Initiality for Martin-Löf Type Theory

Peter LeFanu Lumsdaine   Guillaume Brunerie
(joint work with Menno de Boer, Anders Mörtberg)

Stockholm University

HoTTEST, 10 September 2020

Some after-talk fixes + additions.
Video: https://youtu.be/1ogUFFUfU_M

Initiality for dependent type theories

By initiality for a type theory, mean a statement like:

Template

The syntactic category of dependent type theory with Σ, Π, and Id-types forms the initial contextual category with Σ, Π, and Id-structure.

Initiality for dependent type theories

By initiality for a type theory, mean a statement like:

Template

The syntactic category of dependent type theory with Σ, Π, and Id-types forms the initial contextual category with Σ, Π, and Id-structure.

- Justifies categorical-algebraic definition of “models of DTT with XYZ-types” as “contextual cats with XYZ-structure”
- Packages the bureaucracy of interpreting syntax into such structures
- Should hold “robustly” for “all reasonable” type theories: not rely on “fragile” properties like normalisation
- Variations: could state with CwA’s, CwF’s, C-systems, etc.; with various different presentations of the type theory; with 2-categorical initiality; …

Status

- **Thesis: Initiality is established**
  - Proven by Streicher (1991 book) for Calculus of Constructions; adapted+refined by Hofmann (1995 thesis) for DTT with Π, Id, $\mathbb{N}$.
  - Proof extends straightforwardly + robustly to other type theories.
  - (NB: Other presentations in literature (that I'm aware of) use techniques specific to certain type theories, don't extend robustly; or use substantially different syntax; or handwave many details.)

Status

- **Thesis: Initiality is established**
  - Proven by Streicher (1991 book) for Calculus of Constructions; adapted+refined by Hofmann (1995 thesis) for DTT with Π, Id, $\mathbb{N}$.
  - Proof extends straightforwardly + robustly to other type theories.
  - (NB: Other presentations in literature (that I'm aware of) use techniques specific to certain type theories, don't extend robustly; or use substantially different syntax; or handwave many details.)
- **Antithesis: Initiality is an open problem**
  - Extension not really straightforward at all!
  - What type theories is it even supposed to hold for? It fails for some!

Status

- **Thesis: Initiality is established**
  - Proven by Streicher (1991 book) for Calculus of Constructions; adapted+refined by Hofmann (1995 thesis) for DTT with Π, Id, $\mathbb{N}$.
  - Proof extends straightforwardly + robustly to other type theories.
  - (NB: Other presentations in literature (that I'm aware of) use techniques specific to certain type theories, don't extend robustly; or use substantially different syntax; or handwave many details.)
- **Antithesis: Initiality is an open problem**
  - Extension not really straightforward at all!
  - What type theories is it even supposed to hold for? It fails for some!
- **Synthesis: Initiality is heuristically well-understood**
  - "Experts" do understand what kinds of type theories it holds for, and how to extend Streicher–Hofmann proof.
  - But: this understanding not clearly articulated anywhere, rigorously or even heuristically.
  - Extension of proof mostly straightforward — minor tweaks needed, no substantial new ideas — but carefully making sure of this involves checking a lot of details.

Status

- **Thesis: Initiality is established**
  - Proven by Streicher (1991 book) for Calculus of Constructions; adapted+refined by Hofmann (1995 thesis) for DTT with Π, Id, ℕ.
  - Proof extends straightforwardly + robustly to other type theories.
  - (NB: Other presentations in literature (that I'm aware of) use techniques specific to certain type theories, don't extend robustly; or use substantially different syntax; or handwave many details.)
- **Antithesis: Initiality is an open problem**
  - Extension not really straightforward at all!
  - What type theories is it even supposed to hold for? It fails for some!
- **Synthesis: Initiality is heuristically well-understood**
  - "Experts" do understand what kinds of type theories it holds for, and how to extend Streicher–Hofmann proof.
  - But: this understanding not clearly articulated anywhere, rigorously or even heuristically.
  - Extension of proof mostly straightforward — minor tweaks needed, no substantial new ideas — but carefully making sure of this involves checking a lot of details. Not obviously straightforward!

Solution proposals

Long-term solution

Define general class of dependent type theories; state and prove initiality for these.

Solution proposals

Long-term solution

Define general class of dependent type theories; state and prove initiality for these.

- Define rigorously a general class of dependent type theories…
- …yielding as many specific theories of interest as possible…
- …modulo differences in presentation, as minor as possible.
- Define corresponding categorical-algebraic structures…
- …yielding the established definitions, as closely as possible…
- …and prove initiality with respect to these.

Solution proposals

Long-term solution

Define general class of dependent type theories; state and prove initiality for these.

- Define rigorously a general class of dependent type theories…
- …yielding as many specific theories of interest as possible…
- …modulo differences in presentation, as minor as possible.
- Define corresponding categorical-algebraic structures…
- …yielding the established definitions, as closely as possible…
- …and prove initiality with respect to these.

Various proposals: Bauer–Lumsdaine–Haselwarter; Brunerie; Uemura; Isaev, Capriotti.

See Peter's HoTTEST, June 2020: https://youtu.be/kQe0knDuZqg

Solution proposals

Long-term solution

Define general class of dependent type theories; state and prove initiality for these.

- Define rigorously a general class of dependent type theories…
- …yielding as many specific theories of interest as possible…
- …modulo differences in presentation, as minor as possible.
- Define corresponding categorical-algebraic structures…
- …yielding the established definitions, as closely as possible…
- …and prove initiality with respect to these.

Various proposals: Bauer–Lumsdaine–Haselwarter; Brunerie; Uemura; Isaev, Capriotti.

See Peter’s HoTTEST, June 2020: https://youtu.be/kQe0knDuZqg

Short-term solution

Just damn well prove it for more non-trivial theories of interest!

Objection!

Frequently asked question

If it’s so dreadfully hard, why not use a different syntax/semantics that makes it easier?

Objection!

Frequently asked question

If it’s so dreadfully hard, why not use a different syntax/semantics that makes it easier?

- It’s not “dreadfully hard”; it’s just not too trivial to be worth proving!

Objection!

Frequently asked question

If it’s so dreadfully hard, why not use a different syntax/semantics that makes it easier?

- It’s not “dreadfully hard”; it’s just not too trivial to be worth proving!
- Off-the-shelf essentially algebraic syntax for CwF’s:
  forgets much of the structure of the type theory;
  so, doesn’t support the other motivating metatheorems.

Objection!

Frequently asked question

If it's so dreadfully hard, why not use a different syntax/semantics that makes it easier?

- It's not "dreadfully hard"; it's just not too trivial to be worth proving!
- Off-the-shelf essentially algebraic syntax for CwF's:
  forgets much of the structure of the type theory;
  so, doesn't support the other motivating metatheorems.
- Logical framework presentations:
  pre-Uemura: couldn't see how they suffice for other motivations;
  linking the "naïve" and LF syntaxes: clearest proof I know (Hofmann) uses initiality for naïve syntax.

Objection!

Frequently asked question

If it’s so dreadfully hard, why not use a different syntax/semantics that makes it easier?

- It’s not “dreadfully hard”; it’s just not too trivial to be worth proving!
- Off-the-shelf essentially algebraic syntax for CwF’s:
  forgets much of the structure of the type theory;
  so, doesn’t support the other motivating metatheorems.
- Logical framework presentations:
  pre-Uemura: couldn’t see how they suffice for other motivations;
  linking the “naïve” and LF syntaxes: clearest proof I know (Hofmann) uses initiality for naïve syntax.
- And various other options, e.g. QIITs. Generally: each has some advantages; but traditional “naïve” syntax still one important approach to understand.

Stockholm initiality formalisations

Goal

Prove initiality formally, for some specific type theory, ideally approaching “book HoTT”.

Stockholm initiality formalisations

Goal

Prove initiality formally, for some specific type theory, ideally approaching "book HoTT".

Two parallel developments, planned together: Brunerie–de Boer in Agda, Lumsdaine–Mörtberg in Coq.

Small type theory at first: Π-types, a dependent family of base types.

Key design criterion: robust extensibility. Avoid doing anything that wouldn't extend to "arbitrary" constructors/rules.

Stockholm initiality formalisations

Goal

Prove initiality formally, for some specific type theory, ideally approaching “book HoTT”.

Two parallel developments, planned together: Brunerie–de Boer in Agda, Lumsdaine–Mörtberg in Coq.

Small type theory at first: Π-types, a dependent family of base types.

Key design criterion: robust extensibility. Avoid doing anything that wouldn’t extend to “arbitrary” constructors/rules.

“We can have this done within a week.” — PLL, 19 October 2018

Stockholm initiality formalisations

Goal

Prove initiality formally, for some specific type theory, ideally approaching “book HoTT”.

Two parallel developments, planned together: Brunerie–de Boer in Agda, Lumsdaine–Mörtberg in Coq.

Small type theory at first: Π-types, a dependent family of base types.

Key design criterion: robust extensibility. Avoid doing anything that wouldn’t extend to “arbitrary” constructors/rules.

“We can have this done within a week.” — PLL, 19 October 2018

Developments begun 22 October; Brunerie-de Boer initiality attained 19 November!

Categories with families proof

Type theory under consideration

Raw syntax with binding: de Bruijn indices, well-scoped, fully annotated.

Definition

Raw expressions: family of sets $\mathrm{Expr}^{\mathrm{ty}}(n)$, $\mathrm{Expr}^{\mathrm{tm}}(n)$ inductively generated by variables and constructors:

- for $i \in \{1, \ldots, n\}$, have $x_i \in \mathrm{Expr}^{\mathrm{tm}}(n)$;
- for $A \in \mathrm{Expr}^{\mathrm{ty}}(n)$, $B \in \mathrm{Expr}^{\mathrm{ty}}(n+1)$, have $\Pi(A, B) \in \mathrm{Expr}^{\mathrm{ty}}(n)$;
- for $A \in \mathrm{Expr}^{\mathrm{ty}}(n)$, $B \in \mathrm{Expr}^{\mathrm{ty}}(n+1)$, $f, a \in \mathrm{Expr}^{\mathrm{tm}}(n)$, have $\mathrm{app}(A, B, f, a) \in \mathrm{Expr}^{\mathrm{ty}}(n)$
- similar clauses for each constructor of the type theory

Basic operations (e.g. substitution) and lemmas all defined+proven recursively+inductively from these.

Judgements, derivability

Definition

Raw context Γ of length n: sequence of type expressions in scope n.

Judgements: tuples of expressions of the following forms

$$\Gamma \vdash A \ \mathsf{type} \qquad\qquad \Gamma \vdash a : A$$
$$\Gamma \vdash A \equiv B \ \mathsf{type} \qquad\qquad \Gamma \vdash a \equiv b : A$$

Judgements, derivability

Definition

Raw context Γ of length n: sequence of type expressions in scope n.

Judgements: tuples of expressions of the following forms

$$\begin{array}{ll} \Gamma \vdash A \text{ type} & \Gamma \vdash a : A \\ \Gamma \vdash A \equiv B \text{ type} & \Gamma \vdash a \equiv b : A \end{array}$$

Definition

Derivability: inductively defined relation on judgements, by usual rules:

- structural rules;
- logical + congruence rules for the specific constructors.

Judgements, derivability

Definition

Raw context Γ of length n: sequence of type expressions in scope n.

Judgements: tuples of expressions of the following forms

$$\begin{array}{ll} \Gamma \vdash A \text{ type} & \Gamma \vdash a : A \\ \Gamma \vdash A \equiv B \text{ type} & \Gamma \vdash a \equiv b : A \end{array}$$

Definition

Derivability: inductively defined relation on judgements, by usual rules:

- structural rules;
- logical + congruence rules for the specific constructors.

Note: no context or context-equality judgement, either primitive or used in rules. Can show: this doesn't affect derivability.

Categorical models

Definition (Category with families)

- category $\mathbf{C}$ of contexts;
- presheaf of types: $\mathrm{Ty}(\Gamma)$, for $\Gamma \in \mathbf{C}$;
- presheaf of terms: $\mathrm{Tm}(\Gamma, A)$, for $A \in \mathrm{Ty}(\Gamma)$;
- various operations, properties

Categorical models

Definition (Category with families)

- category $\mathbf{C}$ of contexts;
- presheaf of types: $\mathrm{Ty}(\Gamma)$, for $\Gamma \in \mathbf{C}$;
- presheaf of terms: $\mathrm{Tm}(\Gamma, A)$, for $A \in \mathrm{Ty}(\Gamma)$;
- various operations, properties

(Closely equivalent: categories with attributes, split type-categories, …)

Definition

Logical structure on a CwA: operations for all constructors + rules of the type theory.

Environments; partial interpretation

Definition

Environment for scope n in $\mathbf{C}$:

object $\Gamma \in \mathbf{C}$ and partial map $E : \{1, \ldots, n\} \longrightarrow \sum_{A \in \mathrm{Ty}(\Gamma)} \mathrm{Tm}(\Gamma, A)$.

Idea: Γ the interpretation of a context; E gives its types and variables.

Environments; partial interpretation

Definition

Environment for scope n in $\mathbf{C}$:

object $\Gamma \in \mathbf{C}$ and partial map $E : \{1, \ldots, n\} \longrightarrow \sum_{A \in \mathrm{Ty}(\Gamma)} \mathrm{Tm}(\Gamma, A)$.

Idea: Γ the interpretation of a context; E gives its types and variables.

Definition

Given $\mathbf{C}$, environment (Γ, E) for scope n, and expression e in scope n, get partial interpretation $[\![e]\!]^{\Gamma,E}$, suitable type or term, defined by recursion on e.

Interpretation lemmas

Lemma

Partial interpretation is stable under reindexing of environments.

Interpretation lemmas

Lemma

Partial interpretation is stable under reindexing of environments.

Proof.

Structural induction on the expression. □

Interpretation lemmas

Lemma

Partial interpretation is stable under reindexing of environments.

Proof.

Structural induction on the expression. $\square$

Lemma

Partial interpretation interprets syntactic substitution as semantic substitution.

Interpretation lemmas

Lemma

Partial interpretation is stable under reindexing of environments.

Proof.

Structural induction on the expression. □

Lemma

Partial interpretation interprets syntactic substitution as semantic substitution.

Proof.

Structural induction on the expression. □

Interpretation lemmas

Lemma

Partial interpretation is stable under reindexing of environments.

Proof.

Structural induction on the expression. □

Lemma

Partial interpretation interprets syntactic substitution as semantic substitution.

Proof.

Structural induction on the expression. □

Totality

Lemma

Interpretation is total on derivable judgements.

Totality

Lemma

Interpretation is total on derivable judgements.

Proof.

Structural induction on the derivation. □

Totality

Lemma

Interpretation is total on derivable judgements.

Proof.

Structural induction on the derivation. □

Lemma

Partial interpretation is functorial in (weak) maps of CwF's.

Totality

Lemma

Interpretation is total on derivable judgements.

Proof.

Structural induction on the derivation. □

Lemma

Partial interpretation is functorial in (weak) maps of CwF's.

Proof.

Structural induction on the expression. □

Syntactic category

Definition

Syntactic CwF: constructed from syntax, modulo judgemental equality.

Syntactic category

Definition

Syntactic CwF: constructed from syntax, modulo judgemental equality.

Theorem (Initiality)

The syntactic CwF is initial among CwF's with logical structure.

Syntactic category

Definition

Syntactic CwF: constructed from syntax, modulo judgemental equality.

Theorem (Initiality)

The syntactic CwF is initial among CwF's with logical structure.

Proof.

Existence: by totality of the interpretation.
Uniqueness: by functoriality of the interpretation. □

- All inductions structural.

Syntactic category

Definition

Syntactic CwF: constructed from syntax, modulo judgemental equality.

Theorem (Initiality)

The syntactic CwF is initial among CwF's with logical structure.

Proof.

Existence: by totality of the interpretation.
Uniqueness: by functoriality of the interpretation. □

- All inductions structural.
- No mention of equality on objects.

Coq formalisation

Background

Approach based in part on previous attempt by PLL from 2014–15 (in joint development Gylterud–Lumsdaine–Palmgren).

Attempt foundered due to combination of several design choices making life hard:

- use of named variables
- use of setoids for the target models
- started with slightly overambitious type theory
- …

Very useful experience to build on — both the good and the bad aspects…

Design choices

This time round:

- Proof assistant: Coq; specifically, over UniMath. (Mainly: for a well-developed category theory library that both authors were familiar with.)
- Models: Categories with attributes, not assuming objects form a set (so, CwA a 2-category); for 1-categorical initiality, contextual categories as CwA's plus contextuality axiom (implying objects a set).
- Variables in raw syntax: using de Bruijn indices. Raw syntax: well-scoped. These enable:
- All inductions purely structural, over either raw syntax or derivations. No size measures, auxiliary well-founded relations, etc.
- Context and context-equality judgements subsidiary, not primitive, don't appear in derivations. Substitution admissible, not a primitive rule. These enable:
- Interpretation fuction (partial + totality): into arbitrary CwA's. No use of equality on objects/contexts needed.

Experience

Good news

1. Interpretation function (partial + total): went very smoothly.

Experience

Good news

1. Interpretation function (partial + total): went very smoothly.
2. Admissibility of substitution, etc: went surprisingly smoothly.

Experience

Good news

1. Interpretation function (partial + total): went very smoothly.
2. Admissibility of substitution, etc: went surprisingly smoothly.
3. Categorical operations on syntax: clean to define, derive properties as judgemental equalities, etc.

Experience

Good news

1. Interpretation function (partial + total): went very smoothly.
2. Admissibility of substitution, etc: went surprisingly smoothly.
3. Categorical operations on syntax: clean to define, derive properties as judgemental equalities, etc.

Bad news

1. Quotients

Experience

Good news

1. Interpretation function (partial + total): went very smoothly.
2. Admissibility of substitution, etc: went surprisingly smoothly.
3. Categorical operations on syntax: clean to define, derive properties as judgemental equalities, etc.

Bad news

1. Quotients
2. Quotients

Experience

Good news

1. Interpretation function (partial + total): went very smoothly.
2. Admissibility of substitution, etc: went surprisingly smoothly.
3. Categorical operations on syntax: clean to define, derive properties as judgemental equalities, etc.

Bad news

1. Quotients
2. Quotients
3. Quotients

Experience

Good news

1. Interpretation function (partial + total): went very smoothly.
2. Admissibility of substitution, etc: went surprisingly smoothly.
3. Categorical operations on syntax: clean to define, derive properties as judgemental equalities, etc.

Bad news

Specifically, interaction of 2 issues:

1. Syntactic CwA: dependently-typed (maps depend on objects), with objects quotiented (contexts, up to judgemental equality)
2. In UniMath's quotients, the dependent eliminator doesn't compute judgementally.

Together: ends up at least as painful as using setoids.

Status

1. Complete: “Pre-quotient” parts. Contains most mathematically interesting parts, and most useful for applications.
   - definition of type theory, main syntactic lemmas;
   - definition of suitably structured CwA’s, basic lemmas;
   - interpretation function into suitable CwA’s (pre-quotient part of existence for initiality)
   - functoriality of interpretation under CwA (pseudo-)maps (pre-quotient ingredient of uniqueness for initiality)

   Around 4,000 lines of code (not including libraries).
2. Incomplete: “post-quotient” parts, i.e. assembling into the syntactic CwA and functors thereon. Mathematically mostly less interesting, but hard to formalise.
   - Done: syntactic category; most of CwA structure thereon; some logical structure; most of underlying functor of interpretation map.
   - Remaining: rest of CwA structure, and logical structure; interpretation as a structure-preserving map of CwA’s; uniqueness of the interpretation map.

   Arround 3,000 lines of code, so far!

Over to Guillaume!