Initiality for Martin-Löf type theory (in Agda)

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

HoTTEST seminar
10 September 2020

Initiality

Some features of the initiality theorem we proved.

- Initiality for Martin-Löf type theory with Π, Σ, Id, $\mathbb{N}$, $+$, $\perp$, $\top$, U_i, El.
- Syntax is fully annotated, with Tarski-style universes, and substitution is a defined meta-operation.
- Models are contextual categories with extra structure (seen as an essentially algebraic theory).
- Formalized[1] in Agda 2.6.1 with Prop (+ function extensionality, propositional extensionality, and quotients).

[1] https://github.com/guillaumebrunerie/initiality/tree/v2.0

Prop

For convenience, we use the type Prop of strict propositions[1]:

If A : Prop and $u, v : A$, then u and v are judgmentally equal.

- The identity type is Prop-valued,
- a *partial element* of a type A is a pair (P, f) with P : Prop and $f : P \to A$,
- an *equivalence relation* on a type A is $\sim : A \to A \to \mathrm{Prop}$ which is reflexive, symmetric and transitive,
- derivability of judgments is an inductive family in Prop.

Inductive types in Prop cannot be eliminated into arbitrary types, but this hasn't been an issue for this project.

[1] *Definitional Proof-Irrelevance without K*
G. Gilbert, J. Cockx, M. Sozeau, N. Tabareau

Essentially algebraic theories

An *essentially algebraic theory* consists of

- a collection of *sorts*,
- a collection of *function symbols*, each of them having a type

  $$(x_1 : s_1) \dots (x_n : s_n)\ e_1 \ \cdots \ e_m \to s$$

  where $s_1, \dots, s_n, s$ are sorts, and $e_1, \dots, e_m$ are equations involving variables and previously declared function symbols,
- a collection of *equations*.

Given an essentially algebraic theory, it has a category of models

- A *model* is given by a set for each sort, a partial function for each function symbol, satisfying all the equations.
- A *morphism* between models is given by maps between the corresponding sets, which commute with the partial functions.

Categories

The essentially algebraic theory of *categories* consists of

- two sorts Ob and Mor
- domain, codomain, identity, composition:

$$\partial_0 : \mathrm{Mor} \to \mathrm{Ob}, \quad \partial_1 : \mathrm{Mor} \to \mathrm{Ob}, \quad \mathrm{id} : \mathrm{Ob} \to \mathrm{Mor},$$

$$\mathrm{comp} : (g : \mathrm{Mor})(f : \mathrm{Mor})(p : \partial_1(f) = \partial_0(g)) \to \mathrm{Mor}$$

- seven equations

$$\mathrm{id}_0 : \partial_0(\mathrm{id}(X)) = X \quad (\text{for } X : \mathrm{Ob}),$$

and id_1, comp_0, comp_1, id-left, id-right, assoc.

Contextual categories for type theory

Given a type theory, we can define a category where

- The *objects* are the derivable contexts, up to judgmental equality,
- The *morphisms* are the derivable context morphisms / total substitutions, up to judgmental equality,
- Objects are graded by their length,
- There is a *father* operation sending $\vdash (\Gamma, A)$ to $\vdash \Gamma$,
- And operations corresponding to substitution, variables, etc.

A type A in context Γ is seen as the object (Γ, A) whose father is Γ.

A term u of type A in context Γ is seen as the context morphism $(\mathrm{id}_\Gamma, u) : \Gamma \to (\Gamma, A)$, which is such that the composition $\Gamma \to (\Gamma, A) \to \Gamma$ is the identity.

Contextual categories[1]

Contextual categories are categories where objects are graded by natural numbers, and together with:

- $\mathbb{N} \sqcup \mathbb{N}^2$ sorts: Ob_n and $\mathrm{Mor}_{n,m}$
- seven new operations

$$\mathrm{ft} : \mathrm{Ob}_{n+1} \to \mathrm{Ob}_n \qquad \mathrm{pp} : \mathrm{Ob}_{n+1} \to \mathrm{Mor}_{n+1,n}$$

$$\mathrm{star} : (f : \mathrm{Mor}_{m,n})(X : \mathrm{Ob}_{n+1})(p : \partial_1(f) = \mathrm{ft}(X)) \to \mathrm{Ob}_{m+1}$$

$$\mathrm{qq} : (f : \mathrm{Mor}_{m,n})(X : \mathrm{Ob}_{n+1})(p : \partial_1(f) = \mathrm{ft}(X)) \to \mathrm{Mor}_{m+1,n+1}$$

$$\mathrm{ss} : \mathrm{Mor}_{m,n+1} \to \mathrm{Mor}_{m,m+1}$$

$$\mathrm{pt} : \mathrm{Ob}_0 \qquad \mathrm{pt\text{-}mor} : \mathrm{Ob}_n \to \mathrm{Mor}_{n,0}$$

- nineteen new equations

[1] `contextualcat.agda#CCat`

Structured contextual categories[1]: type formers

For every type former we add one new operation and one new equation. For instance for Π-formation:

$$\begin{aligned} \mathrm{PiStr} : (\Gamma : \mathrm{Ob}_n)(A : \mathrm{Ob}_{n+1})(A_{\mathrm{ft}} : \mathrm{ft}(A) = \Gamma) \\ (B : \mathrm{Ob}_{n+2})(B_{\mathrm{ft}} : \mathrm{ft}(B) = A) \to \mathrm{Ob}_{n+1} \\ \mathrm{PiStr}_{\mathrm{ft}} : (\Gamma\ A\ A_{\mathrm{ft}}\ B\ B_{\mathrm{ft}} : [\cdots]) \to \mathrm{ft}(\mathrm{PiStr}(\Gamma, A, A_{\mathrm{ft}}, B, B_{\mathrm{ft}})) = \Gamma \end{aligned}$$

corresponding to

$$\frac{\vdash \Gamma \qquad \Gamma \vdash A \qquad \Gamma, x : A \vdash B}{\Gamma \vdash \Pi_{x:A} B}$$

[1] `contextualcat.agda#StructuredCCat`

Structured contextual categories[1]: term formers

For every term former we add one new operation and two new equations. For instance for the successor on natural numbers:

$$\mathsf{sucStr} : (\Gamma : \mathsf{Ob}_n)(u : \mathsf{Mor}_{n,n+1})(u_s : \text{is-term}(u))(u_1 : \partial_1(u) = \mathsf{NatStr}(\Gamma)) \\ \to \mathsf{Mor}_{n,n+1}$$

$$\mathsf{sucStr}_s : (\Gamma\ u\ u_s\ u_1 : [\cdots]) \to \text{is-term}(\mathsf{sucStr}(\Gamma, u, u_s, u_1))$$

$$\mathsf{sucStr}_1 : (\Gamma\ u\ u_s\ u_1 : [\cdots]) \to \partial_1(\mathsf{sucStr}(\Gamma, u, u_s, u_1)) = \mathsf{NatStr}(\Gamma)$$

corresponding to

$$\frac{\vdash \Gamma \qquad \Gamma \vdash u : \mathbb{N}}{\Gamma \vdash \mathsf{suc}(u) : \mathbb{N}}$$

(where is-term(u) is $\mathsf{comp}(\mathsf{pp}(\partial_1(u)), u) = \mathsf{id}(\partial_0(u))$)

[1] `contextualcat.agda#StructuredCCat`

Structured contextual categories[1]: naturality

Substitution commutes with type/term-formers. We add one new such equation for every type/term-former. For instance:

$$\begin{aligned} \mathsf{PiStrNat} : \mathsf{star}(\delta, \mathsf{PiStr}(\Delta, A, A_{ft}, B, B_{ft})) \\ = \mathsf{PiStr}(\Gamma, \mathsf{star}(\delta, A), \_, \mathsf{star}^{+}(\delta, B), \_) \\ \mathsf{sucStrNat} : \mathsf{starTm}(\delta, \mathsf{sucStr}(\Delta, u, u_s, u_1)) \\ = \mathsf{sucStr}(\Gamma, \mathsf{starTm}(\delta, u), \_, \_) \end{aligned}$$

corresponding to

$$(\Pi_{x:A} B)[\delta] = \Pi_{x:A[\delta]} B[\delta^{+}]$$

$$\mathsf{suc}(u)[\delta] = \mathsf{suc}(u[\delta])$$

[1] `contextualcat.agda#StructuredCCat`

Structured contextual categories[1]: equalities

Finally, we add the appropriate equalities corresponding to judgmental equality rules (e.g., β/η).

We now have an essentially algebraic theory corresponding to models of our type theory, and hence a 1-category of models.

[1] `contextualcat.agda#StructuredCCat`

Quotients[1]

Quotients are postulated like a higher inductive type.

Given a type A and a Prop-valued equivalence relation $\sim$ on A, the quotient $A/\sim$ has two constructors

- $\mathsf{proj} : A \to A/\sim$
- $\mathsf{eq} : \{a\ b : A\}(r : a \sim b) \to \mathsf{proj}(a) = \mathsf{proj}(b)$

together with a dependent elimination rule and a judgmental reduction rule for proj.

[1] `quotients.agda`

Effectiveness of quotients[1]

Lemma

Given $a, b : A$*, if* $\mathrm{proj}(a) = \mathrm{proj}(b)$*, then there exists* $r : a \sim b$*.*

Proof (encode-decode).

Given $a : A$, we define $P : A/{\sim} \to \mathrm{Prop}$ by

$$
\begin{aligned}
P(\mathrm{proj}(b)) &= a \sim b \\
\mathrm{ap}_P(\mathrm{eq}(r)) &= [\ldots] : (a \sim b) = (a \sim c) \quad (\text{where } r : b \sim c)
\end{aligned}
$$

(requires propositional extensionality)

Now we prove that given $p : \mathrm{proj}(a) = x$, then $P(x)$ holds (by path induction on p).

Finally, we can apply it to $x = \mathrm{proj}(b)$. □

[1] `quotients.agda#reflect`

The term model[1]

Quotienting contexts and context morphisms by judgmental equality gives us the term model.

For instance for composition of morphisms:

- Assume we have two morphisms d and t, satisfying $\partial_1(d) = \partial_0(t)$
- Take representatives of the equivalence classes, $\Gamma \vdash \delta : \Delta$ for d and $\Delta' \vdash \theta : \Theta$ for t. We have that $\mathrm{proj}(\Delta) = \mathrm{proj}(\Delta')$.
- By effectiveness, we get that $\vdash \Delta = \Delta'$
- Therefore the composition of δ and θ is well-typed and we can project it back to the quotient to get $t \circ d$.

[1] `termmodel.agda`

Partial interpretation function[1]

Partial functions from A to B are seen as element of the type

$$A \to \mathrm{Partial}(B)$$

where

$$\mathrm{Partial}(X) = \Sigma_{P:\mathrm{Prop}}(P \to X)$$

Given a type-expression A, a term-expression u, and $X : \mathrm{Ob}_n$, we define the partial interpretation function (by structural induction)

$$[\![A]\!]_X : \mathrm{Partial}(\mathrm{Ob}_{n+1})$$

$$[\![u]\!]_X : \mathrm{Partial}(\mathrm{Mor}_{n,n+1})$$

satisfying

$$\mathrm{ft}([\![A]\!]_X) = X \qquad \text{is-term}([\![u]\!]_X) \qquad \partial_0([\![u]\!]_X) = X$$

[1] `partialinterpretation.agda`

Example[1]

```
⟦_⟧Ty_ : TyExpr n → Ob n → Partial (Ob (suc n))
⟦_⟧Tm_ : TmExpr n → Ob n → Partial (Mor n (suc n))

⟦ pi A B ⟧Ty X = do
  [A]  ← ⟦ A ⟧Ty X
  [A]= ← assume (ft [A] ≡ X)
  [B]  ← ⟦ B ⟧Ty [A]
  [B]= ← assume (ft [B] ≡ [A])
  return (PiStr X [A] [A]= [B] [B]=)

⟦ suc u ⟧Tm X = do
  [u]   ← ⟦ u ⟧Tm X
  [u]ₛ  ← assume (is-term [u])
  [u]₁  ← assume (∂₁ [u] ≡ NatStr X)
  return (sucStr [u] [u]ₛ [u]₁)
```

[1] partialinterpretation.agda

Totality[1]

(where relevant we assume that $[\![\Gamma]\!]$ and $[\![\Delta]\!]$ are defined, and we write X and Y for their interpretation)

Theorem

- *If* $\vdash \Gamma$, *then* $[\![\Gamma]\!]$ *is defined.*
- *If* $\Gamma \vdash A$, *then* $[\![A]\!]_X$ *is defined.*
- *If* $\Gamma \vdash u : A$, *then* $[\![u]\!]_X$ *is defined and* $\partial_1([\![u]\!]_X) = [\![A]\!]_X$.
- *If* $\Gamma \vdash \delta : \Delta$, *then* $[\![\delta]\!]_{X,Y}$ *is defined and* $\partial_{0/1}([\![\delta]\!]_{X,Y}) = X/Y$.
- *If* $\vdash \Gamma = \Gamma'$, *then* $[\![\Gamma]\!] = [\![\Gamma']\!]$ *(if both are defined).*
- *If* $\Gamma \vdash A = A'$, *then* $[\![A]\!]_X = [\![A']\!]_X$ *(if both are defined).*
- *If* $\Gamma \vdash u = u' : A$, *then* $[\![u]\!]_X = [\![u']\!]_X$ *(if both are defined).*
- *If* $\Gamma \vdash \delta = \delta' : \Delta$, *then* $[\![\delta]\!]_{X,Y} = [\![\delta']\!]_{X,Y}$ *(if both are defined).*

[1] `totality.agda`

Interpretation of substitutions[1]

Theorem

If $\Delta \vdash A$ *and* $\Gamma \vdash \delta : \Delta$*, then* $[\![A[\delta]]\!]_X$ *is defined and moreover*

$$[\![A[\delta]]\!]_X = \mathsf{star}([\![\delta]\!]_{X,Y}, [\![A]\!]_Y)$$

If $\Delta \vdash u : A$ *and* $\Gamma \vdash \delta : \Delta$*, then* $[\![u[\delta]]\!]_X$ *is defined and moreover*

$$[\![u[\delta]]\!]_X = \mathsf{starTm}([\![\delta]\!]_{X,Y}, [\![u]\!]_Y)$$

[1] `interpretationsubstitution.agda`

Initiality (existence)[1]

Given an arbitrary structured contextual category $\mathcal{C}$, we want to construct a morphism from the term model to $\mathcal{C}$.

- $\mathrm{Ob}_n \to \mathrm{Ob}_n^{\mathcal{C}}$: use the partial interpretation of contexts, the fact that it is actually total, and that it respects judgmental equalities,
- $\mathrm{Mor}_{n,m} \to \mathrm{Mor}_{n,m}^{\mathcal{C}}$: same for context morphisms,
- contextual category structure: use the appropriate lemmas, e.g. the substitution lemma, $[\![\mathrm{id}_\Gamma]\!]_{X,X} = \mathrm{id}_X$, and so on,
- additional operations corresponding to type/term formers: use the fact that the partial interpretation function is appropriately defined.

[1] `initiality-existence.agda`

Initiality (uniqueness)[1]

Given two morphisms f, g from the term model to $\mathcal{C}$, we want to prove that they are equal.

Lemma (uniqueness for types)

Given a type A in a context Γ, if $f(\Gamma) = g(\Gamma)$, then $f(\Gamma, A) = g(\Gamma, A)$.

Proved by structural induction on A, for instance

$$\begin{aligned} f(\Gamma, \Pi_A B) &= f(\mathsf{PiStr}(\Gamma, (\Gamma, A), (\Gamma, A, B))) \\ &= \mathsf{PiStr}^{\mathcal{C}}(f(\Gamma), f(\Gamma, A), f(\Gamma, A, B)) \\ &= \mathsf{PiStr}^{\mathcal{C}}(g(\Gamma), g(\Gamma, A), g(\Gamma, A, B)) \\ &= g(\mathsf{PiStr}(\Gamma, (\Gamma, A), (\Gamma, A, B))) = g(\Gamma, \Pi_A B) \end{aligned}$$

[1] `initiality-uniqueness.agda`

Initiality (uniqueness)[1]

Lemma (uniqueness for terms)

Given a term u in a context Γ, if $f(\Gamma) = g(\Gamma)$, then $f(\mathrm{id}_\Gamma, u) = g(\mathrm{id}_\Gamma, u)$ (proved by structural induction on u).

Theorem (for objects)

For any context Γ we have $f(\Gamma) = g(\Gamma)$ (follows from uniqueness for types).

Theorem (for morphisms)

For any context morphism δ we have $f(\delta) = g(\delta)$.

$$\begin{aligned} f(\delta, u) &= f((\delta, x) \circ (\mathrm{id}, u)) \\ &= \mathrm{qq}^{\mathcal{C}}(f(\delta)) \circ^{\mathcal{C}} f(\mathrm{id}, u) \\ &= \mathrm{qq}^{\mathcal{C}}(g(\delta)) \circ^{\mathcal{C}} g(\mathrm{id}, u) = g(\delta, u) \end{aligned}$$

[1] `initiality-uniqueness.agda`

Future directions

- Performance issues in memory usage and type-checking time, this has to be fixed.
- Add even more type formers? For instance we haven't implemented W-types.
- Prove initiality for an "arbitrary" type theory.