Something about commutativity and well-orders

Anonymous

January 20, 2024

In this talk, we study free monoids, free commutative monoids, and their connections with sorting and well-orders, using univalent type theory, implemented in Cubical Agda.

Background

First, we review the basics of universal algebra, free algebras and their universal property. We write Set for the category of hSets and functions. A signature σ is given by a type of operations with an arity function: $(\mathsf{op} : \mathsf{Set}) \times (\mathsf{ar}\colon \mathsf{op} \to \mathsf{Set})$. This gives a signature endofunctor $F_\sigma(X) :\equiv \sum_{(f\colon \mathsf{op})} X^{\mathsf{ar}(f)}$ on Set. A σ-structure $\mathfrak{X}$ is an F_σ-algebra: $(X : \mathsf{Set}) \times (\alpha_X\colon F_\sigma(X) \to X)$, with carrier set X, and a morphism of σ-structures is a F_σ-algebra morphism, giving the category of σ-algebras $\sigma\text{-}\mathsf{Alg}$.

The forgetful functor $U_\sigma\colon \sigma\text{-}\mathsf{Alg}$ to Set admits a left adjoint, giving the free σ-algebra construction on a carrier set. As is standard, this construction is given by an inductive type of trees $\mathsf{Tr}_\sigma(V)$, generated by two constructors, $\mathsf{leaf}\colon V \to \mathsf{Tr}_\sigma(V)$ and $\mathsf{node}\colon \Sigma_\sigma(\mathsf{Tr}_\sigma(V)) \to \mathsf{Tr}_\sigma(V)$. $\mathsf{Tr}_\sigma(V)$ is canonically a σ-algebra $\mathfrak{T}(V) = (\mathsf{Tr}_\sigma(V), \mathsf{node})$, with the universal map $\eta_V : V \to \mathsf{Tr}_\sigma(V)$ given by leaf. The universal property states that, given any σ-structure $\mathfrak{X}$, composition with η_V is an equivalence: $(-) \circ \eta_V\colon \sigma\text{-}\mathsf{Alg}(\mathfrak{T}(V), \mathfrak{X}) \xrightarrow{\sim} (V \to X)$. The inverse to this map is the extension operation $(-)^\sharp$, which extends a map $f\colon V \to X$ to a homomorphism $f^\sharp\colon \mathfrak{T}(V) \to \mathfrak{X}$.

An equational signature ε is given by a type of equations with an arity of free variables: $(\mathsf{eq} : \mathsf{Set}) \times (\mathsf{fv}\colon \mathsf{eq} \to \mathsf{Set})$. A system of equations (or a theory T) over (σ, ε) is given by a pair of trees on the set of free variables, for each equation: $\mathscr{l}, \mathscr{r}\colon (e : \mathsf{eq}) \to \mathsf{Tr}_\sigma(\mathsf{fv}(e))$. A σ-structure $\mathfrak{X}$ satisfies T, written $\mathfrak{X} \vDash T$, if, for each equation $e : \mathsf{eq}$ and $\rho\colon \mathsf{fv}(e) \to X$, $\rho^\sharp(\mathscr{l}(e)) = \rho^\sharp(\mathscr{r}(e))$. The full subcategory of $\sigma\text{-}\mathsf{Alg}$ given by σ-structures satisfying T is the variety of T-algebras in Set. Similarly, the forgetful functor to Set admits a left adjoint, which is classically constructed by quotienting the free σ-algebra by the congruence relation generated by T. However, we do not give the general construction for it, since it requires non-constructive principles [**blass**], and instead consider the varieties of monoids and commutative monoids.

Monoids and commutativity

The signature for monoids σ_{Mon} is given by two operations (unit and multiplication) of arity 0 and 2, respectively, written as $(\mathsf{Fin}(2), \{0 \mapsto \mathsf{Fin}(0); 1 \mapsto \mathsf{Fin}(2)\})$. The equational signature for monoids $\varepsilon_{\mathsf{Mon}}$ is given by three equations (left unit, right unit, associativity) of free variable arity 1, 1, and 3, respectively, written as $(\mathsf{Fin}(3), \{0 \mapsto \mathsf{Fin}(1); 1 \mapsto \mathsf{Fin}(1); 2 \mapsto \mathsf{Fin}(3)\})$. The theory of monoids T_{Mon} is given by the pairs of left and right trees, using the free variables for each equation. Commutative monoids are given by the same signature of operations, but additionally include the commutativity equation, which uses 2 free variables.

We study various constructions of free monoids and free commutative monoids, using HITs and quotients, and prove the universal property for each construction. We construct:

- $\mathsf{FreeMon}$ and $\mathsf{FreeCMon}$ HITs, given by generators for operations and higher generators for equations,
- List, SList, CList, given by cons-lists, cons-lists with adjacent swaps, cons-lists with a commutation relation, respectively.

Using quotients, we consider various commutativity relations on presentations of free monoids. Given a free monoid construction: $A \xrightarrow{\eta} \mathscr{L}(A)$, a commutativity relation is a binary relation $\approx$ on $\mathscr{L}(A)$ such that, $A \xrightarrow{\eta} \mathscr{L}(A) \xrightarrow{q} \mathscr{L}(A)/{\approx}$ is a free commutative monoid construction. From this we construct:

- PList, a quotient of List by various permutation relations,

- Bag, a quotient of $\mathsf{Array}(A) = (n : \mathbb{N}) \times (A^{\mathsf{Fin}(n)})$ by $(n, f) \sim (m, g) :\equiv (\sigma \colon \mathsf{Fin}(n) \simeq \mathsf{Fin}(m)) \times (f = g \circ \sigma)$.

Further, using the universal property we study various properties of these constructions:

- characterizations of the path spaces of each type,
- combinatorial properties, such as, $\mathscr{L}(A + B) \simeq \mathscr{L}(A) + \mathscr{L}(B)$, $\mathscr{M}(A + B) \simeq \mathscr{M}(A) \times \mathscr{M}(B)$,
- injectivity of $\mathsf{cons}_A(x, -)\colon \mathscr{L}(A) \to \mathscr{L}(A)$ and $\mathscr{M}(A) \to \mathscr{M}(A)$, for any $x : A$.

Total orders and Sorting

Finally, we use this framework to study sorting and total orders.

Definition 1. *Given a section $s : SList\ X \to List\ X$ to the canonical map $List\ X \to SList\ X$, xs is said to be sorted if xs is in the image of s.*

We define the proposition *is-sorted* $: List\ X \to \mathscr{U}$ to be $\lambda xs.\ \exists(ys : SList\ X).\ s(ys) = xs$. We also use the universal property of free monoid to define membership proofs for $List\ X$ and $SList\ X$. We do so by noting propositions form a commutative monoid under $\vee$, which allow us to define membership proof for an element x using the extension operation $(-)^\sharp$ by lifting the function $\lambda y.\ x = y$ from $X \to Prop$ to $List\ X \to Prop$ and $SList\ X \to Prop$ respectively.

Proposition 1. *A section $s : SList\ X \to List\ X$ to the canonical map from the free monoid to the free commutative monoid on set X implies a total order on set X iff $\forall\, x\ y\ xs.$ is-sorted$(x :: xs) \to y \in x :: xs \to$ is-sorted$([x, y])$.*

It is well known that a total order on set X would imply a sort function on *List* X. It should follow that a sort function on set X would imply a total order on X. We can formalize the notion of a sort function as a section to the canonical map from *List* to *SList*, which can be thought of as a function which picks a canonical representation from an unordered list, thereby sorting the list in the process. However, we cannot prove transitivity purely from s being a section, one example being a function $s : SList\ \mathbb{N} \to List\ \mathbb{N}$ which sorts ascendingly given an odd-lengthed *SList* and descendingly given an even-lengthed *SList*. Hence, a stronger assumption is needed to fully construct a total order.

Definition 2. *Given a section $s : SList\ X \to List\ X$ to the canonical map $List\ X \to SList\ X$, we define a relation $\leq$: if x is the head of $s(\{x, y\})$, $x \leq y$.*

We note that $x \leq y$ iff *is-sorted*$([x, y])$.

Lemma 1. *Given a section $s : SList\ X \to List\ X$ to the canonical map $List\ X \to SList\ X$, $s(\{x, y\})$ must either be $[x, y]$ or $[y, x]$.*

We note that the canonical map $q : List\ X \to SList\ X$ preserves length and preserves membership, $x \in xs$ iff $x \in q(xs)$. Since $q(s(\{x, y\})) = \{x, y\}$ by definition, $s(\{x, y\})$ must therefore have length 2, and $x, y \in s(\{x, y\})$. Let $q(\{x, y\})$ be $[u, v]$, we perform a proof by cases. For case $x = u, y = v$ and $x = v, y = u$ the proof is trivial. For case $x = u, y = u$ and $x = v, y = v$, we obtain a proof $x = y$. Since $s(\{x, x\}) = [u, v]$ by assumption, and $q([u, v]) = \{x, x\}$ by definition of s, $u, v \in \{x, x\}$, therefore u, v must equal to x. Since $u = v = x = y$, and $s(\{x, y\}) = [u, v]$, $s(\{x, y\}) = [x, y]$.

With this lemma we can then prove $\leq$ does indeed satisfy all axioms of total order.

Proposition 2. *$\leq$ is reflexive.*

By Lemma 1, $s(\{x, x\})$ must either be $[x, x]$ or $[x, x]$. Either case we obtain a proof that $s(\{x, x\}) = [x, x]$. Since x is the head of $[x, x]$, $x \leq x$.

Proposition 3. *$\leq$ is antisymmetric.*

Given $x \leq y$ and $y \leq x$, we want to show $x = y$. By Lemma 1, $s(\{x, y\})$ must either be $[x, y]$ or $[y, x]$. In the case $s(\{x, y\}) = [x, y]$, since $y \leq x$, y is the head of $[x, y]$, and we obtain a proof $y = x$. In the case $s(\{x, y\}) = [y, x]$, we invert x and y in the previous proof and obtain $x = y$. Either case we obtain $x = y$.

Proposition 4. *$\leq$ is total.*

We want to show for any x and y, either $x \leq y$ or $y \leq x$. By Lemma 1, $s(\{x, y\})$ must either be $[x, y]$ or $[y, x]$, therefore either x or y is the head of $s(\{x, y\})$. We obtain either $x \leq y$ or $y \leq x$.

Proposition 5. *$\leq$ is transitive.*

Given $x \leq y$ and $y \leq z$, we want to show $x \leq z$. We first note that the head of $s(\{x, y, z\})$ must either be x, y, or z. For case x, by assumption $[x, z]$ is sorted, therefore $x \leq z$. For case y, by assumption $[y, x]$ is sorted, therefore $y \leq x$, and since $x \leq y$ by assumption, by antisymmetry $x = y$, and by assumption $y \leq z$, therefore $x \leq z$. For case z, by assumption $[z, y]$ is sorted, therefore $z \leq y$, and since $y \leq z$ by assumption, by antisymmetry $y = z$, and by assumption $x \leq y$, therefore $x \leq z$.

Proposition 6. *A section $s : SList\ X \to List\ X$ to the canonical map from the free monoid to the free commutative monoid on set X satisfying $\forall\, x\, y\, xs.\ \textit{is-sorted}(x :: xs) \to y \in x :: xs \to \textit{is-sorted}([x, y])$ implies a total order on X.*

We show that $\leq$ satisfies all axioms of total order above.

Proposition 7. *A total order on X implies a section $s : SList\ X \to List\ X$ to the canonical map from the free monoid to the free commutative monoid on set X satisfying $\forall\, x\, y\, xs.\ \textit{is-sorted}(x :: xs) \to y \in x :: xs \to \textit{is-sorted}([x, y])$.*

We can define such a section as a sort function on *SList X*. We show that the sort function satifies the property as follow: since x is the minimal element in $x :: xs$, we prove by induction on xs, to obtain a proof that $x \leq y$, and use it to show $is - sorted([x, y])$.

With this we obtain a proof a section $s : SList\ X \to List\ X$ to the canonical map from the free monoid to the free commutative monoid on set X satisfying $\forall\, x\, y\, xs.\ \textit{is-sorted}(x :: xs) \to y \in x :: xs \to \textit{is-sorted}([x, y])$ iff there is a total order on X.

Conjecture 1. *SList and Bag are free symmetric monoidal groupoid*

Previous works already established *SList* and *Bag* are free commutative monoids when 0-truncated, and it is folklore that when 1-truncated they should be free symmetric monoidal groupoid, although this has not yet been shown internally in HoTT.

In the case of *SList*, higher path constructors are needed. To establish the coherence of *swap* we need a higher *hexagon* path constructor. However to define the *hexagon* constructor it would involve compositions of *swap* which leads to a regularity problem. To avoid this we need to split the *hexagon* equations as below:

```
hexagon- : (a b c : A) (cs : SList A)
-> a :: b :: c :: cs = c :: b :: a :: cs
hexagon↑ : (a b c : A) (cs : SList A)
-> Square (\i -> b :: swap a c cs i) (hexagon- a b c cs)
(swap b a (c :: cs)) (swap b c (a :: cs))
hexagon↓ : (a b c : A) (cs : SList A)
-> Square (hexagon- a b c cs) (swap a c (b :: cs))
(\i -> a :: swap b c cs i) (\i -> c :: swap b a cs i)
```

In the case of *Bag*, we first need to show that *Array* itself is a free monoidal groupoid. We then need to show a free monoidal groupoid quotiented by a permutation relation would be a free symmetric monoidal groupoid, and show that by quotienting *Array* with an isomorphism on the index, we would indeed get a free symmetric monoidal groupoid.