Make Three To Throw Away

Frontiers in Homotopical Proof Assistants

Jonathan Sterling

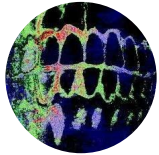
Aarhus University

WITS ’22

Heartfelt thanks to Jesper and Richard for organizing this wonderful meeting.

The RedPRL Development Team (redprl.org)

Carlo Angiuli



Evan Cavallo

Favonia

Robert Harper

Reed Mullanix

Jon Sterling

Our philosophy

We have never been interested in building big, production-quality tools.

Big tools are for people have **more resources to spend** and **less desire for a career**. Success for us is to have our ideas incorporated into the next generation of big tools.

The **RedPRL** project has always been about developing the *empirical* and *grounded-theoretic* basis for our main research project, higher-dimensional types.

Each implementation was meant to develop a specific thought in our search for the shining path to the future of interactive math.

So we built “three to throw away”.

We build proof assistants for **higher-dimensional type theory**. Why?

We build proof assistants for **higher-dimensional type theory**. Why?

We believe that mathematics and computer programming are the same thing.

We build proof assistants for **higher-dimensional type theory**. Why?

We believe that mathematics and computer programming are the same thing.

Martin-Löf type theory great for programming but remains **inadequate** for even the most basic mathematics (bottom half of a "dualist" foundation).

We build proof assistants for **higher-dimensional type theory**. Why?

We believe that mathematics and computer programming are the same thing.

Martin-Löf type theory great for programming but remains **inadequate** for even the most basic mathematics (bottom half of a "dualist" foundation).

Higher-dimensional types pose a "monist" solution to this problem.

Why higher dimensional types?

Non-negotiable treasures of mathematical experience

- **function extensionality**
  there is at most one function encoding each functional relation
- **function comprehension**
  there is at least one function encoding each functional relation
- **propositional extensionality**
  there is at most one proposition encoding each subsingleton type
- **effectivity of equivalence relations**
  original equivalence relation can be extracted from the quotient map

Without the above, general mathematics is **impossible to develop**.

Voevodsky's solution

Voevodsky's proposed the **univalence principle**, an induction rule for *isomorphisms* (*cf*. Paulin-Mohring *J* rule).

Univalence simultaneously solves *all* the old problems, *and also* exposes new concepts to the light of type theory.

Nine years of the Cubical Hypothesis

Computational content of the univalence principle originally dubious.

Cubical semantics proposed by **Coquand** and his collaborators in 2013 as a solution to this problem, leading to subsequent explosion in cubical research:

1. **denotational semantics** (Bezem, Coquand, and Huber, 2014; Cohen, Coquand, Huber, and Mörtberg, 2017; Angiuli, Brunerie, et al., 2021)
2. **syntactic canonicity** (Huber, 2018)
3. **computational semantics** (Angiuli, Hou (Favonia), and Harper, 2018)
4. **standard model in homotopy types** (Awodey, Cavallo, Coquand, Riehl, and Sattler, forthcoming)
5. **normalization & decidability** (Sterling and Angiuli, 2021; Sterling, 2021)

Cubical type theory extends Martin-Löf's judgmental structure.

Cubical type theory extends Martin-Löf's judgmental structure.

- an *abstract interval* $0, 1 : \mathbb{I}$ with no case analysis; elements $r : \mathbb{I}$ called "dimensions"

Cubical type theory extends Martin-Löf's judgmental structure.

- an *abstract interval* $0, 1 : \mathbb{I}$ with no case analysis; elements $r : \mathbb{I}$ called "dimensions"
- a universe $\mathbb{F}$ of proof-irrelevant propositions closed under:
  - dimensional equality $r =_{\mathbb{I}} s$ with equality reflection
  - finite conjunction and disjunction $\wedge, \vee$; elimination into types
  - universal quantification $\forall i : \mathbb{I}.\phi\, i$ for $\phi : \mathbb{I} \to \mathbb{F}$

Path equality

New notion of equality given by *paths*:

$$\frac{\Gamma, i : \mathbb{I} \vdash w : A \qquad \Gamma \vdash [0/i]w \equiv u : A \qquad \Gamma \vdash [1/i]w \equiv v : A}{\Gamma \vdash \langle i \rangle w : \mathsf{Path}_A\, u\, v}$$

Paths commute with function spaces:

$$\begin{aligned} &\mathsf{funext} : ((x : A) \to \mathsf{Path}_B\, (f\, x)\, (g\, x)) \cong \mathsf{Path}_{A \to B}\, f\, g \\ &\mathsf{funext}\, h \coloneqq \langle i \rangle \lambda x. h\, x\, i \end{aligned}$$

A golden age for implementation

Implementation has been at the center of the cubical project since the beginning.

- **Generation 0:** prototype evaluators
  simhu/cubical
- **Generation 1:** prototype typecheckers, interactive derivation builders
  mortberg/cubicaltt, **RedPRL**$^{(*)}$, mortberg/yacctt
- **Generation 2:** interactive proof assistants (elaboration, typechecking, holes)
  Cubical Agda, **redtt**$^{(*)}$, **cooltt**$^{(*)}$, molikto/mlang, others?

We are personally responsible for **RedPRL**, **redtt**, and **cooltt**.

I want to give you my take on our story, both the positive and the negative.

The rest of this talk

1. **RedPRL**: Nuprl/computational type theory in higher dimensions
2. **redtt**: cubical phase distinction and cubical holes
3. **cooltt**: a cubical logical framework; interactive visualization
4. My thoughts and advice for the future

RedPRL (2016–2017)

HOW IT STARTED

RedPRL: a cubical version of Nuprl

Nuprl = using **types** to structure **untyped programs**.

- programs do not come with types ($M : A$); well-typedness ($M \in A$) is a theorem
- judgmental equality eschewed in favor of propositional equality + elim rule (equivalent to equality reflection)
- interactive construction of typing derivations using tactics
- untyped computation rules allow context-free adjustment of goals
- no equations except α-equivalence can be fully automated: computation rules cannot be automatic because they may diverge

RedPRL= Nuprl + cubes + univalence + higher inductive types

```
theorem Trans(#1:lvl) :
  (->
   [ty : (U #1 kan)]
   [p : (line [_] ty)]
   [q : (line [_] ty)]
   [eq : (= ty (@ p 1) (@ q 0))]
   (path [_] ty (@ p 0) (@ q 1)))
by {
  lam ty p q eq => (abs x =>
    `(hcom 0~>1 ty (@ p x)
      [x=0 [_] (@ p 0)]
      [x=1 [z] (@ q z)]));
    auto; assumption
}.
```

What exactly went wrong with RedPRL?

We were motivated by several scientific hypotheses that I no longer believe.

- **Thesis:** *decidable type systems validate too few equations to be viable.*

What exactly went wrong with RedPRL?

We were motivated by several scientific hypotheses that I no longer believe.

- **Thesis:** *decidable type systems validate too few equations to be viable.*
  Fact Check: Automatic β/η is a sweet spot for usability, worlds better than PRL's pure α-equivalence! In either case, additional non-automatic eq'ns needed.

What exactly went wrong with RedPRL?

We were motivated by several scientific hypotheses that I no longer believe.

- **Thesis:** *decidable type systems validate too few equations to be viable.*
  Fact Check: Automatic β/η is a sweet spot for usability, worlds better than PRL's pure α-equivalence! In either case, additional non-automatic eq'ns needed.
- **Thesis**: *type checking, even when decidable, is infeasible in practice.*

What exactly went wrong with RedPRL?

We were motivated by several scientific hypotheses that I no longer believe.

- **Thesis:** *decidable type systems validate too few equations to be viable.*
  Fact Check: Automatic β/η is a sweet spot for usability, worlds better than PRL's pure α-equivalence! In either case, additional non-automatic eq'ns needed.
- **Thesis**: *type checking, even when decidable, is infeasible in practice.*
  Fact Check: Coquand's 1996 typechecking algorithm is blindingly fast in the **usual case** with much room for optimization (*cf.* AndrasKovacs/smalltt).

What exactly went wrong with RedPRL?

We were motivated by several scientific hypotheses that I no longer believe.

- **Thesis:** *decidable type systems validate too few equations to be viable.*
  Fact Check: Automatic β/η is a sweet spot for usability, worlds better than PRL's pure α-equivalence! In either case, additional non-automatic eq'ns needed.
- **Thesis**: *type checking, even when decidable, is infeasible in practice.*
  Fact Check: Coquand's 1996 typechecking algorithm is blindingly fast in the **usual case** with much room for optimization (*cf.* AndrasKovacs/smalltt).
- **Thesis**: *type checking can be simulated in* ****PRL*** *by a tactical heuristic.*
  "Running till I'm dead is the same as looping."

What exactly went wrong with RedPRL?

We were motivated by several scientific hypotheses that I no longer believe.

- **Thesis:** *decidable type systems validate too few equations to be viable.*
  Fact Check: Automatic β/η is a sweet spot for usability, worlds better than PRL's pure α-equivalence! In either case, additional non-automatic eq'ns needed.
- **Thesis**: *type checking, even when decidable, is infeasible in practice.*
  Fact Check: Coquand's 1996 typechecking algorithm is blindingly fast in the **usual case** with much room for optimization (*cf*. AndrasKovacs/smalltt).
- **Thesis**: *type checking can be simulated in* ****PRL*** *by a tactical heuristic.*
  "Running till I'm dead is the same as looping."
  Fact Check: Long inversion phases needed to make heuristic type checking work; sadly, most Nuprl/**RedPRL** rules very non-determistic (but *cf*. Andromeda!).

Retrospective on equality reflection

RedPRL has both path types (for homotopy) and equality types (with reflection):

$$\frac{\Gamma \vdash M : \mathsf{Eq}_A\, a\, b}{\Gamma \vdash a \equiv b : A}\ \text{reflection}$$

I didn't believe people when they told me that equality reflection was "bad". Our experience was eye-opening.

Retrospective on equality reflection

RedPRL has both path types (for homotopy) and equality types (with reflection):

$$\text{reflection}\quad \frac{\Gamma \vdash M : \mathsf{Eq}_A\, a\, b}{\Gamma \vdash a \equiv b : A}$$

I didn't believe people when they told me that equality reflection was "bad". Our experience was eye-opening.

- **Expectation:** I will use equality reflection to easily discharge subgoals that hold by virtue of rich mathematical theorems (*e.g.* quicksort $\equiv$ mergesort).

Retrospective on equality reflection

RedPRL has both path types (for homotopy) and equality types (with reflection):

$$\frac{\Gamma \vdash M : \mathsf{Eq}_A\, a\, b}{\Gamma \vdash a \equiv b : A}\ \text{reflection}$$

I didn't believe people when they told me that equality reflection was "bad". Our experience was eye-opening.

- **Expectation:** I will use equality reflection to easily discharge subgoals that hold by virtue of rich mathematical theorems (*e.g.* quicksort $\equiv$ mergesort).
- **Reality:** I am forced to manually use equality reflection each and every time a goal is off by β- or η-rule (*e.g.* $\mathcal{C}[\lambda x.\langle x.1, x.2\rangle] \equiv \mathcal{C}[\lambda x.x]$ for enormous $\mathcal{C}[-]$). Even worse, I also have to prove that $u : A \times B \to A \times B \gg \mathcal{C}[u]$ *type*.

Retrospective on equality reflection

RedPRL has both path types (for homotopy) and equality types (with reflection):

$$\frac{\Gamma \vdash M : \mathsf{Eq}_A\, a\, b}{\Gamma \vdash a \equiv b : A}\ \text{reflection}$$

I didn't believe people when they told me that equality reflection was "bad". Our experience was eye-opening.

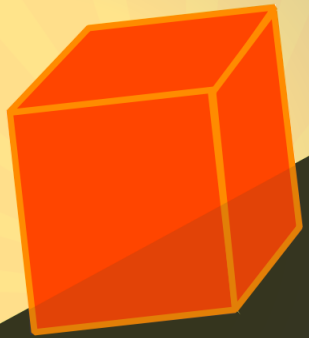
- **Expectation:** I will use equality reflection to easily discharge subgoals that hold by virtue of rich mathematical theorems (*e.g.* quicksort $\equiv$ mergesort).
- **Reality:** I am forced to manually use equality reflection each and every time a goal is off by β- or η-rule (*e.g.* $\mathcal{C}[\lambda x.\langle x.1, x.2\rangle] \equiv \mathcal{C}[\lambda x.x]$ for enormous $\mathcal{C}[-]$). Even worse, I also have to prove that $u : A \times B \to A \times B \gg \mathcal{C}[u]$ *type*.

Back to the drawing board!

redtt (2018–2019)

redtt: cubical proof assistant

between the darkness and the dawn,
a red cube rises...

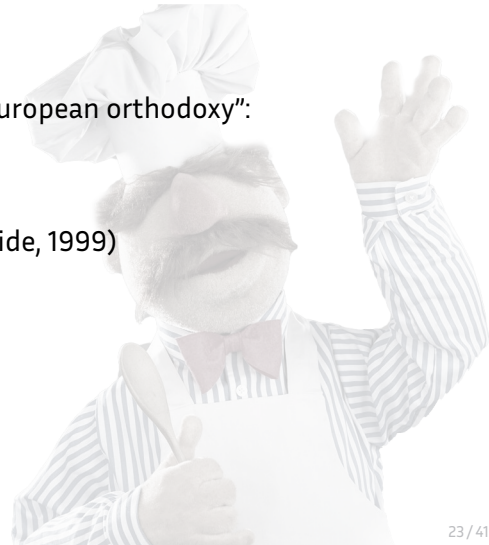


```
def torus→s1×s1 : torus → s1 × s1 =
  elim [
  | pt → (base, base)
  | p/one i → (loop i, base)
  | p/two i → (base, loop i)
  | square i j → (loop j, loop i)
  ]

def s1×s1→torus : (s1 × s1) → torus =
  λ [,] →
  elim [
  | base → elim [ base → pt | loop j → p/two j ]
  | loop i → elim [ base → p/one i | loop j → square j i ]
  ]

def torus→s1×s1/section
  : (t : torus)
  → path torus (s1×s1→torus (torus→s1×s1 t)) t
  =
  λ * → refl
```

redtt: a return to orthodoxy

Our experience with **RedPRL** won us over to the "European orthodoxy":

- normalization by evaluation (Abel, 2013)
- bidirectional type checking (Coquand, 1996)
- proof-states shall always be well-typed (McBride, 1999)



redtt contribution: a cubical phase distinction

Cubical type theory imposed equality reflection for $r =_{\mathbb{I}} s$. Why doesn't this break implementation?

redtt *stratifies* the type theory by a phase distinction between the "cubical layer" and the "Martin-Löf layer":

1. terms of type $\mathbb{F}, \mathbb{I}, \phi$ depend only on contexts involving $i : \mathbb{I}, \phi : \mathbb{F}, \_ : \phi$
2. no such thing as $\mathbb{N} \to \mathbb{I}$ or $(\mathbb{I} \to \mathbb{I}) \to \mathbb{I}$

Because cubical layer is decidable, equality reflection for $r =_{\mathbb{I}} s$ is harmless.

Lesson: failure to stratify leads to incompleteness of the type checking algorithm and unexpected behavior (*cf.* Cubical Agda). **Phase distinction is important!**

The Three Rules of Discipline for interactive proof

The “McBride Doctrine” states that the role of the machine is to assist human beings through the **intermediate steps** of a proof act.

The following Three Rules of Discipline assist in realizing this goal:

1. Always decompose goals into subgoals that jointly entail the main goal, *subject to no side conditions*.
2. Do not spawn a subgoal that is not well-typed.
3. *Send* data inward from the goal and *receive* data outward from the context.

How we violate the Three Rules of Discipline

The masses can tell when we are violating these rules of discipline:

- when they spend in hour coding in Agda and ***for 58 minutes*** *of that hour*, their code is mostly yellow and therefore meaningless,
- when they prove a huge theorem in Coq and receive an error only on **Qed.**

The McBride Doctrine is already challenged by the heavy use of unification in Agda; **cubical type theory** challenges it further via equational side conditions.

The need for cubical holes

Suppose you are trying to define a *path* from u to v in type $(x : A) \times B\,x$.
We start with a hole.

$$\frac{\vdash \boxed{?0} : \mathrm{Path}_{(x:A)\times B\,x}\; u\; v}{\vdash \boxed{?0} : \mathrm{Path}_{(x:A)\times B\,x}\; u\; v}$$

The need for cubical holes

Suppose you are trying to define a *path* from u to v in type $(x : A) \times B\,x$.
We start with a hole. We attempt to abstract a path.

$$\frac{i : \mathbb{I} \vdash \boxed{?0\ \boxed{i}} : (x : A) \times B\,x \qquad \vdash \boxed{?0\ \boxed{0}} \equiv u : (x : A) \times B\,x \qquad \vdash \boxed{?0\ \boxed{1}} \equiv v : (x : A) \times B\,x}{\vdash \langle i \rangle \boxed{?0\ \boxed{i}} : \mathrm{Path}_{(x:A)\times B\,x}\,u\,v}$$

The need for cubical holes

Suppose you are trying to define a *path* from u to v in type $(x : A) \times B\,x$.
We start with a hole. We attempt to abstract a path. But this is a type error.

$$\frac{i : \mathbb{I} \vdash \boxed{?0\ \boxed{i}} : (x : A) \times B\,x \qquad \vdash \boxed{?0\ \boxed{0}} \equiv u : (x : A) \times B\,x \qquad \vdash \boxed{?0\ \boxed{1}} \equiv v : (x : A) \times B\,x}{\vdash \langle i \rangle \boxed{?0\ \boxed{i}} : \mathrm{Path}_{(x:A)\times B\,x}\,u\,v}$$

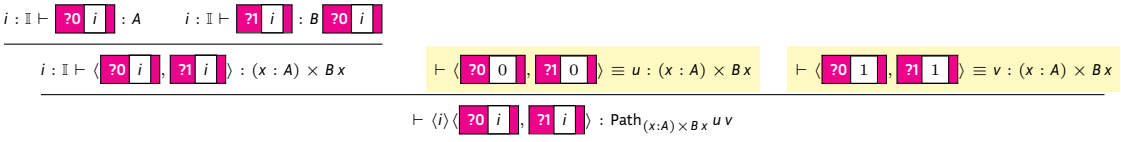
The need for cubical holes

Suppose you are trying to define a *path* from u to v in type $(x : A) \times B\,x$.
We start with a hole. We attempt to abstract a path. But this is a type error. But suppose we reserve judgment and proceed anyway.

$$\frac{i : \mathbb{I} \vdash \boxed{?0}\,\boxed{i} : (x : A) \times B\,x \qquad \vdash \boxed{?0}\,\boxed{0} \equiv u : (x : A) \times B\,x \qquad \vdash \boxed{?0}\,\boxed{1} \equiv v : (x : A) \times B\,x}{\vdash \langle i \rangle \boxed{?0}\,\boxed{i} : \mathrm{Path}_{(x:A)\times B\,x}\,u\,v}$$

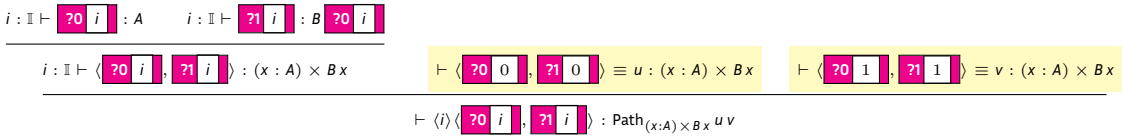
The need for cubical holes

Suppose you are trying to define a *path* from u to v in type $(x : A) \times B\,x$.
We start with a hole. We attempt to abstract a path. But this is a type error. But suppose we reserve judgment and proceed anyway. We introduce a pair of holes.



The need for cubical holes

Suppose you are trying to define a *path* from u to v in type $(x : A) \times B\,x$.
We start with a hole. We attempt to abstract a path. But this is a type error. But suppose we reserve judgment and proceed anyway. We introduce a pair of holes.



The proving process is now de-localized.

The need for cubical holes

Suppose you are trying to define a *path* from u to v in type $(x : A) \times B\,x$.
We start with a hole. We attempt to abstract a path. But this is a type error. But suppose we reserve judgment and proceed anyway. We introduce a pair of holes.

$$\dfrac{i : \mathbb{I} \vdash ?0\ i : A \qquad i : \mathbb{I} \vdash ?1\ i : B\ (?0\ i)}{i : \mathbb{I} \vdash \langle ?0\ i, ?1\ i \rangle : (x : A) \times B\,x} \qquad \vdash \langle ?0\ 0, ?1\ 0 \rangle \equiv u : (x : A) \times B\,x \qquad \vdash \langle ?0\ 1, ?1\ 1 \rangle \equiv v : (x : A) \times B\,x$$

$$\vdash \langle i \rangle \langle ?0\ i, ?1\ i \rangle : \mathrm{Path}_{(x:A)\times B\,x}\ u\ v$$

The proving process is now de-localized. We must wait until the holes are filled, and hope the yellow doesn't turn red.

The need for cubical holes

Suppose you are trying to define a *path* from u to v in type $(x : A) \times B\,x$.
We start with a hole. We attempt to abstract a path. But this is a type error. But suppose we reserve judgment and proceed anyway. We introduce a pair of holes.

$$\dfrac{i : \mathbb{I} \vdash ?0\ i : A \qquad i : \mathbb{I} \vdash ?1\ i : B\ (?0\ i)}{i : \mathbb{I} \vdash \langle ?0\ i, ?1\ i \rangle : (x : A) \times B\,x} \qquad \vdash \langle ?0\ 0, ?1\ 0 \rangle \equiv u : (x : A) \times B\,x \qquad \vdash \langle ?0\ 1, ?1\ 1 \rangle \equiv v : (x : A) \times B\,x$$

$$\vdash \langle i \rangle \langle ?0\ i, ?1\ i \rangle : \mathrm{Path}_{(x:A)\times B\,x}\ u\ v$$

The proving process is now de-localized. We must wait until the holes are filled, and hope the yellow doesn't turn red. **This is not interactive proof!**

redtt's "cubical holes" (2018)

redtt annotates every goal with a *partial element* that we are meant to match. These are **pushed inward** (Third Rule of Discipline) when possible.

$$\frac{\vdash \boxed{?0} : \mathrm{Path}_{(x:A)\times B\,x}\; u\; v \mid \phi \hookrightarrow w}{\vdash \boxed{?0} : \mathrm{Path}_{(x:A)\times B\,x}\; u\; v \mid \phi \hookrightarrow w}$$

redtt's "cubical holes" (2018)

redtt annotates every goal with a *partial element* that we are meant to match. These are **pushed inward** (Third Rule of Discipline) when possible.

$$\frac{i : \mathbb{I} \vdash \boxed{?0\ \ i} : (x : A) \times B\,x \mid \phi \hookrightarrow w@i, i = 0 \hookrightarrow u, i = 1 \hookrightarrow v}{\vdash \langle i \rangle \boxed{?0\ \ i} : \mathsf{Path}_{(x:A)\times B\,x}\,u\,v \mid \phi \hookrightarrow w}$$

redtt's "cubical holes" (2018)

redtt annotates every goal with a *partial element* that we are meant to match. These are **pushed inward** (Third Rule of Discipline) when possible.

$$\frac{i : \mathbb{I} \vdash \boxed{?0\ i} : A \mid \phi \hookrightarrow w@i.1, i = 0 \hookrightarrow u.1, u = 1 \hookrightarrow v.1 \qquad i : \mathbb{I} \vdash \boxed{?1\ i} : B\ \boxed{?0\ i} \mid \phi \hookrightarrow w@i.2, i = 0 \hookrightarrow u.2, u = 1 \hookrightarrow v.2}{\dfrac{i : \mathbb{I} \vdash \langle \boxed{?0\ i}, \boxed{?1\ i} \rangle : (x : A) \times B\,x \mid \phi \hookrightarrow w@i, i = 0 \hookrightarrow u, i = 1 \hookrightarrow v}{\vdash \langle i \rangle \langle \boxed{?0\ i}, \boxed{?1\ i} \rangle : \mathsf{Path}_{(x:A)\times B\,x}\ u\ v \mid \phi \hookrightarrow w}}$$

redtt's "cubical holes" (2018)

redtt annotates every goal with a *partial element* that we are meant to match. These are **pushed inward** (Third Rule of Discipline) when possible.

$$\frac{i : \mathbb{I} \vdash \boxed{?0\ i} : A \mid \phi \hookrightarrow w@i.1, i = 0 \hookrightarrow u.1, u = 1 \hookrightarrow v.1 \qquad i : \mathbb{I} \vdash \boxed{?1\ i} : B\ \boxed{?0\ i} \mid \phi \hookrightarrow w@i.2, i = 0 \hookrightarrow u.2, u = 1 \hookrightarrow v.2}{\dfrac{i : \mathbb{I} \vdash \langle \boxed{?0\ i}, \boxed{?1\ i} \rangle : (x : A) \times B\,x \mid \phi \hookrightarrow w@i, i = 0 \hookrightarrow u, i = 1 \hookrightarrow v}{\vdash \langle i \rangle \langle \boxed{?0\ i}, \boxed{?1\ i} \rangle : \mathsf{Path}_{(x:A)\times B\,x}\ u\ v \mid \phi \hookrightarrow w}}$$

Completely compositional, no yellow !

redtt's "cubical holes" (2018)

redtt annotates every goal with a *partial element* that we are meant to match. These are **pushed inward** (Third Rule of Discipline) when possible.

$$\frac{i : \mathbb{I} \vdash \boxed{?0\ i} : A \mid \phi \hookrightarrow w@i.1, i = 0 \hookrightarrow u.1, u = 1 \hookrightarrow v.1 \qquad i : \mathbb{I} \vdash \boxed{?1\ i} : B\ \boxed{?0\ i} \mid \phi \hookrightarrow w@i.2, i = 0 \hookrightarrow u.2, u = 1 \hookrightarrow v.2}{\dfrac{i : \mathbb{I} \vdash \langle \boxed{?0\ i}, \boxed{?1\ i} \rangle : (x : A) \times B\,x \mid \phi \hookrightarrow w@i, i = 0 \hookrightarrow u, i = 1 \hookrightarrow v}{\vdash \langle i \rangle \langle \boxed{?0\ i}, \boxed{?1\ i} \rangle : \mathsf{Path}_{(x:A)\times B\,x}\ u\ v \mid \phi \hookrightarrow w}}$$

Completely compositional, no yellow !

Similar idea proposed by McBride in TYPES'19; a variant of **redtt**'s cubical holes interface adapted for Cubical Agda in November 2019 by Vezzosi.

cooltt (2020–present)

```
def path (A : type) (a : A) (b : A) : type =
  ext i => A with [i=0 => a | i=1 => b]

def Ω1s1 : type = path circle base base

def loopn : nat -> Ω1s1 =
  elim [
  | zero => _ => base
  | suc {n => loopn} =>
    i =>
    hcom circle 0 1 {∂ i} {k _ =>
      [ k=0 => loopn i
      | i=0 => base
      | i=1 => loop k
      ]
    }
  ]

normalize {loopn 45}
```

cooltt: a cubical logical framework

cooltt is fundamentally in continuity with **redtt**. So what's new?

- All cubical machinery factored into a **logical framework**/logical framework layer containing $\mathbb{F}, \mathbb{I}$ such that $\mathbb{F}$ is closed under $\forall_{\mathbb{I}}, \wedge, \vee$.
- Cubical type theory is just a universe in the LF.
- New and more efficient evaluator for cubical compositions & coercions.
- Arbitrary dimension case splits $[\phi \hookrightarrow M \mid \psi \hookrightarrow N]$ with the full η-law.
  - Typing and equational theory are complete, in contrast to Cubical Agda.
  - Completeness depends on our stratification / cubical phase distinction.
- Coercive subtyping works "deep", *e.g.* $\lambda p.p$ inhabits the following type:

$$(p : Z \to (x : A) \times B\,x) \to (z : Z) \to \{A \mid \top \hookrightarrow (p\,z).1\} \times \{B\,(p\,z).1 \mid \top \hookrightarrow (p\,z).2\}$$

  Effortlessly elaborated to coercions.

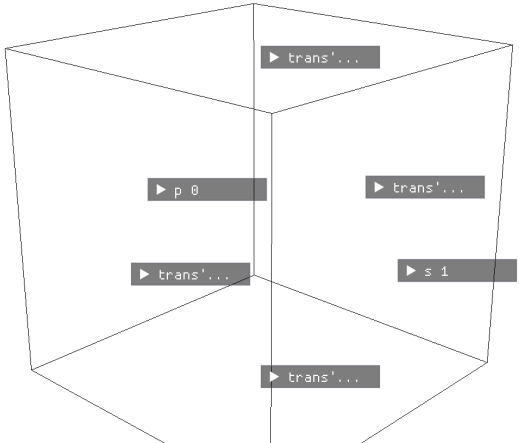
CUBES IN THE ***METAVERSE*** ???


(initial concept)


(revised concept)

(working prototype)

```
▼ Context

A : type
p : (i : 𝕀) → A
q : (i : 𝕀) → sub A {i = 0} {p 1}
r : (i : 𝕀) → sub A {i = 0} {q 1}
s : (i : 𝕀) → sub A {i = 0} {r 1}
i : 𝕀
j : 𝕀
k : 𝕀
|- ? : A
```



Where do we go from here?

Time to get united...

Type theoretic ITP community is small but fragmented: culture of isolation and insulation from related work runs deep into the 1980s.

It is premature to unite around a single project, but we must seize the opportunity to foster deeper connections between our work, unite around shared goals, and nurture the mutual awareness of each other's work. **"Doing our own thing" is not science.**

Time to unite!

Ideas that have proved their worth

Four important ideas that we should uphold and unite around.

1. **Semantic type checking & NbE à la Abel/Coquand are both fast and practical!**
   See recent work by Kovács. Fear of η-laws and unit types is Jurassic.
2. **Interactive theorem proving is about the middle of a proof, not the end.**
   Down with yellow code, uphold the Three Points of Discipline!
3. **Need for smooth handling of algebraic hierarchies.**
   Packed classes à la ssreflect/HierarchyBuilder most scalable and reflective of mathematical practice. Needs language-level support, smooth interaction between pointwise and fibered indexing à la Standard ML and Arend.
4. **Total type theory really is adequate for general-purpose programming.**
   Evinces distinction between computation *qua* judgmental equality and computation *qua* compilation (Bove–Capretta, McBride, Brady, *etc.*).
   See **Lean 4** and **Idris 2**!

Some challenges for higher dimensional types

- Homotopy coherence **vastly better behaved** than proof irrelevance, but **much harder to use**. How to develop usable interfaces to h'topy-coherent structures?
- Resolve the contradiction between *lifting* (cubical) and *pasting* (homotopy.io).
- Develop immersive **visual tools** for higher-dimensional mathematics that can be used by both domain experts and students.
- Develop realistic machine models for cubical computation (*à bas* substitution!).

Some challenges for our community

- **Apply our tools to real math**: emphasis of huge inductions has distortive effect.
  - See progress made by the **Lean**/mathlib community.
  - Many silly debates ("unique choice or not??") have **objectively correct answers** in the presence of applications beyond "progress and preservation".
- **Turn away the crypto influencers!** Blockchain and related trends are currently a hot area of applications, but short-term funding for ITPs is not worth destroying our living environment. Our community must respond quickly and resolutely against this menace to the human life-form.
- **Embrace & learn from the hobbyists.** Fashionable with accomplished academics to say that (*e.g.*) dependent types are impossibly hard, but zoomers on Twitter are thankfully ignoring them and writing their own cubical(!!) implementations!
- **Go down to the countryside.** We speak of DEI, but to achieve it we must reach outside our insular communities and learn from the masses and their needs. ITP has *potential* to bring mathematics to non-collegiate learners, and vice versa.

Thank you, and **have fun**!