Guarded Cubical Type Theory: Path Equality for Guarded Recursion

Birkedal, Bizjak, Clouston
Grathwohl, Spitters and Vezzosi

Aarhus University

Bishop's numerical language and type theory

Bishop: constructive mathematics as a language for numerical computations
State of the art: type theory based proof assistants
Big library (corn/math-classes) for verified exact analysis in Coq:
Reals, metric spaces, simple ODE solver which actually compute in type theory (with O'Connor, then ForMath)
Want better support/semantics for:
(co)inductive definitions, quotients, transport of structures, ...
GCTT is also a step in that direction
Also: need better algorithms (MAP16)
Verifying huge computational proofs in type theory is feasible

Introduction

Two motivations for guarded cubical type theory:

- Path equality for guarded dependent types
  application in computer science
- Add guarded recursion to cubical type theory.

Univalent type theory

New foundation for (constructive) maths based on homotopy types
Extends set theoretic foundation
Analogy between setoids (=Bishop sets) and 0-types
Type of points and at most one proof that they are equal
Richman's families of Bishop sets
Families of setoids correspond to maps $A/= \to SET/\cong$ in the
Hofmann Streicher groupoid model
HS has univalence (isomorphic types may be identified)
Adding more universes leads to ∞-groupoids (=homotopy types)
These can be modeled by simplicial sets
sSet model of univalent type theory Voevodsky

Univalent type theory

sSet model is classical, no computation
Coquand: constructive cubical model of univalent type theory
Cubical type theory and type checker

Cubical type theory

Idea: equalities are paths, maps from abstract interval to the type
Abstract interval $\mathbb{I}$ is modeled by a DeMorgan algebra
distributive lattice with involution sat DeMorgan laws
Involution: $(1 - r)$ Paths between paths are squares, etc

Cubical type theory

$$
\begin{array}{llll}
\Gamma, \Delta & ::= & () \mid \Gamma, x : A & \text{Contexts} \\
t, u, A, B & ::= & x \mid \lambda x : A.t \mid t\, u \mid (x : A) \to B & \Pi\text{-types} \\
& \mid & (t, u) \mid t.1 \mid t.2 \mid (x : A) \times B & \Sigma\text{-types} \\
& \mid & 0 \mid \mathsf{s}\, t \mid \mathsf{natrec}\, t\, u \mid \mathsf{N} & \text{Naturals} \\
& \mid & \mathsf{U} & \text{Universe}
\end{array}
$$

Interval $\mathbb{I}$ (context, not a type)

$$r, s \ ::= \ 0 \mid 1 \mid i \mid 1 - r \mid r \wedge s \mid r \vee s.$$

$$\Gamma, \Delta \ ::= \ \cdots \mid \Gamma, i : \mathbb{I}.$$

Path types

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash t : A \qquad \Gamma \vdash u : A}{\Gamma \vdash \mathsf{Path}\, A\ t\ u}$$

$$\frac{\Gamma \vdash A \qquad \Gamma, i : \mathbb{I} \vdash t : A}{\Gamma \vdash \langle i \rangle\, t : \mathsf{Path}\, A\ t[0/i]\ t[1/i]} \qquad \frac{\Gamma \vdash t : \mathsf{Path}\, A\ u\ s \qquad \Gamma \vdash r : \mathbb{I}}{\Gamma \vdash t\, r : A}$$

Figure: Typing rules for path types.

Funext for Path

Proof term for functional extensionality:

$$\text{funext}\ f\ g \triangleq \lambda p.\langle i\rangle\,\lambda x.\,p\,x\,i \quad : \quad ((x : A) \to \text{Path}\,B\ (f\,x)\ (g\,x)) \to \text{Path}\ (A \to B)\ f\ g.$$

Face lattice

Free distributive lattice on the symbols $(i = 0)$ and $(i = 1)$ for all names i, quotiented by the relation $(i = 0) \wedge (i = 1) = 0_{\mathbb{F}}$.

$$\varphi, \psi \ ::= \ 0_{\mathbb{F}} \mid 1_{\mathbb{F}} \mid (i = 0) \mid (i = 1) \mid \varphi \wedge \psi \mid \varphi \vee \psi.$$

Restriction of a context to a face:

$$\Gamma, \Delta \ ::= \ \cdots \mid \Gamma, \varphi.$$

For example, $\Gamma, \varphi \vdash \psi_1 = \psi_2 : \mathbb{F}$ is equivalent to
$\Gamma \vdash \varphi \wedge \psi_1 = \varphi \wedge \psi_2 : \mathbb{F}$
$\Gamma \vdash t : A[\varphi \mapsto u]$ abbreviates $\Gamma \vdash t : A$ and $\Gamma, \varphi \vdash t = u : A$

Composition

$$\frac{\Gamma \vdash \varphi : \mathbb{F} \quad \Gamma, i : \mathbb{I} \vdash A \qquad \Gamma, \varphi, i : \mathbb{I} \vdash u : A \qquad \Gamma \vdash a_0 : A[0/i][\varphi \mapsto u[0/i]]}{\Gamma \vdash \mathsf{comp}^i\ A\ [\varphi \mapsto u]\ a_0 : A[1/i][\varphi \mapsto u[1/i]]}.$$

Transport operation for Path types

$$\mathsf{transp}^i\ A\ a\ \triangleq\ \mathsf{comp}^i\ A\ [0_{\mathbb{F}} \mapsto []]\ a\ :\ A[1/i].$$

where a has type $A[0/i]$.
Example of the use of systems is a proof that Path is transitive; given $p\ :\ \mathsf{Path}\, A\ a\ b$ and $q\ :\ \mathsf{Path}\, A\ b\ c$ we can define

$$\mathsf{trans}\ p\, q \triangleq \langle i \rangle \mathsf{comp}^j\ A\, [(i = 0) \mapsto a, (i = 1) \mapsto q\, j]\ (p\, i)\ :\ \mathsf{Path}\, A\, a\, c$$

This builds a path between the appropriate endpoints because:
$\mathsf{comp}^j\ A\ [1_{\mathbb{F}} \mapsto a]\ (p\, 0) = a$
$\mathsf{comp}^j\ A\ [1_{\mathbb{F}} \mapsto q\, j]\ (p\, 1) = q\, 1 = c$.

Cubical type theory

Also: Glue, universe, ...

CTT is an extension of MLTT with functional extensionality and univalence.

(Precise: For strict J one uses a modified equality Id (Swan))

Guarded recursion

A way of defining infinite objects using self-reference
E.g. streams
New type former $\triangleright$ ('later', data available tomorrow)

$$\mathsf{Str}_A \ = \ A \times \triangleright \mathsf{Str}_A$$

$\mathit{fix} : (\triangleright A \to A) \to A$ for solving domain equations
Also used to model program logics (concurrency, ...)

Semantics of $\triangleright$

Guarded recursion (GDTT) modeled in $\widehat{\omega}$ (topos of trees).

$$(\triangleright X)(n) = \begin{cases} \{\star\} & \text{if } n = 0 \\ X(m) & \text{if } n = m + 1 \end{cases}$$

GDTT is an extensional type theory.
Want computation, so intensional variant.
Need a computational interpretation for the proof that bisimular streams are equal. More generally:
$\triangleright \mathsf{Id}\, A\, t\, u \to \mathsf{Id}\, (\triangleright A)\, (\mathsf{next}\, t)\, (\mathsf{next}\, u)$
Compare: funext
$(x : A) \to \mathsf{Id}\, B\, (fx)\, (gx) \to \mathsf{Id}\, (A \to B)\, f\, g$

Later

$$\frac{\Gamma \vdash}{\vdash \cdot : \Gamma \rightarrowtriangle \cdot} \qquad \frac{\vdash \xi : \Gamma \rightarrowtriangle \Gamma' \qquad \Gamma \vdash t : \triangleright \xi.A}{\vdash \xi\,[x \leftarrow t] : \Gamma \rightarrowtriangle \Gamma', x : A}$$

Figure: Formation rules for delayed substitutions.

Do notation, applicative functor

$$\frac{\Gamma, \Gamma' \vdash A \qquad \vdash \xi : \Gamma \rightarrowtriangle \Gamma'}{\Gamma \vdash \triangleright \xi.A} \qquad \frac{\Gamma, \Gamma' \vdash A : \mathsf{U} \qquad \vdash \xi : \Gamma \rightarrowtriangle \Gamma'}{\Gamma \vdash \triangleright \xi.A : \mathsf{U}}$$

$$\frac{\Gamma, \Gamma' \vdash t : A \qquad \vdash \xi : \Gamma \rightarrowtriangle \Gamma'}{\Gamma \vdash \mathsf{next}\, \xi.\, t : \triangleright \xi.A}$$

Figure: Typing rules for later types.

$$\frac{\vdash \xi\,[x \leftarrow t] : \Gamma \multimap \Gamma', x : B \qquad \Gamma, \Gamma' \vdash A}{\Gamma \vdash \rhd\xi\,[x \leftarrow t]\,.A = \rhd\xi.A}$$

$$\frac{\begin{array}{c}\vdash \xi\,[x \leftarrow t, y \leftarrow u]\,\xi' : \Gamma \multimap \Gamma', x : B, y : C, \Gamma'' \\ \Gamma, \Gamma' \vdash C \qquad \Gamma, \Gamma', x : B, y : C, \Gamma'' \vdash A\end{array}}{\Gamma \vdash \rhd\xi\,[x \leftarrow t, y \leftarrow u]\,\xi'.A = \rhd\xi\,[y \leftarrow u, x \leftarrow t]\,\xi'.A}$$

$$\frac{\vdash \xi : \Gamma \multimap \Gamma' \qquad \Gamma, \Gamma', x : B \vdash A \qquad \Gamma, \Gamma' \vdash t : B}{\Gamma \vdash \rhd\xi\,[x \leftarrow \mathsf{next}\,\xi.\,t]\,.A = \rhd\xi.A[t/x]}$$

Figure: Type equality rules for later types

$$\frac{\vdash \xi\,[x \leftarrow t] : \Gamma \rightarrowtriangle \Gamma', x : B \qquad \Gamma, \Gamma' \vdash u : A}{\Gamma \vdash \mathsf{next}\,\xi\,[x \leftarrow t]\,.\,u = \mathsf{next}\,\xi.\,u : \rhd\xi.A}$$

$$\frac{\vdash \xi\,[x \leftarrow t, y \leftarrow u]\,\xi' : \Gamma \rightarrowtriangle \Gamma', x : B, y : C, \Gamma'' \qquad \Gamma, \Gamma' \vdash C \qquad \Gamma, \Gamma', x : B, y : C, \Gamma'' \vdash v : A}{\Gamma \vdash \mathsf{next}\,\xi\,[x \leftarrow t, y \leftarrow u]\,\xi'.\,v = \mathsf{next}\,\xi\,[y \leftarrow u, x \leftarrow t]\,\xi'.\,v : \rhd\xi\,[x \leftarrow t, y \leftarrow u]\,\xi'.A}$$

$$\frac{\vdash \xi : \Gamma \rightarrowtriangle \Gamma' \qquad \Gamma, \Gamma', x : B \vdash u : A \qquad \Gamma, \Gamma' \vdash t : B}{\Gamma \vdash \mathsf{next}\,\xi\,[x \leftarrow \mathsf{next}\,\xi.\,t]\,.\,u = \mathsf{next}\,\xi.\,u[t/x] : \rhd\xi.A[t/x]}$$

$$\frac{\Gamma \vdash t : \rhd\xi.A}{\Gamma \vdash \mathsf{next}\,\xi\,[x \leftarrow t]\,.\,x = t : \rhd\xi.A}$$

Figure: Term equality rules for later types.

Example

Similar to funext, we now have in GCTT

$$\lambda p.\langle i\rangle \operatorname{next}\xi[p' \leftarrow p].\, p'\, i \;:\; (\rhd\xi.\, \operatorname{Path} A\, t\, u) \to \quad \operatorname{Path}\,(\rhd\xi.A)\,(\operatorname{next}\xi.\, t)\,(\operatorname{next}\xi.\, u).$$

This improves on the equality reflection that was needed before.
Cor: bisimilar streams are equal

Fixpoints

In GDTT: $\mathrm{fix}\, x.t = t[\mathrm{next}\,\mathrm{fix}\, x.t/x]$
(breaks decidable type checking).
Instead, we have a delay fixed point (dfix).
A path from the fixed point ($dfix^0$) to its unfolding ($dfix^1$).

$$\frac{\Gamma \vdash r : \mathbb{I} \qquad \Gamma, x : \rhd A \vdash t : A}{\Gamma \vdash \mathrm{dfix}^r\, x.t : \rhd A}$$

$$\frac{\Gamma, x : \rhd A \vdash t : A}{\Gamma \vdash \mathrm{dfix}^1\, x.t = \mathrm{next}\, t[\mathrm{dfix}^0\, x.t/x] : \rhd A}$$

Proposition (Unique guarded fixed points)

Any guarded fixed-point of $f : \rhd A \to A$ is path equal to $\mathrm{fix}\, x.f\, x$.

Examples

- If $f : A \to A \to B$ is commutative, then zipWith $f : \mathrm{Str}_A \to \mathrm{Str}_A \to \mathrm{Str}_B$ is commutative.
- Let

$$\begin{array}{lcllcl} \mathrm{Rec}_A & \triangleq & \mathrm{fix}\, x.(\rhd[x' \leftarrow x].x') \to A & & \\ \Delta & \triangleq & \lambda x.f(\mathrm{next}[x' \leftarrow x].\,((\mathrm{unfold}\, x')x)) & : & \rhd \mathrm{Rec}_A \to A \\ \mathrm{Y} & \triangleq & \lambda f.\Delta(\mathrm{next}\,\mathrm{fold}\,\Delta) & : & (\rhd A \to A) \to A \end{array}$$

where fold and unfold are the transports along the path between Rec_A and $\rhd \mathrm{Rec}_A \to A$.
Y is a guarded fixed-point combinator.

Semantics of GCTT

Intuitively, cubical model *in* the topos of trees.
(iterated forcing)
Existing theory does not directly work due to strictness and universes. A more concrete construction.
Semantics in $\widehat{\square \times \omega}$.
$\square$ is the opposite of the Kleisli category of the free De Morgan algebra monad on finite sets. (=Lawvere theory of De Morgan algebras).
More concretely, given a countably infinite set of names $i, j, k, \ldots$, $\mathcal{C}$ has as objects finite sets of names I, J. A morphism $I \to J \in \mathcal{C}$ is a function $J \to \mathbf{DM}(I)$, where $\mathbf{DM}(I)$ is the free De Morgan algebra with generators I.

Semantics of GCTT

Theorem

A presheaf topos with a non-trivial ($0 \neq 1$) internal DeMorgan algebra with the disjunction property ($a \vee b = 1 \vdash a = 1 \vee b = 1$) models CTT *(without universe and gluing).*
In particular, $\mathcal{C} \times \mathbb{C}$ for any category $\mathbb{C}$ models CTT*.*

Hence $\widehat{\square \times \omega}$ models CTT.
$\rhd$ can be defined explicitly.

$$(\rhd(X))(I, n) \begin{cases} \{\star\} & \text{if } n = 0 \\ X(I, m) & \text{if } n = m + 1 \end{cases}$$

Key observation: $\rhd$ preserves fibrancy.
Conclusion: $\widehat{\square \times \omega}$ models GCTT.

Glue and universe

If the presheaf topos also has a fibrant universe and $\forall : \mathbb{F}^{\mathbb{I}} \to \mathbb{F}$, then we can model the full CTT.

In particular, $\mathcal{C} \times \mathbb{C}$ for any category $\mathbb{C}$ with an initial object can be used to give semantics to the entire cubical type theory.

Implementation

Prototype build on top of cubical
Hope to integrate into agda

Conclusions

- New type theory GCTT with a model in $\widehat{\mathcal{C} \times \omega}$
  Path equality for guarded dependent type theory
  (Application of HoTT to CS)
- Adding guarded recursion to cubical type theory
- Axiomatic treatment of the cubical model using the internal logic. ‘new’ class of models.

TODO:
canonicity of GCTT(from CTT)