CICM 2016

Incorporating Quotation and Evaluation Into Church's Type Theory: Syntax and Semantics

William M. Farmer

Department of Computing and Software
McMaster University

26 July 2016

Outline

- Motivation.
- Syntax and semantics of $\mathrm{CTT}_{\mathrm{qe}}$.
- Examples.
- Sketch of a proof system.
- Conclusion.

MKM Challenge: Schemas

- How can we express a schema in a proof assistant?
- Example: The induction schema

$$(\varphi[x \mapsto 0] \wedge \forall\, x\,.\,(\varphi \supset \varphi[x \mapsto S(x)])) \supset \forall\, x\,.\,\varphi$$

  represents an infinite collection of axioms where φ ranges over a set of (open) first-order formulas.
- Note that φ ranges over syntactic expressions, not over semantic values.
- The induction schema is used to define the first-order theories of both Presburger arithmetic and Peano arithmetic with φ ranging over different sets of formulas.
- What happens to the induction schema after a new constant is defined?
  - φ ranges over the same set?
  - φ ranges over an extended set?

Approach 1: Replace with a Single Axiom

- The induction schema is replaced with the second-order induction axiom

$$\forall P \,.\, (P(0) \wedge \forall x \,.\, (P(x) \supset P(S(x)))) \supset \forall x \,.\, P(x)$$

  where P ranges over unary predicates of natural numbers.
- Advantages:
  1. The induction axiom is a single formula.
  2. The induction axiom is stronger than the induction schema.
- Disadvantages:
  1. The induction axiom is not expressible in first-order logic.
  2. Presburger arithmetic and Peano arithmetic cannot be defined using the induction axiom.
- This approach is cheating!

Approach 2: Implemented as a Rule of Inference

- The induction schema is implemented as a rule of inference.
- Advantages:
  1. Instances of the induction schema can be used in proofs.
  2. Presburger arithmetic and Peano arithmetic can be defined.
- Disadvantages:
  1. The induction schema is expressed in the proof assistant's metalogic, but not in its logic.
  2. Presburger arithmetic and Peano arithmetic cannot be defined independently of the proof assistant's proof system.

Approach 3: Local Reflection

- The induction schema is expressed in the proof assistant's logic using the following infrastructure:
  1. An inductive type of syntactic values that represent the syntactic structures of the formulas in a language L_{nat}.
  2. A quotation operator in the metalogic that maps a formula in L_{nat} to the syntactic value that represents it.
  3. An evaluation operator in the logic that maps a syntactic value e to the value of the formula in L_{nat} that e represents.
- Advantages:
  1. The induction schema is expressed as a single formula.
  2. Presburger arithmetic and Peano arithmetic can be defined.
- Disadvantages:
  1. The evaluation operator may not be definable in the logic.
  2. The infrastructure is local; a new infrastructure is needed for each new kind of schema.
  3. The infrastructure must be expanded for new defined constants.

Is there a better approach for problems like these?

Replete Approach: Global Reflection

- The following infrastructure is added to the logic:
  1. An inductive type of syntactic values that represent all the expressions in the language of the logic.
  2. Global quotation ($\ulcorner \cdot \urcorner$) and evaluation ($[\![\cdot]\!]$) operators.
- This approach is employed in Lisp and other programming languages that support metaprogramming with reflection.
- Advantages:
  1. We can reason directly about the syntax of the entire language of the logic in the logic itself.
  2. The infrastructure thus provides a foundation for metareasoning with reflection.
  3. The infrastructure does not have to be augmented or expanded.
- Disadvantages:
  1. The proof assistant's logic must be modified.
  2. Several problems make the modification of the logic challenging.

Problems Confronting the Replete Approach

- Evaluation Problem. The liar paradox can be expressed in the logic if the evaluation operator is not restricted.
- Variable Problem. Syntactic notions — like whether a variable is free in an expression — can depend on the semantics of the expression as well as on its syntax.
  - For example, if $c = \ulcorner x + 3 \urcorner$, then x is free in $[\![c]\!]$ since

$$[\![c]\!] = [\![\ulcorner x + 3 \urcorner]\!] = x + 3.$$

- Double Substitution Problem. Substitution of an expression e for a variable x occurring in $[\![e']\!]$ may require two substitutions.
  - For example, if the value of x is $\ulcorner x \urcorner$, then

$$[\![x]\!] = [\![\ulcorner x \urcorner]\!] = x = \ulcorner x \urcorner.$$

Can metareasoning with reflection be implemented in a traditional logic using the replete approach?

This is largely an open question!

Our Research Plan

1. Develop a version of Church's type theory called $\mathrm{CTT}_{\mathrm{qe}}$ that is engineered to support the replete approach.
   - $\mathrm{CTT}_{\mathrm{qe}}$ is based on $\mathcal{Q}_0$, Peter Andrews' elegant version of Church's type theory.
2. Develop a proof system for $\mathrm{CTT}_{\mathrm{qe}}$.
3. Implement $\mathrm{CTT}_{\mathrm{qe}}$.
4. Demonstrate the utility of $\mathrm{CTT}_{\mathrm{qe}}$ by using the implementation to formalize a series of examples that involve the interplay of syntax and semantics.

Syntax: Types

A type of $\mathrm{CTT}_{\mathrm{qe}}$ is defined inductively as follows:

1. Type of individuals: ι is a type.
2. Type of truth values: o is a type.
3. Type of constructions: ϵ is a type.
4. Function type: If α and β are types, then $(\alpha \to \beta)$ is a type.

Syntax: Logical Constants

$=_{\alpha\to\alpha\to o}$ for all $\alpha \in \mathcal{T}$

$\mathsf{is\text{-}var}_{\epsilon\to o}$

$\mathsf{is\text{-}con}_{\epsilon\to o}$

$\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}$

$\mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}$

$\mathsf{quo}_{\epsilon\to\epsilon}$

$\mathsf{is\text{-}expr}^{\alpha}_{\epsilon\to o}$ for all $\alpha \in \mathcal{T}$

Syntax: Expressions

An expression of type α of $\mathrm{CTT}_{\mathrm{qe}}$ is defined inductively as follows:

1. Variable: $\mathbf{x}_\alpha$ is an expression of type α.
2. Constant: $\mathbf{c}_\alpha$ is an expression of type α.
3. Function application: $(\mathbf{F}_{\alpha\to\beta}\,\mathbf{A}_\alpha)$ is an expression of type β.
4. Function abstraction: $(\lambda\,\mathbf{x}_\alpha\;.\;\mathbf{B}_\beta)$ is an expression of type $\alpha\to\beta$.
5. Quotation: $\ulcorner\mathbf{A}_\alpha\urcorner$ is an expression of type ϵ if $\mathbf{A}_\alpha$ is eval-free.
6. Evaluation: $[\![\mathbf{A}_\epsilon]\!]_{\mathbf{B}_\beta}$ is an expression of type β.

Syntax: Constructions

A construction of $\mathrm{CTT}_{\mathrm{qe}}$ is an expression of type ϵ defined inductively as follows:

1. $\ulcorner \mathbf{x}_\alpha \urcorner$ is a construction.
2. $\ulcorner \mathbf{c}_\alpha \urcorner$ is a construction.
3. If $\mathbf{A}_\epsilon$ and $\mathbf{B}_\epsilon$ are constructions, then $\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\,\mathbf{A}_\epsilon\,\mathbf{B}_\epsilon$, $\mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}\,\mathbf{A}_\epsilon\,\mathbf{B}_\epsilon$, and $\mathsf{quo}_{\epsilon\to\epsilon}\,\mathbf{A}_\epsilon$ are constructions.

Let $\mathcal{E}$ be the function mapping eval-free expressions to constructions that is defined inductively as follows:

1. $\mathcal{E}(\mathbf{x}_\alpha) = \ulcorner \mathbf{x}_\alpha \urcorner$.
2. $\mathcal{E}(\mathbf{c}_\alpha) = \ulcorner \mathbf{c}_\alpha \urcorner$.
3. $\mathcal{E}(\mathbf{F}_{\alpha\to\beta}\,\mathbf{A}_\alpha) = \mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\,\mathcal{E}(\mathbf{F}_{\alpha\to\beta})\,\mathcal{E}(\mathbf{A}_\alpha)$.
4. $\mathcal{E}(\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_\beta) = \mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}\,\mathcal{E}(\mathbf{x}_\alpha)\,\mathcal{E}(\mathbf{B}_\beta)$.
5. $\mathcal{E}(\ulcorner \mathbf{A}_\alpha \urcorner) = \mathsf{quo}_{\epsilon\to\epsilon}\,\mathcal{E}(\mathbf{A}_\alpha)$.

Syntax: Five Kinds of Eval-Free Expressions

Kind	Syntax	Syntactic Values
Variable	$\mathbf{x}_\alpha$	$\ulcorner \mathbf{x}_\alpha \urcorner$
Constant	$\mathbf{c}_\alpha$	$\ulcorner \mathbf{c}_\alpha \urcorner$
Function application	$\mathbf{F}_{\alpha\to\beta}\,\mathbf{A}_\alpha$	$\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\,\mathcal{E}(\mathbf{F}_{\alpha\to\beta})\,\mathcal{E}(\mathbf{A}_\alpha)$
Function abstraction	$\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_\beta$	$\mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}\,\mathcal{E}(\mathbf{x}_\alpha)\,\mathcal{E}(\mathbf{B}_\beta)$
Quotation	$\ulcorner \mathbf{A}_\alpha \urcorner$	$\mathsf{quo}_{\epsilon\to\epsilon}\,\mathcal{E}(\mathbf{A}_\alpha)$

Syntax: Definitions and Abbreviations

$(\mathbf{A}_\alpha = \mathbf{B}_\alpha)$	stands for	$=_{\alpha\to\alpha\to o}\ \mathbf{A}_\alpha\,\mathbf{B}_\alpha$.
T_o	stands for	$=_{o\to o\to o} = =_{o\to o\to o}$.
F_o	stands for	$(\lambda\, x_o\ .\ T_o) = (\lambda\, x_o\ .\ x_o)$.
$(\forall\, \mathbf{x}_\alpha\ .\ \mathbf{A}_o)$	stands for	$(\lambda\, \mathbf{x}_\alpha\ .\ T_o) = (\lambda\, \mathbf{x}_\alpha\ .\ \mathbf{A}_o)$.
$\wedge_{o\to o\to o}$	stands for	$\lambda\, x_o\ .\ \lambda\, y_o\ .$ $((\lambda\, g_{o\to o\to o}\ .\ g_{o\to o\to o}\ T_o\ T_o) =$ $(\lambda\, g_{o\to o\to o}\ .\ g_{o\to o\to o}\ x_o\ y_o))$.
$(\mathbf{A}_o \wedge \mathbf{B}_o)$	stands for	$\wedge_{o\to o\to o}\,\mathbf{A}_o\,\mathbf{B}_o$.
$\supset_{o\to o\to o}$	stands for	$\lambda\, x_o\ .\ \lambda\, y_o\ .\ (x_o = (x_o \wedge y_o))$.
$(\mathbf{A}_o \supset \mathbf{B}_o)$	stands for	$\supset_{o\to o\to o}\,\mathbf{A}_o\,\mathbf{B}_o$.
$\neg_{o\to o}$	stands for	$=_{o\to o\to o}\,F_o$.
$(\neg\mathbf{A}_o)$	stands for	$\neg_{o\to o}\,\mathbf{A}_o$.
$\vee_{o\to o\to o}$	stands for	$\lambda\, x_o\ .\ \lambda\, y_o\ .\ \neg(\neg x_o \wedge \neg y_o)$.
$(\mathbf{A}_o \vee \mathbf{B}_o)$	stands for	$\vee_{o\to o\to o}\,\mathbf{A}_o\,\mathbf{B}_o$.
$(\exists\, \mathbf{x}_\alpha\ .\ \mathbf{A}_o)$	stands for	$\neg(\forall\, \mathbf{x}_\alpha\ .\ \neg\mathbf{A}_o)$.
$[\![\mathbf{A}_\epsilon]\!]_\beta$	stands for	$[\![\mathbf{A}_\epsilon]\!]_{\mathbf{B}_\beta}$.

Semantics: Frames and Interpretations

A frame of $\mathrm{CTT}_{\mathrm{qe}}$ is a collection $\{D_\alpha \mid \alpha \in \mathcal{T}\}$ of domains such that:

1. D_ι is a nonempty set of values (called individuals).
2. $D_o = \{\mathrm{T}, \mathrm{F}\}$, the set of standard truth values.
3. D_ϵ is the set of constructions of $\mathrm{CTT}_{\mathrm{qe}}$.
4. For $\alpha, \beta \in \mathcal{T}$, $D_{\alpha\to\beta}$ is the set of total functions from D_α to D_β.

An interpretation of $\mathrm{CTT}_{\mathrm{qe}}$ is a pair $(\{D_\alpha \mid \alpha \in \mathcal{T}\}, I)$ consisting of a frame and an interpretation function I that maps each constant in $\mathcal{C}$ of type α to an element of D_α such that $I(\mathbf{c}_\alpha)$ is an appropriate fixed meaning when $\mathbf{c}_\alpha$ is a logical constant.

Semantics: Models

An interpretation $\mathcal{M} = (\{D_\alpha \mid \alpha \in \mathcal{T}), I\}$ is a model for $\mathrm{CTT}_{\mathrm{qe}}$ if there is a binary valuation function $V^{\mathcal{M}}$ such that, for all assignments $\varphi \in \mathrm{assign}(\mathcal{M})$ and expressions $\mathbf{C}_\gamma$, $V^{\mathcal{M}}_\varphi(\mathbf{C}_\gamma) \in D_\gamma$ and each of the following conditions is satisfied:

1. If $\mathbf{C}_\gamma \in \mathcal{V}$, then $V^{\mathcal{M}}_\varphi(\mathbf{C}_\gamma) = \varphi(\mathbf{C}_\gamma)$.
2. If $\mathbf{C}_\gamma \in \mathcal{C}$, then $V^{\mathcal{M}}_\varphi(\mathbf{C}_\gamma) = I(\mathbf{C}_\gamma)$.
3. If $\mathbf{C}_\gamma$ is $\mathbf{F}_{\alpha\to\beta}\,\mathbf{A}_\alpha$, then $V^{\mathcal{M}}_\varphi(\mathbf{C}_\gamma) = V^{\mathcal{M}}_\varphi(\mathbf{F}_{\alpha\to\beta})(V^{\mathcal{M}}_\varphi(\mathbf{A}_\alpha))$.
4. If $\mathbf{C}_\gamma$ is $\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_\beta$, then $V^{\mathcal{M}}_\varphi(\mathbf{C}_\gamma)$ is the function $f \in D_{\alpha\to\beta}$ such that, for each $d \in D_\alpha$, $f(d) = V^{\mathcal{M}}_{\varphi[\mathbf{x}_\alpha \mapsto d]}(\mathbf{B}_\beta)$.
5. If $\mathbf{C}_\gamma$ is $\ulcorner \mathbf{A}_\alpha \urcorner$, then $V^{\mathcal{M}}_\varphi(\mathbf{C}_\gamma) = \mathcal{E}(\mathbf{A}_\alpha)$.
6. If $\mathbf{C}_\gamma$ is $[\![\mathbf{A}_\epsilon]\!]_\beta$ and $V^{\mathcal{M}}_\varphi(\text{is-expr}^\beta_{\epsilon\to o}\,\mathbf{A}_\epsilon) = \mathrm{T}$, then

$$V^{\mathcal{M}}_\varphi(\mathbf{C}_\gamma) = V^{\mathcal{M}}_\varphi(\mathcal{E}^{-1}(V^{\mathcal{M}}_\varphi(\mathbf{A}_\epsilon))).$$

Basic Theorems

- Theorem (Law of Quotation). $\ulcorner \mathbf{A}_\alpha \urcorner = \mathcal{E}(\mathbf{A}_\alpha)$ is valid in every model of $\mathrm{CTT}_{\mathrm{qe}}$.

- Theorem (Law of Disquotation). $[\![\ulcorner \mathbf{A}_\alpha \urcorner]\!]_\alpha = \mathbf{A}_\alpha$ is valid in every model of $\mathrm{CTT}_{\mathrm{qe}}$.
  - Thus the Evaluation Problem is not an issue.

Example 1: Induction Schema

- The induction schema for Peano arithmetic can be expressed in $\mathrm{CTT}_{\mathrm{qe}}$ as:

$$\forall f_\epsilon \,.\, \text{is-expr}^{\iota\to o}_{\epsilon\to o}\, f_\epsilon \supset \\ ((\llbracket f_\epsilon \rrbracket_{\iota\to o}\, 0 \wedge (\forall x_\iota \,.\, \llbracket f_\epsilon \rrbracket_{\iota\to o}\, x_\iota \supset \llbracket f_\epsilon \rrbracket_{\iota\to o}\, (\mathsf{S}_{\iota\to\iota}\, x_\iota))) \supset \\ \forall x_\iota \,.\, \llbracket f_\epsilon \rrbracket_{\iota\to o}\, x_\iota).$$

- The induction schema for Presburger arithmetic can be expressed in $\mathrm{CTT}_{\mathrm{qe}}$ as:

$$\forall f_\epsilon \,.\, \text{is-expr}^{\iota\to o}_{\epsilon\to(\epsilon\to o)\to o}\, f_\epsilon\, \text{presburger}_{\epsilon\to o} \supset \\ ((\llbracket f_\epsilon \rrbracket_{\iota\to o}\, 0 \wedge (\forall x_\iota \,.\, \llbracket f_\epsilon \rrbracket_{\iota\to o}\, x_\iota \supset \llbracket f_\epsilon \rrbracket_{\iota\to o}\, (\mathsf{S}_{\iota\to\iota}\, x_\iota))) \supset \\ \forall x_\iota \,.\, \llbracket f_\epsilon \rrbracket_{\iota\to o}\, x_\iota).$$

Comments about $\mathrm{CTT}_{\mathrm{qe}}$

- Quasiquotation comes for free.
  - The quasiquotation $\ulcorner \lfloor \mathbf{B}_\epsilon \rfloor \wedge_{ooo} \lfloor \mathbf{C}_\epsilon \rfloor \urcorner$ is expressed by

$$\mathsf{app}_{\epsilon\epsilon\epsilon}\,[\mathsf{app}_{\epsilon\epsilon\epsilon}\ulcorner \wedge_{ooo} \urcorner\, \mathbf{B}_\epsilon]\, \mathbf{C}_\epsilon.$$

- $\mathrm{CTT}_{\mathrm{qe}}$ is simpler but less expressive than $\mathcal{Q}_0^{\mathrm{uqe}}$, a version of Church's type theory that supports the replete approach with syntax values for all expressions of the language.
- If the syntax values of $\mathrm{CTT}_{\mathrm{qe}}$ are restricted to representing only closed eval-free expressions, the three problems given above do not come into play, but the utility of the logic is greatly reduced.

Another MKM Challenge: Meaning Formulas

- A syntax-based mathematical algorithm A is a symbolic algorithm that manipulates mathematical expressions in a mathematically meaningful way.
  - Example: A symbolic differentiation algorithm.
- The computational behavior of A is the relationship between the input and output expressions of A.
- The mathematical meaning of A is the relationship between what the input and output expressions of A mean mathematically.
- A meaning formula for A is a statement that expresses the mathematical meaning of A.
  - Involves the interplay of syntax and semantics.
  - Difficult to express in a traditional logic.
- How can we express, prove, and apply a meaning formula in a proof assistant's logic?

Example 2: Polynomial Differentiation

- Let pdiff be the symbolic differentiation algorithm defined by the usual differentiation rules for polynomials.
  - Example: $\mathsf{pdiff}(u \cdot v, x) = \mathsf{pdiff}(u, x) \cdot v + u \cdot \mathsf{pdiff}(v, x)$.
- Informally, the meaning formula for pdiff is:

$$\forall\, u : \mathsf{Poly} \;.\; \mathsf{deriv}(\lambda\, x : \mathbb{R} \;.\; u) = \lambda\, x : \mathbb{R} \;.\; \mathsf{pdiff}(u, x).$$

- Notice that undefinedness is not an issue since both polynomial functions and their derivatives are total.
- Notice also the lack of precision in the meaning formula:
  - $\forall\, u : \mathsf{Poly} \;.\; \mathsf{deriv}(\lambda\, x : \mathbb{R} \;.\; \boxed{\mathsf{u}}) = \lambda\, x : \mathbb{R} \;.\; \boxed{\mathsf{pdiff}(\mathsf{u}, \mathsf{x})}$.
- This imprecision can be removed using quotation and evaluation:

$$\forall\, u : \mathsf{Poly} \;.\; \mathsf{deriv}(\lambda\, x : \mathbb{R} \;.\; [\![u]\!]) = \lambda\, x : \mathbb{R} \;.\; [\![\mathsf{pdiff}(u, \ulcorner x \urcorner)]\!].$$

Proof System: Requirements

To be useful, a proof system for $\mathrm{CTT}_{\mathrm{qe}}$ needs to satisfy the following requirements:

R1. The proof system is sound, i.e., it only proves valid formulas.

R2. The proof system is complete with respect to the Henkin general models semantics for $\mathrm{CTT}_{\mathrm{qe}}$ for eval-free formulas.

R3. The proof system can be used to reason about quotations and other expressions of type ϵ that denote constructions.

R4. The proof system can instantiate free variables that occur in the first argument of an evaluation as found in formulas that represent axiom schemas and meaning formulas.

R5. The proof system can prove formulas, such as those that represent meaning formulas, in which free variables occur in the first argument of an evaluation.

Related Work

- Programming languages that support metaprogramming.
  - Agda, Archon, Elixir, F#, Lisp, MetaML, MetaOCaml, reFLect, Template Haskell.
- Applications of local reflection in formal logics.
  - Coq, Agda, ...
- Work on global reflection.
  - "Implementing Reflection in Nuprl" [Barzilay 2006]
  - "Towards Practical Reflection for Formal Mathematics" [Giese, Buchberger 2007].
  - "On the Semantics of ReFLect as a Basis for a Reflective Theorem Prover" [Melham, Cohn, Childs 2013].

Conclusion

- Quotation and evaluation provide a basis for reasoning about the interplay of syntax and semantics in a traditional logic.
- We have presented the syntax and semantics of $\mathrm{CTT}_{\mathrm{qe}}$, a version of Andrews' $\mathcal{Q}_0$ with quotation and evaluation, and shown how schemas and meaning formulas can be expressed in it.
- We are working on developing a proof system for $\mathrm{CTT}_{\mathrm{qe}}$ and implementing $\mathrm{CTT}_{\mathrm{qe}}$, possibly by extending HOL Light.
- $\mathrm{CTT}_{\mathrm{qe}}$ is simpler and easier to implement than $\mathcal{Q}_0^{\mathrm{uqe}}$ but much less expressive.
- We believe quotation and evaluation can be incorporated in other traditional logics in a similar way.