Strictly Associative Group Theory using Univalence

Alex Rice

March 26, 2023

Often when proving basic properties about groups, we end up proving equations using equational reasoning. Unfortunately when formalising this in a proof assistant, most steps of the proof are for managing brackets and identities around our expression, which is annoying to work with and obscures the actual meaningful steps of the proof.

The usual approaches to these problems involve either putting up with the problem, using a variety of tactics to make the manipulations less painful, or having an array of helper lemmas. None of these methods mirror the approach used in traditional mathematics where it would be very rare to see a bracket written out in a proof.

As an example consider the following (cubical) Agda proof that proves that the inverse of any element in some group G is unique:

```
InvUniqueLeft : ∀ {ℓ} (𝒢 : Group ℓ) → Type ℓ
InvUniqueLeft 𝒢 = ∀ g h → h · g ≡ 1g → h ≡ inv g
  where
    open GroupStr (𝒢 .snd)

inv-unique-left : ∀ {ℓ} (𝒢 : Group ℓ) → InvUniqueLeft 𝒢
inv-unique-left 𝒢 g h p =
  h               ≡⟨ sym (·IdR h) ⟩
  h · 1g          ≡⟨ cong (h ·_) (sym (·InvR g)) ⟩
  h · (g · inv g) ≡⟨ ·Assoc h g (inv g) ⟩
  (h · g) · inv g ≡⟨ cong (_· inv g) p ⟩
  1g · inv g      ≡⟨ ·IdL (inv g) ⟩
  inv g    □
  where
    open GroupStr (𝒢 .snd)
```

Most of the proof above consists of bureaucracy of moving around brackets and introducing and deleting identities.

In this work, we present an alternative solution. For every group $\mathcal{G}$ we are able to find an isomorphic group for which associativity and unitality hold judgmentally. This leverages Cayley's Theorem, which states that every group is isomorphic to a subgroup of a permutation group. By representing permutations as functions, we are able to ensure that associativity and unitality hold "on the nose". I will discuss how this can be done and some of the challenges that one faces.

This then allows us to use univalence to obtain that any group is not only isomorphic but actually equal to a group with strict associativity and unitality, which allows us to prove theorems about groups by only proving the theorems for the strictified group. As an example the code above can be reduced to the following:

```
inv-unique-left-strict : ∀ {ℓ} (𝒢 : Group ℓ) → InvUniqueLeft 𝒢
inv-unique-left-strict 𝒢 = strictify InvUniqueLeft
  λ g h p → begin
    h ∘⌊⌋                ≈˘⌊ ·InvR′ g ⌋
    ⌊ h ∘ g ⌋∘ g ⁻¹  ≈⌊ p ⌋
    g ⁻¹     ∎′
  where
    open import Groups.Reasoning 𝒢
```

Here we no longer need to explicitly introduce identities or move brackets around. The simplicity of what remains also allows us to give equational reasoning tools similar to those in the Agda standard library, while also removing the need to use the cong function (note that the syntax is slightly different to avoid naming clashes). The function strictify above transports the proof in the strict group back to a proof for an arbitrary group.

In the strictified group the following equations hold definitionally:

- $a(bc) = (ab)c$,
- $a1 = a = 1a$,
- $a^{-1^{-1}} = a$,
- and $(fg)^{-1} = g^{-1} \cdot f^{-1}$.

We believe this allows proofs about groups to be written in a more natural way and that this work contains a novel way to use univalence to simplifying standard mathematical proofs. The full code for this development can be found at `https://github.com/alexarice/GroupsUF`. Although it is written in cubical Agda, it does not rely on the computational properties of cubical, and should work in any proof assistant with Univalence.