Evaluating Notions of Finite Sets in Homotopy Type Theory

Giovanni Fincato de Loureiro

Research Question

What is the difference in HoTT implementations when using Kuratowski-finite notions of finite sets versus Bishop-finite notions?

Homotopy Type Theory

HoTT is a development on type theory that seeks to replace the current ZFC set theory with a constructive mathematics foundation[1].

Univalence: Isomorphic structures are not just equivalent, but identical (Equivalence is equivalent to identity)[2]

Figure 1: Coffee mug is equivalent to a taurus[3]

Higher Inductive Types: Inductive types that allow for path constructors[4]

Inductive Type nat : Type :=
| 0 : nat
| S : nat $\rightarrow$ nat

Higher Inductive Type interval : Type :=
| zero : interval
| one : interval
| segment : zero = one

Finite Sets

HoTT must have the computational facilities for finite types

The problem: Constructive mathematics has more than one way of defining whether a type is finite [5]

Bishop-finite: A set is finite if it is equivalent to a canonical finite set {0, … n} for some natural number n
For type A:
$isBf(A) := \Sigma(n : \mathbb{N}), \|A \simeq [n]\|$

The problem: Requires underlying type to have decidable equality [6]

The solution: Definition of finite without needing decidable equality

K-finite: If there is a K-finite set containing all the members of the type, the type is finite
For type A:
$isKf(A) := \Sigma(X : K(A)), \Pi(a : A), a \in X$ [7]

Higher Inductive Type K(A: Type) :=
| $\varnothing$: K(A)
| $\{\cdot\} : A \rightarrow K(A)$
| $\cup : K(A) \rightarrow K(A) \rightarrow K(A)$
| nl : $\Pi(x : K(A)), \varnothing \cup x = x$
| nr : $\Pi(x : K(A)), x \cup \varnothing = x$
| idem : $(x : A), \{x\} \cup \{x\} = \{x\}$
| assoc : $\Pi(x,y,z : K(A)), x \cup (y \cup z) = (x \cup y) \cup z$
| com : $\Pi(x,y : K(A)), x \cup y = y \cup x$
| trunc : $\Pi(x, y : K(A)), \Pi(p,q : x = y), p = q$

In classical mathematics, with LEM, these notions are identical. In HoTT, not necessarily.

Examples

B-finite types are a proper subset of K-finite[7]:

$$isBf(A) \rightarrow isKf(A)$$

Use example of type A such that

$$isKf(A) \wedge \neg isBf(A)$$

Circles: S^1

Higher Inductive Type S^1 : Type :=
| base : S^1
| loop : base = base

Loop cannot be algorithmically compared
→ circle does not have decidable equality
→ circle type is not B-finite: $\neg isBf(S^1)$

Take set {base}, singleton containing only base
Use truncation $\|base=base\| \rightarrow \Pi(x : S^1), x \in \{base\}$
→ circle type is contained in a singleton
→ all singletons are K-finite
→ $isKf(S^1)$

2-Sphere S^2 [8]

Higher Inductive Type S^2 : Type :=
| base2 : S^2
| surf2 : $refl_{base} = refl_{base}$ in base = base

Surf2 also cannot be algorithmically compared
Truncation $\|base=base\| \rightarrow \Pi(x : S^2), x \in \{base2\}$

$$isKf(S^2) \wedge \neg isBf(S^2)$$

General n-spheres: $base^n$, loop $\Omega^n(S^n, base^n)$[8]
For all n, we get same result of
$$isKf(S^n) \wedge \neg isBf(S^n)$$

Conclusions

K-finite notion is more flexible than B-finite

B-finite notion in HoTT leads to unintuitive results, such as singletons which are not finite

K-finite notions provide the computational facilities expected of finite sets ($\cup$, $\in$, $\cap$)

Implement as list data type $L(A) \simeq K(A)$
Intuitively builds from nil and {a : A}
By using "for" loops to iterate through items in the list, can easily build equivalent functions for $\cup$, $\in$, $\cap$ in lists

References

[1] Awodey, S. (2013). Structuralism, invariance, and Univalence. Philosophia Mathematica, 22(1), 1–11.
[2] AWODEY, STEVE, et al. "Introduction – from Type Theory and Homotopy Theory to Univalent Foundations." Mathematical Structures in Computer Science, vol. 25, no. 5, 2015
[3] Henry Segerman, www.youtube.com/watch?v=9NlqYr6-TpA
[4] Kraus, Nicolai, and Jakob Von Raumer. "Path Spaces of Higher Inductive Types in Homotopy Type Theory." 2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)
[5] Atkey, Robert, and Neelakantan Krishnaswami. "Proceedings 6th Workshop on Mathematically Structured Functional Programming." Electronic Proceedings in Theoretical Computer Science, vol. 207, 2016, doi:10.4204/eptcs.207.0.
[6] Bishop, Errett, and Douglas Bridges. "Constructive Analysis." Grundlehren Der Mathematischen Wissenschaften, 1985
[7] Frumin, Dan, et al. "Finite Sets in Homotopy Type Theory." Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs, 2018
[8] Institute for Advanced Study. (2013). Homotopy type theory: Univalent foundations of mathematics.