Higher Inductive Types Via Impredicative Encodings

M.Sc. thesis supervised by Dr Benno van den Berg

Stefano Volpe

Institute for Logic, Language and Computation

August 27, 2025

Prerequisites

From the **type theory/FP/LoVe** course:

- inductive types;
- dependent types.

Outline

1. Higher Inductive Types
2. Impredicative encodings
3. My thesis

1.1 Intuition

dependent type theory	type	term
homotopy theory	space	point

1.1 Intuition

dependent type theory	type	term
homotopy theory	space	point

An inductive type is freely generated by the **(point) constructors** in its **signature**.

```
data ℕ : Type where
  𝕆 : ℕ
  𝕊 : ℕ → ℕ
```

1.1 Intuition

dependent type theory	type	term
homotopy theory	space	point

An inductive type is freely generated by the **(point) constructors** in its **signature**.

```
data ℕ : Type where
  𝟘 : ℕ
  𝕊 : ℕ → ℕ
```

dependent type theory	identity type	identity proof
homotopy theory	path space	path

1.1 Intuition

dependent type theory	type	term
homotopy theory	space	point

An inductive type is freely generated by the **(point) constructors** in its **signature**.

```
data ℕ : Type where
  𝟘 : ℕ
  𝕊 : ℕ → ℕ
```

dependent type theory	identity type	identity proof
homotopy theory	path space	path

Higher inductive types are freely generated by point and **path constructors**.

1.2 Homotopy Theory via HITs

```
data 𝕀 : Type where -- 𝕀₀ ∘————————∘ 𝕀₁
  𝕀₀ : 𝕀             --        seg
  𝕀₁ : 𝕀
  seg : 𝕀₀ ≡ 𝕀₁
```

1.2 Homotopy Theory via HITs

```agda
data 𝕀 : Type where -- 𝕀₀ ∘────────∘ 𝕀₁
  𝕀₀ : 𝕀              --      seg
  𝕀₁ : 𝕀
  seg : 𝕀₀ ≡ 𝕀₁

data S¹ : Type where --             loop
  base : S¹          --
  loop : base ≡ base --        base
```

1.3 Algebra via HITs

If a free algebraic object has no computable forms for its terms, how do we construct it?

```
data FreeSemigroup (A : Type) {h : isSet A} : Type where
  η : A → FreeSemigroup A
  _∘_ : FreeSemigroup A {h} → FreeSemigroup A {h} → FreeSemigroup A
  associative : (a b c : FreeSemigroup A {h}) → (a ∘ b) ∘ c ≡ a ∘ (b ∘ c)
  truncated : isSet (FreeSemigroup A)
```

1.4 Programming via HITs

```
data ℕ/3ℕ : Type where
  𝟘 : ℕ/3ℕ
  𝕊 : ℕ/3ℕ → ℕ/3ℕ
  mod : 𝟘 ≡ 𝕊 (𝕊 (𝕊 𝟘))
  truncated : isSet ℕ/3ℕ
```

1.4 Programming via HITs

```
data ℕ/3ℕ : Type where
  𝟘 : ℕ/3ℕ
  𝕊 : ℕ/3ℕ → ℕ/3ℕ
  mod : 𝟘 ≡ 𝕊 (𝕊 (𝕊 𝟘))
  truncated : isSet ℕ/3ℕ

data Fin (A : Type) : Type where
  ∅ : Fin A
  L : A → Fin A
  _∪_ : Fin A  → Fin A → Fin A
  assoc : (x y z : Fin A) → x ∪ (y ∪ z) ≡ (x ∪ y) ∪ z
  identityᵣ : (x : Fin A) → x ∪ ∅ ≡ x
  identityₗ : (x : Fin A) → ∅ ∪ x ≡ x
  commutativity : (x y : Fin A) → x ∪ y ≡ y ∪ x
  idempotence : (x : A) → L x ∪ L x ≡ L x
  truncated : isSet (Fin A)
```

Outline

2.1 Encodings?

Rather than assuming its existence, can we **encode** a generic HIT, together with its point and path constructors, as

2.1 Encodings?

Rather than assuming its existence, can we **encode** a generic HIT, together with its point and path constructors, as

- an algebra $(D, c, p) : \mathrm{Alg}$ within our theory and,

2.1 Encodings?

Rather than assuming its existence, can we **encode** a generic HIT, together with its point and path constructors, as

- an algebra $(D, c, p) : \mathrm{Alg}$ within our theory and,
- for all $(E, d, q) : \mathrm{Alg}$, a function $\mathrm{rec}_{(E,d,q)} : D \to E$?

2.1 Encodings?

β-rules $\leftrightarrow$ "$\mathrm{rec}_{(E,d,q)}$ is an algebra morphism for any algebra (E, d, q)"

2.1 Encodings?

β-rules $\leftrightarrow$ "$\mathrm{rec}_{(E,d,q)}$ is an algebra morphism for any algebra (E, d, q)"

η-rule $\leftrightarrow$ "$\mathrm{rec}_{(E,d,q)}$ is the **only** algebra morphism for any algebra (E, d, q)"

2.2 Impredicative?

A construction is called **“impredicative”** if it quantifies over a type universe including the type being defined.

Outline

3.1 The System

A dependent type theory with

- Π-types,
- Σ-types,
- intensional identity, and
- function extensionality

whose bottom universe is impredicative:

$$\frac{\Gamma, a : A \vdash B : \mathcal{U}_0}{\Gamma \vdash \prod_{x:A} B : \mathcal{U}_0}.$$

3.2 Working Definition of HIT

To date, there is no agreement on a general signature definition for HITs. We encode two:

- Van der Weide's HITs;
- $\mathcal{W}$-suspensions.

3.3 Set-Truncated Van der Weide HITs

Set-truncated type

Type A is set-truncated $:\equiv \prod_{x,y:A}$ ("$x =_A y$" is proof-irrelevant)

3.3 Set-Truncated Van der Weide HITs

Encoding idea:

1. naïve encoding: $D :\equiv \prod_{(E,e,q) : \text{Alg}} E$

3.3 Set-Truncated Van der Weide HITs

Encoding idea:

1. naïve encoding: $D :\equiv \prod_{(E,e,q):\,\mathrm{Alg}} E$
2. + naturality, i.e. equipping $x : D$ with a witness of

$$f(x(E,e,q)) =_{E'} x(E',e',q')$$

   for any morphism $f : (E,e,q) \to (E',e',q')$.

3.4 General W-Suspensions

Algebras can encode **recursive definitions**, i.e. eliminations into a type.

3.4 General W-Suspensions

Algebras can encode **recursive definitions**, i.e. eliminations into a type.

Fibered algebras can encode **inductive proofs**, i.e. eliminations into a type family.

3.4 General W-Suspensions

Encoding idea:

1. naïve encoding: $D :\equiv \prod_{(E,e,q):\ \mathrm{Alg}} E$

3.4 General W-Suspensions

Encoding idea:

1. naïve encoding: $D :\equiv \prod_{(E,e,q) :\, \mathrm{Alg}} E$
2. + inductivity, i.e. equipping $x : D$ with a witness of

$$\prod_{(E',e',q') :\, \mathrm{FibAlg}} E' x$$

3.4 General W-Suspensions

Encoding idea:

1. naïve encoding: $D :\equiv \prod_{(E,e,q):\ \mathrm{Alg}} E$
2. + inductivity, i.e. equipping $x : D$ with a witness of

$$\prod_{(E',e',q'):\ \mathrm{FibAlg}} E'x$$

3. + inductivity (again)

3.5 Contributions

1. encodings of **Van der Weide's HITs** that eliminate into **set-truncated** types of the impr. universe;

3.5 Contributions

1. encodings of **Van der Weide's HITs** that eliminate into **set-truncated** types of the impr. universe;
2. **full formalisation** of (1) in Agda;

3.5 Contributions

1. encodings of **Van der Weide's HITs** that eliminate into **set-truncated** types of the impr. universe;
2. **full formalisation** of (1) in Agda;
3. encodings of **$\mathcal{W}$-suspensions** that eliminate into the impr. universe.

3.6 Limitations

As per usual: no big elimination. Even if you omit higher universes, our encodings still cannot index type families.

Thank you!

3.7 References

[1] Agda Community. 2024. Agda. Retrieved from https://wiki.portal.chalmers.se/agda

[2] Agda Community. 2024. Agda Standard Library. Retrieved from https://github.com/agda/agda-stdlib

[3] Agda Community. 2025. Cubical Agda Library. Retrieved from https://github.com/agda/cubical

[4] Steve Awodey, Jonas Frey, and Sam Speight. 2018. Impredicative Encodings of (Higher) Inductive Types. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '18)*, July 2018. ACM, 76–85. https://doi.org/10.1145/3209108.3209130

3.7 References

[5] Steve Awodey, Nicola Gambino, and Kristina Sojakova. 2012. Inductive Types in Homotopy Type Theory. In *Logic in Computer Science, Symposium on*, June 2012. IEEE Computer Society, 95–104. https://doi.org/10.1109/LICS.2012.21

[6] Steve Awodey, Nicola Gambino, and Kristina Sojakova. 2017. Homotopy-Initial Algebras in Type Theory. *J. ACM* 63, 6 (January 2017). https://doi.org/10.1145/3006383

[7] Henning Basold, Herman Geuvers, and Niels van der Weide. 2017. Higher Inductive Types in Programming. *23*, 63–88. https://doi.org/10.3217/jucs-023-01-0063

3.7 References

[8] Steven Bronsveld, Herman Geuvers, and Niels van der Weide. 2025. Impredicative Encodings of Inductive and Coinductive Types. 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. https://doi.org/10.4230/LIPICS.FSCD.2025.11

[9] Steven Bronsveld. 2024. Impredicative Encodings of Inductive and Coinductive Types. Master's thesis. https://doi.org/10.48550/arXiv.2505.13495

[10] Paolo Capriotti and Nicolai Kraus. 2017. Univalent higher categories via complete Semi-Segal types. *Proceedings of the ACM on Programming Languages* 2, POPL (December 2017), 1–29. https://doi.org/10.1145/3158132

[11] Floris van Doorn. 2016. Constructing the propositional truncation using non-recursive HITs. In *Proceedings of the 5th ACM SIGPLAN Conference on Certified*

3.7 References

Programs and Proofs (CPP 2016), January 2016. ACM, 122–129. https://doi.org/10.1145/2854065.2854076

[12] Dan Frumin, Herman Geuvers, Léon Gondelman, and Niels van der Weide. 2018. Finite sets in homotopy type theory. In *Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP 2018)*, 2018. Association for Computing Machinery, 201–214. https://doi.org/10.1145/3167085

[13] Ambrus Kaposi and András Kovács. 2020. Signatures and Induction Principles for Higher Inductive-Inductive Types. *Logical Methods in Computer Science* (February 2020). https://doi.org/10.23638/lmcs-16(1:10)2020

3.7 References

[14] Ambrus Kaposi, András Kovács, and Thorsten Altenkirch. 2019. Constructing quotient inductive-inductive types. *Proceedings of the ACM on Programming Languages* 3, POPL (January 2019), 1–24. https://doi.org/10.1145/3290315

[15] Nicolai Kraus. 2016. Constructions with Non-Recursive Higher Inductive Types. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '16)*, July 2016. ACM, 595–604. https://doi.org/10.1145/2933575.2933586

[16] Tom Leinster. 2014. *Basic Category Theory*. Cambridge University Press. https://doi.org/10.1017/CBO9781107360068

3.7 References

[17] Per Martin-Löf. 1975. An Intuitionistic Theory of Types: Predicative Part. In *Logic Colloquium '73, Proceedings of the Logic Colloquium*. Elsevier, 73–118. https://doi.org/10.1016/s0049-237x(08)71945-1

[18] Per Martin-Löf. 1982. Constructive Mathematics and Computer Programming. In *Logic, Methodology and Philosophy of Science VI, Proceedings of the Sixth International Congress of Logic, Methodology and Philosophy of Science*. Elsevier, 153–175. https://doi.org/10.1016/s0049-237x(09)70189-2

[19] Egbert Rijke. 2017. The join construction. https://doi.org/10.48550/ARXIV.1701.07538

[20] Egbert Rijke. 2025. *Introduction to Homotopy Type Theory*. Cambridge University Press. Retrieved from https://www.cambridge.org/core/books/

3.7 References

introduction-to-homotopy-type-theory/0DD31EC06C80797A50ACE807251E80B6

[21] Xavier Ripoll Echeveste. 2023. Alternative Impredicative Encodings of Inductive Types. Master's thesis. Retrieved from https://eprints.illc.uva.nl/id/eprint/2272

[22] Michael Shulman. 2014. Splitting Idempotents. *Homotopy Type Theory*. Retrieved from https://homotopytypetheory.org/2014/12/08/splitting-idempotents/

[23] Michael Shulman. 2014. Splitting Idempotents, II. *Homotopy Type Theory*. Retrieved from https://homotopytypetheory.org/2014/12/22/splitting-idempotents-ii/

3.7 References

[24] Michael Shulman. 2017. Idempotents in intensional type theory. *Logical Methods in Computer Science* (April 2017). https://doi.org/10.2168/lmcs-12(3:9)2016

[25] Michael Shulman. 2018. Impredicative Encodings, Part 3. *Homotopy Type Theory*. Retrieved from https://homotopytypetheory.org/2018/11/26/impredicative-encodings-part-3

[26] Kristina Sojakova. 2015. Higher Inductive Types as Homotopy-Initial Algebras. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '15)*, January 2015. ACM, 31–42. https://doi.org/10.1145/2676726.2676983

3.7 References

[27] Sam Speight. 2018. Impredicative Encodings of Inductive Types in Homotopy Type Theory. Master's thesis. Retrieved from https://www.cs.ox.ac.uk/people/sam.speight/publications/sams-hott-thesis.pdf

[28] Morten Heine Sørensen and Pawel Urzyczyn. 2006. *Lectures on the Curry-Howard isomorphism*. Elsevier. Retrieved from https://www.sciencedirect.com/bookseries/studies-in-logic-and-the-foundations-of-mathematics/vol/149/suppl/C

[29] The Univalent Foundations Program. 2013. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Retrieved from https://homotopytypetheory.org/book

3.7 References

[30] Andrea Vezzosi, Anders Mörtberg, and Andreas Abel. 2021. Cubical Agda: A dependently typed programming language with univalence and higher inductive types. *Journal of Functional Programming* 31, (2021), e8. https://doi.org/10.1017/S0956796821000034

[31] Niels van der Weide and Herman Geuvers. 2019. The Construction of Set-Truncated Higher Inductive Types. *Electronic Notes in Theoretical Computer Science* 347, (November 2019), 261–280. https://doi.org/10.1016/j.entcs.2019.09.014

[32] Niels van der Weide. 2020. Constructing Higher Inductive Types as Groupoid Quotients. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '20)*, July 2020. ACM, 929–943. https://doi.org/10.1145/3373718.3394803

3.7 References