Cubical Type Theory: From $i_0$ to $i_1$

Gergő Érdi
http://gergo.erd.hu/

Haskell.SG
December 2018.
How many Agda programmers does it take to change a lightbulb?
How many Agda programmers does it take to change a lightbulb?

Are you kidding me? It takes two PhD’s six months just to prove that the socket and the bulb are wound in the same direction!
1. Martin-Löf Type Theory
Type Theory

- Single unified language for objects and propositions (c.f. ZF set theory + FOL)
- Dependent types give us predicate logic (via Curry-Howard)
- Type formers, eliminators, $\beta$-rules
MLTT types

- \( U \): the type of types (called \( \text{Set} \) in Agda)
- \( \bot, \top, \text{Bool} \)
- \( \Pi, \Sigma \)
- Inductive datatypes (e.g. \( \mathbb{N} \))
Equality in MLTT

\[ \text{Id} \ A \ x \ y : U \]

Its sole constructor is \text{refl} : \forall x \rightarrow \text{Id} \ x \ x

Definitional equality: everything can only be equal to itself.
Properties of \( \text{Id} \)

Axiom \( J \): eliminator for identity type

\[
J : (P : (x y : A) \rightarrow \text{Id} \ x \ y \rightarrow \text{Set}) \rightarrow \\
(\forall x \rightarrow P \ x \ x \ (\text{refl} \ x)) \rightarrow \\
\forall \{x \ y : A\} \ (p : \text{Id} \ x \ y) \rightarrow P \ x \ y \ p
\]

From this, we can prove that \( \text{Id} \) is an equivalence relation.
Properties of $\text{Id}$ (cont.d)

Uniqueness of identity types:

$$\text{UIP} : \{x, y : A\} \rightarrow (p, q : \text{Id} x y) \rightarrow \text{Id} p q$$

Axiom K: equivalent to UIP

$$\text{K} : \forall (x : A) \rightarrow (P : \text{Id} x x \rightarrow \text{Set}) \rightarrow$$

$$P (\text{refl} x) \rightarrow$$

$$\forall (p : \text{Id} x x) \rightarrow P p$$

UIP / K are independent of (but compatible with) MLTT.
Properties of $\text{Id}$ (cont.d)

Function extensionality:

$$\text{funExt} : (f \ g : (x : A) \to B \ x) \to$$

$$(\forall x \to \text{Id} \ (f \ x) \ (g \ x)) \to$$

$$\text{Id} \ f \ g$$

Function extensionality is independent of (but compatible with) MLTT.
2. Topological homotopies
Spaces and paths

In some topological space $A$ and two points $x, y : A$, a path $p$ from $x$ to $y$ (or, $p : x \leadsto y$) is:

$p : [0, 1] \to A, p \in C \text{ s.t.}$
$p(0) = x, p(1) = y$
If $f, g : A \to B, f, g \in C$, then a homotopy $H$ between $f$ and $g$ is:

$$H : A \times [0, 1] \to B, H \in C \text{ s.t.}$$

$$H(x, 0) = f(x)$$
$$H(x, 1) = g(x)$$
Homotopies between paths

If $p, q : x \rightsquigarrow y$, then as a special case, a homotopy $H$ between $p$ and $q$ is:

$$H : [0, 1] \times [0, 1] \rightarrow A, H \in C \text{ s.t.}$$

$$H(i, 0) = p(i)$$
$$H(i, 1) = q(i)$$
$$H(0, j) = x$$
$$H(1, j) = y$$

This can be iterated.
Paths as equalities?

Paths between points are a bit like equalities between them: they are reflexive (trivial path), symmetric (just go backwards) and transitive (concatenation).

But what does that mean?
3. Homotopy Type Theory
Basic idea: types are spaces, and the paths in that space (written \(_\equiv_\)) correspond to equalities.

- This only makes sense if all functions are continuous
  - Trivially true for discrete spaces
- Paths have structure, so UIP doesn’t hold
- Paths are purely synthetic, we’re not putting \([0, 1] \subseteq \mathbb{R}\) at the base of our formal system...
Are there any non-discrete spaces?

- **U** is a type, so some types A and B are points in that space. When is there a path between them?

- *Univalence axiom*: the paths in **U** are equivalent to *equivalences*, i.e. invertible functions modulo paths. This is highly desirable!

- Different equivalences yield different paths (e.g. *id* vs. *not* for *Bool*)

- Function extensionality can be proven from UA
Non-discrete spaces by fiat

Might as well use this rich structure of paths!

*Higher inductive type:* similar to an inductive datatype, but constructors for not only points, but paths, paths between paths, etc.

```haskell
data Circle : Set where
  base : Circle
  loop : base ≡ base
```

This *generates* a space via the algebra of paths; e.g.

```
trans loop loop : base ≡ base.
```
We can represent the integers \( \mathbb{Z} \) as \( \mathbb{N} \times \mathbb{N} / \sim \) where 
\[
(x, y) \sim (x', y') := (x + y') \equiv (x' + y).
\]
HIT example: \( \mathbb{Z} \)

Written out as a HIT:

\[
\text{Same} : \mathbb{N} \to \mathbb{N} \to \mathbb{N} \to \mathbb{N} \to _-
\]
\[
\text{Same } x \ y \ x' \ y' = x + y' \equiv x' + y
\]

\text{data } \mathbb{Z} : \text{Set where}

\[
_-_- : \mathbb{N} \to \mathbb{N} \to \mathbb{Z}
\]
\[
\text{quot} : \forall x \ y \ x' \ y' \to \text{Same } x \ y \ x' \ y'
\to x - y \equiv x' - y'
\]
Continuity in this space: representation-invariance.

Enforced by the type system: functions are defined over points and paths at the same time.

For example, if we want to do doubling:

\[
\text{double} : \mathbb{Z} \rightarrow \mathbb{Z} \\
\text{double} (x - y)) = 2 \ast x - 2 \ast y
\]

we also have to give

\[
\text{double} (\text{quot } x \ y \ x' \ y' \ \text{eq}) = \\
\text{quot } (2 \ast x) (2 \ast y) (2 \ast x') (2 \ast y') \text{ arithmetic-prf}
\]
Summary

• MLTT, paths as equality, no $K$
• Univalence added as an axiom
• All functions continuous by construction
• Function extensionality is a theorem
• Higher inductive types (and more...)

Big **BUT:**
Summary

- MLTT, paths as equality, no $K$
- Univalence added as an axiom
- All functions continuous by construction
- Function extensionality is a theorem
- Higher inductive types (and more...)

**Big **BUT: HoTT postulates the Univalence Axiom with no computational content
4. Cubical Type Theory
Representations of paths

- **Topology:** $p : [0, 1] \rightarrow A, p \in C$:
  “continuously-infinitely detailed”, $p\left(\frac{1}{\pi}\right)$ etc.

- **Homotopy Type Theory:** $p : \{0, 1\} \rightarrow A$? But no UIP, so it does have structure? But not enough to support computation?
Representations of paths

- **Topology:** $p : [0, 1] \rightarrow A, p \in C$: “continuously-infinitely detailed”, $p\left(\frac{1}{\pi}\right)$ etc.

- **Homotopy Type Theory:** $p : \{0, 1\} \rightarrow A$? But no UIP, so it does have structure? But not enough to support computation?

- **Cubical Type Theory:** $p : I \rightarrow A$, where $I$ is some formal version of $[0, 1]$
Paths, algebraically

$I$ is the free distributive lattice (of countably infinite, distinct direction variables):

\[
\begin{align*}
i_0 & : I \\
\sim & : I \to I \\
\_ \lor _{} & : I \to I \to I \\
\_ \land _{} & : I \to I \to I \\
\end{align*}
\]

This has decidable equality!
Paths, algebraically

$I$ is the free distributive lattice (of countably infinite, distinct direction variables):

\[
i0, i1 : I
\]

\[
\sim : I \to I
\]

\[
\_ \_ \lor : I \to I \to I
\]

\[
\_ \_ \land : I \to I \to I
\]

This has decidable equality!

We then represent a path $p : x \equiv y$ by a function $p : I \to A$ s.t. $p i0 = x$ and $p i1 = y$. 
refl and sym are easy theorems

Unlike in HoTT, path reflexivity and symmetry are no longer axioms:

\[
\text{refl} : \{x : A\} \to x \equiv x \\
\text{refl } \{x\} = \lambda i \to x
\]

\[
\text{sym} : \forall \{x \ y : A\} \to x \equiv y \to y \equiv x \\
\text{sym } p = \lambda i \to p (~ i)
\]
Equality-like behaviour

\[
\text{cong} : (f : A \rightarrow B) \{x \ y : A\} \rightarrow x \equiv y \rightarrow f \ x \equiv f \ y \\
\text{cong } f \ p = \lambda \ i \rightarrow f \ (p \ i)
\]
Equality-like behaviour

\[
\text{cong} : (f : (x : A) \to B \, x) \{x \, y : A\} \to \\
(p : x \equiv y) \to \text{PathP} \, (\lambda \, i \to B \, (p \, i)) \, (f \, x) \, (f \, y) \\
\text{cong} \, f \, p = \lambda \, i \to f \, (p \, i)
\]
Equality-like behaviour

funExt : \{f \; g : (x : A) \rightarrow B \; x\} \rightarrow
(\forall \; x \rightarrow f \; x \equiv g \; x) \rightarrow f \equiv g

funExt \; p = \lambda \; i \rightarrow (\lambda \; x \rightarrow p \; x \; i)
What about transitivity?

If \( p : x \equiv y \) and \( q : y \equiv z \), how do we make

\[
\text{trans } p \ q = \lambda \ i \ \rightarrow \begin{cases} 
  p(2i) & \text{if } i \leq 0.5 \\
  q(2i - 1) & \text{if } i \geq 0.5 
\end{cases}
\]
Path composition

The primitive operation that supports transitivity, and many other ways of composing paths, is: given the bottom of a “box“, and a system of consistent sides, we can construct the lid.
Transitivity via \( \text{comp} \)

\[
\text{trans} : x \equiv y \rightarrow y \equiv z \rightarrow x \equiv z
\]

\[
\text{trans } p \ q \ i = \text{comp} \ (\lambda \_ \rightarrow A)
\]

\[
(\lambda \{ \ j \ (i = i0) \rightarrow x \\
; j \ (i = i1) \rightarrow q \ j \\
\} \ \\
(\text{inc} \ (p \ i))
\]
A sliding version

\[
\text{slidingLid} : (p_0 : x \equiv y) \ (p_1 : x' \equiv y') \ (q : x \equiv x') \rightarrow \\
\forall i \rightarrow p_0 \ i \equiv p_1 \ i
\]

\[
\text{slidingLid} \ p_0 \ p_1 \ q \ i \ j = \text{comp} \ (\lambda \_ \rightarrow A) \\
(\lambda \{ k \ (j = i0) \rightarrow p_0 \ (i \land k) \\
; \ k \ (j = i1) \rightarrow p_1 \ (i \land k) \\
; \ k \ (i = i0) \rightarrow q \ j \\
}) \\
(\text{inc} \ (q \ j))
\]
double, cubically

double : \mathbb{Z} \rightarrow \mathbb{Z}

double (x - y) = (2 \times x) - (2 \times y)

double (\text{quot } x \ y \ x' \ y' \ p \ i) =

\text{quot } (2 \times x) (2 \times y) (2 \times x') (2 \times y') p' \ i

where

p' : 2 \times x + 2 \times y' \equiv 2 \times x' + 2 \times y

p' = \text{arithmetic-proof } x \ y \ p
double, cubically

\[
\begin{align*}
  x - y & \quad \text{quot} \quad p \quad i \quad \rightarrow \quad x' - y' \\
  2x - 2y & \quad \text{quot} \quad p' \quad i' \quad \rightarrow \quad 2x' - 2y'
\end{align*}
\]
A non-unary example: $\mathbb{Z}$ addition
A problem:

What if there is no way to continuously deform

\text{slidingLid } p_0 \ p_1 \ q_0 \ i1

(a homotopically transformed proof)

into

\[ q_1 \]

(an arithmetic proof about natural numbers)
We define \(\mathbb{Z}\) not to have any holes by adding a third constructor (à la HoTT §6.10):

\[
\text{data } \mathbb{Z} : \text{Set} \ \text{where}
\]
\[
\_\_\_ : \mathbb{N} \to \mathbb{N} \to \mathbb{Z}
\]
\[
\text{quot} : \forall \ x \ y \ x' \ y' \to \text{Same } x \ y \ x' \ y' \to x - y \equiv x' - y'
\]
\[
\text{trunc} : \forall \ \{x \ y : \mathbb{Z}\} \to (p \ q : x \equiv y) \to p \equiv q
\]

More cases to handle in functions, but more possibilities in constructing results.
We didn’t talk about

- Details of equivalences
- Univalence (a *theorem* in CTT) and glueing in general
Future project ideas

- Prove \((\mathbb{Z}, +)\) is an Abelian group
- Prove \(\mathbb{Z} \simeq \text{Int}\) (from the standard library)
- Prove \(\mathbb{Z} \simeq \text{base} \equiv \text{base}\) (in Circle)