Cubical Type Theory and Cubical Agda

Anders Mörtberg

EPIT – April 15, ~~2020~~ 2021

Identity types

The central inductive type in HoTT is the identity type:

```
data _≡_ {A : Type} (x : A) : A → Type where
  refl : x ≡ x
```

Identity types

The central inductive type in HoTT is the identity type:

```
data _≡_ {A : Type} (x : A) : A → Type where
  refl : x ≡ x
```

This type is crucial to express equations and specifications:

$$1 + 1 \equiv 2$$

$$(m\ n : \mathbb{N}) \to m + n \equiv n + m$$

$$\{A\ B : \mathsf{Type}\} \to (f : A \to B) \to \{x\ y : A\} \to x \equiv y \to f\ x \equiv f\ y$$

$$\ldots$$

Some are provable by refl because of judgmental/definitional equality, some need more elaborate arguments using the induction principle J

Identity types

Pre-HoTT problem: _≡_ is not extensional enough, we cannot prove:

$$\mathsf{funExt} : \{A\ B : \mathsf{Type}\}\ (f\ g : A \to B) \to ((x : A) \to f\,x \equiv g\,x) \to f \equiv g$$
$$\mathsf{funExt}\ f\ g\ p = ?$$

It's also impossible to prove propositional extensionality or, more generally, univalence. It's also difficult to handle quotient types...

Identity types

Pre-HoTT problem: _≡_ is not extensional enough, we cannot prove:

$$\mathsf{funExt} : \{A\ B : \mathsf{Type}\}\ (f\ g : A \to B) \to ((x : A) \to f\,x \equiv g\,x) \to f \equiv g$$
$$\mathsf{funExt}\ f\ g\ p = ?$$

It's also impossible to prove propositional extensionality or, more generally, univalence. It's also difficult to handle quotient types...

HoTT: add them as axioms to TT[1]

[1]Justified by simplicial set model

Identity types

Pre-HoTT problem: $\_\equiv\_$ is not extensional enough, we cannot prove:

$$\mathsf{funExt} : \{A\ B : \mathsf{Type}\}\ (f\ g : A \to B) \to ((x : A) \to f\,x \equiv g\,x) \to f \equiv g$$
$$\mathsf{funExt}\ f\ g\ p = ?$$

It's also impossible to prove propositional extensionality or, more generally, univalence. It's also difficult to handle quotient types...

HoTT: add them as axioms to TT[1] **breaks canonicity** ☹

[1]Justified by simplicial set model

Identity types

Pre-HoTT problem: _≡_ is not extensional enough, we cannot prove:

$$\mathsf{funExt} : \{A\ B : \mathsf{Type}\}\ (f\ g : A \to B) \to ((x : A) \to f\,x \equiv g\,x) \to f \equiv g$$
$$\mathsf{funExt}\ f\ g\ p = ?$$

It's also impossible to prove propositional extensionality or, more generally, univalence. It's also difficult to handle quotient types...

HoTT: add them as axioms to TT[1] **breaks canonicity** ☹

Cubical: extend TT and make them provable[2]

[1] Justified by simplicial set model
[2] Inspired by *cubical* set models

Identity types

Pre-HoTT problem: $\_\equiv\_$ is not extensional enough, we cannot prove:

$$\mathsf{funExt} : \{A\ B : \mathsf{Type}\}\ (f\ g : A \longrightarrow B) \longrightarrow ((x : A) \longrightarrow f\,x \equiv g\,x) \longrightarrow f \equiv g$$
$$\mathsf{funExt}\ f\ g\ p = ?$$

It's also impossible to prove propositional extensionality or, more generally, univalence. It's also difficult to handle quotient types...

HoTT: add them as axioms to TT[1] **breaks canonicity** ☹

Cubical: extend TT and make them provable[2] **preserves canonicity** ☺

[1] Justified by simplicial set model
[2] Inspired by *cubical* set models

Cubical type theory

Key idea: take HoTT very literally and replace inductive _≡_ with paths

[3] We write $x \equiv y$ for path-equality and $x = y$ for definitonal/judgmental equality

Cubical type theory

Key idea: take HoTT very literally and replace inductive $\_\equiv\_$ with paths

A path $p : x \equiv y$ is a function $p : \mathrm{I} \to A$ with endpoints x and y:[3]

$$p \ \mathrm{i0} = x \qquad\qquad p \ \mathrm{i1} = y$$

Get cubes by iteration: $p : \mathrm{I} \to \mathrm{I} \to A$ is a square, $q : \mathrm{I} \to \mathrm{I} \to \mathrm{I} \to A$ is a cube, etc...

[3] We write $x \equiv y$ for path-equality and $x = y$ for definitonal/judgmental equality

Cubical type theory

Key idea: take HoTT very literally and replace inductive _≡_ with paths

A path $p : x \equiv y$ is a function $p : \mathbb{I} \to A$ with endpoints x and y:[3]

$$p\ \mathrm{i0} = x \qquad\qquad p\ \mathrm{i1} = y$$

Get cubes by iteration: $p : \mathbb{I} \to \mathbb{I} \to A$ is a square, $q : \mathbb{I} \to \mathbb{I} \to \mathbb{I} \to A$ is a cube, etc...

This gives us funext, univalence, quotients without sacrificing canonicity!

[3] We write $x{\equiv}y$ for path-equality and $x = y$ for definitonal/judgmental equality

Cubical type theory

Key idea: take HoTT very literally and replace inductive _≡_ with paths

A path $p : x \equiv y$ is a function $p : \mathbb{I} \to A$ with endpoints x and y:[3]

$$p\ \mathrm{i0} = x \qquad\qquad p\ \mathrm{i1} = y$$

Get cubes by iteration: $p : \mathbb{I} \to \mathbb{I} \to A$ is a square, $q : \mathbb{I} \to \mathbb{I} \to \mathbb{I} \to A$ is a cube, etc...

This gives us funext, univalence, quotients without sacrificing canonicity!

This simple idea is the basis of `Cubical Agda`

[3] We write $x \equiv y$ for path-equality and $x = y$ for definitonal/judgmental equality

Cubical proof assistants

`Cubical Agda` was implemented by Andrea Vezzosi, building on a series of experimental typecheckers developed at Chalmers: `cubical`, `cubicaltt`...

There are also many other cubical and cubically-inspired systems: `Arend`, **`RedPRL`**, **`redtt`**, **`cooltt`**, `yacctt`, `mlang`...

Cubical proof assistants

`Cubical Agda` was implemented by Andrea Vezzosi, building on a series of experimental typecheckers developed at Chalmers: `cubical`, `cubicaltt`...

There are also many other cubical and cubically-inspired systems: `Arend`, **RedPRL**, **redtt**, **cooltt**, `yacctt`, `mlang`...

These build on various cubical models and cubical type theories developed by many people over multiple years. The particular flavor that `Cubical Agda` builds on is based on the "CCHM" cubical type theory of

Cubical Type Theory: a constructive interpretation of the univalence axiom (2015)
Cyril Cohen, Thierry Coquand, Simon Huber, Anders Mörtberg
`https://arxiv.org/abs/1611.02108`

Cubical Agda

The cubical mode has been part of Agda since version 2.6.0 (April 2019)

To activate it just open an `.agda` file and add

```
{-# OPTIONS --cubical #-}
```

Cubical Agda

The cubical mode has been part of Agda since version 2.6.0 (April 2019)

To activate it just open an `.agda` file and add

```
{-# OPTIONS --cubical #-}
```

Since October 2018 Andrea Vezzosi and I have been maintaining the `agda/cubical` library:

`https://github.com/agda/cubical/`

By now 52 contributors, 56k LOC, 500 files

New features that we will look closer at today:

- Interval (pre-)type I with endpoints $\mathsf{i0} : \mathsf{I}$ and $\mathsf{i1} : \mathsf{I}$
- Kan operations (transp and hcomp)
- Computational univalence (via Glue types)
- General schema for higher inductive types

`Cubical` Agda time!