Evaluating Scopes of Different Implementations of Finite Sets in Homotopy Type Theory

Giovanni Fincato de Loureiro
Supervisor(s): Kobe Wullaert, Benedikt Ahrens
EEMCS, Delft University of Technology, The Netherlands

June 19, 2022

A Dissertation Submitted to EEMCS faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering

Abstract

This report explores the differences in implementations of homotopy type theory using different definitions of finite sets. The expressive ability of homotopy theory is explored when using the newly established definition and implementation of Kuratowski finite sets. This implementation is then compared to that when using the previously established definition and implementation of Bishop finite sets in homotopy type theory.

The implementation of finite sets in homotopy type theory is a topic of extensive research. Recent advances have come up with an implementation of Kuratowski-finite sets which promises to be more general and flexible than previously established implementations of finite sets via Bishop-finite sets and enumerated types.

This paper will explore examples of types that satisfy Kuratowki-finite notions of finite sets, but not Bishop-finite notions. Through these examples, it will be proven that Kuratowski-finite notions are strictly more flexible than Bishop-finite notions. It will then be proven that by using Kuratowski-notions, the familiar computations involving sets can be used in the same way as with Bishop-finite sets; there is no loss in computational facilities.

1 Introduction

Currently, tools such as Coq are used to automatically verify proofs under the basis of type theory, proposed by Martin-Löf as an alternative to set theory[1]. Unfortunately, the foundational principles of type theory are troublesome to translate into the programming language of automated proof assistants[2]. Homotopy type theory (henceforth refered to as "HoTT") which uses constructive foundations, was proposed by Vladimir Voevodsky as an alternative to standard ZFC set theory[3].

The mathematical foundations of HoTT are a development of intuitionistic Martin-Löf type theory that employs higher inductive types, henceforth referred to as "HIT"s, and the univalence axiom[2]. The univalence axiom relates the concept of equality in mathematical propositions to the concept of equivalence in homotopy theory. Under the univalence axiom, isomorphic structures are treated as equals[4]. HIT's, meanwhile, are similar to inductive types, but allow for path constructors as well as point constructors[12]. A simple example of a higher inductive type is the circle, which is defined by a simple point constructor for the base of the circle, and a path constructor describing the loop which begins and ends at the base to create the circle S^1[5]

```
Inductive S¹ : Type :=
    | base : S¹
    | loop : base = base
```

Figure 1: The constructor for circles as a higher inductive type

In essence, HoTT explains not only that "A = B" but also how to get from A to B, while in standard set theory this would have to be defined additionally[6]. This approach promises to make mathematical proofs easier to translate into computer programs, allowing for more complex proofs to be checked automatically. The development and research of HoTT is done with the objective of (among others) making mathematical proofs to be more easily verified by computers.

If HoTT aims at providing a system for reasoning, its application to different mathematical structures must be developed further. The application of HoTT to finite sets and

finite types is a topic of ongoing research. As of the writing of this paper, there is no formally accepted and established method for defining finite sets within HoTT that provides the computational facilities and the proof principles expected of finite sets - such as union and cardinality operations, and extensional equality for sets[7].

Different approaches to implementing finite sets in HoTT have been explored[7]. In constructive mathematics, there are different forms of defining finite sets[8]. The most well known is the definition that a set is finite if it is equivalent to a canonical finite set {0, ..., n} for some natural number n[9]. This definition of finite sets was popularized by Bishop, and is referred to as "Bishop-finite". However, using the Bishop-finite notion of finite sets has computational limitations. Canonical finite sets require the underlying type to have *decidable equality*[10], meaning two instances of the type are either equal or not equal. In HoTT, this is not always the case. This limits how flexible the notion is when used to define finite sets.

A 2018 paper by van der Weide et al. [7] explores the possibility of instead using the notion of Kuratowski-finiteness. The Kuratowski-finite notion of finite sets denotes that a type is finite if there exists a Kuratowski-finite subset of the type that contains all instances of said type. Although similar at face This approach is inspired by topos theory, and van der Weide's paper claims this notion to be more general and flexible than that of Bishop-finiteness, as Kuratowski-finiteness does not require the underlying type to have decidable equality, and instead only requires *decidable mere equality*[7], which is further explained in section 3.2.

This paper will compare the computational scope of implementations between the Bishop-finite notion and the Kuratowski-finite notion implementations (from van der Weide et al.[7]) of finite sets in HoTT. As HoTT is developed, it is crucial that the computational facilities of finite sets are provided by HoTT implementations. The research question being investigated is "What is the difference in HoTT implementations when using Kuratowski-finite notions of finite sets versus Bishop-finite notions?"

This paper is structured as follows:

- Section 2 covers definitions of terminology and some fundamental aspects of different finite set notions
- Section 3 evaluates examples of types that prove that the notions of finite sets are not equivalent; that is, examples that are covered by Kuratowski-finite notions but not Bishop-finite notions
- Section 4 establishes the scope of each notion in-depth, along with the difference in each
- Section 5 explores the implications of the differences between the notions to HoTT implementations by showing what an actual implementation of Kuratowski-finite notions would look like
- Section 6 includes concluding statements and suggests where future research into the field should be directed

2 Definitions

HoTT is a refinement of constructive set theory, and in constructive mathematics there is more than one way of stating that a set has a finite number of elements[8]. In this paper,

the two main notions of finiteness explored are that of Bishop-finiteness (henceforth referred to as "*B-finiteness*") and Kuratowski-finiteness (henceforth referred to as "*K-finiteness*"). K-finite and B-finite sets over some type $\mathcal{A}$ will be denoted $\mathcal{K}(\mathcal{A})$ and $\mathcal{B}(\mathcal{A})$, respectively.

2.1 Bishop-finite sets

The most straightforward and intuitive way to establish that a set is finite is to simply count the elements in the set: an approach which leads to the notion of B-finiteness, the most well-known notion of finiteness. A type $\mathcal{A}$ is B-finite if there is a natural number n such that $\mathcal{A}$ is merely equivalent to a canonical cardinal of size n[9]. Formally, the type $\mathcal{A}$ is B-finite if it satisfies the proposition $isBf(\mathcal{A})$, where[7]:

Definition 2.1:

$$isBf(\mathcal{A}) := \Sigma(n : \mathbb{N}), ||\mathcal{A} \simeq [n]||$$

B-finiteness has been studied previously in the setting of HoTT, and the notion has been formalized[11]. The constructs using the B-finite notion are limited in that B-finiteness can be restrictive. The first issue is that B-finite types are not, in general, closed under disjoint union[7]. Furthermore the definition of B-finiteness depends on finding mere-equivalence of a set over type $\mathcal{A}$ to finite cardinals, such as $[n] = \{0, ..., n-1\}$[9]. This can be problematic because all finite-cardinals have decidable equality[13], meaning all B-finite sets must have decidable equality as well. Within the context of HoTT, a constructive approach, this is rather restrictive, as decidable equality is not guaranteed for many types. As is explored in section 3, some types which are in fact finite unfortunately do not have decidable equality[7], which has led researchers to search for other notions of finite sets which prove to be more flexible.

2.2 Kuratowski-finite sets

The notion of K-finiteness was proposed by Kuratowski as a replacement for B-finiteness, promising to be more flexible and general [14]. At the risk of oversimplifying, a type is considered K-finite if there is some finite set that contains all instances of said type. Formally, the type $\mathcal{A}$ is K-finite if it satisfies $isKf(\mathcal{A})$, where[7]:

Definition 2.2:

$$isKf(\mathcal{A}) := \Sigma(X : \mathcal{K}(\mathcal{A})), \Pi(a : \mathcal{A}), a \in X$$

In the paper by van der Weide et al., the K-finite set is defined as a higher inductive type[7]:

Higher Inductive Type $\mathcal{K}(\mathcal{A})$:=
1 | $\emptyset$: $\mathcal{K}(\mathcal{A})$
2 | $\{\cdot\}$: $\mathcal{A} \to \mathcal{K}(\mathcal{A})$
3 | $\cup$: $\mathcal{K}(\mathcal{A}) \to \mathcal{K}(\mathcal{A}) \to \mathcal{K}(\mathcal{A})$
4 | nl : $\Pi(x : \mathcal{K}(\mathcal{A})), \emptyset \cup x = x$
5 | nr : $\Pi(x : \mathcal{K}(\mathcal{A})), x \cup \emptyset = x$
6 | idem : $\Pi(x : \mathcal{A}), \{x\} \cup \{x\} = \{x\}$
7 | assoc : $\Pi(x, y, z : \mathcal{K}(\mathcal{A})), x \cup (y \cup z) = (x \cup y) \cup z$
8 | com : $\Pi(x, y : \mathcal{K}(\mathcal{A})), x \cup y = y \cup x$
9 | trunc : $\Pi(x, y : \mathcal{K}(\mathcal{A})), \Pi(p, q : x = y), p = q$

Figure 2: The constructor for K-finite as a higher inductive type. Note the numbers on the leftmost side indicate only line numbers, and do not form part of the constructor

The first three lines of the type are point constructors, indicating that the empty set is K-finite (line 1), as are singletons (line 2), and the rule that the union between a K-finite set and another K-finite is also K-finite (line 3). Lines 4 onward are path constructors, a unique feature of higher inductive types. Line 4 and 5 indicate the union between a set and the empty set is identical to the original set. Line 6, 7, and 8 are paths between points and refer to basic rules of sets, such as associative and commutative laws. Line 9, however, is unique to HoTT; trunc is a function between two paths that forces $\mathcal{K}(\mathcal{A})$ to be a HSet. Simply put, a HSet is a set where all paths between two points are equal. Note that in HoTT, the terms "HSet" and "set" are often used interchangeably[15].

2.3 Similarities between finite set notions

Before comparing the differences between these two notions of finiteness, it is crucial to explore their similarities. Firstly, it must be noted that all B-finite types are also K-finite, although further sections will show that the inverse is not true in general. This can be intuitively determined from the analysis of the constructor of $\mathcal{K}(\mathcal{A})$ in *Figure 2* - it is clear that any set X which satisfies *isBf(X)* can be built from successive unions of singletons. A formal proof is provided by van der Weide et al.[7]

It is interesting to note that the two notions of finiteness are equivalent to each other if the underlying type has *decidable equality*. A type having decidable equality implies that any two objects of the type are either equal or not equal. Formally, a type $\mathcal{A}$ has decidable equality if and only if it satisfies $hasDecEq(\mathcal{A})$ where:

Definition 2.3:

$$hasDecEq(\mathcal{A}) := \Pi(a, b : A), (a = b) + (a \neq b)$$

In classical mathematics, where the Law of Excluded Middle (LEM) is assumed, this is evidently always true. Two objects are either equal or not equal. However, this is only true assuming LEM - which is not necessarily the case in HoTT. Without the LEM, there are in fact types which do not have decidable equality - that is, it is mathematically impossible to determine whether two instances of the type they are equal or not.

3 Example types analysis

In order to explore the difference between K-finite and B-finite notions, we will explore examples of types that fit K-finite notions of finiteness but not B-finite.

3.1 Example type study: Real numbers

Perhaps the simplest example can be found in the real numbers, $\mathbb{R}$. The classical real numbers - that is, the set[1] containing both rational and irrational numbers - do not have decidable equality, and representations of real numbers often rely on approximating them towards a rational number[17]. The non-decidability of real numbers is generally well known, and an

[1]The word "set" here is used informally. In fact, as they are classically defined, the real numbers are not a set at all[2]

intuitive explanation describes how, in an effective topos[2], a real number is represented by a Turing machine that computes arbitrarily good rational approximations; attempting to establish an equality between two of these types would clearly not be computable. However, consider a set X with only one real number contained, $\{x\}$ for some $x \in \mathbb{R}$. By definition this singleton has only one element, yet it cannot be considered a finite set by the B-finite notion - since B-finite types require decidable equality in the underlying type.

Despite being a singleton (a set with a single inhabitant), by B-finite notions of finite sets, the singleton set is in fact not considered finite if the underlying type does not have decidable equality. This is not only very counter-intuitive, but also problematic when trying to reason about such a set. Set operations cannot be performed on such a set while assuming finiteness, despite the set having only one inhabitant. This is what led to the idea of K-finiteness being used as the notion of finiteness in HoTT. Under K-finiteness, the singleton set is always finite by definition, which is much less constraining. More examples of types which meet the K-finite but not B-finite notions will be explored to better understand the non-equality between the two.

3.2 Example type study: Circles

Such types that do not have decidable equality without assuming LEM include the circle, the constructor for which is shown in *Figure 1.* A paper by Shulman et al. [5] goes in depth into the fundamental group of the circle in HoTT. The loop in the constructor cannot be algorithmically compared, meaning the circle type lacks decidable equality. Furthermore, it can be proven that the circle type S^1 is not a B-finite set.

First, let it be established that if X is a B-finite set, then all singleton subobjects of it must be B-finite[7]. Knowing that a B-finite set is equivalent to canonical finite set $\{0, ..., n\}$ for some $n \in \mathbb{N}$, this is a rather intuitive conclusion; any singleton subset $\{x\}$ of a B-finite set X contains a single instance of a type with decidable equality (as implied by the encapsulating set X being B-finite itself) and the singleton is therefore equivalent to set $\{0, ..., n\}$ where $n = 0$ (i.e. singleton set $\{0\}$)[3].

The singleton sub-object of the circle type S^1 is not B-finite, however, due to the circle lacking decidable equality. From this fact, we can therefore conclude that the circle type S^1 is not a B-finite set. It can be further proven that S^1 is actually not a set at all. First, consider the identity type commonly used in HoTT. It has a single constructor **refl** of type $\Pi(x : A), (x = x)$. This identity type plays a crucial role in all of HoTT[16], but here we will use it to prove that the circle type S^1 is not a set. to borrow a proof from the Homotopy Type Theory book[2]: If $\mathbf{loop} = \mathbf{refl}_{base}$, then we could use some $x : A$ and $p : x = x$ to create a function

$$f : S^1 \to A$$

for any type A, with f being $f(\mathbf{base}) :\equiv x$ and $f(\mathbf{loop}) := p$. This would lead to the following equation:

$$p = f(\mathbf{loop}) = f(\mathbf{refl}_{base}) = \mathbf{refl}_x$$

However, for this equation to stand, A has to be a set. Since A represents an arbitrary type, this equation then implies that every type is a set, which is definitively untrue[2]. This leads to the conclusion that $\mathbf{loop} \neq \mathbf{refl}_{base}$.

[2]At the risk of egregiously oversimplifying the concept, a topos is best understood here as "a context under which one does mathematics"

[3]A formal proof is provided by van der Weide et al.[7]

Despite not having decidable equality (and therefore not being B-finite), the circle type S^1 does have what is referred to by van der Weide et al.[7] as *decidable mere equality*. Decidable mere equality is similar to decidable equality, but has truncation on the two equalities. Formally, the type A has decidable mere equality if and only if it satisfies $hasMereDecEq(X)$ where:

Definition 3.1:

$$hasMereDecEq(X) := \Pi(a, b : A), ||a = b|| + ||(a \neq b)||$$

the circle type S^1 does have decidable mere equality[7], and this can be used to prove that S^1 is a K-finite type. For $X = \{\mathbf{base}\}$ (a singleton containing only **base**)[4] , path induction on S^1 shows $\Pi(x : S^1), x \in \{\mathbf{base}\}$ using the truncation $||\mathbf{base} = \mathbf{base}||$. Using the same truncation, $x = \mathbf{base}$ is a mere proposition[5] - that is, a type where all inhabitants are equal [2]. From this proof that the type S^1 is contained in a singleton, it is shown that S^1 is by definition K-finite, as the K-finite notion of finite sets includes singletons by definition.

Furthermore, we now have a more useful example of K-finite and B-finite types not being equal. The circle type S^1 is encapsulated in a singleton, making it K-finite, but is not B-finite due to the lack of decidable equality. From the two proofs, we can come to a general conclusion: types which are not sets, but are encapsulated in singletons, are K-finite, but not B-finite. This notion can be further explored by examining the 2-sphere.

3.3 Example type study: 2-Spheres

The 2-sphere higher inductive type S^2 follows from the circle type, with a point base S^2 and a 2-dimensional path.

Inductive S^2 : Type :=
| base : S^2
| surf2 : $\mathbf{refl}_{base} = \mathbf{refl}_{base}$ in $base = base$

Figure 3: The constructor for circles as a higher inductive type[2]

It is important to note that the path $surf2 : \mathbf{refl}_{base} = \mathbf{refl}_{base}$ contains $\mathbf{refl}_{base}$, however, this does not imply that circles within the 2-sphere have decidable equality. The $\mathbf{refl}_{base}$ reference is not equivalent to the *loop* path which is found in the constructor for circles. Similar to the circle type S^1, the 2-sphere type S^2 also does not have decidable equality, and this can be shown with a similar proof to that for S^1: If $\mathbf{surf2} = \mathbf{refl}_{\mathbf{refl}_{base2}}$, then we could use some $x : A$ and $p : x = x$ to create a function

$$f : S^2 \to A$$

for any type A, with f being $f(\mathbf{base2}) :\equiv x$ and $f(\mathbf{loop}) := p$. This would lead to the following equation:

$$p = f(\mathbf{surf2}) = f(\mathbf{refl}_{base2}) = \mathbf{refl}_x$$

However, for this equation to stand, A has to be a set. Since A represents an arbitrary type, this equation then implies that every type is a set, which is definitively untrue[2]. This leads to the conclusion that $\mathbf{loop} \neq \mathbf{refl}_{base}$.

[4]Curly brackets in this paper are used to explicitly indicate sets and their contents. In particular, curly brackets containing a single item a, as in $\{a\}$ indicate a singleton containing a

[5]Referred to as a HProp in van der Weide et al.[7]

Similar to the circle, the 2-sphere is also a K-finite type. The mere equality of the circle can be raised in dimension to the sphere as well. We can prove that $hasMereDecEq(S^2)$ through a similar method: Assume a singleton with $X = \{\mathbf{base}\}$, path induction on S^2 shows $\Pi(x : S^2), x \in \{\mathbf{base2}\}$ using the same truncation of $||base = base||$ used for circles, but working downwards from the dimension of **surf2**. The 2-sphere contains its own equality of $||base2 = base2||$ from **surf2**, as $\mathbf{refl}_{base} = \mathbf{refl}_{base}$ is itself built upon $base = base$. The truncation of $||base2 = base2||$ allows us to contain all **base2** instances into the singleton $X = \{\mathbf{base}\}$, proving the 2-sphere to be contained in a singleton - and therefore K-finite. Like the real numbers and the circle, the 2-sphere is a K-finite type that is not B-finite.

3.4 Example type study: n-Spheres

Notice the pattern of circles/spheres having a base point, and an equality between **base**. The circle has an equality from base to base in **loop**, while the sphere has an equality from $\mathbf{refl}_{base2}$ to $\mathbf{refl}_{base2}$ in **surf2**, which can be interpreted as a higher-dimensional **loop**. From the definition of the circle type S^1 and the 2-sphere type S^2, we can therefore reason about a definition of spheres in a more general level in higher dimensions - the n-spheres, as in, a sphere at the n dimension.

Firstly, just like the circle had a base S^1 and the sphere had a base S^2, the n-sphere will have a base point S^n. Note that point bases S^n are simply a point in space, but at the n dimensional level. The loop is not as trivial to generalize to higher dimensions, but we can reason about an n-loop with an equality $\Omega^n(S^n, \mathbf{base}^n)$ creating the sphere itself. The notion behind this definition is that of a "dependent n-loop", with each loop an equality building upon the previous dimension loop[2].

The n-sphere S^n also does not have decidable equality, and this can be shown with a similar proof to that for S^1 and S^2. Every $\Omega^n(S^n, \mathbf{base}^n)$ will be built upon the equality between $\Omega^{n-1}(S^{n-1}, \mathbf{base}^{n-1})$. By repeated path induction on Ω^n to Ω^{n-1}, it will always reduce to a path $base = base$, as found in S^1. By this reasoning, any n-dimensional sphere can be proven to have non-decidable equality through the same reasoning of $\mathbf{loop} \neq \mathbf{refl}_{base}$ used in section 3.4.

Similarly, by reducing $\Omega^n(S^{n-1}, \mathbf{base})$ to get $base = base$ by repeated path induction, we can truncate $||base = base||$ to prove that the **base** in S^n will always have $\Pi(x : S^n), x \in \{\mathbf{base}\}$ for any $n \in \mathbb{N}$, such that any n-sphere can be contained in a singleton.

From this we can conclude that any n-sphere is not B-finite, because they lack decidable equality. They are K-finite, however, as they can all be contained within a singleton.

4 Comparison

We can now begin to establish the difference between B-finite and K-finite notions in a more intuitive way.

Let us establish two functions. Firstly, $\mathcal{K}(\mathcal{A})$ implies that some given type $\mathcal{A}$ is K-finite, while $\mathcal{B}(\mathcal{A})$ implies that a type is B-finite. From section 2.3, it is established that all B-finite types are K-finite. From this fact, and the functions $isBf$ and $isKf$ from sections 2.1 and 2.2 respectively, we can formally establish:

Proposition 4.1:

$$\Pi(\mathcal{A} : \text{Type}), isBf(\mathcal{A}) \Rightarrow isKf(\mathcal{A})$$

Meaning that B-finiteness implies K-finiteness. Through exploring the examples in section 3, it has been established that there are types which are K-finite but not B-finite

Proposition 4.2:

$$\Sigma(\mathcal{A} : \text{Type}), isKf(\mathcal{A}) \times \neg isBf(\mathcal{A})$$

Another way to say this is that the set of B-finite types is a proper subset of the set of K-finite types. The examples explored in section 3 ($\mathbb{R}$, Circles, 2-Spheres, n-Spheres) were those which are K-finite, but not B-finite. For these types, the method to prove they were K-finite but not B-finite was to reduce the type to a singleton. Essentially, the examples we are looking for is those types $\mathcal{A}$ where a singleton containting $\{a\}$ for some $a \in \mathcal{A}$. From analysis of the example types, the method used was to prove the the type is contained within a singleton (which is K-finite by definition), and then prove the singleton is not B-finite. B-finiteness requires decidable equality in the underlying type, and to reduce a type to a singleton we require decidable mere equality. Recall the functions $hasDecEq$ and $hasMereDecEq$ from sections 2.3 and 3.2 respectively, which represent decidable equality and mere decidable equality. Stating that a type can be reduced to a singleton, but that it is not decidably equal, is therefore equivalent to the type being K-finite but not B-finite. For any given type $\mathcal{A}$, we can declare:

Proposition 4.3:

$$\Pi(\mathcal{A} : \text{Type}), isKf(\mathcal{A}) \times \neg isBf(\mathcal{A}) \simeq hasMereDecEq(\mathcal{A}) \times \neg hasDecEq(\mathcal{A})$$

From the analysed examples, types such as $\mathbb{R}$ and S^n do not have decidable equality, but do have decidable mere equality.

Given the examples studied, we can now begin to explore actual implications of these differences. Firstly let us define subobjects for B-finiteness and K-finiteness. The K-finite subobject is already defined in figure 2 as $\mathcal{K}(\mathcal{A})$; in order to define a B-finite subobject, we need to define the notion of a mere proposition, or **HProp**[7]. A type $\mathcal{A}$ is an **HProp** if it satisfies $isHProp(\mathcal{A})$ where

Definition 4.1:

$$isHProp(\mathcal{A}) := \Pi(x, y : \mathcal{A}), x = y$$

An intuitive way to understand this is $isHProp(\mathcal{A})$ implies all inhabitants of $\mathcal{A}$ are equal. The type **HProp** itself will therefore be defined as

Definition 4.2:

$$\textbf{HProp} := \Sigma(\mathcal{A} : \text{Type}), isHProp(\mathcal{A})$$

Let us furthermore define a subobject $P : \mathcal{A} \to \textbf{HProp}$. This object is B-finite if the subset $\Sigma(x : \mathcal{A}), P(x)$ is also B-finite. The intuition behind defining this type is so we can create a B-finite subobject for B-finiteness, the same way we have $\mathcal{K}(\mathcal{A})$ as K-finite subobjects for K-finite types. This B-finite subobject is defined as[18]

Proposition 4.4:

$$\Sigma(X : \mathcal{A} \to \text{HProp}), isBf(X)$$

B-finite objects will be referred to as $\mathcal{B}(\mathcal{A})$ as a B-finite equivalent of $\mathcal{K}(\mathcal{A})$.

From these definitions we can begin to compare how set operations differ between $\mathcal{B}(\mathcal{A})$ and $\mathcal{K}(\mathcal{A})$ subobjects. If we are to use K-finite notions rather than B-finite notions of finiteness in HoTT, then $\mathcal{K}(\mathcal{A})$ should possess the same computational functions on finite sets that we expect from $\mathcal{B}(\mathcal{A})$. The strategy for this will be to build up the two types in a similar way as done in Figure 2; beginning with the empty set $\emptyset$, proceeding to the singleton $\{\cdot\}$, then defining behaviour for union operation $\cup$, intersection operation $\cap$, and finally the (decidable) member function $\in_d$.

Let us establish that empty set $\emptyset$ is both $\mathcal{K}(\mathcal{A})$ and $\mathcal{B}(\mathcal{A})$. The empty set satisfies $isBf(\emptyset)$, since the empty set is a canonical set with cardinality 0. It is also K-finite, as this is defined in the constructor for $\mathcal{K}(\mathcal{A})$.

Next, the singletons. Any singleton is $\mathcal{K}(\mathcal{A})$ by definition, as the singleton it is a point constructor in the $\mathcal{K}(\mathcal{A})$ HIT. However, for a singleton to be $\mathcal{B}(\mathcal{A})$, the underlying type must not have higher paths[18]; as in, it must satisfy $isSet$ where

Definition 4.3:

$$isSet(\mathcal{A}) := \Pi(x, y : \mathcal{A}), \Pi(p, q : x = y), p = q$$

alternatively, an equivalent definition[15] is

Definition 4.4:

$$isSet(\mathcal{A}) := \Pi(x : \mathcal{A}), \Pi(p : x = x), p = \mathrm{id}_x$$

where id_x is the identity type for x. Notice none of the examples explored in section 3 satisfy $isSet$. The real numbers $\mathbb{R}$, the circle S^1, and the n-spheres S^n, lack decidable equality and are not sets at all.

The decidable membership function must be established for both $\mathcal{K}(\mathcal{A})$ and $\mathcal{B}(\mathcal{A})$. Note that unlike equality, which in the case of $\mathcal{K}(\mathcal{A})$ is not necessarily decidable, the membership function must always be decidable. That means:

Proposition 4.3:

$$\Pi(X : \mathcal{B}(\mathcal{A})), \Pi(a : A), (a \in X) + (a \notin X)$$

Proposition 4.4:

$$\Pi(X : \mathcal{K}(\mathcal{A})), \Pi(a : A), (a \in X) + (a \notin X)$$

Let us take a simple function to carry out the membership function on a set, where $\{\mathcal{A}\}$ denotes a set containing members of type $\mathcal{A}$:

```
1    def ∈_d (a : A, X : K(A)) : Boolean :=
2        for(x in X):
3            if(x == a):
4                return true;
5        return false;
```

Figure 3: Decidable membership function $\in_d$. Note the numbers at the beginning of each line denotes only the line number, and are not part of the pseudocode function

Notice the equality function in line 3. In order to decide whether an object is a member of a set, it is required to equate the object to the members of the set. With decidable

equality, this is very straightforward - meaning $\mathcal{B}(\mathcal{A})$ ensures functionality of the membership function. However, mere decidable equality alone is actually enough to carry out the membership function[18], as the truncated equality (as found in mere decidable types) is enough to establish membership. This means that, since $\mathcal{K}(\mathcal{A})$ has (or rather, requires) $hasMereDecEq(\mathcal{A})$, the membership function can be carried out for any $\mathcal{K}(\mathcal{A})$. There is, therefore, no loss of functionality in the decidable membership function when using $\mathcal{K}(\mathcal{A})$ instead of $\mathcal{B}(\mathcal{A})$.

The union operation $\cup$ is more complex. $\mathcal{K}(\mathcal{A})$ is closed under union by definition (see figure 2). $\mathcal{B}(\mathcal{A})$, however, is not. This is rather counter intuitive, as any two $\mathcal{B}(\mathcal{A})$ are each equivalent to some canonical finite set $[n]$ for some $n \in \mathbb{N}$. Say we have two sets $X, Y : \mathcal{B}(\mathcal{A})$, where $|X| = n$ and $|Y| = m$. It is reasonable to assume that a union between the sets would result in a set with cardinality of $n + m - |X \cap Y|$; however, as has been established, the cardinality of the set is not the only factor in determining finiteness when using $\mathcal{B}(\mathcal{A})$. Take a simple pseucode function for $\cup$, using the funciton $\in_d$ from figure 3:

```
1    def ∪(X : K(A), Y : K(A)) : K(A) :=
2        var Z = X;
3        for(y in Y):
4            if ¬(y ∈_d X):
5                Z = Z ∪ {y};
6        return Z;
```

Figure 4: Union function $\cup$. Note the numbers at the beginning of each line denotes only the line number, and are not part of the function

Notice the usage of $\in_d$ in line 4. In order to conduct a union operation, we necessarily require the decidable membership predicate $\in_d$. This means that for the union operation to be carried out, we are under the same restrictions as we are for the $\in_d$ function; that is, decidable equality is required for $\mathcal{B}(\mathcal{A})$, and only mere decidable equality for $\mathcal{K}(\mathcal{A})$.

Finally, the intersection operation $\cap$ on $\mathcal{K}(\mathcal{A})$ and $\mathcal{B}(\mathcal{A})$. Take a simple intersection function:

```
1    def ∩(X : K(A), Y : K(A)) : {A} :=
2        var Z = ∅;
3        for(x in X):
4            if(x ∈_d Y):
5                Z = Z ∪ x
6        return Z;
```

Figure 5: Intersection function $\cap$. Note the numbers at the beginning of each line denotes only the line number, and are not part of the function

In line 4, as with the $\cup$ function, the intersection function also requires $\in_d$.

It follows from reasoning that with using $\mathcal{K}(\mathcal{A})$ there is no loss of set functionality compared to using $\mathcal{B}(\mathcal{A})$. That is, any set operation which can be done on $\mathcal{B}(\mathcal{A})$ can also be done on $\mathcal{K}(\mathcal{A})$, with the difference that the underlying type only requires decidable mere equality. The simple conclusion to draw from this is that $\mathcal{K}(\mathcal{A})$ is strictly more flexible than $\mathcal{B}(\mathcal{A})$ when it comes to operating on sets.

5 implementation

One of the ways in which to gauge the implications of using $\mathcal{K}(\mathcal{A})$ is to implement a list type using $\mathcal{K}(\mathcal{A})$ and evaluate the functions. Let us call this $\mathcal{K}(\mathcal{A})$ implementation of a list $\mathcal{L}(\mathcal{A})$. Transforming $\mathcal{K}(\mathcal{A})$ into a list is fairly straightforward. One of the ways to do so is through this simple example of a function $\mathcal{F} : \mathcal{K}(\mathcal{A}) \rightarrow \mathcal{L}(\mathcal{A})$, with which we can prove any $\mathcal{K}(\mathcal{A}$ can be converted into a list:

```
def F(K : K(A)) : L(A) :=
    var L = nil;
    for (x in K):
        L ::: x;
    return L;
```

Figure 6: Function $\mathcal{F}$ for that returns a list given a $\mathcal{K}(\mathcal{A})$. Note that nil denotes an empty list, and ::: is the list concatenation function

We can prove that $\mathcal{L}(\mathcal{A})$ is equivalent to $\mathcal{K}(\mathcal{A})$ by showing a bijection between the two types. We already have $\mathcal{F} : \mathcal{K}(\mathcal{A}) \rightarrow \mathcal{L}(\mathcal{A})$, so now let us define the inverse function $\mathcal{F}^{-1} : \mathcal{L}(\mathcal{A}) \rightarrow \mathcal{K}(\mathcal{A})$:

```
def F^-1(L : L(A)) : K(A) :=
    var X = ∅;
    for (a in L):
        X = X ∪ {a};
    return X;
```

Figure 7: Function $\mathcal{F}^{-1}$ that returns a set $\mathcal{K}(\mathcal{A})$ given a list $\mathcal{L}(\mathcal{A})$. Note that $\{a\}$ denotes a singleton containing only element a.

The other functions defined for sets $\mathcal{K}(\mathcal{A})$ in figures 3, 4, 5 are very straightforward to apply to lists, requiring only simple changes in typing. The decidable membership function $\in_d$ in figure 3 requires only a trivial type change from $\mathcal{K}(\mathcal{A})$ to $\mathcal{L}(\mathcal{A})$:

```
def ∈_d (a : A, X : L(A)) : Boolean :=
    for(x in X):
        if(x == a):
            return true;
    return false;
```

Figure 8: Function $\in_d (a : \mathcal{A}, X : \mathcal{L}(\mathcal{A})) : Boolean$
The union function in figure 4:

```
def ∪(X : L(A), Y : L(A)) : L(A) :=
    var Z = X;
    for(y in Y):
        if ¬(y ∈_d X):
            Z = Z ::: y;
    return Z;
```

Figure 9: Function $\cup(X : \mathcal{L}(\mathcal{A}), Y : \mathcal{L}(\mathcal{A})) : \mathcal{L}(\mathcal{A})$

and the intersection function in figure 5:

```
def ∩(X : L(A), Y : L(A)) : L(A) :=
    var Z = nil;
    for(x in X):
        if(x ∈_d Y):
            Z = Z ::: x
    return Z;
```

Figure 10: Function $\cap(X : \mathcal{L}(\mathcal{A}), Y : \mathcal{L}(\mathcal{A})) : \mathcal{L}(\mathcal{A})$

The $\mathcal{K}(\mathcal{A})$ finite set definition is actually very convenient to work with, as it is easily convertible back and forth from an abstract set type to a list type $\mathcal{L}(\mathcal{A})$.

6 Conclusion

In section 2 it was established that all B-finite sets are also K-finite, meaning that the K-finite notions is at least as flexible as the B-finite notions. In section 3, by exploring different examples of types within HoTT, it was established that there are types which form provably K-finite sets, but do not satisfy B-finite notions of finiteness. The K-finite notion is therefore strictly more flexible than the B-finite.

The implications of this research is that within HoTT, the K-finite notions of finiteness provide a more flexible and intuitive definition of finiteness within sets. When using HoTT, we should be able to reason about finite sets with the same intuition that we do outside of HoTT; that is, a set with a limited number of elements.

In the opinion of the author, the $\mathcal{K}(\mathcal{A})$ is also more intuitive to understand from the HIT definition. Further research in the field should focus on using the $\mathcal{K}(\mathcal{A})$ as a construction outside of HoTT, and explore its usefulness in a classical mathematics standpoint. The concept of a finite set being built up from a base cases $\emptyset$ and $\{\bullet\}$ and built up from union operations is more intuitive to understand than the standard $\mathcal{B}(\mathcal{A})$ definition of finite sets, in which even some singletons are not considered finite.

Further research is also required to formally define a general definition of $\mathcal{K}(\mathcal{A})$ types that do not satisfy $\mathcal{B}(\mathcal{A})$. These types are interesting in that they behave differently in the context of HoTT versus in classical mathematics.

7 Responsible Research

There are no major ethical concerns for this research project, as the project was based on pure mathematics and no experiments were carried out. HoTT as a foundation for mathematics can be used in proof-checking algorithms; however, this paper did not employ the use of any software. As such, ethical issues of reproducing experimental results, having code available, etc. do not apply.

Where they are used, the theorems of other research papers are clearly referenced such that credit is given to the responsible researchers.

References

[1] Sambin, G. & Smith, J. (1998). An intuitionistic theory of types. Twenty Five Years of Constructive Type Theory. https://doi.org/10.1093/oso/9780198501275.003.0010

[2] Institute for Advanced Study. (2013). Homotopy type theory: Univalent foundations of mathematics.

[3] Awodey, S. (2013). Structuralism, invariance, and Univalence. Philosophia Mathematica, 221, 1-11. https://doi.org/10.1093/philmat/nkt030

[4] Awodey, S., Pelayo, Á. & Warren, M. A. (2013). Voevodsky's Univalence Axiom in homotopy type theory. Notices of the American Mathematical Society, 60(09), 1164. https://doi.org/10.1090/noti1043

[5] Licata, D. R. & Shulman, M. (2013). Calculating the fundamental group of the circle in homotopy type theory. 2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science. https://doi.org/10.1109/lics.2013.28

[6] A new foundation for Mathematics. Research ; Development World. (2014, September 3). Retrieved May 19, 2022, from https://www.rdworldonline.com/a-new-foundation-for-mathematics/

[7] Frumin, D., Geuvers, H., Gondelman, L. & van der Weide, N. (2018). Finite sets in homotopy type theory. Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs. https://doi.org/10.1145/3167085

[8] Lambán Laureano, Romero, A. & Rubio, J. (2010). Constructively Finite? In Contribuciones científicas en honor de Mirian Andrés Gómez (pp. 217−230). essay, Universidad de La Rioja. Servicio de Publicaciones.

[9] Bishop, E. & Bridges, D. (1985). Constructive analysis. Grundlehren Der Mathematischen Wissenschaften. https://doi.org/10.1007/978-3-642-61667-9

[10] Yorgey, B. A., Weirich, S., Zdancewic, S., Carette, J., Pierce, B. & Tannen, V. (2014). Combinatorial species and labelled structures (dissertation). University of Pennsylvania.

[11] Bauer, A., Gross, J., Lumsdaine, P. L. F., Shulman, M., Sozeau, M. & Spitters, B. (2017). The Hott Library: A formalization of homotopy type theory in Coq. Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs. https://doi.org/10.1145/3018610.3018615

[12] Kraus, N. & von Raumer, J. (2019). Path spaces of higher inductive types in homotopy type theory. 2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS). https://doi.org/10.1109/lics.2019.8785661

[13] Beeson, M. (2021). Intuitionistic NF Set Theory. https://doi.org/10.48550/arXiv.2104.00506

[14] Kuratowski, C. (1920). Sur la notion d'ensemble fini. Fundamenta Mathematicae, 1(1).

[15] Rijke, E., & Spitters, B. (2015). Sets in homotopy type theory. Mathematical Structures in Computer Science, 25(5), 1172−1202. https://doi.org/10.1017/s0960129514000553

[16] Ladyman, J., & Presnell, S. (2015). Identity in homotopy type theory, part I: The justification of path induction. Philosophia Mathematica, 23(3). https://doi.org/10.1093/philmat/nkv014

[17] Gilbert, G. (2017). Formalising real numbers in homotopy type theory. Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs. https://doi.org/10.1145/3018610.3018614

[18] Frumin, D. (2018, January). Finite Sets in Homotopy Type Theory. CPP. Nijmegen; Radboud Universiteit. https://cs.ru.nl/ dfrumin/pdf/t/finsets_cpp18.pdf