Dependent type theory as the initial category with families

Internship at Chalmers University of Technology, with Peter Dybjer and Thierry Coquand

Simon Castellan

December 8, 2014

Introduction

Initiality:

- a term of ring theory (eg. $1+1$) $\rightarrow$ a unique object in any ring.
- a simply typed λ-term $\rightarrow$ a unique object in a CCC

Goal: extension of this result to dependent type theory

- Main problem: several derivations for a typing judgement $\rightarrow$ *coherence problem*

Contribution: an original way of solving this problem

Overview

Coherence problem already solved by [Str91] and [Cur93].
Streicher's way:

1. Define an **annotated syntax**
2. Solve the coherence problem there
3. Prove the equivalence with the usual syntax.

Problem with this approach:

1. Definition on untyped terms
2. Annotations are *ad-hoc*.

Our way:

1. Define a *fully* annotated syntax
2. Solve **completely** the problem (as in [Cur93], but less technical)
3. Prove the equivalence.

Table of contents

Martin-Löf's Logical Framework

- Extension of simply type λ-calculus with *dependant types*, namely:
  - dependent product: $\Pi(x : A)B$ or $\Pi(A, B)$
  - universe: type set and a decoding function $\mathrm{el}(x)$.
  - polymorphism: $\Pi(x : set)(el(x) \Rightarrow el(x))$
- Extends Curry-Howard to first order predicate logic
- Terms appear in types (via el) $\Rightarrow$ computation at the level of types
- Type casting: $t : A$ and $A = A'$ then $t : A'$
- Typing judgement $\Gamma \vdash t : A$ along with equality judgement $\Gamma \vdash t = t' : A$

Explicit substitutions

Application for dependent product

$$\frac{\Gamma \vdash t : \Pi(x : A)B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\ u : B\{u/x\}}$$

$\Rightarrow$ Substitutions becomes part of the syntax.

- **Substitution**: $\Gamma \vdash f : \Delta$ "f implements Δ in Γ".
- Key operations of substitutions:
  1. **projection**: $\Gamma \cdot A \vdash p : \Gamma$
  2. **extension**: $f : \Gamma \to \Delta$ and $\Gamma \vdash t : A \to \langle f, a \rangle : \Gamma \to \Delta \cdot A$
- Contravariance: $\Delta \vdash t : A + \Gamma \vdash f : \Delta \Rightarrow \Gamma \vdash t[f] : A[f]$.

How much annotations

Traditional typing rule:

$$\frac{\Gamma \cdot A \vdash t : B}{\Gamma \vdash \lambda(t) : A \to B}$$

Γ, A, B are implicit. Fully explicit rule:

$$\frac{\Gamma \vdash \qquad \Gamma \vdash A \qquad \Gamma \cdot A \vdash B \qquad \Gamma \cdot A \vdash t : B}{\Gamma \vdash \lambda(\Gamma, A, B, t) : A \to B}$$

- Less space for derivations.

Syntax of our calculus

- 8 judgements: typing and equality for contexts, types, terms, substitutions.

Type constructors:

- $\mathsf{set}(\Gamma)$ (universe)
- $\Pi(\Gamma, A, B)$ (dependent product without variable)
- $A[f]^{\Gamma}_{\Delta}$

Syntax of our calculus

- 8 judgements: typing and equality for contexts, types, terms, substitutions.

Type constructors:

- $\mathrm{set}(\Gamma)$ (universe)
- $\Pi(\Gamma, A, B)$ (dependent product without variable)
- $A[f]^{\Gamma}_{\Delta}$

Typing rule for dependent product

$$\frac{\Gamma \vdash \qquad \Gamma \vdash A \qquad \Gamma \cdot A \vdash B}{\Gamma \vdash \Pi(\Gamma, A, B)}$$

Syntax of our calculus

- 8 judgements: typing and equality for contexts, types, terms, substitutions.

Type constructors:

- set(Γ) (universe)
- $\Pi(\Gamma, A, B)$ (dependent product without variable)
- $A[f]^{\Gamma}_{\Delta}$

Typing rule for substitutions on types

$$\frac{\Gamma \vdash \qquad \Delta \vdash \qquad \Delta \vdash A \qquad \Gamma \vdash f : \Delta}{\Gamma \vdash A[f]}$$

Syntax of our calculus

- 8 judgements: typing and equality for contexts, types, terms, substitutions.

Term constructors:

- $\lambda(\Gamma, A, B, t)$ (λ-abstraction)
- $\mathsf{ap}(\Gamma, A, B, t)$ (unary application)
- $q(\Gamma, A)$ (zeroth de Bruijn variable)
- $(t : A)[f]^{\Gamma}_{\Delta}$ (substitution)

Syntax of our calculus

- 8 judgements: typing and equality for contexts, types, terms, substitutions.

Term constructors:

- $\lambda(\Gamma, A, B, t)$ (λ-abstraction)
- $\mathrm{ap}(\Gamma, A, B, t)$ (unary application)
- $q(\Gamma, A)$ (zeroth de Bruijn variable)
- $(t : A)[f]^{\Gamma}_{\Delta}$ (substitution)

Typing rule for λ-abstraction

$$\frac{\Gamma \vdash \qquad \Gamma \vdash A \qquad \Gamma \cdot A \vdash B \qquad \Gamma \cdot A \vdash t : B}{\Gamma \vdash \lambda(\Gamma, A, B, t) : \Pi(\Gamma, A, B)}$$

Syntax of our calculus

- 8 judgements: typing and equality for contexts, types, terms, substitutions.

Term constructors:

- $\lambda(\Gamma, A, B, t)$ (λ-abstraction)
- $\mathrm{ap}(\Gamma, A, B, t)$ (unary application)
- $q(\Gamma, A)$ (zeroth de Bruijn variable)
- $(t : A)[f]^{\Gamma}_{\Delta}$ (substitution)

Type casting

$$\frac{\Gamma = \Gamma' \vdash \qquad \Gamma \vdash A = A' \qquad \Gamma \vdash t : A}{\Gamma' \vdash t : A'}$$

Syntax of our calculus

- 8 judgements: typing and equality for contexts, types, terms, substitutions.

Term constructors:

- $\lambda(\Gamma, A, B, t)$ (λ-abstraction)
- $\mathsf{ap}(\Gamma, A, B, t)$ (unary application)
- $q(\Gamma, A)$ (zeroth de Bruijn variable)
- $(t : A)[f]^{\Gamma}_{\Delta}$ (substitution)

Term equality (β)

$$\frac{\Gamma \cdot A \vdash t : B}{\Gamma \cdot A \vdash t = \mathsf{ap}(\lambda(t)) : B}$$

Syntax of our calculus

- 8 judgements: typing and equality for contexts, types, terms, substitutions.

Term constructors:

- $\lambda(\Gamma, A, B, t)$ (λ-abstraction)
- $\mathsf{ap}(\Gamma, A, B, t)$ (unary application)
- $q(\Gamma, A)$ (zeroth de Bruijn variable)
- $(t : A)[f]^{\Gamma}_{\Delta}$ (substitution)

Term equality (η)

$$\frac{\Gamma \vdash t : \Pi(\Gamma, A, B)}{\Gamma \vdash t = \lambda(\mathsf{ap}(t)) : \Pi(\Gamma, A, B)}$$

Compressing derivations

- $\delta \mapsto \delta^z$: compressing derivations by
  1. transitivity of equality

$$\dfrac{\dfrac{\dfrac{\vdots}{\Gamma'' \vdash A} \qquad \Gamma' = \Gamma'' \vdash}{\Gamma' \vdash A} \qquad \Gamma = \Gamma' \vdash}{\Gamma \vdash A} \to \dfrac{\dfrac{\vdots}{\Gamma'' \vdash A} \qquad \Gamma = \Gamma''}{\Gamma \vdash A}$$

  2.

@2

reflexivity

$$\dfrac{\dfrac{\vdots}{\Gamma \vdash A} \qquad \Gamma = \Gamma \vdash}{\Gamma \vdash A} \to \dfrac{\vdots}{\Gamma \vdash A}$$

Theorem

Let δ and δ' be two derivations of a judgement J. We have $\delta^z \equiv \delta'^z$.

Coherence lemma

Goal: a definition on derivations $\rightarrow$ definition on judgements.
Interpretation: A map $\varphi : \mathscr{D} \rightarrow X$ such that

$$\varphi\left(\frac{\delta : \Gamma \vdash t : A \qquad \Gamma \vdash A = A'}{\Gamma \vdash t : A'}\right) = \varphi(\delta)$$

Theorem

Any interpretation $\varphi : \mathscr{D} \rightarrow X$ defined on derivations yields a map $\bar{\varphi} : \mathscr{J} \rightarrow X$ defined on typing judgements such that whenever $\delta : J$ then $\varphi(\delta) = \bar{\varphi}(J)$

Categories with families (CwF)

- Categorical semantics centered around contexts and substitutions as morphisms between contexts: definitional equality becomes equality in a CwF
- Category of CwFs
- Example: term model $\mathbb{T}$: quotient of syntax by definitional equality.
- **Goal**: initiality of $\mathbb{T}$

Initiality of $\mathbb{T}$

Let $\mathscr{C}$ be a CwF.

1. Interpretation in any CwF: a map $[\![\cdot]\!]$ from the syntax to $\mathscr{C}$

$$\left[\!\!\left[\frac{\delta_\Gamma :\ \Gamma \vdash \qquad \delta_A :\ \Gamma \vdash A \qquad \delta_B :\ \Gamma \cdot A \vdash B}{\Gamma \vdash \Pi(\Gamma, A, B)}\right]\!\!\right] = \Pi([\![\delta_\Gamma]\!], [\![\delta_A]\!], [\![\delta_B]\!])$$

2. Extends to a morphism of CwFs: $[\![\cdot]\!] : \mathbb{T} \to \mathscr{C}$
   for instance $F([\Gamma \vdash]) = [\![\Gamma \vdash]\!]$
3. Uniqueness: there is a unique map from $\mathbb{T}$ to $\mathscr{C}$.

$\Rightarrow$ $\mathbb{T}$ is an initial object.

Syntax and term model

- We now consider the same calculus but without the extra annotations.

Type constructors:

- set (universe)
- $\Pi(A, B)$ (dependent product without variable)
- $A[f]$ (substitution)

Syntax and term model

- We now consider the same calculus but without the extra annotations.

Type constructors:

- set (universe)
- $\Pi(A, B)$ (dependent product without variable)
- $A[f]$ (substitution)

Syntax and term model

- We now consider the same calculus but without the extra annotations.

Term constructors:

- $\lambda(t)$ (λ-abstraction)
- $\mathrm{ap}(t)$ (unary application)
- q (variable)
- $t[f]$ (substitution)

Syntax and term model

- We now consider the same calculus but without the extra annotations.

Term constructors:

- $\lambda(t)$ (λ-abstraction)
- $\mathrm{ap}(t)$ (unary application)
- q (variable)
- $t[f]$ (substitution)

- $\mathbb{T}^i$: the implicit term model
- Stripping operator s from $\mathbb{T}$ to $\mathbb{T}^i$
- **Goal**: $s : \mathbb{T} \cong \mathbb{T}^i$

s is one-to-one

- Injectivity of s: if $s(\Gamma) = s(\Gamma')$ then $\Gamma = \Gamma' \vdash$.
- hard part, reflexivity case: if $s(\Gamma) \equiv s(\Gamma')$ then $\Gamma = \Gamma' \vdash$.
- We need **normalisation**, because of the substitution rule:

$$\frac{\Gamma \vdash f : \Delta \qquad \Delta \vdash t : A}{\Gamma \vdash t[f] : A[f]}$$

  No Δ in conclusion.

  1. Prove the result for normal term which only substitutions in specific situtions.
  2. Prove that the result extend to non-normal terms.

- s has an inverse $\mathbb{T}^i \to \mathbb{T}$.

  1. By induction: build a right inverse $t : \mathbb{T}^i \to \mathbb{T}$ ($s \circ t = \mathrm{Id}_{\mathbb{T}^i}$)
  2. By initiality of T, we know that $t \circ s = \mathrm{Id}_{\mathbb{T}}$

$\to \mathbb{T}^i$ is initial.

Conclusion

- Original method: fully annotated syntax
- Extension to other dialects (and GAT)
- Third initial CwF: semantic domain (normalization by evaluation)

Biblio

P.L. Curien.
Substitution up to isomorphism.
Fundamenta Informaticae, 19(1-2):51–85, 1993.

T. Streicher.
Semantics of type theory: correctness, completeness, and independence results.
Birkhauser Boston Inc., 1991.