Extensional Finite Sets and Multisets in Type Theory *

Clemens Kupke, Fredrik Nordvall Forsberg, and Sean Watters

University of Strathclyde, UK
{clemens.kupke, fredrik.nordvall-forsberg, sean.watters}@strath.ac.uk

The type of lists is one of the most elementary inductive data types. It has been studied and used extensively by computer scientists and programmers for decades. Two conceptually similar structures are those of finite sets and multisets, which can be thought of as unordered analogues to lists. However, capturing unordered structures in a data type while maintaining desirable properties such as decidable equality and the correct equational theory is challenging.

The usual approach to formalise unordered structures in mathematics is to represent them as functions (with finite support): finite sets as $X \to 2$, and finite multisets as $X \to \mathbb{N}$, respectively. However, these representations do not enjoy decidable equality, even if the underlying type X does.

Meanwhile the approach taken in most programming languages is to pretend — one uses a list (or another ordered structure for efficiency) internally, but hides it and any invariants behind a layer of abstraction provided by an API. However, each set or multiset can then be represented by many different lists, meaning that the equational theory might not be correct. This is a problem in a dependently typed setting, where having equality as a first-class type allows us to distinguish between different representations of the same set.

In the setting of homotopy type theory (HoTT) [14], we can use higher inductive types (HITs) to define the identities on an inductive type simultaneously with its elements. This allows us to define a data type which enjoys both decidable equality and the right equational theory, as demonstrated by Choudhury and Fiore [3]. However, many proof assistants today do not support HITs; thus, the main question we set out to answer in this work is whether it is possible in ordinary dependent type theory to define data types of finite sets and multisets, which:

(i) have decidable equality iff the underlying set has decidable equality; and

(ii) satisfy the equational theories of finite sets and multisets.

For property (ii), we take as our success criteria the facts that the type of finite sets is the free idempotent commutative monoid [7], and that finite multisets are the free commutative monoid. Thus, we are really aiming to find data types for the free idempotent commutative monoid and free commutative monoid, which satisfy the above property (i). We accomplish this by restricting our attention to only those sets with decidable equality that can be totally ordered. We can then form a type of sorted lists over such a set. Provided we treat the existence of the ordering data carefully, this type turns out to give us exactly finite sets when the order is strict, and finite multisets when it is non-strict.

We show that our constructions satisfy universal properties, in the sense that they are left adjoints to forgetful functors — this is the standard way to state freeness in the language of category theory. However, note that the notion of freeness is with respect to e.g. totally ordered monoids, rather than all monoids. For proving the universal properties and for defining the categories involved, we need function extensionality. However we stress that the constructions themselves work in ordinary dependent type theory, without function extensionality.

*This is an extended abstract of a paper that was published at APLAS 2023 [10]. All results are formalised in Agda and are available at: https://www.seanwatters.uk/agda/fresh-lists/.

Fresh Lists Fresh lists, the key inductive data type of this work, were first introduced by C. Coquand to represent contexts in the simply typed lambda calculus [4]. The type of fresh lists is a parameterised data type similar to the type of ordinary lists, with the additional requirement that in order to adjoin a new element x to a list xs, that element x must be "fresh" with respect to all other elements already present in the list xs. We follow the Agda standard library [1] in considering a generalised notion of freshness, given by an arbitrary binary relation on the carrier set. We can recover Coquand's original notion of freshness by choosing inequality as our freshness relation.

Finite Sets as Sorted Lists Our candidate representation for finite sets satisfying the above properties (i) and (ii) is the type of sorted lists without duplicates. We obtain this by the appropriate instantiation of the type of fresh lists; namely, $\mathsf{FList}(A, <)$ for some type $A : \mathsf{Set}$ and a strict total order $< : A \to A \to \mathsf{Prop}$. We then prove an extensionality principle analogous to set extensionality which allows us to show that $\mathsf{FList}(A, <)$ is an idempotent commutative monoid with the empty list as the unit, and the operation which merges two sorted lists as the multiplication.

To establish (ii), we would like to show that this type is the *free* idempotent commutative monoid. However, there is a wrinkle — the domain of the sorted list functor cannot be simply the category of sets Set, since we require that the underlying set is equipped with a strict total order in order to form the type of sorted lists. Assuming that any set can be equipped with such an order is a strongly classical axiom called the Ordering Principle which is strictly weaker than the well-ordering principle [8, Ch. 5 §5], but still implies LEM [13]. Therefore to remain constructive, we must restrict the domain of the functor to strictly totally ordered sets. Thus, we define the categories STO of strictly totally ordered sets, and OICMon of *ordered* idempotent commutative monoids (ordering data is also required for the monoids so that it can be preserved by the forgetful functor; this is satisfied for $\mathsf{FList}(A, <)$ via the lexicographic ordering). With the categories in place, we can prove that the type of sorted lists is functorial, and left adjoint to the forgetful functor $\mathcal{U} : \mathsf{OICMon} \to \mathsf{STO}$, giving us the desired universal property.

Other Free Algebraic Structures The choice to implement sorted lists as an instantiation of the type of fresh lists reveals further paths to explore; what happens for other instantiations of the freshness relation? It turns out that different choices each yield a different free structure.

In particular, it should come as no surprise that finite multisets are represented by sorted lists with duplicates (i.e., fresh lists over a total order $\leqslant$). The proof of the adjunction is very similar to the previous case, however we obtain a different extensionality principle: since the membership relation for multisets is valued in Set rather than Prop, we must prove an isomorphism rather than merely a bi-implication. Other such results are summarised in Table 1.

Freshness Relation	Free Algebraic Structure	Data Structure
$\leqslant$, a total order	Ordered Commutative Monoid	Sorted lists
$<$, a strict total order	Ordered Idempotent Comm. Monoid	Sorted lists w/o duplicates
$\lambda x.\lambda y.\bot$	Pointed Set	Maybe
$\lambda x.\lambda y.\top$	Monoid	List
$\neq$	Left-Regular Band Monoid	Lists without duplicates
$=$	Reflexive Partial Monoid	$1 + (A \times \mathbb{N}^{>0})$

Table 1: Free algebraic structures as instantiations of freshlists (carrier set A)

Related Work Appel and Leroy [2] recently introduced canonical binary tries as an extensional representation of finite *maps*. These can be used to construct finite sets with elements from the index type. Krebbers [9] extended this technique to form extensional finite maps over arbitrary countable sets of keys.

The technique of using underlying ordering data to construct extensional data structures is not new, and has been employed in a number of Coq libraries for many years [5][6][12][11].

References

[1] The Agda Community. Agda standard library, 2023. https://github.com/agda/agda-stdlib.

[2] Andrew W. Appel and Xavier Leroy. Efficient extensional binary tries. *Journal of Automated Reasoning*, 67(1):8, 2023.

[3] Vikraman Choudhury and Marcelo Fiore. Free commutative monoids in Homotopy Type Theory. In Justin Hsu and Christine Tasson, editors, *Mathematical Foundations of Programming Semantics (MFPS '22)*, volume 1 of *Electronic Notes in Theoretical Informatics and Computer Science*, 2023.

[4] Catarina Coquand. A formalised proof of the soundness and completeness of a simply typed lambda-calculus with explicit substitutions. *Higher Order Symbolic Computation*, 15(1):57–90, 2002.

[5] Arthur Azevedo de Amorim et al. `extructures`. Coq library available at https://github.com/arthuraa/extructures.

[6] Cyril Cohen et al. `finmap`. Coq library available at https://github.com/math-comp/finmap.

[7] Dan Frumin, Herman Geuvers, Léon Gondelman, and Niels van der Weide. Finite sets in homotopy type theory. In *International Conference on Certified Programs and Proofs CPP '18*, pages 201–214. Association for Computing Machinery, 2018.

[8] Thomas Jech. *The Axiom of Choice*. North-Holland, 1973.

[9] Robbert Krebbers. Efficient, extensional, and generic finite maps in coq-std++. In *The Coq Workshop 2023*, 2023.

[10] Clemens Kupke, Fredrik Nordvall Forsberg, and Sean Watters. A fresh look at commutativity: Free algebraic structures via fresh lists. In *Programming Languages and Systems - 21st Asian Symposium, APLAS 2023, Taipei, Taiwan, November 26-29, 2023, Proceedings*, volume 14405 of *Lecture Notes in Computer Science*, pages 135–154. Springer, 2023.

[11] Christian Strub. `ssrmisc`. Coq library available at https://github.com/strub/ssrmisc/blob/master/fset.v.

[12] Pierre-Yves Strub. `fset`. Coq library available at https://www.ps.uni-saarland.de/formalizations/fset/html/libs.fset.html.

[13] Andrew Swan. `irreflexive-extensional-order-on-every-set-gives-excluded-middle`. Agda formalisation by Tom De Jong available at https://www.cs.bham.ac.uk/~mhe/TypeTopology/Ordinals.WellOrderingTaboo.html.

[14] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study, 2013.