# The Case of Cubical Type Theory

Bruno Barras
j.w.w. Valentin Maestracci and Luc Chabassier

June 24, 2022

# Why Cubical Type Theory?

Encoding of Cubical Type Theory:

- Having a 21st century formalism in Dedukti
- (Encoding HoTT is not fun enough)
- Raises challenging problems of broad interest

# Encoding Type Theories in Dedukti

Encoding a Type Theory $T$ usually consists of:

- a Dedukti theory $D(T)$,
- a translation that turns $T$-term $t$ into a Dedukti term $[\![t]\!]$ in theory $D(T)$.

Expected property (in this talk):

- Soundness: a well-typed term in $T$ leads to a well-typed term

$$\vdash_T M : A \quad \Rightarrow \quad D(T) \vdash_{\lambda\Pi/\mathcal{R}} [\![M]\!] : [\![A]\!]$$

# Definitional Equality: the nice case

Type theories usually use definitional equality through a conversion rule

$$\frac{\vdash_T M : A \quad \vdash_T A = B}{\vdash_T M : B}$$

It is convenient to have that whenever $\vdash_T A = B$, we have the corresponding conversion $\vdash_{\lambda\Pi/\mathcal{R}} [\![A]\!] = [\![B]\!]$.

Issue: cases where do not know how to turn equality of $T$ into rewriting rules and have this property?

# And when it is not possible?

Encode definitional equality of $T$ as a propositional equality in Dedukti

Conversion rule is justified by inserting transports

Eventually ends up in "transport hell"

A general framework has been introduced to help in this situation: Two Layers Type Theories (2LTT, Annenkov, Kraus et al)

# Overview

# Cubical Type Theory

Geometric (higher dimensional) interpretation of TT

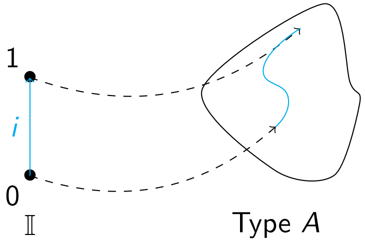
- Introduces an interval “pretype” $\mathbb{I}$ with endpoints 0 and 1 (actually a de Morgan algebra)

Paths as $\mathbb{I}$-indexed functions

- Path construction and application: similar to non-dependent product
- Path induction takes the form of a composition operation
- That’s cubical magic!

# Dimension 1...

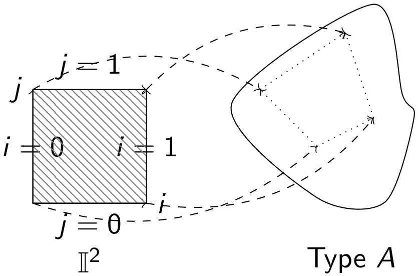
Judgement $i : \mathbb{I} \vdash M : A$ represents a (continuous) path in $A$:




(heterogeneous path if $A$ depends on $i$)

# Dimension 2 and beyond...

Judgement $i : \mathbb{I}, j : \mathbb{I} \vdash M : A$ represents a square in $A$:



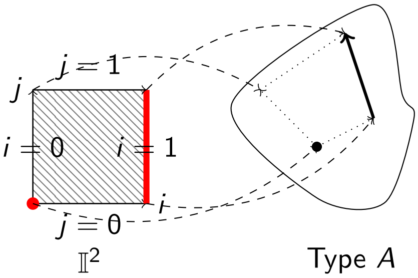

Judgement $i : \mathbb{I}, j : \mathbb{I}, k : \mathbb{I} \vdash M : A$ represents a cube in $A$, etc.

# Faces

Judgement contexts may be restricted to a set of faces of an hypercube: $\Gamma, \phi \vdash$

Judgement $\vdash \phi : \mathbb{F}$ expresses that face $\phi$ is well-formed

Example: $i = 1 \vee i = 0 \wedge j = 0 \vdash M : A$



# Typing Rules

cf CCHM paper (CHM would do as well).

Path introduction and application:

$$\frac{i : \mathbb{I} \vdash M : A}{\vdash \langle i \rangle\, M : \mathrm{Path}_A\, M(i0)\, M(i1)} \qquad \frac{\vdash p : \mathrm{Path}_A\, t\, u \quad \vdash r : \mathbb{I}}{\vdash p\, r : A}$$

$$\frac{}{\vdash \langle i \rangle M\ r = M[i \backslash r]} \quad \frac{\vdash p : \mathrm{Path}_A\, t\, u}{\vdash p\, 0 = t \quad \vdash p\, 1 = u}$$

Composition:

$$\frac{\begin{array}{c} i : \mathbb{I} \vdash A \text{ type} \quad \vdash \phi : \mathbb{F} \\ \phi,\ i : \mathbb{I} \vdash u : A(i) \quad \vdash a_0 : A(i0)[\phi \Rightarrow u(i0)] \end{array}}{\vdash \mathrm{comp}^i\, A\, \phi\, u\, a_0\ :\ A(i1)[\phi \Rightarrow u(i1)]}$$

(where $\vdash M : A[\phi \Rightarrow N]$ means $\vdash M : A$ and $\phi \vdash M = N$)

# Challenges

Encode a de Morgan algebra:

- Associative Commutative
- Non-linear rules (complement)

Path reduction rules

- Some rules require typing information

Expressing the coherence condition of composition

- Composition rule cannot be expressed by a mere $\lambda\Pi/\mathcal{R}$ type
- Face witnesses are in the context (shallow encoding)

$\Rightarrow$ Coherence condition has to be materialized by a type

# Overview

# Two Layers Type Theories

2 copies of MLTT: internal and external layers

In Dedukti:

```
    T   : Type.     (; code for internal types ;)
def El  : T -> Type (; decoding fun ;)

    xT  : Type.     (; code for external types ;)
def xEl : xT -> Type.
```

...and the types constructors (the usual suspects):

| | | | | | |
|---|---|---|---|---|---|
| False, | True, | Pi, | Sig, | Nat, | Eq |
| xFalse, | xTrue, | xPi, | xSig, | xNat, | xEq |

# Embedding the inner layer inside the outer one

Every internal type has an isomorphic external type

```
def c  :  T -> xT.

(; iso between El A and xEl (c A) ;)
def cUp :    A : T -> El A       -> xEl (c A).
def cDown : A : T -> xEl (c A) -> El A.
[A, a] cDown A (cUp A a) --> a.
[A, a] cUp A (cDown A a) --> a.
```

No alignment of internal/external type constructors:

- `xEq` is assumed to enjoy Uniqueness of Identity Proofs (UIP) and functional extensionality (FunExt)
- `Eq` (renamed `Path`) satisfies Univalence

# Overview

Cubical Type Theory

Encoding 2LTTs in Dedukti

Encoding Cubical TT as a 2LTT

Facing the Transport Hell

# Cubical as a Two Layer TT

The external layer encodes judgements:

- Cubical's definitional equality is an external (propositional) equality

```
def cubEq := A : T => a : El A => b : El A =>
    xEq (c A) (cUp A a) (cUp A b).
```

- The interval $\mathbb{I}$ and face type $\mathbb{F}$ are external types

```
I : xT.   0  : xEl I.  1 :  xEl I.
F : xT.   0f : xEl F.  1f : xEl F.
          (; more operations to come ;)
```

# de Morgan algebra

Symbols and rewrite rules related to a de Morgan algebra

```
def Imin : xEl I -> xEl I -> xEl I.
def Imax : xEl I -> xEl I -> xEl I.
def Ineg : xEl I -> xEl I.

Ineg 0 --> 1.          Ineg 1 --> 0.

[a] Imin a 0 --> 0.   [a] Imax a 0 --> a.
[a] Imin 0 a --> 0.   [a] Imax 0 a --> a.
[a] Imin a 1 --> a.   [a] Imax a 1 --> 1.
[a] Imin 1 a --> a.   [a] Imax 1 a --> 1.
[a] Imin a a --> a.   [a] Imax a a --> a.
```

Adding ACD makes the system non-confluent ($a \vee a \wedge b = a$)

$\Rightarrow$ ACD remain at the external level
(deemed less useful than the above laws)

# Encoding paths

```
def Path :  A : T -> El A -> El A -> T.

def lam : A : T ->
          p : (xEl I -> El A) ->
          El (Path A (p 0) (p 1)).

def app : A : T -> u : El A -> v : El A ->
        El (Path A u v) -> xEl I -> El A.

[A, p, u, v, r]  app A u v (lam A p) r --> (p r).
[A, p, u, v]     app A u v p 0 --> u.
[A, p, u, v]     app A u v p 1 --> v.
   (; use annotations to avoid typed reduction ;)
```

# Face operations

Some implementations reuse $\mathbb{I}$ for elements of $\mathbb{F}$. We don't.

```
def eq0 : xEl I -> xEl F.  (; hyperplane ;)
def eq1 : xEl I -> xEl F.  (; hyperplane ;)
def Fmin: xEl F -> xEl F -> xEl F.(;intersection;)
def Fmax: xEl F -> xEl F -> xEl F. (; union ;)
```

eq0 is intended to be applied to variables

Those rewrite rules explain what happens when an interval variable is substituted:

```
[f, g] eq0 (Fmin f g) --> Fmax (eq0 f) (eq0 g).
(; ... more similar rules ... ;)
```

# Face constraints in the context

How do we represent contexts $\Gamma, \phi$ ?

$\phi$ is turned into a proposition that tells which points belong to the face. Ideally it should be irrelevant (a la SProp). Plan B is to use the external equality...

```
def faceType :  xEl F -> xT.
```

This type is isomorphic to its Curry-Howard representation:

$$\begin{aligned}
\texttt{faceType (eq0 i)} &\approx \texttt{xEq I i 0} \\
\texttt{faceType (eq1 i)} &\approx \texttt{xEq I i 1} \\
\texttt{faceType (Fmin f g)} &\approx \texttt{faceType f} \times \texttt{faceType g} \\
\texttt{faceType (Fmax f g)} &\approx || \texttt{faceType f} + \texttt{faceType g} ||
\end{aligned}$$

Up to isomorphism otherwise we lose confluence:

$$\texttt{facType f} \times 1 \leftarrow \texttt{faceType(Fmin f 1f)} \rightarrow \texttt{faceType f}$$

# Compositions

$$\frac{i : \mathbb{I} \vdash A \text{ type} \quad \phi, i : \mathbb{I} \vdash u : A(i) \quad \vdash a_0 : A(i0)[\phi \Rightarrow u(i0)]}{\vdash \mathrm{comp}^i\, A\, \phi\, u\, a_0 \ : \ A(i1)[\phi \Rightarrow u(i1)]}$$

(hiding `El` and `xEl`)

```
def Comp :
  phi : F -> A : (I -> T) ->
  u   : (faceType phi -> I -> A) ->
  a0  : A 0 ->
  (e : faceType phi -> cubEq (A 0) a0 (u e 0)) ->
  A 1.

def CompEq : ... phi A u a0 coh ...
  e : faceType phi ->
  cubEq (A 1) (Comp phi A u a0 coh) (u e 1).
```

# Glueing

Glueing is the feature of CTT from which Univalence is derived. It constructs paths between isomorphic types.

Not yet implemented

(Does not look good)

# Overview

# Usefulness of the encoding

Can we actually encode Cubical terms in our encoding?

- Each non-structural rule of Cubical is represented by a Dedukti symbol
- Conversion steps are represented by a transport steps

Translation $[\![t]\!]$ has the same shape as $t$ + transports at any place

Transport hell:

- Simple transports when type do not match because of a conversion is manageable
- Hell comes from the fact that a single term may be decorated by transports in many ways
  They are all provably equal

Such transports are hardly manageable by humans

# Generating transports

This problem is similar to that of translating Extensional Type Theory to Intensional Type Theory

Previous work (Winterhalter et al):

- ETT can be translated to ITT + UIP + FunExt
- implemented in MetaCoq

Chabassier ported this work to Dedukti

- generates directly well-typed terms

Bad news: both attempts do not scale to non-trivial terms

# Conclusions

Many challenges, few victories

However a solution exists in principle thanks to 2LTT

Possible paths to explore

- Try to optimize transports
  Strong incentive to squeeze external equality
- Extend Dedukti with useful adhoc structures (AC, ACUI, ACDUI)
- Extend Dedukti with simple decision procedures