the underlying type of P_X is a set, the underlying type of any X-set in the connected component of P_X will also be a set. As a consequence, the above type can slightly be simplified, by dropping the requirement that A should be a set:

Proposition 5.2.3. The type $\Sigma(A : \mathcal{U}).\Sigma(f : X \to A \to A).\|\,\mathrm{P}_X = (A, f)\|_{-1}$ is a delooping of G.

For instance, the delooping (3) of $\mathbb{Z}_n$ can slightly be simplified as

$$\Sigma((A, f) : \mathcal{U}^{\circlearrowleft}).\|(\mathbb{Z}_n, s) = (A, f)\|_{-1}$$

where

$$\mathcal{U}^{\circlearrowleft} \quad := \quad \Sigma(A : \mathcal{U}).A \to A$$

is the type of all endomorphisms of the universe.

6 Future works

We have presented our formalization in cubical Agda of the classical construction of groups deloopings in homotopy type theory based on torsors, and have shown that this construction can be simplified in practice when a generating set is known for the group.

This work is part of a larger investigation of "efficient" models of groups deloopings, in the sense that we can compute effectively with those. In particular, the construction of the infinite real projective space performed by Buchholtz and Rijke [3], provides a cellular description of $\mathrm{B}\,\mathbb{Z}_2$ (which is better than the usual ones obtained by generic methods such as torsors or HITs). We plan to generalize their construction in order to construct lens spaces

and thus obtain a cellular version of $\mathrm{B}\,\mathbb{Z}_n$ for every natural number n, as well as efficient representations of deloopings of other classical groups [10]. More generally, the formalization of group theory in univalent foundations is still under heavy development [1], and we aim at developing general techniques to construct efficient representations of (internal) groups in homotopy type theory, which would open the way to cohomological computations or the definition of group actions on higher types (as a generalization of group actions on sets). Also, we believe that the theory of G-torsors (and its simplifications when we have a generating set X) can be generalized to the setting of double deloopings of groups; we should be able to define some notion of 2-G-torsors (and 2-X-torsors) in order to have a model for the double delooping of G, $\mathrm{B}^2\,G$.

References

[1] Marc Bezem, Ulrik Buchholtz, Pierre Cagne, Bjørn Ian Dundas, and Daniel R. Grayson. Symmetry. URL: `https://github.com/UniMath/SymmetryBook`.

[2] Marc Bezem, Ulrik Buchholtz, Daniel R Grayson, and Michael Shulman. Construction of the circle in UniMath. *Journal of Pure and Applied Algebra*, 225(10):106687, 2021. `doi:10.1016/j.jpaa.2021.106687`.

[3] Ulrik Buchholtz and Egbert Rijke. The real projective spaces in homotopy type theory. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–8. IEEE, 2017. `arXiv:1704.05770`, `doi:10.5555/3329995.3330081`.

[4] Camil Champin. Delooping generated groups. URL: `https://github.com/camilchp/generated-deloopings`.

[5] Martín Hötzel Escardó. Introduction to univalent foundations of mathematics with Agda. Preprint, 2019. `arXiv:1911.00580`.

[6] Hugo Herbelin. A dependently-typed construction of semi-simplicial types. *Mathematical Structures in Computer Science*, 25(5):1116–1131, 2015. `doi:10.1017/S0960129514000528`.

[7] Krzysztof Kapulkin and Peter LeFanu Lumsdaine. The simplicial model of Univalent Foundations (after Voevodsky). *Journal of the European Mathematical Society*, 23(6):2071–2126, 2021. `arXiv:1211.2851`, `doi:10.4171/jems/1050`.

[8] Daniel R Licata and Eric Finster. Eilenberg-MacLane spaces in homotopy type theory. In *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–9, 2014. `doi:10.1145/2603088.2603153`.

[9] Per Martin-Löf. *Intuitionistic type theory*, volume 9. Bibliopolis Naples, 1984.

[10] Émile Oleon and Samuel Mimram. Delooping cyclic groups with lens spaces in homotopy type theory. In preparation, 2023.

[11] Henri Poincaré. *Analysis situs*. Gauthier-Villars Paris, France, 1895.

[12] Egbert Rijke, Elisabeth Bonnevier, Jonathan Prieto-Cubides, Fredrik Bakke, and others. The agda-unimath library. URL: `https://github.com/UniMath/agda-unimath/`.

[13] The Agda Community. Cubical Agda Library, July 2023. URL: `https://github.com/agda/cubical`.

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

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

[16] Andrea Vezzosi, Anders Mörtberg, and Andreas Abel. Cubical Agda: A dependently typed programming language with univalence and higher inductive types. *Journal of Functional Programming*, 31, 2021. `doi:10.1145/3341691`.

[17] David Wärn. Eilenberg-MacLane spaces and stabilisation in homotopy type theory. Preprint, 2023. `arXiv:2301.03685`.