Cubical Type Theory: From `i0` to `i1`

Gergő Érdi
http://gergo.erdi.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, β-rules

MLTT types

- U: the type of types (called Set in Agda)
- $\bot$, $\top$, Bool
- Π, Σ
- Inductive datatypes (e.g. $\mathbb{N}$)

Equality in MLTT

$\mathsf{Id}\ A\ x\ y : \mathsf{U}$

Its sole constructor is $\mathsf{refl} : \forall\ \mathsf{x} \longrightarrow \mathsf{Id}\ \mathsf{x}\ \mathsf{x}$

Definitional equality: everything can only be equal to itself.

Properties of Id

Axiom J: eliminator for identity type

$$
\begin{aligned}
&\mathsf{J} : (P : (x\ y : A) \longrightarrow \mathsf{Id}\ x\ y \longrightarrow \mathsf{Set}) \longrightarrow \\
&\quad (\forall\ x \longrightarrow P\ x\ x\ (\mathsf{refl}\ x)) \longrightarrow \\
&\quad \forall\ \{x\ y : A\}\ (p : \mathsf{Id}\ x\ y) \longrightarrow P\ x\ y\ p
\end{aligned}
$$

From this, we can prove that Id is an equivalence relation.

Properties of Id (*cont.d*)

Uniqueness of identity types:

$$\mathsf{UIP} : \{x\ y : A\} \longrightarrow (p\ q : \mathsf{Id}\ x\ y) \longrightarrow \mathsf{Id}\ p\ q$$

Axiom K: equivalent to UIP

$$\begin{aligned}&\mathsf{K} : \forall\,(x : A) \longrightarrow (P : \mathsf{Id}\ x\ x \longrightarrow \mathsf{Set}) \longrightarrow\\&\quad P\,(\mathsf{refl}\ x) \longrightarrow\\&\quad \forall\,(p : \mathsf{Id}\ x\ x) \longrightarrow P\ p\end{aligned}$$

UIP / K are independent of (but compatible with) MLTT.

Properties of Id (*cont.d*)

Function extensionality:

$$\begin{aligned}&\mathsf{funExt} : (f\,g : (x : A) \longrightarrow B\,x) \longrightarrow\\&\quad (\forall\, x \longrightarrow \mathsf{Id}\,(f\,x)\,(g\,x)) \longrightarrow\\&\quad \mathsf{Id}\,f\,g\end{aligned}$$

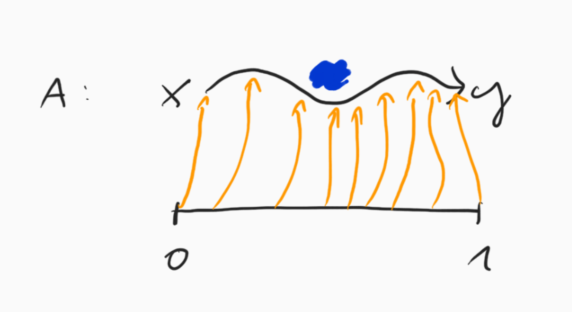
Function extensionality is independent of (but compatible with) MLTT.

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, $p : x \rightsquigarrow y$) is:

$p : [0, 1] \to A, p \in C$ s.t.
$p(0) = x, p(1) = y$



Homotopies

If $f, g : A \to B$, $f, g \in C$, then a homotopy H between f and g is:

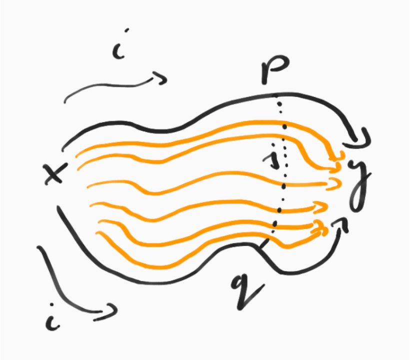
$H : A \times [0, 1] \to B, H \in C$ s.t.
$H(x, 0) = f(x)$
$H(x, 1) = g(x)$

Homotopies between paths

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

$H : [0,1] \times [0,1] \to A, H \in C$ 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

Type Theory with Paths

Basic idea: types are spaces, and the paths in that space (written _≡_) 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?

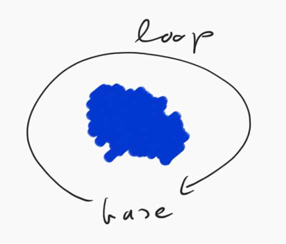
- U is a type, so some types *A* and *B* are points in that space. When is there a path between them?
- *Univalence axiom*: the paths in U are equivalent to *equivalences*, i.e. invertible functions modulo paths. This is highy desirable!
- Different equivalences yield different paths (e.g. *id* vs. *not* 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.

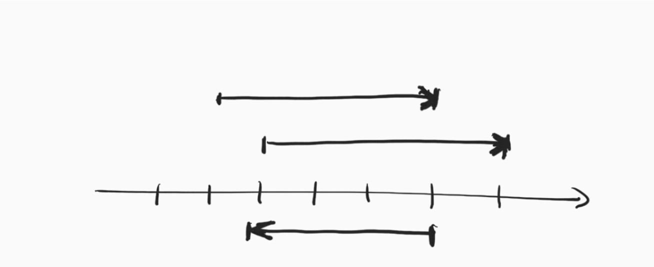
```
data Circle : Set where
  base : Circle
  loop : base ≡ base
```



This *generates* a space via the algebra of paths; e.g. trans loop loop : base ≡ base.

HIT example: $\mathbb{Z}$

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: ℤ

Written out as a HIT:

```
Same : ℕ → ℕ → ℕ → ℕ → _
Same x y x′ y′ = x + y′ ≡ x′ + y

data ℤ : Set where
  _-_ : ℕ → ℕ → ℤ
  quot : ∀ x y x′ y′ → Same x y x′ y′
    → x - y ≡ x′ - y′
```

Functions over $\mathbb{Z}$

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:

$$
\begin{aligned}
&\text{double} \ : \ \mathbb{Z} \ \rightarrow \mathbb{Z} \\
&\text{double}\ (x \text{ - } y)) \ = \ 2 * x \text{ - } 2 * y
\end{aligned}
$$

we also have to give

$$
\begin{aligned}
&\text{double}\ (\text{quot}\ x\ y\ x'\ y'\ eq) \ = \\
&\quad \text{quot}\ (2 * x)\ (2 * y)\ (2 * x')\ (2 * y')\ \textit{arithmetic-prf}
\end{aligned}
$$

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: $p : [0,1] \to A, p \in C$:
  "continuously-infinitely detailed", $p(\frac{1}{\pi})$ etc.
- Homotopy Type Theory: $p : \{0,1\} \to A$? But no UIP, so it does have structure? But not enough to support computation?

Representations of paths

- Topology: $p : [0,1] \to A, p \in C$: "continuously-infinitely detailed", $p(\frac{1}{\pi})$ etc.
- Homotopy Type Theory: $p : \{0,1\} \to A$? But no UIP, so it does have structure? But not enough to support computation?
- Cubical Type Theory: $p : I \to A$, where I is some formal version of $[0,1]$

Paths, algebraically

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

```
i0 i1 : I

~_  : I → I
_∨_ : I → I → I
_∧_ : I → I → I
```

This has decidable equality!

Paths, algebraically

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

```
i0 i1 : I

~_  : I → I
_∨_ : I → I → I
_∧_ : I → I → I
```

This has decidable equality!

We then represent a path $p : x \equiv y$ by a function p : I → *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:

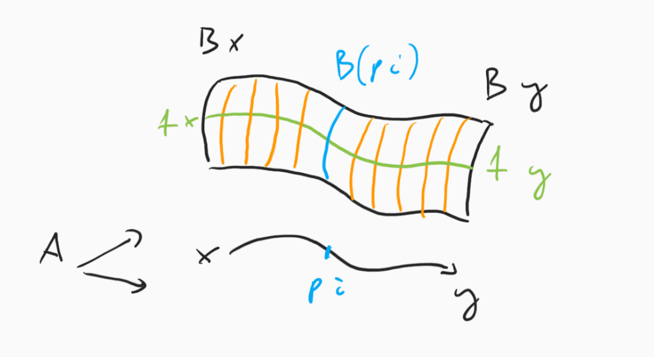
```
refl : {x : A} → x ≡ x
refl {x} = λ i → x

sym : ∀ {x y : A} → x ≡ y → y ≡ x
sym p = λ i → p (~ i)
```

Equality-like behaviour

$$\mathsf{cong} : (f : A \longrightarrow B)\ \{x\ y : A\} \longrightarrow x \equiv y \longrightarrow f\,x \equiv f\,y$$
$$\mathsf{cong}\ f\ p = \lambda\ i \longrightarrow f\ (p\ i)$$

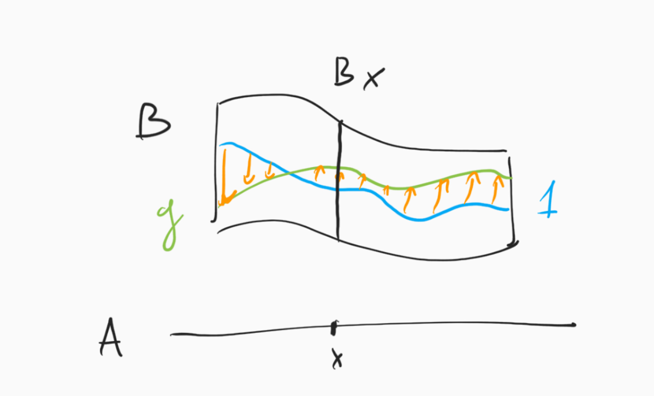
Equality-like behaviour

```
cong : (f : (x : A) → B x) {x y : A} →
  (p : x ≡ y) → PathP (λ i → B (p i)) (f x) (f y)
cong f p = λ i → f (p i)
```



Equality-like behaviour

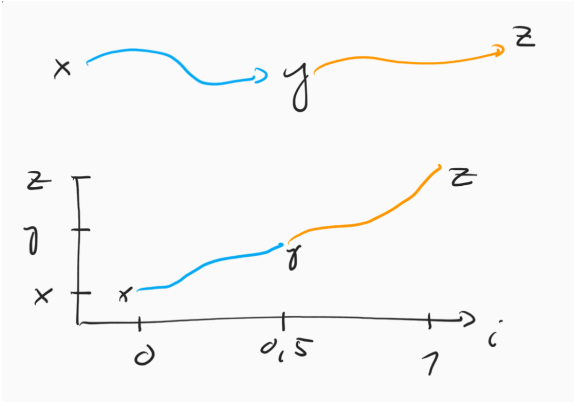
$$\text{funExt} : \{f\ g : (x : A) \longrightarrow B\ x\} \longrightarrow$$
$$(\forall\ x \longrightarrow f\ x \equiv g\ x) \longrightarrow f \equiv g$$
$$\text{funExt}\ p = \lambda\ i \longrightarrow (\lambda\ x \longrightarrow p\ x\ i)$$



What about transitivity?

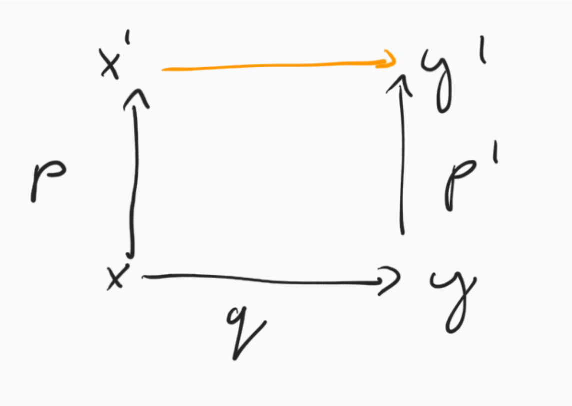
If $p : x \equiv y$ and $q : y \equiv z$, how do we make

$$\text{trans}\ p\ q = \lambda\ i \to \begin{cases} p(2i) & \text{if } i \leq 0.5 \\ q(2i - 1) & \text{if } i \geq 0.5 \end{cases}$$



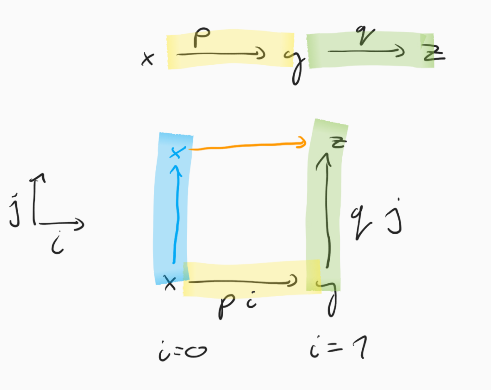
Path composition

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.

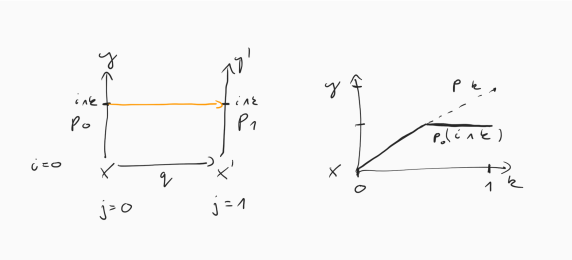


Transitivity via comp

```
trans : x ≡ y → y ≡ z → x ≡ z
trans p q i = comp (λ _ → A)
  (λ { j (i = i0) → x
     ; j (i = i1) → q j
     })
  (inc (p i))
```



A sliding version

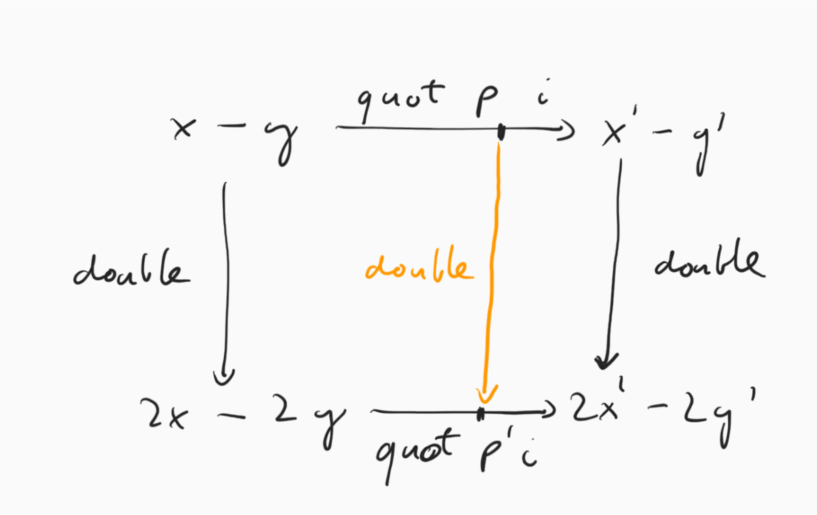



```
slidingLid : (p₀ : x ≡ y) (p₁ : x′ ≡ y′) (q : x ≡ x′) →
  ∀ i → p₀ i ≡ p₁ i
slidingLid p₀ p₁ q i j = comp (λ _ → A)
  (λ { k (j = i0) → p₀ (i ∧ k)
     ; k (j = i1) → p₁ (i ∧ k)
     ; k (i = i0) → q j
     })
  (inc (q j))
```

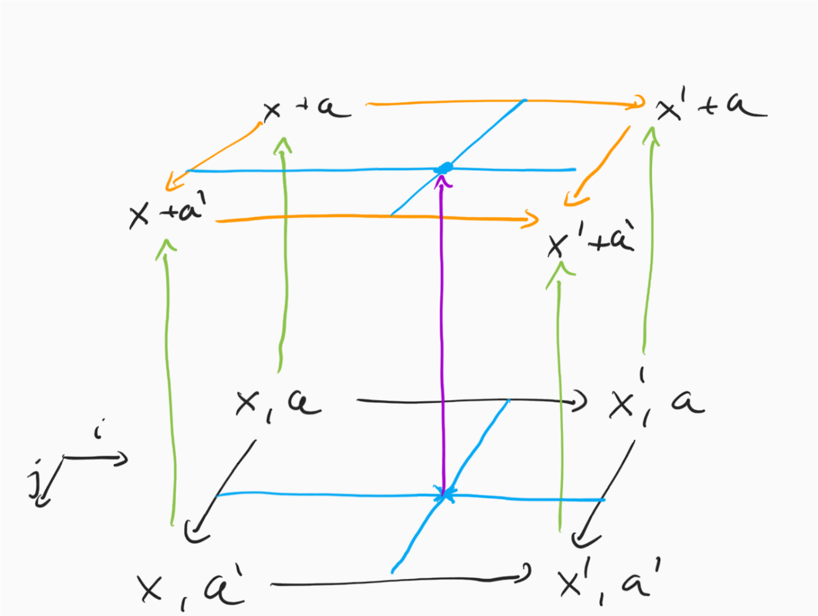
double, cubically

```
double : ℤ → ℤ
double (x - y) = (2 * x) - (2 * y)
double (quot x y x′ y′ p i) =
  quot (2 * x) (2 * y) (2 * x′) (2 * y′) p′ i
  where
    p′ : 2 * x + 2 * y′ ≡ 2 * x′ + 2 * y
    p′ = arithmetic-proof x y p
```

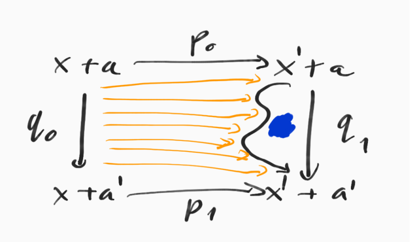
double, cubically



A non-unary example: $\mathbb{Z}$ addition



A problem:



What if there is no way to continuously deform

slidingLid p_0 p_1 q_0 $i1$

(a homotopically transformed proof)

into

q_1

(an arithmetic proof about natural numbers)

Solution: set-truncating

We *define* $\mathbb{Z}$ not to have any holes by adding a third constructor (à la HoTT §6.10):

$$
\begin{aligned}
&\text{data } \mathbb{Z} : \text{Set where} \\
&\quad \_\text{-}\_ : \mathbb{N} \longrightarrow \mathbb{N} \longrightarrow \mathbb{Z} \\
&\quad \text{quot} : \forall\, x\; y\; x'\; y' \longrightarrow \text{Same}\; x\; y\; x'\; y' \longrightarrow x \text{ - } y \equiv x' \text{ - } y' \\
&\quad \text{trunc} : \forall\, \{x\; y : \mathbb{Z}\} \longrightarrow (p\; q : x \equiv y) \longrightarrow p \equiv q
\end{aligned}
$$

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 $\mathbb{Z} \simeq$ Int (from the standard library)
- Prove $\mathbb{Z} \simeq$ base ≡ base (in Circle)