Finiteness in Cubical Type Theory

Donnacha Oisín Kidney[1] and Gregory Provan[2]

[1] University College Cork o.kidney@cs.ucc.ie
[2] University College Cork g.provan@cs.ucc.ie

Abstract. We study five different notions of finiteness in Cubical Type Theory and prove the relationship between them. In particular we show that any totally ordered Kuratowski finite type is manifestly Bishop finite.
We also prove closure properties for each finite type, and classify them topos-theoretically. For instance we show that the category of decidable Kuratowski finite sets (also called the category of cardinal finite sets) form a Π-pretopos.
We then develop a parallel classification for the countably infinite types, as well as a proof of the countability of $A^\star$ for a countable type A.
We formalise our work in Cubical Agda, and we implement a library for proof search (including combinators for level-polymorphic fully generic currying), and demonstrate how it can be used to both prove properties and synthesise full functions given desired properties.

Keywords: Agda · Homotopy Type Theory · Cubical Type Theory · Dependent Types · Finiteness · Topos · Kuratowski finite

1 Introduction

1.1 Foreword

In constructive mathematics we are often preoccupied with *why* something is true. Take finiteness, for example. There are a handful of ways to demonstrate some type is finite: we could provide a surjection from another finite type; we could show that any collection of its elements larger than some bound contains duplicates; or that any stream of its elements contain duplicates.

Classically, all of these proofs end up proving the same thing: that our type is finite. Constructively (in Martin-Löf Type Theory [2] at least), however, all three of the statements above construct a different version of finiteness. *How* we show that some type is finite has a significant impact on the type of finiteness we end up dealing with.

Homotopy Type Theory [4] adds another wrinkle to the story. Firstly, in HoTT we cannot assume that every type is a (homotopy) set: this means that the finiteness predicates above can be further split into versions which apply to sets only, and those that apply to all types. Secondly, HoTT gives us a principled and powerful way to construct quotients, allowing us to regain some of the flexibility

of classical definitions by "forgetting" the parts of a proof we would be forced to remember in MLTT.

Finally, the other important property of constructive mathematics is that we can actually *compute* with proofs. Cubical Type Theory [1], and its implementation in Cubical Agda [5], realise this property even in the presence of univalence, giving computational content to programs written in HoTT.

1.2 Contributions

In this work we will examine five notions of finiteness in Homotopy Type Theory, the relationships between them, and their topos-theoretic characterisation. We also briefly examine a definition for countable sets, comparing it to the manifestly Bishop finite sets. Our work is formalised in Cubical Agda, where we also develop a library for proof search based on the finiteness predicates.

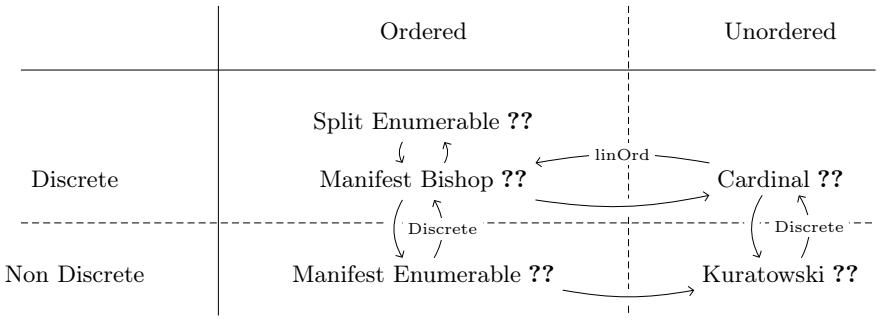

Fig. 1: Classification of finiteness predicates according to whether they are discrete (imply decidable equality) and whether they induce a linear order.

The finiteness predicates we are interested in are organised in Figure 1. We will explore two aspects of each predicate: its relation to the other predicates, and its topos-theoretic classification.

When we say "relation" we are referring to the arrows in Figure 1. The proofs, then, amount to a function which inhabits each arrow. Each unlabelled arrow is a weakening: i.e., every manifest Bishop finite set is manifest enumerable. The labelled arrows are strengthening proofs: i.e., every manifest enumerable set *with decidable equality* is manifest Bishop finite. Our most significant result here is the proof that a cardinal finite set with a decidable total order is manifestly Bishop finite.

We will then characterise each predicate as some form of topos. Our proofs follow the structure of [4, Chapters 9, 10] and [3]. This means we will define first the precategory of sets, and then the category of sets, and prove the required closures and limits to get us to a topos. We say "some form of" topos here because of course the category of sets in HoTT do *not*, in fact, form a topos,

rather a ΠW-pretopos. Our main result here is that the category of decidable Kuratowski finite sets forms a Π-pretopos.

After the finite predicates, we will briefly look at the infinite countable types, and classify them in a parallel way to the finite predicates. We here show that countably finite sets form a W-pretopos.

All of our work is formalised in Cubical Agda [5]. We will make mention of the few occasions where the formalisation of some proof is of interest, but the main place where we will discuss the code is in the final section, where we implement a library for proof search, based on omniscience and exhaustibility. We also use this section to demonstrate some *computation* using univalence: our proof search library can deal seamlessly with functions, testing them for equality, synthesising them, etc.

1.3 Related Work

1.4 Notation

Basic MLTT Notation: Π, Σ, etc

Paths, Univalence, isomorphisms

HITs

2 Finiteness Predicates

While our primary interest is in decidable Kuratowski finiteness, we found it useful to examine the other finiteness predicates as well, if only as a way to construct proofs which can then be transferred to proof on Kuratowski finiteness. This, in turn, is why the relation proofs are so important: in HoTT, if two types are proved equivalent, we get access to the extremely powerful univalence axiom, which allows us to switch between types freely depending on what is more convenient. The Π-topos proof, for instance, is defined on cardinal finiteness, not decidable Kuratowski finiteness: an equivalence proof between the two types means we can transfer it over, formally, and with good justification. Similarly, many of the closure proofs are more naturally defined over the enumerated finiteness predicates.

2.1 Kuratowski Finiteness

The first thing we must define is a representation of subsets.

Definition 1 (Kuratowski Finite Subset)**.** $\mathcal{K}(A)$ is the type of Kuratowski-finite subsets of A.

$$\begin{aligned}\mathcal{K}(A) = {} & [] : \mathcal{K}(A);\\ & \mid \cdot :: \cdot : A \to \mathcal{K}(A) \to \mathcal{K}(A);\\ & \mid \mathrm{com} : \Pi(x, y : A), \Pi(xs : \mathcal{K}(A)), x :: y :: xs \equiv y :: x :: xs; \qquad (1)\\ & \mid \mathrm{dup} : \Pi(x : A), \Pi(xs : \mathcal{K}(A)), x :: x :: xs \equiv x :: xs;\\ & \mid \mathrm{trunc} : \Pi(xs, ys : \mathcal{K}(A)), \Pi(p, q : xs \equiv ys), p \equiv q;\end{aligned}$$

The Kuratowski finite subset is a free join semilattice (or, equivalently, a free commutative idempotent monoid). From a programming perspective, it could well be seen as a finite set (in the sense of the data structure). In most languages, it might be represented by a hash table or some balanced tree: HITs allow us to represent precisely the notion of a finite set, without putting any constraints on the contents, or choosing any internal data structure.

Though it would be possible to use this type as-is, we found it easier to work with the following type as it has fewer path constructors. What we've done here is basically pulled out the "monoid" part of "free commutative idempotent monoid": the first two constructors now construct a list, which is a representation of the free monoid *without* quotient types. This obviates the need for the associative and identity constructors. We have proven both types equivalent in our formalisation.

Next, we need a way to say that a type is Kuratowski finite, rather than just some subset. For that, we will need to define membership of $\mathcal{K}$.

Definition 2 (Membership of K)**.** First, we need to provide equations for the two point constructors. $x \in [] = \bot$, and $x \in y :: xs = \|(x \equiv y) + (x \in xs)\|$. The com and dup constructors are handled by proving that the truncated form of $+$ is itself commutative and idempotent. The type of propositions is itself a set, satisfying the trunc constructor.

Definition 3 (Kuratowski Finiteness)**.** A type is Kuratowski finite iff there exists a Kuratowski finite subset which contains all of its elements.

$$\mathcal{K}^f(A) = \Sigma(xs : \mathcal{K}(A)), \Pi(x : A), x \in xs \tag{2}$$

2.2 Split Enumerable

Definition 4 (Split Enumerable Set)**.**

$$\mathcal{E}!(A) = \Sigma(xs : \mathbf{List}(A)), \Pi(x : A), x \in xs \tag{3}$$

We call the first component of this pair the "support" list, and the second component the "cover" proof.

In this, and the rest of the definitions in the paper, we tend to prefer list-based notions of finiteness. This is purely a matter of perspective, however: the definition above is precisely equivalent to a split surjection from a finite prefix of the natural numbers.

Lemma 1.

$$\mathcal{E}!(A) \simeq \Sigma(n : \mathbb{N}), (\mathbf{Fin}(n) \twoheadrightarrow! A) \tag{4}$$

We have proven this statement in our formalisation, but it is worth pointing out that the proof is nearly trivial: after inlining the definitions of **List**, $\in$, and $\twoheadrightarrow!$, it becomes clear that the two terms are basically the same anyway, modulo the reassociation of Σ.

Split enumerability is quite a strong predicate, implying both decidable equality and a total order on the underlying type. This

2.3 Manifest Bishop

2.4 Manifest Enumerable

2.5 Cardinal

3 Relations Between Each Finiteness Definition

4 Topos

– Complete (pullbacks + terminal object)

5 Countably Infinite Types

6 Search

Add funding ack:
This work has been supported by the Science Foundation Ireland under the following grant: 13/RC/2D94 to Irish Software Research Centre.

References

1. Cohen, C., Coquand, T., Huber, S., Mörtberg, A.: Cubical Type Theory: A constructive interpretation of the univalence axiom. arXiv:1611.02108 [cs, math] p. 34 (Nov 2016)
2. Martin-Löf, P.: Intuitionistic Type Theory. Padua (Jun 1980)
3. Rijke, E., Spitters, B.: Sets in homotopy type theory. Mathematical Structures in Computer Science **25**(5), 1172–1202 (Jun 2015). https://doi.org/10.1017/S0960129514000553
4. Univalent Foundations Program, T.: Homotopy Type Theory: Univalent Foundations of Mathematics. https://homotopytypetheory.org/book, Institute for Advanced Study (2013)
5. Vezzosi, A., Mörtberg, A., Abel, A.: Cubical Agda: A Dependently Typed Programming Language with Univalence and Higher Inductive Types. Proc. ACM Program. Lang. **3**(ICFP), 87:1–87:29 (Jul 2019). https://doi.org/10.1145/3341691