Incorporating Quotation and Evaluation Into Church's Type Theory*

William M. Farmer†

11 April 2018

Abstract

CTT_{qe} is a version of Church's type theory that includes quotation and evaluation operators that are similar to quote and eval in the Lisp programming language. With quotation and evaluation it is possible to reason in CTT_{qe} about the interplay of the syntax and semantics of expressions and, as a result, to formalize syntax-based mathematical algorithms. We present the syntax and semantics of CTT_{qe} as well as a proof system for CTT_{qe}. The proof system is shown to be sound for all formulas and complete for formulas that do not contain evaluations. We give several examples that illustrate the usefulness of having quotation and evaluation in CTT_{qe}.

Keywords: Church's type theory, simple type theory, metareasoning, reflection, quotation, evaluation, quasiquotation, reasoning about syntax, schemas, symbolic computation, meaning formulas, substitution.

*This paper is a greatly extended version of [56]. This research was supported by NSERC.

†Address: Department of Computing and Software, McMaster University, 1280 Main Street West, Hamilton, Ontario L8S 4K1, Canada. E-mail: wmfarmer@mcmaster.ca.

Contents

List of Tables

1 Introduction

The Lisp programming language is famous for its use of *quotation* and *evaluation*.[1] From code the Lisp quotation operator called *quote* produces meta-level data (i.e., S-expressions) that represents the code, and from this data the Lisp evaluation operator called *eval* produces the code that the data represents. In Lisp, *metaprogramming* (i.e., programming at the meta-level) is performed by manipulating S-expressions and is *reflected* (i.e., integrated) into object-level programming by the use of quote and eval.

Metaprogramming with reflection is a very powerful programming tool. Besides Lisp, several other programming languages employ quotation and evaluation mechanisms to enable metaprogramming with reflection. Examples include Agda [97, 98, 120], Archon [113], Elixir [103], F# [119], Idris [32, 33, 34], MetaML [115], MetaOCaml [109], reFLect [72], Scala [99, 110], and Template Haskell [111]. Indeed nearly all major programming languages today provide some level of support for metaprogramming. Quotation is not crucial for metaprogramming since strings or abstract syntax trees (ASTs) can be used directly as data representing code. However, quotation is convenient for connecting code to its representation.

Analogous to metaprogramming in a programming language, *metareasoning* is performed in a logic by manipulating meta-level values (e.g., ASTs) that represent expressions in the logic and is *reflected* into object-level reasoning using quotation and evaluation[2] mechanisms [42]. Metareasoning with reflection has been used in several proof assistants — including Agda [120, 121], Coq [14, 17, 31, 69, 71, 81, 100], Idris [32, 33, 34], Isabelle/HOL [30], Lean [45], Maude [36], Nqthm/ACL1 [15, 80], Nuprl/MetaPRL [2, 7, 37, 79, 85, 96, 125], PVS [122], reFLect [89], and Theorema [63] — for formalizing metalogical techniques (logical reflection) and incorporating symbolic computation into proofs (computational reflection) [54, 75].

In proof assistants such as Coq and Agda, metareasoning with reflection is implemented in the logic by defining an infrastructure consisting of (1) an *inductive type of syntactic values* that represent certain object-level expressions, (2) an *informal quotation operator* (residing only at the meta-level) that maps these object-level expressions to syntactic values, and (3) a *formal evaluation operator* (residing at the object-level) that maps syntactic values to the values of the object-level expressions that they represent. (The three components of this approach form an instance of a *syntax framework* [60], a mathematical structure that models systems for reasoning about the syntax of an interpreted language.)

The reflection infrastructures that have been employed in today's proof assistants are usually *local* in the sense that the syntactic values of the inductive type represent only the expressions of the logic that are relevant to a particular problem, the quotation operator can only be applied to these expressions, and

[1] Lisp is also famous for its use of *quasiquotation*, a more powerful form of quotation which we discuss in section 4.

[2] Evaluation in this context is also called unquoting, interpretation, dereferencing, and dereification.

the evaluation operator can only be applied to the syntactic values of this inductive type. Can metareasoning with reflection be implemented in a traditional logic like first-order logic or simple type theory [52] using a *global* infrastructure for the entire set of expressions of the logic with global quotation and evaluation operators like Lisp's quote and eval? This is largely an open question.

We have proposed a version of NBG set theory named Chiron [53] and a version of Alonzo Church's type theory [35][3] named $\mathcal{Q}_0^{\mathrm{uqe}}$ [55] that include global quotation and evaluation operators, but these logics have a high level of complexity and are not easy to implement. This paper presents a logic named $\mathrm{CTT}_{\mathrm{qe}}$, a version of Church's type theory with quotation and evaluation which is much simpler than $\mathcal{Q}_0^{\mathrm{uqe}}$. We believe $\mathrm{CTT}_{\mathrm{qe}}$ is the first readily implementable version of simple type theory that includes global quotation and evaluation. See [63] for research in a similar direction.

Several challenging design problems face the logic engineer who seeks to incorporate global quotation and evaluation into a traditional logic. The three design problems that most concern us are the following. We will write the quotation and evaluation operators applied to an expression e as $\ulcorner e \urcorner$ and $[\![e]\!]$, respectively.

1. *Evaluation Problem.* An evaluation operator is applicable to syntactic values that represent formulas and thus is effectively a truth predicate. Hence, by the proof of Alfred Tarski's theorem on the undefinability of truth [116, 117, 118], if the evaluation operator is total in the context of a sufficiently strong theory like first-order Peano arithmetic, then it is possible to express the liar paradox using the quotation and evaluation operators. Therefore, the evaluation operator must be partial and the law of disquotation[4] cannot hold universally (i.e., for some expressions e, $[\![\ulcorner e \urcorner]\!] \neq e$). As a result, reasoning with evaluation is cumbersome and leads to undefined expressions.

2. *Variable Problem.* The variable x is not free in the expression $\ulcorner x + 3 \urcorner$ (or in any quotation). However, x is free in $[\![\ulcorner x + 3 \urcorner]\!]$ because $[\![\ulcorner x + 3 \urcorner]\!] = x + 3$. If the value of a constant c is $\ulcorner x + 3 \urcorner$, then x is free in $[\![c]\!]$ because $[\![c]\!] = [\![\ulcorner x + 3 \urcorner]\!] = x + 3$. Hence, in the presence of an evaluation operator, whether or not a variable is free in an expression may depend on the values of the expression's components. As a consequence, the substitution of an expression for the free occurrences of a variable in another expression depends on the semantics (as well as the syntax) of the expressions involved and must be integrated with the proof system for the logic. That is, a logic with quotation and evaluation requires a semantics-dependent form of substitution in which side conditions, like whether a

[3] Church's type theory [35] is a version of simple type theory with lambda notation. It is a classical form of type theory in contrast to constructive type theories, like Martin-Löf type theory [88] and the calculus of constructions [41].

[4] There are different meanings for the law of disquotation depending, for example, on how free variables in a quotation are handled. We are assuming a law of disquotation in which $[\![\ulcorner e \urcorner]\!]$ has the same value as e for any variable assignment.

variable is free in an expression, are proved within the proof system. This is a major departure from traditional logic.

3. *Double Substitution Problem.* By the semantics of evaluation, the value of $[\![e]\!]$ is the *value* of the expression whose syntax tree is represented by the *value* of e. Hence the semantics of evaluation involves a double valuation (see condition 6 of the definition of a general model in subsection 3.3). If the value of a variable x is $\ulcorner x \urcorner$, then $[\![x]\!] = [\![\ulcorner x \urcorner]\!] = x = \ulcorner x \urcorner$ (see Proposition 8.2.1). Hence the substitution of $\ulcorner x \urcorner$ for x in $[\![x]\!]$ requires one substitution inside the argument of the evaluation operator and another substitution after the evaluation operator is eliminated. This double substitution is another major departure from traditional logic.

To solve the Evaluation Problem, it is necessary to restrict the application of either the quotation operator or the evaluation operator. In $\mathcal{Q}_0^{\mathrm{uqe}}$, $\ulcorner e \urcorner$ is defined for every expression e, but $[\![\ulcorner e \urcorner]\!]$ is defined only if e is an expression in which every occurrence of the evaluation operator is within a quotation. The Variable and Double Substitution Problems are then solved by defining an explicit substitution operator that operates on syntactic values. This results in a proof system for $\mathcal{Q}_0^{\mathrm{uqe}}$ that is very complex and very difficult to implement. On the other hand, if $\ulcorner e \urcorner$ is defined only when e is a closed evaluation-free expression, then all three of the design problem disappear. However, most of the usefulness of having quotation and evaluation in a logic also disappear — which is illustrated by the examples in section 4.

The logic $\mathrm{CTT}_{\mathrm{qe}}$ takes a middle path to solve the three design problems: $\ulcorner e \urcorner$ is defined only if e is a (possible open) expression that does not contain the evaluation operator. It is much simpler than $\mathcal{Q}_0^{\mathrm{uqe}}$, but also much more useful than a logic in which only closed expressions can be quoted. Like $\mathcal{Q}_0^{\mathrm{uqe}}$, $\mathrm{CTT}_{\mathrm{qe}}$ is based on $\mathcal{Q}_0$ [3], Peter Andrews' version of Church's type theory. In this paper, we present the syntax and semantics of $\mathrm{CTT}_{\mathrm{qe}}$ as well as a proof system for $\mathrm{CTT}_{\mathrm{qe}}$. We also give several examples that demonstrate the benefits of having quotation and evaluation in $\mathrm{CTT}_{\mathrm{qe}}$.

The rest of the paper is organized as follows. The syntax of $\mathrm{CTT}_{\mathrm{qe}}$ is defined in section 2. A Henkin-style general models semantics for $\mathrm{CTT}_{\mathrm{qe}}$ is defined in section 3. Four examples that illustrate the utility of the quotation and evaluation facility in $\mathrm{CTT}_{\mathrm{qe}}$ are presented in section 4. A proof system for $\mathrm{CTT}_{\mathrm{qe}}$ is given in section 5. Various proof-theoretic results about the proof system are proved in section 6. The proof system is proved in sections 7 and 8 to be, respectively, sound with respect to the semantics of $\mathrm{CTT}_{\mathrm{qe}}$ and complete with respect to the semantics of $\mathrm{CTT}_{\mathrm{qe}}$ for evaluation-free formulas. In section 9 the results stated about the examples discussed in section 4 are proved within the proof system for $\mathrm{CTT}_{\mathrm{qe}}$. The extensive body of literature related to $\mathrm{CTT}_{\mathrm{qe}}$ is briefly surveyed in section 10. And the paper ends with some final remarks in section 11 including a brief discussion on future work.

2 Syntax

CTT_{qe} has the syntax of Church's type theory plus an inductive type of syntactic values, a partial quotation operator, and a typed evaluation operator. The syntax of CTT_{qe} is very similar to the syntax of $\mathcal{Q}_0$ [3, pp. 210–211]. Like $\mathcal{Q}_0$, the propositional connectives and quantifiers are defined using function application, function abstraction, and equality. For the sake of simplicity, CTT_{qe} does not contain, as in $\mathcal{Q}_0$, a definite description operator or, as in the logic of the HOL proof assistant [70], an indefinite description (choice) operator and type variables.

2.1 Types

A *type* of CTT_{qe} is a string of symbols defined inductively by the following formation rules:

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.[5]

Let $\mathcal{T}$ denote the set of types of CTT_{qe}. $\alpha, \beta, \gamma, \ldots$ are syntactic variables ranging over types. When there is no loss of meaning, matching pairs of parentheses in types may be omitted. We assume that function type formation associates to the right so that a type of the form $(\alpha \to (\beta \to \gamma))$ may be written as $\alpha \to \beta \to \gamma$.

We will see in the next subsection that in CTT_{qe} types are directly assigned to variables and constants and thereby indirectly assigned to expressions.

2.2 Expressions

A *typed symbol* is a symbol with a subscript from $\mathcal{T}$. Let $\mathcal{V}$ be a set of typed symbols such that, for each $\alpha \in \mathcal{T}$, $\mathcal{V}$ contains denumerably many typed symbols with subscript α. A *variable of type* α of CTT_{qe} is a member of $\mathcal{V}$ with subscript α. $\mathbf{f}_\alpha, \mathbf{g}_\alpha, \mathbf{h}_\alpha, \mathbf{u}_\alpha, \mathbf{v}_\alpha, \mathbf{w}_\alpha, \mathbf{x}_\alpha, \mathbf{y}_\alpha, \mathbf{z}_\alpha, \ldots$ are syntactic variables ranging over variables of type α. We will assume that $f_\alpha, g_\alpha, h_\alpha, u_\alpha, v_\alpha, w_\alpha, x_\alpha, y_\alpha, z_\alpha, \ldots$ are actual variables of type α of CTT_{qe}.

Let $\mathcal{C}$ be a set of typed symbols disjoint from $\mathcal{V}$ that includes the typed symbols in Table 1. A *constant of type* α of CTT_{qe} is a member of $\mathcal{C}$ with subscript α. The typed symbols in Table 1 are the *logical constants* of CTT_{qe}. $\mathbf{c}_\alpha, \mathbf{d}_\alpha, \ldots$ are syntactic variables ranging over constants of type α.

An *expression of type* α of CTT_{qe} is a string of symbols defined inductively by the formation rules below. $\mathbf{A}_\alpha, \mathbf{B}_\alpha, \mathbf{C}_\alpha, \ldots$ are syntactic variables ranging

[5] In Andrews' $\mathcal{Q}_0$ [3] and Church's original system [35], the function type $(\alpha \to \beta)$ is written as $(\beta\alpha)$.

$=_{\alpha\to\alpha\to o}$	for all $\alpha \in \mathcal{T}$
$\mathsf{is\text{-}var}_{\epsilon\to o}$	
$\mathsf{is\text{-}var}^{\alpha}_{\epsilon\to o}$	for all $\alpha \in \mathcal{T}$
$\mathsf{is\text{-}con}_{\epsilon\to o}$	
$\mathsf{is\text{-}con}^{\alpha}_{\epsilon\to o}$	for all $\alpha \in \mathcal{T}$
$\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}$	
$\mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}$	
$\mathsf{quo}_{\epsilon\to\epsilon}$	
$\mathsf{is\text{-}expr}_{\epsilon\to o}$	
$\mathsf{is\text{-}expr}^{\alpha}_{\epsilon\to o}$	for all $\alpha \in \mathcal{T}$
$\sqsubset_{\epsilon\to\epsilon\to o}$	
$\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}$	

Table 1: Logical Constants

over expressions of type α. An expression is *eval-free* if it is constructed using just the first five formation rules.

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.

The sole purpose of the second component $\mathbf{B}_\beta$ in an evaluation $[\![\mathbf{A}_\epsilon]\!]_{\mathbf{B}_\beta}$ is to establish the type of the evaluation. A *formula* is an expression of type o. When there is no loss of meaning, matching pairs of parentheses in expressions may be omitted. We assume that function application formation associates to the left so that an expression of the form $((\mathbf{G}_{\alpha\to\beta\to\gamma}\,\mathbf{A}_\alpha)\,\mathbf{B}_\beta)$ may be written as $\mathbf{G}_{\alpha\to\beta\to\gamma}\,\mathbf{A}_\alpha\,\mathbf{B}_\beta$.

Let $\mathbf{A}_\alpha$ and $\mathbf{B}_\beta$ be eval-free expressions. An occurrence of a variable $\mathbf{x}_\alpha$ in $\mathbf{B}_\beta$ is *bound* [*free*] if (1) it is not in a quotation and (2) it is [not] in a subexpression of $\mathbf{B}_\beta$ of the form $\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{C}_\gamma$. A variable $\mathbf{x}_\alpha$ is *bound* [*free*] *in* $\mathbf{B}_\beta$ if there is a bound [free] occurrence of $\mathbf{x}_\alpha$ in $\mathbf{B}_\beta$. $\mathbf{A}_\alpha$ is *free for* $\boldsymbol{x}_\alpha$ *in* $\mathbf{B}_\beta$ if no free occurrence of $\mathbf{x}_\alpha$ in $\mathbf{B}_\beta$ is within a subexpression of $\mathbf{B}_\beta$ of the form $\lambda\,\mathbf{y}_\gamma\,.\,\mathbf{C}_\delta$ such that $\mathbf{y}_\gamma$ is free in $\mathbf{A}_\alpha$.

Remark 2.2.1 (Bound and Free Variables) For expressions that are not eval-free (i.e., contain one or more evaluations), it is not possible to define in a purely syntactic way the notion of a bound or free variable due to the Variable Problem (see section 1). Hence notions concerning bound and free variables, such as substitution and alpha-equivalence, cannot be readily extended to non-eval-free expressions.

2.3 Constructions

A *construction* of CTT_{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.

The set of constructions is thus an inductive type whose base elements are quotations of variables and constants and whose constructors are $\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}$, $\mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}$, and $\mathsf{quo}_{\epsilon\to\epsilon}$. We will call these three constants *syntax constructors*.

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. $\mathcal{E}(\ulcorner \mathbf{A}_\alpha \urcorner) = \mathsf{quo}_{\epsilon\to\epsilon}\,\mathcal{E}(\mathbf{A}_\alpha)$.

$\mathcal{E}$ is clearly injective. When $\mathbf{A}_\alpha$ is eval-free, $\mathcal{E}(\mathbf{A}_\alpha)$ is the unique construction that represents the syntax tree of $\mathbf{A}_\alpha$. That is, $\mathcal{E}(\mathbf{A}_\alpha)$ is a syntactic value that represents how $\mathbf{A}_\alpha$ is syntactically constructed. For every eval-free expression, there is a construction that represents its syntax tree, but not every construction represents the syntax tree of an eval-free expression. For example, $\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\ulcorner \mathbf{x}_\alpha \urcorner\ulcorner \mathbf{x}_\alpha \urcorner$ represents the syntax tree of $(\mathbf{x}_\alpha\,\mathbf{x}_\alpha)$ which is not an expression of CTT_{qe} since the types are mismatched. A construction is *proper* if it is in the range of $\mathcal{E}$, i.e., it represents the syntax tree of an eval-free expression. Whether a construction is proper is easily decided by examining it syntactic structure.

The five kinds of eval-free expressions and the syntactic values that represent their syntax trees are given in Table 2. The logical constants $\mathsf{is\text{-}var}_{\epsilon\to o}$, $\mathsf{is\text{-}var}^{\alpha}_{\epsilon\to o}$, $\mathsf{is\text{-}con}_{\epsilon\to o}$, $\mathsf{is\text{-}con}^{\alpha}_{\epsilon\to o}$, $\mathsf{is\text{-}expr}_{\epsilon\to o}$, $\mathsf{is\text{-}expr}^{\alpha}_{\epsilon\to o}$, $\sqsubset_{\epsilon\to\epsilon\to o}$, and $\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}$ are used to make assertions about the expressions that constructors represent. Their meanings are given in subsection 3.2. $\sqsubset_{\epsilon\to\epsilon\to o}$ is needed to express the induction principle for constructions (Axiom B6 in section 5).

Remark 2.3.1 (Type of Constructions) All constructions have the same type ϵ. Thus, $\mathcal{E}(A_\alpha)$ and $\mathcal{E}(B_\beta)$ both have type ϵ even when A_α and B_β are eval-free expressions with different types. An alternate approach would be to parameterize ϵ so that all constructors of the form $\mathcal{E}(A_\alpha)$ would have the type ϵ_α. Instead of parameterizing ϵ by the type of expressions, we have chosen to partition ϵ by the set $\{\mathsf{is\text{-}expr}^{\alpha}_{\epsilon\to o} \mid \alpha \in \mathcal{T}\}$ of unary predicates on ϵ.

2.4 Theories

Let $\mathcal{D} \subseteq \mathcal{C}$. An expression $\mathbf{A}_\alpha$ of CTT_{qe} is a *$\mathcal{D}$-expression* if each constant occurring in $\mathbf{A}_\alpha$ is a member of $\mathcal{D}$. Let $L_\mathcal{D}$ be the set of all $\mathcal{D}$-expressions. A *language* of CTT_{qe} is $L_\mathcal{D}$ for some $\mathcal{D} \subseteq \mathcal{C}$ such that $\mathcal{D}$ contains all the logical constants of CTT_{qe}. A *theory* of CTT_{qe} is a pair $T = (L, \Gamma)$ where L is a language of CTT_{qe} and Γ is a set of formulas in L. The *theory of the logic* is the theory $T_{\text{logic}} = (L_\mathcal{C}, \emptyset)$. A theory $T = (L, \Gamma)$ is *eval-free* if each member of Γ is eval-free. $\mathbf{A}_\alpha$ is an *expression of a theory* $T = (L, \Gamma)$ if $\mathbf{A}_\alpha \in L$.

2.5 Definitions and Abbreviations

As Andrews does in [3, p. 212], we introduce in Table 3 several defined logical constants and abbreviations. The former includes constants for true and false and the propositional connectives. The latter includes notation for equality, the propositional connectives, universal and existential quantification, $\sqsubset_{\epsilon\to\epsilon\to o}$ as an infix operator, and a simplified notation for evaluations.

3 Semantics

The semantics of CTT_{qe} extends the semantics of $\mathcal{Q}_0$ [3, pp. 238–239] by defining the domain of the type ϵ and what quotations and evaluations mean.

3.1 Frames

A *frame* of CTT_{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. $D_o = \{\text{T}, \text{F}\}$, the set of standard *truth values*.
3. D_ϵ is the set of *constructions* of CTT_{qe}.
4. For $\alpha, \beta \in \mathcal{T}$, $D_{\alpha\to\beta}$ is some set of *total functions* from D_α to D_β.

$\mathcal{D}_\iota$ is the *domain of individuals*, $\mathcal{D}_o$ is the *domain of truth values*, $\mathcal{D}_\epsilon$ is the *domain of constructions*, and, for $\alpha, \beta \in \mathcal{T}$, $\mathcal{D}_{\alpha\to\beta}$ is a *function domain*.

Kind	**Syntax**	**Syntactic Value**
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	$\ulcorner \mathbf{A}_\alpha \urcorner$	$\mathsf{quo}_{\epsilon\to\epsilon}\, \mathcal{E}(\mathbf{A}_\alpha)$

Table 2: Five Kinds of Eval-Free Expressions

$(\mathbf{A}_\alpha = \mathbf{B}_\alpha)$	stands for	$=_{\alpha\to\alpha\to o}\ \mathbf{A}_\alpha\,\mathbf{B}_\alpha$.
$(\mathbf{A}_o \equiv \mathbf{B}_o)$	stands for	$=_{o\to o\to o}\ \mathbf{A}_o\,\mathbf{B}_o$.
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	$\neg(\forall\, \mathbf{x}_\alpha\ .\ \neg \mathbf{A}_o)$.
$(\mathbf{A}_\alpha \neq \mathbf{B}_\alpha)$	stands for	$\neg(\mathbf{A}_\alpha = \mathbf{B}_\alpha)$.
$\mathbf{A}_\epsilon \sqsubset_{\epsilon\to\epsilon\to o} \mathbf{B}_\epsilon$	stands for	$\sqsubset_{\epsilon\to\epsilon\to o}\ \mathbf{A}_\epsilon\, \mathbf{B}_\epsilon$.
$[\![\mathbf{A}_\epsilon]\!]_\beta$	stands for	$[\![\mathbf{A}_\epsilon]\!]_{\mathbf{B}_\beta}$.

Table 3: Definitions and Abbreviations

3.2 Interpretations

An *interpretation* of CTT_{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:

1. For all $\alpha \in \mathcal{T}$, $I(=_{\alpha\to\alpha\to o})$ is the function $f \in D_{\alpha\to\alpha\to o}$ such that, for all $d_1, d_2 \in D_\alpha$, $f(d_1)(d_2) = \text{T}$ iff $d_1 = d_2$. That is, $I(=_{\alpha\to\alpha\to o})$ is the identity relation on D_α.

2. $I(\mathsf{is\text{-}var}_{\epsilon\to o})$ is the function $f \in D_{\epsilon\to o}$ such that, for all constructions $\mathbf{A}_\epsilon \in D_\epsilon$, $f(\mathbf{A}_\epsilon) = \text{T}$ iff $\mathbf{A}_\epsilon = \ulcorner \mathbf{x}_\alpha \urcorner$ for some variable $\mathbf{x}_\alpha \in \mathcal{V}$ (where α can be any type).

3. For all $\alpha \in \mathcal{T}$, $I(\mathsf{is\text{-}var}^\alpha_{\epsilon\to o})$ is the function $f \in D_{\epsilon\to o}$ such that, for all constructions $\mathbf{A}_\epsilon \in D_\epsilon$, $f(\mathbf{A}_\epsilon) = \text{T}$ iff $\mathbf{A}_\epsilon = \ulcorner \mathbf{x}_\alpha \urcorner$ for some variable $\mathbf{x}_\alpha \in \mathcal{V}$.

4. $I(\mathsf{is\text{-}con}_{\epsilon\to o})$ is the function $f \in D_{\epsilon\to o}$ such that, for all constructions $\mathbf{A}_\epsilon \in D_\epsilon$, $f(\mathbf{A}_\epsilon) = \text{T}$ iff $\mathbf{A}_\epsilon = \ulcorner \mathbf{c}_\alpha \urcorner$ for some constant $\mathbf{c}_\alpha \in \mathcal{C}$ (where α can be any type).

5. For all $\alpha \in \mathcal{T}$, $I(\mathsf{is\text{-}con}^\alpha_{\epsilon\to o})$ is the function $f \in D_{\epsilon\to o}$ such that, for all constructions $\mathbf{A}_\epsilon \in D_\epsilon$, $f(\mathbf{A}_\epsilon) = \text{T}$ iff $\mathbf{A}_\epsilon = \ulcorner \mathbf{c}_\alpha \urcorner$ for some constant $\mathbf{c}_\alpha \in \mathcal{C}$.

6. $I(\mathsf{app}_{\epsilon\to\epsilon\to\epsilon})$ is the function $f \in D_{\epsilon\to\epsilon\to\epsilon}$ such that, for all constructions $\mathbf{A}_\epsilon, \mathbf{B}_\epsilon \in D_\epsilon$, $f(\mathbf{A}_\epsilon)(\mathbf{B}_\epsilon)$ is the construction $\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\, \mathbf{A}_\epsilon\, \mathbf{B}_\epsilon$.

7. $I(\mathsf{abs}_{\epsilon\to\epsilon\to\epsilon})$ is the function $f \in D_{\epsilon\to\epsilon\to\epsilon}$ such that, for all constructions $\mathbf{A}_\epsilon, \mathbf{B}_\epsilon \in D_\epsilon$, $f(\mathbf{A}_\epsilon)(\mathbf{B}_\epsilon)$ is the construction $\mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}\,\mathbf{A}_\epsilon\,\mathbf{B}_\epsilon$.

8. $I(\mathsf{quo}_{\epsilon\to\epsilon})$ is the function $f \in D_{\epsilon\to\epsilon}$ such that, for all constructions $\mathbf{A}_\epsilon \in D_\epsilon$, $f(\mathbf{A}_\epsilon)$ is the construction $\mathsf{quo}_{\epsilon\to\epsilon}\,\mathbf{A}_\epsilon$.

9. $I(\mathsf{is\text{-}expr}_{\epsilon\to o})$ is the function $f \in D_{\epsilon\to o}$ such that, for all constructions $\mathbf{A}_\epsilon \in D_\epsilon$, $f(\mathbf{A}_\epsilon) = \mathrm{T}$ iff $\mathbf{A}_\epsilon = \mathcal{E}(\mathbf{B}_\alpha)$ for some (eval-free) expression $\mathbf{B}_\alpha$ (where α can be any type).

10. For all $\alpha \in \mathcal{T}$, $I(\mathsf{is\text{-}expr}^\alpha_{\epsilon\to o})$ is the function $f \in D_{\epsilon\to o}$ such that, for all constructions $\mathbf{A}_\epsilon \in D_\epsilon$, $f(\mathbf{A}_\epsilon) = \mathrm{T}$ iff $\mathbf{A}_\epsilon = \mathcal{E}(\mathbf{B}_\alpha)$ for some (eval-free) expression $\mathbf{B}_\alpha$.

11. $I(\sqsubset_{\epsilon\to\epsilon\to o})$ is the function $f \in D_{\epsilon\to\epsilon\to o}$ such that, for all constructions $\mathbf{A}_\epsilon, \mathbf{B}_\epsilon \in D_\epsilon$, $f(\mathbf{A}_\epsilon)(\mathbf{B}_\epsilon) = \mathrm{T}$ iff $\mathbf{A}_\epsilon$ is a proper subexpression of $\mathbf{B}_\epsilon$.

12. $I(\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o})$ is the function $f \in D_{\epsilon\to\epsilon\to o}$ such that, for all constructions $\mathbf{A}_\epsilon, \mathbf{B}_\epsilon \in D_\epsilon$, $f(\mathbf{A}_\epsilon)(\mathbf{B}_\epsilon) = \mathrm{T}$ iff $\mathbf{A}_\epsilon = \ulcorner \mathbf{x}_\alpha \urcorner$ for some $\mathbf{x}_\alpha \in \mathcal{V}$, $\mathbf{B}_\epsilon = \mathcal{E}(\mathbf{C}_\beta)$ for some (eval-free) expression $\mathbf{C}_\beta$, and $\mathbf{x}_\alpha$ is free in $\mathbf{C}_\beta$.

Remark 3.2.1 (Domain of Constructions) We would prefer that D_ϵ contains just the set of proper constructions because we need only proper constructions to represent the syntax trees of eval-free expressions. However, then the natural interpretations of the three syntax constructors — $\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}$, $\mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}$, and $\mathsf{quo}_{\epsilon\to\epsilon}$ — would be partial functions. Since $\mathrm{CTT_{qe}}$ admits only total functions, it is more convenient to allow D_ϵ to include improper constructions than to interpret the syntax constructors as total functions that represent partial functions.

An *assignment* into a frame $\{D_\alpha \mid \alpha \in \mathcal{T}\}$ is a function φ whose domain is $\mathcal{V}$ such that $\varphi(\mathbf{x}_\alpha) \in D_\alpha$ for each $\mathbf{x}_\alpha \in \mathcal{V}$. Given an assignment φ, $\mathbf{x}_\alpha \in \mathcal{V}$, and $d \in D_\alpha$, let $\varphi[\mathbf{x}_\alpha \mapsto d]$ be the assignment ψ such that $\psi(\mathbf{x}_\alpha) = d$ and $\psi(\mathbf{y}_\beta) = \varphi(\mathbf{y}_\beta)$ for all variables $\mathbf{y}_\beta$ distinct from $\mathbf{x}_\alpha$. Given an interpretation $\mathcal{M} = (\{D_\alpha \mid \alpha \in \mathcal{T}\}, I)$, $\mathsf{assign}(\mathcal{M})$ is the set of assignments into the frame of $\mathcal{M}$.

3.3 General Models

We are now ready to define the notion of a "general model" for $\mathrm{CTT_{qe}}$ in which function domains need not contain all possible total functions. General models were introduced by Leon Henkin in [77].

An interpretation $\mathcal{M} = (\{D_\alpha \mid \alpha \in \mathcal{T}), I\}$ is a *general model* for $\mathrm{CTT_{qe}}$ if there is a binary valuation function $V^{\mathcal{M}}$ such that, for all assignments $\varphi \in \mathsf{assign}(\mathcal{M})$ and expressions $\mathbf{C}_\gamma$, $V^{\mathcal{M}}_\varphi(\mathbf{C}_\gamma) \in D_\gamma$[6] and each of the following conditions is satisfied:

[6] We write $V^{\mathcal{M}}_\varphi(\mathbf{C}_\gamma)$ instead of $V^{\mathcal{M}}(\varphi, \mathbf{C}_\gamma)$.

1. If $\mathbf{C}_\gamma \in \mathcal{V}$, then $V_\varphi^{\mathcal{M}}(\mathbf{C}_\gamma) = \varphi(\mathbf{C}_\gamma)$.
2. If $\mathbf{C}_\gamma \in \mathcal{C}$, then $V_\varphi^{\mathcal{M}}(\mathbf{C}_\gamma) = I(\mathbf{C}_\gamma)$.
3. If $\mathbf{C}_\gamma$ is $\mathbf{F}_{\alpha\to\beta}\,\mathbf{A}_\alpha$, then $V_\varphi^{\mathcal{M}}(\mathbf{C}_\gamma) = V_\varphi^{\mathcal{M}}(\mathbf{F}_{\alpha\to\beta})(V_\varphi^{\mathcal{M}}(\mathbf{A}_\alpha))$.
4. If $\mathbf{C}_\gamma$ is $\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_\beta$, then $V_\varphi^{\mathcal{M}}(\mathbf{C}_\gamma)$ 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 $V_\varphi^{\mathcal{M}}(\mathbf{C}_\gamma) = \mathcal{E}(\mathbf{A}_\alpha)$.
6. If $\mathbf{C}_\gamma$ is $[\![\mathbf{A}_\epsilon]\!]_\beta$ and $V_\varphi^{\mathcal{M}}(\mathsf{is\text{-}expr}^\beta_{\epsilon\to o}\,\mathbf{A}_\epsilon) = \mathrm{T}$, then
$$V_\varphi^{\mathcal{M}}(\mathbf{C}_\gamma) = V_\varphi^{\mathcal{M}}(\mathcal{E}^{-1}(V_\varphi^{\mathcal{M}}(\mathbf{A}_\epsilon))).$$
7. For each $\beta \in \mathcal{T}$, there is some $d_\beta \in D_\beta$ such that, if $\mathbf{C}_\gamma$ is $[\![\mathbf{A}_\epsilon]\!]_\beta$ and $V_\varphi^{\mathcal{M}}(\mathsf{is\text{-}expr}^\beta_{\epsilon\to o}\,\mathbf{A}_\epsilon) = \mathrm{F}$, then $V_\varphi^{\mathcal{M}}(\mathbf{C}_\gamma) = d_\beta$.

Proposition 3.3.1 *General models for* $\mathrm{CTT_{qe}}$ *exist.*

Proof It is easy to construct an interpretation $\mathcal{M} = (\{\mathcal{D}_\alpha \mid \alpha \in \mathcal{T}\}, I)$ such that $\mathcal{D}_{\alpha\to\beta}$ is the set of all total functions from $\mathcal{D}_\alpha$ to $\mathcal{D}_\beta$ for all $\alpha, \beta \in \mathcal{T}$. It is also easy to define by induction on the structure of expressions a valuation function $V^{\mathcal{M}}$ such that $\mathcal{M}$ is a general model for $\mathrm{CTT_{qe}}$. □

Remark 3.3.2 (Semantics of Evaluations) When $V_\varphi^{\mathcal{M}}(\mathsf{is\text{-}expr}^\beta_{\epsilon\to o}\,\mathbf{A}_\epsilon) = \mathrm{T}$, the semantics of $V_\varphi^{\mathcal{M}}([\![\mathbf{A}_\epsilon]\!]_\beta)$ involves a double valuation as mentioned in the Double Substitution Problem (see section 1).

Remark 3.3.3 (Undefined Evaluations) Suppose $V_\varphi^{\mathcal{M}}(\mathbf{A}_\epsilon)$ is an improper construction. Then $V_\varphi^{\mathcal{M}}(\mathcal{E}^{-1}(V_\varphi^{\mathcal{M}}(\mathbf{A}_\epsilon)))$ is undefined and $V_\varphi^{\mathcal{M}}([\![\mathbf{A}_\epsilon]\!]_\beta)$ has no natural value. Since $\mathrm{CTT_{qe}}$ does not admit undefined expressions, $V_\varphi^{\mathcal{M}}([\![\mathbf{A}_\epsilon]\!]_\beta)$ is defined to be d_β, an unspecified error value for the type β. Similarly, if $V_\varphi^{\mathcal{M}}(\mathbf{A}_\epsilon)$ is a proper construction of the form $\mathcal{E}(\mathbf{B}_\gamma)$ with $\gamma \neq \beta$, $V_\varphi^{\mathcal{M}}([\![\mathbf{A}_\epsilon]\!]_\beta) = d_\beta$.

Let $\mathcal{M}$ be a general model for $\mathrm{CTT_{qe}}$. $\mathbf{A}_o$ is *valid in* $\mathcal{M}$, written $\mathcal{M} \vDash \mathbf{A}_o$, if $V_\varphi^{\mathcal{M}}(\mathbf{A}_o) = \mathrm{T}$ for all $\varphi \in \mathsf{assign}(\mathcal{M})$. $\mathbf{A}_o$ is *valid in* $\mathrm{CTT_{qe}}$, written $\vDash \mathbf{A}_o$, if $\mathbf{A}_o$ is valid in every general model for $\mathrm{CTT_{qe}}$.

Let $T = (L, \Gamma)$ be a theory of $\mathrm{CTT_{qe}}$, $\mathbf{A}_o$ be a formula of T, and $\mathcal{H}$ be a set of formulas of T. A *general model for* T is a general model $\mathcal{M}$ for $\mathrm{CTT_{qe}}$ such that $\mathcal{M} \vDash \mathbf{A}_o$ for all $\mathbf{A}_o \in \Gamma$. $\mathbf{A}_o$ is *valid in* T, written $T \vDash \mathbf{A}_o$, if $\mathbf{A}_o$ is valid in every general model for T. $\mathcal{H}$ *entails* $\boldsymbol{A}_o$ *in* T, written $T, \mathcal{H} \vDash \mathbf{A}_o$, if, for all general models $\mathcal{M}$ for T and all $\varphi \in \mathsf{assign}(\mathcal{M})$, $V_\varphi^{\mathcal{M}}(\mathbf{H}_o) = \mathrm{T}$ for all $\mathbf{H}_o \in \mathcal{H}$ implies $V_\varphi^{\mathcal{M}}(\mathbf{A}_o) = \mathrm{T}$.

Proposition 3.3.4 *Let* $\mathcal{M}$ *be a general model for* $\mathrm{CTT_{qe}}$, $\boldsymbol{A}_\epsilon$ *be a construction, and* $\varphi \in \mathsf{assign}(\mathcal{M})$. *Then* $V_\varphi^{\mathcal{M}}(\boldsymbol{A}_\epsilon) = \boldsymbol{A}_\epsilon$.

Proof Follows immediately from conditions 6–8 of the definition of an interpretation and conditions 3 and 5 of the definition of a general model. □

Theorem 3.3.5 (Law of Quotation) $\ulcorner \boldsymbol{A}_\alpha \urcorner = \mathcal{E}(\boldsymbol{A}_\alpha)$ *is valid in* CTT_{qe}.

Proof Let $\mathcal{M}$ be a general model for CTT_{qe} and $\varphi \in \mathsf{assign}(\mathcal{M})$. Then

$$V_\varphi^{\mathcal{M}}(\ulcorner \mathbf{A}_\alpha \urcorner) \tag{1}$$
$$= \mathcal{E}(\mathbf{A}_\alpha) \tag{2}$$
$$= V_\varphi^{\mathcal{M}}(\mathcal{E}(\mathbf{A}_\alpha)) \tag{3}$$

(2) follows from condition 5 of the definition of a general model and (3) follows from Proposition 3.3.4. Hence $V_\varphi^{\mathcal{M}}(\ulcorner \mathbf{A}_\alpha \urcorner) = V_\varphi^{\mathcal{M}}(\mathcal{E}(\mathbf{A}_\alpha))$ for all $\varphi \in \mathsf{assign}(\mathcal{M})$, and so $\ulcorner \mathbf{A}_\alpha \urcorner = \mathcal{E}(\mathbf{A}_\alpha)$ is valid in every general model for CTT_{qe}. □

Theorem 3.3.6 (Law of Disquotation) $[\![\ulcorner \boldsymbol{A}_\alpha \urcorner]\!]_\alpha = \boldsymbol{A}_\alpha$ *is valid in* CTT_{qe}.

Proof

Let $\mathcal{M}$ be a general model for CTT_{qe} and $\varphi \in \mathsf{assign}(\mathcal{M})$. Then

$$V_\varphi^{\mathcal{M}}([\![\ulcorner \mathbf{A}_\alpha \urcorner]\!]_\alpha) \tag{1}$$
$$= V_\varphi^{\mathcal{M}}(\mathcal{E}^{-1}(V_\varphi^{\mathcal{M}}(\ulcorner \mathbf{A}_\alpha \urcorner))) \tag{2}$$
$$= V_\varphi^{\mathcal{M}}(\mathcal{E}^{-1}(\mathcal{E}(\mathbf{A}_\alpha))) \tag{3}$$
$$= V_\varphi^{\mathcal{M}}(\mathbf{A}_\alpha) \tag{4}$$

(a) $V_\varphi^{\mathcal{M}}(\ulcorner \mathbf{A}_\alpha \urcorner) = \mathcal{E}(\mathbf{A}_\alpha)$ is by condition 5 of the definition of a general model. (b) $V_\varphi^{\mathcal{M}}(\mathsf{is\text{-}expr}^{\alpha}_{\epsilon \to o} \ulcorner \mathbf{A}_\alpha \urcorner) = \text{T}$ follows from (a). (2) follows from (b) and condition 6 of the definition of a general model; (3) follows from (a); and (4) is immediate. Hence $V_\varphi^{\mathcal{M}}([\![\ulcorner \mathbf{A}_\alpha \urcorner]\!]_\alpha) = V_\varphi^{\mathcal{M}}(\mathbf{A}_\alpha)$ for all $\varphi \in \mathsf{assign}(\mathcal{M})$, and so $[\![\ulcorner \mathbf{A}_\alpha \urcorner]\!]_\alpha = \mathbf{A}_\alpha$ is valid in every general model for CTT_{qe}.

□

Remark 3.3.7 (Evaluation Problem) Theorem 3.3.6 shows that disquotation holds universally in CTT_{qe} contrary to the Evaluation Problem (see section 1). We have avoided the Evaluation Problem in CTT_{qe} by admitting only quotations of eval-free expressions and thus making it impossible to express the liar paradox. If quotations of non-eval-free expressions were allowed in CTT_{qe}, the logic would be significantly more expressive, but also much more complicated, as seen in $\mathcal{Q}_0^{\text{uqe}}$ [55].

3.4 Standard Models

An interpretation $\mathcal{M} = (\{\mathcal{D}_\alpha \mid \alpha \in \mathcal{T}), I\}$ is a *standard model* for CTT_{qe} if $\mathcal{D}_{\alpha\to\beta}$ is the set of all total functions from $\mathcal{D}_\alpha$ to $\mathcal{D}_\beta$ for all $\alpha, \beta \in \mathcal{T}$.

Lemma 3.4.1 *Standard models for* CTT_{qe} *exist, and every standard model for* CTT_{qe} *is also a general model for* CTT_{qe}.

Proof By the proof of Proposition 3.3.1. □

A general model for CTT_{qe} is a *nonstandard model* for CTT_{qe} if it is not a standard model.

4 Examples

We will present in this section four examples that illustrate the utility of the quotation and evaluation facility in CTT_{qe}.

4.1 Reasoning about Syntax

Reasoning about the syntax of expressions is normally performed in the metalogic, but in CTT_{qe} reasoning about the syntax of eval-free expressions can be performed in the logic itself. This is done by reasoning about constructions (which represent the syntax trees of eval-free expressions) using quotation and the machinery of constructions. Algorithms that manipulate eval-free expressions can be formalized as functions that manipulate constructions. The functions can be executed using beta-reduction, rewriting, and other kinds of simplification.

As an example, consider the constant $\mathsf{make\text{-}implication}_{\epsilon\to\epsilon\to\epsilon}$ defined as

$$\lambda\, x_\epsilon \,.\, \lambda\, y_\epsilon \,.\, (\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\, (\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\, \ulcorner \supset_{o\to o\to o} \urcorner\, x_\epsilon)\, y_\epsilon).$$

It can be used to build constructions that represent implications. As another example, consider the constant $\mathsf{is\text{-}app}_{\epsilon\to o}$ defined as

$$\lambda\, x_\epsilon \,.\, \exists\, y_\epsilon \,.\, \exists\, z_\epsilon \,.\, x_\epsilon = (\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\, y_\epsilon\, z_\epsilon).$$

It can be used to test whether a construction represents a function application.

Reasoning about syntax is a two-step process: First, a construction is built using quotation and the machinery of constructions, and second, the construction is employed using evaluation. Continuing the example above,

$$\mathsf{make\text{-}implication}_{\epsilon\to\epsilon\to\epsilon}\, \ulcorner \mathbf{A}_o \urcorner\, \ulcorner \mathbf{B}_o \urcorner$$

builds a construction equivalent to the quotation $\ulcorner \mathbf{A}_o \supset \mathbf{B}_o \urcorner$ and

$$[\![\mathsf{make\text{-}implication}_{\epsilon\to\epsilon\to\epsilon}\, \ulcorner \mathbf{A}_o \urcorner\, \ulcorner \mathbf{B}_o \urcorner]\!]_o$$

employs the construction as the implication $\mathbf{A}_o \supset \mathbf{B}_o$. We prove these two results within the proof system for CTT_{qe} in subsection 9.1 (see Propositions 9.1.1 and 9.1.2).

Using this mixture of quotation and evaluation, it is possible to express the interplay of syntax and semantics that is needed to formalize syntax-based algorithms that are commonly used in mathematics [54]. See subsection 4.4 for an example.

4.2 Quasiquotation

Quasiquotation is a parameterized form of quotation in which the parameters serve as holes in a quotation that are filled with expressions that denote syntactic values. It is a very powerful syntactic device for specifying expressions and defining macros. Quasiquotation was introduced by Willard Van Orman Quine in 1940 in the first version of his book *Mathematical Logic* [105]. It has been extensively employed in the Lisp family of programming languages [8].[7]

In CTT_{qe}, constructing a large quotation from smaller quotations can be tedious because it requires many applications of syntax constructors. Quasiquotation provides a convenient way to construct big quotations from little quotations. It can be defined straightforwardly in CTT_{qe}.

A *quasi-expression* of CTT_{qe} is defined inductively as follows:

1. $\lfloor \mathbf{A}_\epsilon \rfloor$ is a quasi-expression called an *antiquotation*.
2. $\mathbf{x}_\alpha$ is a quasi-expression.
3. $\mathbf{c}_\alpha$ is a quasi-expression.
4. If M and N are quasi-expressions, then $(M\,N)$, $(\lambda\,\mathbf{x}_\alpha\,.\,N)$, $(\lambda\,\lfloor \mathbf{A}_\epsilon \rfloor\,.\,N)$, and $\ulcorner M \urcorner$ are quasi-expressions.

A quasi-expression is thus an eval-free expression where one or more subexpressions have been replaced by antiquotations. For example, $\neg(\mathbf{A}_o \wedge \lfloor \mathbf{B}_\epsilon \rfloor)$ is a quasi-expression. Obviously, every eval-free expression is a quasi-expression.

Let $\mathcal{E}'$ be the function mapping quasi-expressions to expressions of type ϵ that is defined inductively as follows:

1. $\mathcal{E}'(\lfloor \mathbf{A}_\epsilon \rfloor) = \mathbf{A}_\epsilon$.
2. $\mathcal{E}'(\mathbf{x}_\alpha) = \ulcorner \mathbf{x}_\alpha \urcorner$.
3. $\mathcal{E}'(\mathbf{c}_\alpha) = \ulcorner \mathbf{c}_\alpha \urcorner$.
4. $\mathcal{E}'(M\,N) = \mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\,\mathcal{E}'(M)\,\mathcal{E}'(N)$.
5. $\mathcal{E}'(\lambda\,M\,.\,N) = \mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}\,\mathcal{E}'(M)\,\mathcal{E}'(N)$.
6. $\mathcal{E}(\ulcorner M \urcorner) = \mathsf{quo}_{\epsilon\to\epsilon}\,\mathcal{E}'(M)$.

Notice that $\mathcal{E}'(M) = \mathcal{E}(M)$ when M is an eval-free expression. Continuing our example above, $\mathcal{E}'(\neg(\mathbf{A}_o \wedge \lfloor \mathbf{B}_\epsilon \rfloor)) =$

$$\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\,\ulcorner \neg_{o\to o} \urcorner\,(\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\,(\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\,\ulcorner \wedge_{o\to o\to o} \urcorner\,\mathcal{E}'(\mathbf{A}_o))\,\mathbf{B}_\epsilon).$$

A *quasiquotation* is an expression of the form $\ulcorner M \urcorner$ where M is a quasi-expression. Thus every quotation is a quasiquotation. The quasiquotation $\ulcorner M \urcorner$

[7] In Lisp, the standard symbol for quasiquotation is the backquote (‘) symbol, and thus in Lisp, quasiquotation is usually called *backquote*.

serves as an alternate notation for the expression $\mathcal{E}'(M)$. So $\ulcorner\neg(\mathbf{A}_o \wedge \lfloor\mathbf{B}_\epsilon\rfloor)\urcorner$ stands for the significantly more verbose expression in the previous paragraph. It represents the syntax tree of a negated conjunction in which the part of the tree corresponding to the second conjunct is replaced by the syntax tree represented by $\mathbf{B}_\epsilon$. If $\mathbf{B}_\epsilon$ is a quotation $\ulcorner\mathbf{C}_o\urcorner$, then the quasiquotation $\ulcorner\neg(\mathbf{A}_o \wedge \lfloor\ulcorner\mathbf{C}_o\urcorner\rfloor)\urcorner$ is equivalent to the quotation $\ulcorner\neg(\mathbf{A}_o \wedge \mathbf{C}_o)\urcorner$.

The use of quasiquotation is further illustrated in the next two subsections.

4.3 Schemas

A *schema* is a metalogical expression containing syntactic variables. An instance of a schema is a logical expression obtained by replacing the syntactic variables with appropriate logical expressions. In CTT_{qe}, a schema can be formalized as a single logical expression.

For example, consider the *law of excluded middle (LEM)* that is expressed as the formula schema $A \vee \neg A$ where A is a syntactic variable ranging over all formulas. LEM can be formalized in CTT_{qe} as the universal statement

$$\forall\, x_\epsilon \,.\, \mathsf{is\text{-}expr}^{o}_{\epsilon\to o}\, x_\epsilon \supset [\![x_\epsilon]\!]_o \vee \neg[\![x_\epsilon]\!]_o.$$

An instance of this formalization of LEM is any instance of the universal statement. Using quasiquotation, LEM could also be formalized in CTT_{qe} as

$$\forall\, x_\epsilon \,.\, \mathsf{is\text{-}expr}^{o}_{\epsilon\to o}\, x_\epsilon \supset [\![\ulcorner\lfloor x_\epsilon\rfloor \vee \neg\lfloor x_\epsilon\rfloor\urcorner]\!]_o.$$

In subsection 9.2, we prove the first formalization of LEM within the proof system for CTT_{qe}, and we show how instances of LEM can be derived by instantiating this formalization (see Theorem 9.2.1 and Proposition 9.2.2).

If we assume that the domain of the type ι is the natural numbers and $\mathcal{C}$ includes the usual constants of natural number arithmetic (including a constant $S_{\iota\to\iota}$ representing the successor function), then the (first-order) *induction schema for Peano arithmetic* can be formalized in CTT_{qe} as

$$\begin{aligned}&\forall\, f_\epsilon \,.\, ((\mathsf{is\text{-}expr}^{\iota\to o}_{\epsilon\to o}\, f_\epsilon \wedge \mathsf{is\text{-}peano}_{\epsilon\to o}\, f_\epsilon) \supset \\ &\quad ((([\![f_\epsilon]\!]_{\iota\to o}\, 0 \wedge (\forall\, x_\iota \,.\, [\![f_\epsilon]\!]_{\iota\to o}\, x_\iota \supset [\![f_\epsilon]\!]_{\iota\to o}\, (S_{\iota\to\iota}\, x_\iota))) \supset \forall\, x_\iota \,.\, [\![f_\epsilon]\!]_{\iota\to o}\, x_\iota))\end{aligned}$$

where $\mathsf{is\text{-}peano}_{\epsilon\to o}\, f_\epsilon$ holds iff f_ϵ represents the syntax tree of a predicate of first-order Peano arithmetic. The *induction schema for Presburger arithmetic* is exactly the same as the induction schema for Peano arithmetic except that the predicate $\mathsf{is\text{-}peano}_{\epsilon\to o}$ is replaced by an appropriate predicate $\mathsf{is\text{-}presburger}_{\epsilon\to o}$. Hence it is possible to directly express both first-order Peano arithmetic and Presburger arithmetic as theories in CTT_{qe}.

4.4 Meaning Formulas

Many symbolic algorithms work by manipulating mathematical expressions in a mathematically meaningful way. A *meaning formula* for such an algorithm

is a statement that captures the mathematical relationship between the input and output expressions of the algorithm. For example, consider a symbolic differentiation algorithm that takes as input an expression representing a function (say x^2), repeatedly applies syntactic differentiation rules to the expression, and then returns as output the final expression that is produced ($2x$). The intended meaning formula of this algorithm states that the function $(\lambda\, x : \mathbb{R} \,.\, 2x)$ represented by the output expression $2x$ is the derivative of the function $(\lambda\, x : \mathbb{R} \,.\, x^2)$ represented by the input expression x^2.

Meaning formulas are difficult to express in a traditional logic like first-order logic or simple type theory since there is no way to directly refer to the syntactic structure of the expressions in the logic [54]. However, meaning formulas can be easily expressed in CTT_{qe}.

Consider the following example on the differentiation of polynomials. Let $T_{\mathbb{R}} = (L_{\mathcal{D}}, \Gamma)$ be a theory of CTT_{qe} in which the type ι is specified to be the real numbers (see [52, p. 276–277]). We assume that $\mathcal{D}$ contains the following constants:

1. Constants for zero and one: 0_ι and 1_ι.
2. The usual constants for the negation, addition, and multiplication of real numbers: $-_{\iota\to\iota}$, $+_{\iota\to\iota\to\iota}$, and $*_{\iota\to\iota\to\iota}$.
3. The usual constants for the negation, addition, and multiplication of functions: $-_{(\iota\to\iota)\to(\iota\to\iota)}$, $+_{(\iota\to\iota)\to(\iota\to\iota)\to(\iota\to\iota)}$, and $*_{(\iota\to\iota)\to(\iota\to\iota)\to(\iota\to\iota)}$.
4. $\mathsf{is\text{-}diff}_{(\iota\to\iota)\to o}$ such that $\mathsf{is\text{-}diff}_{(\iota\to\iota)\to o}\, \mathbf{F}_{\iota\to\iota}$ holds iff the function $\mathbf{F}_{\iota\to\iota}$ is differentiable (at every point in the domain of ι).
5. $\mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}$ such that $\mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}\, \mathbf{F}_{\iota\to\iota}$ denotes the derivative of the function $\mathbf{F}_{\iota\to\iota}$.
6. $\mathsf{is\text{-}poly}_{\epsilon\to o}$ such that $\mathsf{is\text{-}poly}_{\epsilon\to o}\, \mathbf{A}_\epsilon$ holds iff $\mathbf{A}_\epsilon$ represents a syntax tree of an expression of type ι that has the form of a polynomial with variables from $\{x_\iota, y_\iota\}$ and constants from $\{0_\iota, 1_\iota\}$.
7. $\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}$ such that $\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, \mathbf{A}_\epsilon \ulcorner \mathbf{x}_\iota \urcorner$ denotes the result of applying the usual differentiation rules for polynomials to $\mathbf{A}_\epsilon$ with respect to $\mathbf{x}_\iota$.

When n is a natural number with $n \geq 2$, let n_ι be an abbreviation defined by:

1. 2_ι stands for $1_\iota + 1_\iota$.
2. $(n+1)_\iota$ stands for $n_\iota + 1_\iota$.

To improve readability, $+_{\iota\to\iota\to\iota}$, $*_{\iota\to\iota\to\iota}$, $+_{(\iota\to\iota)\to(\iota\to\iota)\to(\iota\to\iota)}$, and $*_{(\iota\to\iota)\to(\iota\to\iota)\to(\iota\to\iota)}$ are written as infix operators. $\mathsf{is\text{-}poly}_{\epsilon\to o}$ is recursively

defined in $T_{\mathbb{R}}$ (using quasiquotation) as:

$$\begin{aligned}
\lambda\, u_\epsilon \,.\, u_\epsilon = \ulcorner x_\iota \urcorner \vee u_\epsilon = \ulcorner y_\iota \urcorner \vee u_\epsilon = \ulcorner 0_\iota \urcorner \vee u_\epsilon = \ulcorner 1_\iota \urcorner \vee \\
\quad \exists\, v_\epsilon \,.\, (\mathsf{is\text{-}poly}_{\epsilon\to o}\, v_\epsilon \wedge u_\epsilon = \ulcorner -_{\iota\to\iota} \lfloor v_\epsilon \rfloor \urcorner) \vee \\
\quad \exists\, v_\epsilon \,.\, \exists\, w_\epsilon \,.\, (\mathsf{is\text{-}poly}_{\epsilon\to o}\, v_\epsilon \wedge \mathsf{is\text{-}poly}_{\epsilon\to o}\, w_\epsilon \wedge \\
\quad\quad u_\epsilon = \ulcorner \lfloor v_\epsilon \rfloor +_{\iota\to\iota\to\iota} \lfloor w_\epsilon \rfloor \urcorner) \vee \\
\quad \exists\, v_\epsilon \,.\, \exists\, w_\epsilon \,.\, (\mathsf{is\text{-}poly}_{\epsilon\to o}\, v_\epsilon \wedge \mathsf{is\text{-}poly}_{\epsilon\to o}\, w_\epsilon \wedge \\
\quad\quad u_\epsilon = \ulcorner \lfloor v_\epsilon \rfloor *_{\iota\to\iota\to\iota} \lfloor w_\epsilon \rfloor \urcorner)
\end{aligned}$$

When n is a natural number, let $\mathbf{x}_\iota^n$ be an abbreviation defined by:

1. $\mathbf{x}_\iota^0$ stands for 1_ι.
2. $\mathbf{x}_\iota^{m+1}$ stands for $\mathbf{x}_\iota^m * \mathbf{x}_\iota$.

$\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}$ is specified in $T_{\mathbb{R}}$ (using quasiquotation) by the following formulas:

1. $\mathsf{is\text{-}var}^\iota_{\epsilon\to o}\, x_\epsilon \supset \mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, x_\epsilon\, x_\epsilon = \ulcorner 1_\iota \urcorner$.
2. $(\mathsf{is\text{-}var}^\iota_{\epsilon\to o}\, x_\epsilon \wedge \mathsf{is\text{-}var}^\iota_{\epsilon\to o}\, y_\epsilon \wedge x_\epsilon \neq y_\epsilon) \supset \mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, x_\epsilon\, y_\epsilon = \ulcorner 0_\iota \urcorner$.
3. $(\mathsf{is\text{-}con}^\iota_{\epsilon\to o}\, x_\epsilon \wedge \mathsf{is\text{-}var}^\iota_{\epsilon\to o}\, y_\epsilon) \supset \mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, x_\epsilon\, y_\epsilon = \ulcorner 0_\iota \urcorner$.
4. $(\mathsf{is\text{-}poly}_{\epsilon\to o}\, x_\epsilon \wedge \mathsf{is\text{-}var}^\iota_{\epsilon\to o}\, y_\epsilon) \supset$
   $\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon} \ulcorner -_{\iota\to\iota} \lfloor x_\epsilon \rfloor \urcorner\, y_\epsilon = \ulcorner -_{\iota\to\iota} \lfloor \mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, x_\epsilon\, y_\epsilon \rfloor \urcorner)$.
5. $(\mathsf{is\text{-}poly}_{\epsilon\to o}\, x_\epsilon \wedge \mathsf{is\text{-}poly}_{\epsilon\to o}\, y_\epsilon \wedge \mathsf{is\text{-}var}^\iota_{\epsilon\to o}\, z_\epsilon) \supset$
   $\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon} \ulcorner \lfloor x_\epsilon \rfloor +_{\iota\to\iota\to\iota} \lfloor y_\epsilon \rfloor \urcorner\, z_\epsilon =$
   $\ulcorner \lfloor \mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, x_\epsilon\, z_\epsilon \rfloor +_{\iota\to\iota\to\iota} \lfloor \mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, y_\epsilon\, z_\epsilon \rfloor \urcorner$.
6. $(\mathsf{is\text{-}poly}_{\epsilon\to o}\, x_\epsilon \wedge \mathsf{is\text{-}poly}_{\epsilon\to o}\, y_\epsilon \wedge \mathsf{is\text{-}var}^\iota_{\epsilon\to o}\, z_\epsilon) \supset$
   $\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon} \ulcorner \lfloor x_\epsilon \rfloor *_{\iota\to\iota\to\iota} \lfloor y_\epsilon \rfloor \urcorner\, z_\epsilon =$
   $\ulcorner (\lfloor (\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, x_\epsilon\, z_\epsilon) \rfloor *_{\iota\to\iota\to\iota} \lfloor y_\epsilon \rfloor) +_{\iota\to\iota\to\iota}$
   $(\lfloor x_\epsilon \rfloor *_{\iota\to\iota\to\iota} \lfloor (\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, y_\epsilon\, z_\epsilon) \rfloor) \urcorner$.

The value of an application of $\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}$ can be computed using these formulas as conditional rewrite rules.

The meaning formula for $\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}$ can then be given as

$$\begin{aligned}
\forall\, u_\epsilon \,.\, \forall\, v_\epsilon \,.\, ((\mathsf{is\text{-}poly}_{\epsilon\to o}\, u_\epsilon) \wedge \mathsf{is\text{-}var}^\iota_{\epsilon\to o}\, v_\epsilon \supset \\
\quad \mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}(\llbracket \mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}\, v_\epsilon\, u_\epsilon \rrbracket_{\iota\to\iota}) = \\
\quad \llbracket \mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}\, v_\epsilon\, (\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, u_\epsilon\, v_\epsilon) \rrbracket_{\iota\to\iota}).^{8}
\end{aligned}$$

Unfortunately, we are not able to prove this formula in the proof system for CTT_{qe} presented later in the paper. So instead we will define the *meaning formula for* $\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}$ as the formula schema

$$\begin{aligned}
\forall\, u_\epsilon \,.\, (\mathsf{is\text{-}poly}_{\epsilon\to o}\, u_\epsilon \supset \\
\quad \mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}(\lambda\, \mathbf{x}_\iota \,.\, \llbracket u_\epsilon \rrbracket_\iota) = \lambda\, \mathbf{x}_\iota \,.\, \llbracket \mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, u_\epsilon \ulcorner \mathbf{x}_\iota \urcorner \rrbracket_\iota)
\end{aligned}$$

where $\mathbf{x}_\iota$ is either x_ι or y_ι. As expected, the equation

$$\mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}(\lambda\, x_\iota \,.\, x_\iota^2) = \lambda\, x_\iota \,.\, 2 * x_\iota$$

follows from this meaning formula and the definitions of $\mathsf{is\text{-}poly}_{\epsilon\to o}$ and $\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}$.

In subsection 9.3, we prove the meaning formula for $\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}$ and the application of it given above within the proof system for CTT_{qe} (see Theorem 9.3.3 and Proposition 9.3.4).

5 Proof System

We present in this section the proof system for CTT_{qe} which is an extension of Andrews' elegant proof system for $\mathcal{Q}_0$ given in [3, p. 213].

5.1 Requirements

At first glance, it would appear that a proof system for CTT_{qe} could be straightforwardly developed by extending Andrews' proof system for $\mathcal{Q}_0$. We can define $\mathsf{is\text{-}var}_{\epsilon\to o}$, $\mathsf{is\text{-}var}^{\alpha}_{\epsilon\to o}$, $\mathsf{is\text{-}con}_{\epsilon\to o}$, $\mathsf{is\text{-}con}^{\alpha}_{\epsilon\to o}$ by axiom schemas. We can also define $\mathsf{is\text{-}expr}_{\epsilon\to o}$, $\mathsf{is\text{-}expr}^{\alpha}_{\epsilon\to o}$, $\sqsubset_{\epsilon\to\epsilon\to o}$, and $\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}$, by axiom schemas using recursion on the construction of expressions. We can specify that the type ϵ of constructions is an inductive type using a set of axioms that say (1) the constructions are distinct from each other and (2) induction holds for constructions. We can specify quotation using the Law of Quotation $\ulcorner \mathbf{A}_\alpha \urcorner = \mathcal{E}(\mathbf{A}_\alpha)$ (Theorem 3.3.5). And we can specify evaluation using the Law of Disquotation $[\![\ulcorner \mathbf{A}_\alpha \urcorner]\!]_\alpha = \mathbf{A}_\alpha$ (Theorem 3.3.6).

Andrews' proof system with these added axioms would enable simple theorems about closed expressions involving quotation and evaluation to be proved, but the proof system would not be able to substitute expressions for free variables occurring within evaluations. Hence axiom schemas and meaning formulas could be expressed in CTT_{qe}, but they would be useless because they could not be instantiated using the proof system. Clearly, a useful proof system for CTT_{qe} requires some form of substitution that is applicable to evaluations. The proof system we have described above would also not be able to reduce evaluations that contain free variables. Hence it would not be possible to prove schemas and meaning formulas. A useful proof system for CTT_{qe} needs a means to reduce evaluations applied to expressions that are not just quotations.

To be useful, a proof system for CTT_{qe} needs to satisfy the following requirements:

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

[8] We restrict this example to polynomials since polynomial functions and their derivatives are always total. Thus issues of undefinedness do not arise in the formulation of the meaning formula for $\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}$.

2. **R2**. The proof system is complete with respect to the (general models) semantics for CTT_{qe} for eval-free formulas.

3. **R3**. The proof system can be used to reason about constructions that are denoted by quotations and other expressions of type ϵ.

4. **R4**. The proof system can instantiate free variables occurring within evaluations as found in formulas that represent schemas and meaning formulas.

5. **R5**. The proof system can prove formulas, such as those that represent schemas and meaning formulas, in which free variables occur within evaluations.

In this section we present a proof system that is intended to satisfy requirements R1–5. In subsequent sections we will show that this proof system actually does satisfy these requirements.

5.2 Challenges

A proof system for a logic with quotation and evaluation must solve the three design problems presented in section 1. The Evaluation Problem is not an issue in CTT_{qe} since quotations do not contain evaluations. However, the Variable and Double Substitution Problems are serious issues for CTT_{qe}. Due to the Variable Problem, concepts concerning free and bound variables cannot be purely syntactic, and the due to the Double Substitution Problem, a substitution must be performed twice in some cases.

The notion of a variable being free in an expression and the notion of a substitution of an expression for the free occurrences of a variable in another expression are two crucial concepts affected by the Variable Problem. To express the axioms and rules of inference of a proof system for CTT_{qe}, we need a way to say "a variable $\mathbf{x}_\alpha$ is free in an expression $\mathbf{B}_\beta$". We defined this concept in subsection 2.2 in the usual way when $\mathbf{B}_\beta$ is eval-free as a purely syntactic assertion expressed in the metalogic of CTT_{qe}. However, when $\mathbf{B}_\beta$ is not eval-free, this approach does not work because whether $\mathbf{x}_\alpha$ occurs in $\mathbf{B}_\beta$ may depend on the semantics of $\mathbf{B}_\beta$.

An approach taking advantage of CTT_{qe}'s facility for reasoning about the syntax of expressions is to introduce a logical constant $\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}$ such that

$$\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}\ulcorner\mathbf{x}_\alpha\urcorner\ulcorner\mathbf{B}_\beta\urcorner$$

holds iff "$\mathbf{x}_\alpha$ is free in $\mathbf{B}_\beta$". This works fine when $\mathbf{B}_\beta$ is eval-free, but it does not work at all when $\mathbf{B}_\beta$ is not eval-free since once again quotations cannot contain evaluations. (This approach does work in $\mathcal{Q}_0^{\text{uqe}}$ [55] in which quotations in CTT_{qe} may contain evaluations.) Nevertheless, the constant $\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}$ is included in CTT_{qe} so that semantic statements of the form

$$\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}\ulcorner\mathbf{x}_\alpha\urcorner\,\mathbf{A}_\epsilon,$$

can be expressed where $\mathbf{A}_\epsilon$ is not a quotation and may even contain free variables.

So what do we do when $\mathbf{B}_\beta$ is not eval-free? Our approach is to use the more restrictive semantic notion "$\mathbf{x}_\alpha$ is effective in $\mathbf{B}_\beta$" in place of the syntactic notion "$\mathbf{x}_\alpha$ is free in $\mathbf{B}_\beta$" when $\mathbf{B}_\beta$ is possibly not eval-free. "$\mathbf{x}_\alpha$ is effective in $\mathbf{B}_\beta$" means that the value of $\mathbf{B}_\beta$ depends on the value of $\mathbf{x}_\alpha$. Clearly, if $\mathbf{B}_\beta$ is eval-free, "$\mathbf{x}_\alpha$ is effective in $\mathbf{B}_\beta$" implies "$\mathbf{x}_\alpha$ is free in $\mathbf{B}_\beta$". However, "$\mathbf{x}_\alpha$ is effective in B_β" is a refinement of "$\mathbf{x}_\alpha$ is free in $\mathbf{B}_\beta$" on eval-free expressions since $\mathbf{x}_\alpha$ is free in $\mathbf{x}_\alpha = \mathbf{x}_\alpha$, but $\mathbf{x}_\alpha$ is not effective in $\mathbf{x}_\alpha = \mathbf{x}_\alpha$.

We express "$\mathbf{x}_\alpha$ is effective in $\mathbf{B}_\beta$" in CTT_{qe} by the abbreviation

$$\mathsf{IS\text{-}EFFECTIVE\text{-}IN}(\mathbf{x}_\alpha, \mathbf{B}_\beta) \text{ stands for } \exists\, \mathbf{y}_\alpha \,.\, ((\lambda\, \mathbf{x}_\alpha \,.\, \mathbf{B}_\beta)\, \mathbf{y}_\alpha \neq \mathbf{B}_\beta)$$

where $\mathbf{y}_\alpha$ is any variable of type α that differs from $\mathbf{x}_\alpha$. We will prove later (Lemma 7.1.3) that $\mathsf{IS\text{-}EFFECTIVE\text{-}IN}(\mathbf{x}_\alpha, \mathbf{B}_\beta)$ holds precisely when the value of $\mathbf{B}_\beta$ depends on the value of $\mathbf{x}_\alpha$. The following are examples of valid formulas in CTT_{qe} involving $\mathsf{IS\text{-}EFFECTIVE\text{-}IN}$:

1. $(\exists\, \mathbf{u}_\alpha \,.\, \exists\, \mathbf{v}_\alpha \,.\, \mathbf{u}_\alpha \neq \mathbf{v}_\alpha) \supset \mathsf{IS\text{-}EFFECTIVE\text{-}IN}(\mathbf{x}_\alpha, \mathbf{x}_\alpha)$.
2. $\mathsf{IS\text{-}EFFECTIVE\text{-}IN}(\mathbf{x}_\epsilon, [\![\mathbf{x}_\epsilon]\!]_\alpha)$.
3. $((\exists\, \mathbf{u}_\alpha \,.\, \exists\, \mathbf{v}_\alpha \,.\, \mathbf{u}_\alpha \neq \mathbf{v}_\alpha) \wedge \mathbf{y}_\epsilon = \ulcorner \mathbf{x}_\alpha \urcorner) \supset \mathsf{IS\text{-}EFFECTIVE\text{-}IN}(\mathbf{x}_\alpha, [\![\mathbf{y}_\epsilon]\!]_\alpha)$.
4. $\neg \mathsf{IS\text{-}EFFECTIVE\text{-}IN}(\mathbf{x}_\alpha, \lambda\, \mathbf{x}_\alpha \,.\, \mathbf{B}_\beta)$.
5. $\neg \mathsf{IS\text{-}EFFECTIVE\text{-}IN}(\mathbf{x}_\alpha, \mathbf{B}_\beta = \mathbf{B}_\beta)$.
6. $\neg \mathsf{IS\text{-}EFFECTIVE\text{-}IN}(\mathbf{x}_\alpha, \mathbf{B}_\beta \vee \neg \mathbf{B}_\beta)$.

As a consequence of the Variable Problem, substitution involving evaluations cannot be purely syntactic as in a traditional logic. It must be a semantics-dependent operation in which side conditions, like whether a variable is free in an expression, are proved within the proof system. Since CTT_{qe} supports reasoning about syntax, an obvious way forward is to add to $\mathcal{C}$ a logical constant $\mathsf{sub}_{\epsilon\to\epsilon\to\epsilon\to\epsilon}$ such that

$$\mathsf{sub}_{\epsilon\to\epsilon\to\epsilon\to\epsilon} \ulcorner \mathbf{A}_\alpha \urcorner \ulcorner \mathbf{x}_\alpha \urcorner \ulcorner \mathbf{B}_\beta \urcorner = \ulcorner \mathbf{C}_\beta \urcorner$$

holds iff "$\mathbf{C}_\beta$ is the result of substituting $\mathbf{A}_\alpha$ for each free occurrence of $\mathbf{x}_\alpha$ in $\mathbf{B}_\beta$ without any variable captures". $\mathsf{sub}_{\epsilon\to\epsilon\to\epsilon\to\epsilon}$ thus plays the role of an explicit substitution operator [1]. This approach, however, does not work in CTT_{qe} since $\mathbf{B}_\beta$ may contain evaluations, but quotations in CTT_{qe} may not contain evaluations. (Although the approach does work in $\mathcal{Q}_0^{\text{uqe}}$ [55] in which quotations in CTT_{qe} may contain evaluations, it is extremely complicated due to the Evaluation Problem.)

A more promising approach is to add axiom schemas to the five beta-reduction axiom schemas used by Andrews' in his proof system for $\mathcal{Q}_0$ [3, p. 213]

that specify beta-reduction of applications of the form $(\lambda\,\mathbf{x}_\alpha\,.\,\ulcorner\mathbf{B}_\beta\urcorner)\,\mathbf{A}_\alpha$ and $(\lambda\,\mathbf{x}_\alpha\,.\,[\![\mathbf{B}_\epsilon]\!]_\beta)\,\mathbf{A}_\alpha$.

But how do we solve the Double Substitution Problem in applications of the second form? There seems to be no easy way of emulating a double substitution with beta-reduction, so the best approach appears to be to consider only cases that do not require a second substitution, as formalized by the axiom schema B11.2 given below which uses the constant $\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}$. We will see that being able to beta-reduce lambda applications that correspond to double substitutions is not a necessity and that a proof system without this capability can prove many useful theorems involving evaluations.

In summary, we address the problem of stating "$\mathbf{x}_\alpha$ is free in $\mathbf{B}_\beta$" by (1) expressing it either syntactically in the usual way or semantically as $\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}\ulcorner\mathbf{x}_\alpha\urcorner\ulcorner\mathbf{B}_\beta\urcorner$ when $\mathbf{B}_\beta$ is eval-free and (2) expressing it more tightly as $\mathsf{IS\text{-}EFFECTIVE\text{-}IN}(\mathbf{x}_\alpha,\mathbf{B}_\beta)$ when $\mathbf{B}_\beta$ is not restricted. And we address the problem of defining substitution by augmenting the beta-reduction axioms of for $\mathcal{Q}_0$ to cover quotations and evaluations.

5.3 Axioms and Rules of Inference

The proof system for CTT_{qe} consists of the axioms for $\mathcal{Q}_0$ (the A axioms), the single rule of inference for $\mathcal{Q}_0$ named R, and a set of additional axioms (the B axioms).

5.3.1 Andrews' Proof System

Andrews' proof system for $\mathcal{Q}_0$ is complete with respect to the general models semantics for $\mathcal{Q}_0$ [3, 5502 on p. 213]. Its axioms and single rule of inference (Rule R) are the core of the proof system for CTT_{qe}. Axioms A1–3 are used without modification. Andrews' five beta-reduction axiom schemas [3, p. 213] have been replaced with a slightly modified set of axiom schemas. In particular, the syntactic variables in axiom schemas A4.1–6 range over all expressions of CTT_{qe} (but restrictions are placed on the syntactic variables in A4.5). Rule R is modified slightly to work correctly with quotations and evaluations.

A1 (Truth Values)

$$(g_{o\to o}\,T_o \wedge g_{o\to o}\,F_o) = \forall\,x_o\,.\,g_{o\to o}\,x_o.$$

A2 (Leibniz' Law)

$$x_\alpha = y_\alpha \supset h_{\alpha\to o}\,x_\alpha = h_{\alpha\to o}\,y_\alpha.$$

A3 (Extensionality)

$$(f_{\alpha\to\beta} = g_{\alpha\to\beta}) = \forall\,x_\alpha\,.\,(f_{\alpha\to\beta}\,x_\alpha = g_{\alpha\to\beta}\,x_\alpha).$$

A4 (Beta-Reduction)

1. $(\lambda\, \mathbf{x}_\alpha\, .\, \mathbf{y}_\beta)\, \mathbf{A}_\alpha = \mathbf{y}_\beta$ where $\mathbf{x}_\alpha$ and $\mathbf{y}_\beta$ are distinct.
2. $(\lambda\, \mathbf{x}_\alpha\, .\, \mathbf{x}_\alpha)\, \mathbf{A}_\alpha = \mathbf{A}_\alpha$.
3. $(\lambda\, \mathbf{x}_\alpha\, .\, \mathbf{c}_\beta)\, \mathbf{A}_\alpha = \mathbf{c}_\beta$.
4. $(\lambda\, \mathbf{x}_\alpha\, .\, (\mathbf{B}_{\beta\to\gamma}\, \mathbf{C}_\beta))\, \mathbf{A}_\alpha = ((\lambda\, \mathbf{x}_\alpha\, .\, \mathbf{B}_{\beta\to\gamma})\, \mathbf{A}_\alpha)\, ((\lambda\, \mathbf{x}_\alpha\, .\, \mathbf{C}_\beta)\, \mathbf{A}_\alpha)$.
5. $(\lambda\, \mathbf{x}_\alpha\, .\, \lambda\, \mathbf{y}_\beta\, .\, \mathbf{B}_\gamma)\, \mathbf{A}_\alpha = \lambda\, \mathbf{y}_\beta\, .\, ((\lambda\, \mathbf{x}_\alpha\, .\, \mathbf{B}_\gamma)\, \mathbf{A}_\alpha)$ where $\mathbf{x}_\alpha$ and $\mathbf{y}_\beta$ are distinct and either (1) $\mathbf{A}_\alpha$ is eval-free and $\mathbf{y}_\beta$ is not free in $\mathbf{A}_\alpha$ or (2) $\mathbf{B}_\gamma$ is eval-free and $\mathbf{x}_\alpha$ is not free in $\mathbf{B}_\gamma$.
6. $(\lambda\, \mathbf{x}_\alpha\, .\, \lambda\, \mathbf{x}_\alpha\, .\, \mathbf{B}_\beta)\, \mathbf{A}_\alpha = \lambda\, \mathbf{x}_\alpha\, .\, \mathbf{B}_\beta$.

Rule R (Equality Substitution) From $\mathbf{A}_\alpha = \mathbf{B}_\alpha$ and $\mathbf{C}_o$ infer the result of replacing one occurrence of $\mathbf{A}_\alpha$ in $\mathbf{C}_o$ by an occurrence of $\mathbf{B}_\alpha$, provided that the occurrence of $\mathbf{A}_\alpha$ in $\mathbf{C}_o$ is not within a quotation, not the first argument of a function abstraction, and not the second argument of an evaluation.

5.3.2 Axioms for Constructions

The axioms in this part (1) define the logical constants $\mathsf{is\text{-}var}_{\epsilon\to o}$, $\mathsf{is\text{-}var}^{\alpha}_{\epsilon\to o}$, $\mathsf{is\text{-}con}_{\epsilon\to o}$, $\mathsf{is\text{-}con}^{\alpha}_{\epsilon\to o}$, $\mathsf{is\text{-}expr}_{\epsilon\to o}$, and $\mathsf{is\text{-}expr}^{\alpha}_{\epsilon\to o}$, (2) specify the type of constructions as an inductive type, (3) define $\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}$ so that

$$\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}\, \ulcorner \mathbf{x}_\alpha \urcorner\, \ulcorner \mathbf{B}_\beta \urcorner$$

is provable iff $\mathbf{x}_\alpha$ is free in $\mathbf{B}_\beta$ (Lemma 6.5.3). B1.2–4, B2.2–4, B3.1-4, B3.6, and B4.6–7 are axiom schemas; the rest are individual axioms.

By virtue of the axioms given in this part, the proof system for CTT_{qe} satisfies Requirement R3.

B1 (Definitions of $\mathsf{is\text{-}var}_{\epsilon\to o}$ and $\mathsf{is\text{-}var}^{\alpha}_{\epsilon\to o}$)

1. $\mathsf{is\text{-}var}^{\alpha}_{\epsilon\to o}\, x_\epsilon \supset \mathsf{is\text{-}var}_{\epsilon\to o}\, x_\epsilon$.
2. $\mathsf{is\text{-}var}^{\alpha}_{\epsilon\to o}\, \ulcorner \mathbf{x}_\alpha \urcorner$.
3. $\neg(\mathsf{is\text{-}var}^{\alpha}_{\epsilon\to o}\, \ulcorner \mathbf{x}_\beta \urcorner)$ where $\alpha \neq \beta$.
4. $\sim(\mathsf{is\text{-}var}_{\epsilon\to o}\, \mathbf{A}_\epsilon)$ where $\mathbf{A}_\epsilon$ is a construction not of the form $\ulcorner \mathbf{x}_\alpha \urcorner$.

B2 (Definitions of $\mathsf{is\text{-}con}_{\epsilon\to o}$ and $\mathsf{is\text{-}con}^{\alpha}_{\epsilon\to o}$)

1. $\mathsf{is\text{-}con}^{\alpha}_{\epsilon\to o}\, x_\epsilon \supset \mathsf{is\text{-}con}_{\epsilon\to o}\, x_\epsilon$.
2. $\mathsf{is\text{-}con}^{\alpha}_{\epsilon\to o}\, \ulcorner \mathbf{c}_\alpha \urcorner$.
3. $\neg(\mathsf{is\text{-}con}^{\alpha}_{\epsilon\to o}\, \ulcorner \mathbf{c}_\beta \urcorner)$ where $\alpha \neq \beta$.
4. $\neg(\mathsf{is\text{-}con}_{\epsilon\to o}\, \mathbf{A}_\epsilon)$ where $\mathbf{A}_\epsilon$ is a construction not of the form $\ulcorner \mathbf{c}_\alpha \urcorner$.

B3 (Definitions of $\mathsf{is\text{-}expr}_{\epsilon\to o}$ and $\mathsf{is\text{-}expr}^{\alpha}_{\epsilon\to o}$)

1. $\mathsf{is\text{-}expr}^{\alpha}_{\epsilon\to o}\, x_\epsilon \supset \mathsf{is\text{-}expr}_{\epsilon\to o}\, x_\epsilon$.

2. $(\mathsf{is\text{-}var}^{\alpha}_{\epsilon\to o}\, x_\epsilon \vee \mathsf{is\text{-}con}^{\alpha}_{\epsilon\to o}\, x_\epsilon) \supset \mathsf{is\text{-}expr}^{\alpha}_{\epsilon\to o} x_\epsilon$.
3. $(\mathsf{is\text{-}expr}^{\alpha\to\beta}_{\epsilon\to o}\, x_\epsilon \wedge \mathsf{is\text{-}expr}^{\alpha}_{\epsilon\to o}\, y_\epsilon) \supset \mathsf{is\text{-}expr}^{\beta}_{\epsilon\to o}\, (\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\, x_\epsilon\, y_\epsilon)$.
4. $(\mathsf{is\text{-}var}^{\alpha}_{\epsilon\to o}\, x_\epsilon \wedge \mathsf{is\text{-}expr}^{\beta}_{\epsilon\to o}\, y_\epsilon) \supset \mathsf{is\text{-}expr}^{\alpha\to\beta}_{\epsilon\to o}\, (\mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}\, x_\epsilon\, y_\epsilon)$.
5. $\mathsf{is\text{-}expr}_{\epsilon\to o}\, x_\epsilon \supset \mathsf{is\text{-}expr}^{\epsilon}_{\epsilon\to o}\, (\mathsf{quo}_{\epsilon\to\epsilon}\, x_\epsilon)$.
6. $(\mathsf{is\text{-}expr}^{\alpha}_{\epsilon\to o}\, x_\epsilon \wedge \mathsf{is\text{-}expr}^{\beta}_{\epsilon\to o}\, y_\epsilon) \supset \neg(\mathsf{is\text{-}expr}_{\epsilon\to o}\, (\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\, x_\epsilon\, y_\epsilon))$
   where α is not of the form $\beta\to\gamma$.
7. $(\neg(\mathsf{is\text{-}expr}_{\epsilon\to o}\, x_\epsilon) \vee \neg(\mathsf{is\text{-}expr}_{\epsilon\to o}\, y_\epsilon)) \supset \neg(\mathsf{is\text{-}expr}_{\epsilon\to o}\, (\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\, x_\epsilon\, y_\epsilon))$.
8. $(\neg(\mathsf{is\text{-}var}_{\epsilon\to o}\, x_\epsilon) \vee \neg(\mathsf{is\text{-}expr}_{\epsilon\to o}\, y_\epsilon)) \supset \neg(\mathsf{is\text{-}expr}_{\epsilon\to o}\, (\mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}\, x_\epsilon\, y_\epsilon))$.
9. $\neg(\mathsf{is\text{-}expr}_{\epsilon\to o}\, x_\epsilon) \supset \neg(\mathsf{is\text{-}expr}_{\epsilon\to o}\, (\mathsf{quo}_{\epsilon\to\epsilon}\, x_\epsilon))$.

B4 (Constructions are Distinct)

1. $\mathsf{is\text{-}var}_{\epsilon\to o}\, x_\epsilon \supset$
   $(\neg(\mathsf{is\text{-}con}_{\epsilon\to o}\, x_\epsilon) \wedge x_\epsilon \neq \mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\, y_\epsilon\, z_\epsilon \wedge$
   $x_\epsilon \neq \mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}\, y_\epsilon\, z_\epsilon \wedge x_\epsilon \neq \mathsf{quo}_{\epsilon\to\epsilon}\, y_\epsilon)$.
2. $\mathsf{is\text{-}con}_{\epsilon\to o}\, x_\epsilon \supset$
   $(\neg(\mathsf{is\text{-}var}_{\epsilon\to o}\, x_\epsilon) \wedge x_\epsilon \neq \mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\, y_\epsilon\, z_\epsilon \wedge$
   $x_\epsilon \neq \mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}\, y_\epsilon\, z_\epsilon \wedge x_\epsilon \neq \mathsf{quo}_{\epsilon\to\epsilon}\, y_\epsilon)$.
3. $x_\epsilon = \mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\, y_\epsilon\, z_\epsilon \supset$
   $(\neg(\mathsf{is\text{-}var}_{\epsilon\to o}\, x_\epsilon) \wedge \neg(\mathsf{is\text{-}con}_{\epsilon\to o}\, x_\epsilon) \wedge$
   $x_\epsilon \neq \mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}\, u_\epsilon\, v_\epsilon \wedge x_\epsilon \neq \mathsf{quo}_{\epsilon\to\epsilon}\, u_\epsilon)$.
4. $x_\epsilon = \mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}\, y_\epsilon\, z_\epsilon \supset$
   $(\neg(\mathsf{is\text{-}var}_{\epsilon\to o}\, x_\epsilon) \wedge \neg(\mathsf{is\text{-}con}_{\epsilon\to o}\, x_\epsilon) \wedge$
   $x_\epsilon \neq \mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\, u_\epsilon\, v_\epsilon \wedge x_\epsilon \neq \mathsf{quo}_{\epsilon\to\epsilon}\, u_\epsilon)$.
5. $x_\epsilon = \mathsf{quo}_{\epsilon\to\epsilon}\, y_\epsilon \supset$
   $(\neg(\mathsf{is\text{-}var}_{\epsilon\to o}\, x_\epsilon) \wedge \neg(\mathsf{is\text{-}con}_{\epsilon\to o}\, x_\epsilon) \wedge$
   $x_\epsilon \neq \mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\, u_\epsilon\, v_\epsilon \wedge x_\epsilon \neq \mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}\, u_\epsilon\, v_\epsilon)$.
6. $\ulcorner \mathbf{x}_\alpha \urcorner \neq \ulcorner \mathbf{y}_\beta \urcorner$ where $\mathbf{x}_\alpha$ and $\mathbf{y}_\beta$ are distinct.
7. $\ulcorner \mathbf{c}_\alpha \urcorner \neq \ulcorner \mathbf{d}_\beta \urcorner$ where $\mathbf{c}_\alpha$ and $\mathbf{d}_\beta$ are distinct.
8. $\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\, x_\epsilon\, y_\epsilon = \mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\, u_\epsilon\, v_\epsilon \supset (x_\epsilon = u_\epsilon \wedge x_\epsilon = v_\epsilon)$.
9. $\mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}\, x_\epsilon\, y_\epsilon = \mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}\, u_\epsilon\, v_\epsilon \supset (x_\epsilon = u_\epsilon \wedge y_\epsilon = v_\epsilon)$.
10. $\mathsf{quo}_{\epsilon\to\epsilon}\, x_\epsilon = \mathsf{quo}_{\epsilon\to\epsilon}\, u_\epsilon \supset x_\epsilon = u_\epsilon$.

B5 (Definition of $\sqsubset_{\epsilon\to\epsilon\to o}$)

1. $x_\epsilon \sqsubset_{\epsilon\to\epsilon\to o} \mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\, x_\epsilon\, y_\epsilon$.
2. $x_\epsilon \sqsubset_{\epsilon\to\epsilon\to o} \mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\, y_\epsilon\, x_\epsilon$.
3. $x_\epsilon \sqsubset_{\epsilon\to\epsilon\to o} \mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}\, x_\epsilon\, y_\epsilon$.
4. $x_\epsilon \sqsubset_{\epsilon\to\epsilon\to o} \mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}\, y_\epsilon\, x_\epsilon$.

5. $x_\epsilon \sqsubset_{\epsilon\to\epsilon\to o} \mathsf{quo}_{\epsilon\to\epsilon}\, x_\epsilon$.
6. $(x_\epsilon \sqsubset_{\epsilon\to\epsilon\to o} y_\epsilon \wedge y_\epsilon \sqsubset_{\epsilon\to\epsilon\to o} z_\epsilon) \supset x_\epsilon \sqsubset_{\epsilon\to\epsilon\to o} z_\epsilon$.

B6 (Induction Principle for Constructions)

$$\forall\, u_\epsilon \,.\, (\forall\, v_\epsilon \,.\, (v_\epsilon \sqsubset_{\epsilon\to\epsilon\to o} u_\epsilon \supset p_{\epsilon\to o}\, v_\epsilon) \supset p_{\epsilon\to o}\, u_\epsilon) \supset \forall\, u_\epsilon \,.\, p_{\epsilon\to o}\, u_\epsilon.$$

B7 (Definition of $\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}$)

1. $\mathsf{is\text{-}var}_{\epsilon\to o}\, x_\epsilon \supset \mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}\, x_\epsilon\, x_\epsilon$.
2. $(\mathsf{is\text{-}var}_{\epsilon\to o}\, x_\epsilon \wedge \mathsf{is\text{-}var}_{\epsilon\to o}\, y_\epsilon \wedge x_\epsilon \neq y_\epsilon) \supset \neg(\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}\, x_\epsilon\, y_\epsilon)$.
3. $(\mathsf{is\text{-}var}_{\epsilon\to o}\, x_\epsilon \wedge \mathsf{is\text{-}con}_{\epsilon\to o}\, y_\epsilon \supset \neg(\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}\, x_\epsilon\, y_\epsilon)$.
4. $(\mathsf{is\text{-}var}_{\epsilon\to o}\, x_\epsilon \wedge \mathsf{is\text{-}expr}_{\epsilon\to o}\, (\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\, y_\epsilon\, z_\epsilon)) \supset$
   $\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}\, x_\epsilon\, (\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\, y_\epsilon\, z_\epsilon) \equiv$
   $(\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}\, x_\epsilon\, y_\epsilon \vee \mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}\, x_\epsilon\, z_\epsilon)$.
5. $(\mathsf{is\text{-}var}_{\epsilon\to o}\, x_\epsilon \wedge \mathsf{is\text{-}expr}_{\epsilon\to o}\, (\mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}\, y_\epsilon\, z_\epsilon) \wedge x_\epsilon \neq y_\epsilon) \supset$
   $\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}\, x_\epsilon\, (\mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}\, y_\epsilon\, z_\epsilon) \equiv \mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}\, x_\epsilon\, z_\epsilon$.
6. $\mathsf{is\text{-}expr}_{\epsilon\to o}\, (\mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}\, x_\epsilon\, y_\epsilon) \supset \neg(\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}\, x_\epsilon\, (\mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}\, x_\epsilon\, y_\epsilon))$.
7. $(\mathsf{is\text{-}var}_{\epsilon\to o}\, x_\epsilon \wedge \mathsf{is\text{-}expr}_{\epsilon\to o}\, y_\epsilon) \supset \neg(\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}\, x_\epsilon\, (\mathsf{quo}_{\epsilon\to\epsilon}\, y_\epsilon))$.
8. $(\neg(\mathsf{is\text{-}var}_{\epsilon\to o}\, x_\epsilon) \vee \neg(\mathsf{is\text{-}expr}_{\epsilon\to o}\, y_\epsilon)) \supset \neg(\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}\, x_\epsilon\, y_\epsilon)$.

5.3.3 Axioms for Quotation

The axioms for quotation are the instances of the quotation properties expressed below and the instances of the beta-reduction property for applications of the form $(\lambda\, \mathbf{x}_\alpha \,.\, \ulcorner \mathbf{B}_\beta \urcorner)\, \mathbf{A}_\alpha)$.

B8 (Properties of Quotation)

1. $\ulcorner \mathbf{F}_{\alpha\to\beta}\, \mathbf{A}_\alpha \urcorner = \mathsf{app}_{\epsilon\to\epsilon\to\epsilon} \ulcorner \mathbf{F}_{\alpha\to\beta} \urcorner \ulcorner \mathbf{A}_\alpha \urcorner$.
2. $\ulcorner \lambda\, \mathbf{x}_\alpha \,.\, \mathbf{B}_\beta \urcorner = \mathsf{abs}_{\epsilon\to\epsilon\to\epsilon} \ulcorner \mathbf{x}_\alpha \urcorner \ulcorner \mathbf{B}_\beta \urcorner$.
3. $\ulcorner \ulcorner \mathbf{A}_\alpha \urcorner \urcorner = \mathsf{quo}_{\epsilon\to\epsilon} \ulcorner \mathbf{A}_\alpha \urcorner$.

B9 (Beta-Reduction for Quotations)

$$(\lambda\, \mathbf{x}_\alpha \,.\, \ulcorner \mathbf{B}_\beta \urcorner)\, \mathbf{A}_\alpha) = \ulcorner \mathbf{B}_\beta \urcorner.$$

5.3.4 Axioms for Evaluation

The axioms for evaluation are the instances of the evaluation properties expressed below and the instances of the beta-reduction properties for a partial set of applications of the form $(\lambda\,\mathbf{x}_\alpha\,.\,[\![\mathbf{B}_\epsilon]\!]_\beta)\,\mathbf{A}_\alpha$ also expressed below.

By virtue of the axioms for the properties of evaluation and for beta-reduction for evaluations, the proof system for CTT_{qe} satisfies Requirements R5 and R4, respectively.

B10 (Properties of Evaluation)

1. $[\![\ulcorner \mathbf{x}_\alpha \urcorner]\!]_\alpha = \mathbf{x}_\alpha$.
2. $[\![\ulcorner \mathbf{c}_\alpha \urcorner]\!]_\alpha = \mathbf{c}_\alpha$.
3. $(\mathsf{is\text{-}expr}^{\alpha\to\beta}_{\epsilon\to o}\,\mathbf{A}_\epsilon \wedge \mathsf{is\text{-}expr}^{\alpha}_{\epsilon\to o}\,\mathbf{B}_\epsilon) \supset$
   $[\![\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\,\mathbf{A}_\epsilon\,\mathbf{B}_\epsilon]\!]_\beta = [\![\mathbf{A}_\epsilon]\!]_{\alpha\to\beta}\,[\![\mathbf{B}_\epsilon]\!]_\alpha$.
4. $(\mathsf{is\text{-}expr}^{\beta}_{\epsilon\to o}\,\mathbf{A}_\epsilon \wedge \neg(\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}\,\ulcorner \mathbf{x}_\alpha \urcorner\,\ulcorner \mathbf{A}_\epsilon \urcorner)) \supset$
   $[\![\mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}\,\ulcorner \mathbf{x}_\alpha \urcorner\,\mathbf{A}_\epsilon]\!]_{\alpha\to\beta} = \lambda\,\mathbf{x}_\alpha\,.\,[\![\mathbf{A}_\epsilon]\!]_\beta$.
5. $\mathsf{is\text{-}expr}^{\epsilon}_{\epsilon\to o}\,\mathbf{A}_\epsilon \supset [\![\mathsf{quo}_{\epsilon\to\epsilon}\,\mathbf{A}_\epsilon]\!]_\epsilon = \mathbf{A}_\epsilon$.

B11 (Beta-Reduction for Evaluations)

1. $(\lambda\,\mathbf{x}_\alpha\,.\,[\![\mathbf{B}_\epsilon]\!]_\beta)\,\mathbf{x}_\alpha = [\![\mathbf{B}_\epsilon]\!]_\beta$.
2. $(\mathsf{is\text{-}expr}^{\beta}_{\epsilon\to o}\,((\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_\epsilon)\,\mathbf{A}_\alpha) \wedge$
   $\neg(\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}\,\ulcorner \mathbf{x}_\alpha \urcorner\,((\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_\epsilon)\,\mathbf{A}_\alpha))) \supset$
   $(\lambda\,\mathbf{x}_\alpha\,.\,[\![\mathbf{B}_\epsilon]\!]_\beta)\,\mathbf{A}_\alpha = [\![(\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_\epsilon)\,\mathbf{A}_\alpha]\!]_\beta$.

5.3.5 Axioms involving IS-EFFECTIVE-IN

The first axiom in this part says that a variable is not effective in an eval-free expression whenever the variable is not free in the expression. The second axiom strengthens Axiom A4.5 by replacing the side conditions of the form "$\mathbf{B}_\beta$ is eval-free and $\mathbf{x}_\alpha$ is not free in $\mathbf{B}_\beta$" with conditions of the form $\neg\mathsf{IS\text{-}EFFECTIVE\text{-}IN}(\mathbf{x}_\alpha, \mathbf{B}_\beta)$ given in the axiom itself.

B12 ("Not Free In" means "Not Effective In")

$\neg\mathsf{IS\text{-}EFFECTIVE\text{-}IN}(\mathbf{x}_\alpha, \mathbf{B}_\beta)$
where $\mathbf{B}_\beta$ is eval-free and $\mathbf{x}_\alpha$ is not free in $\mathbf{B}_\beta$.

B13 (Axiom A4.5 using "Not Effective In")

$(\neg\mathsf{IS\text{-}EFFECTIVE\text{-}IN}(\mathbf{y}_\beta, \mathbf{A}_\alpha) \vee \neg\mathsf{IS\text{-}EFFECTIVE\text{-}IN}(\mathbf{x}_\alpha, \mathbf{B}_\gamma)) \supset$
$(\lambda\,\mathbf{x}_\alpha\,.\,\lambda\,\mathbf{y}_\beta\,.\,\mathbf{B}_\gamma)\,\mathbf{A}_\alpha = \lambda\,\mathbf{y}_\beta\,.\,((\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_\gamma)\,\mathbf{A}_\alpha)$
where $\mathbf{x}_\alpha$ and $\mathbf{y}_\beta$ are distinct.

5.4 Proofs

Let $T = (L, \Gamma)$ be a theory of CTT_{qe}, $\mathbf{A}_o$ be a formula in L, and $\mathcal{H}$ be a set of eval-free formulas in L. A *proof of* $\boldsymbol{A}_o$ *in* T is a finite sequence of formulas in L ending with $\mathbf{A}_o$ such that every formula in the sequence is one of the axioms of CTT_{qe}, a member of Γ, or is inferred from previous formulas in the sequence by the Rule R. $\mathbf{A}_o$ is a *theorem of* T if there is a proof of $\mathbf{A}_o$ in T. A *proof in* CTT_{qe} is a proof in the theory $T_{\text{logic}} = (L_{\mathcal{C}}, \emptyset)$, and a *theorem of* CTT_{qe} is a theorem of T_{logic}.

$\boldsymbol{A}_o$ *is provable from* $\mathcal{H}$ *in* T, written $T, \mathcal{H} \vdash \mathbf{A}_o$, is defined by the following statements:

1. If $\mathbf{A}_o \in \mathcal{H}$, then $T, \mathcal{H} \vdash \mathbf{A}_o$.
2. If $\mathbf{A}_o$ is a theorem of T, then $T, \mathcal{H} \vdash \mathbf{A}_o$.
3. **Rule R′**. If $T, \mathcal{H} \vdash \mathbf{A}_\alpha = \mathbf{B}_\alpha$; $T, \mathcal{H} \vdash \mathbf{C}_o$; and $\mathbf{D}_o$ is obtained from $\mathbf{C}_o$ by replacing one occurrence of $\mathbf{A}_\alpha$ in $\mathbf{C}_o$ by an occurrence of $\mathbf{B}_\alpha$, then $T, \mathcal{H} \vdash \mathbf{D}_o$ provided that the occurrence of $\mathbf{A}_\alpha$ in $\mathbf{C}_o$ is not within a quotation, not the first argument of a function abstraction, not the second argument of an evaluation, and in a function application $\lambda\, \mathbf{x}_\beta\, .\, \mathbf{E}_\gamma$ only if

   $$T, \mathcal{H} \vdash \neg\mathsf{IS\text{-}EFFECTIVE\text{-}IN}(\mathbf{x}_\beta, \mathbf{A}_\alpha = \mathbf{B}_\alpha)$$

   or

   $$T, \mathcal{H} \vdash \neg\mathsf{IS\text{-}EFFECTIVE\text{-}IN}(\mathbf{x}_\beta, \mathbf{H}_o)$$

   for all $\mathbf{H}_o \in \mathcal{H}$.

Notice that the variables in the members of Γ are treated as if they were universally quantified, while the variables in the members of $\mathcal{H}$ are treated as if they were constants. $T, \emptyset \vdash \mathbf{A}_o$ and $T_{\text{logic}}, \emptyset \vdash \mathbf{A}_o$ are abbreviated as $T \vdash \mathbf{A}_o$ and $\vdash \mathbf{A}_o$, respectively. Note that $T \vdash \mathbf{A}_o$ iff $\mathbf{A}_o$ is a theorem of T since Rule R′ reduces to Rule R when $\mathcal{H} = \emptyset$. T is *consistent* if not $T \vdash F_o$

6 Proof-Theoretic Results

In this section, let T be a theory of CTT_{qe} and $\mathcal{H}$ be a set of formulas of T.

6.1 Equality

Lemma 6.1.1 (Reflexivity of Equality) $\vdash \boldsymbol{A}_\alpha = \boldsymbol{A}_\alpha$.

Proof

$$\vdash (\lambda\, \mathbf{x}_\alpha\, .\, \mathbf{x}_\alpha)\, \mathbf{A}_\alpha = \mathbf{A}_\alpha \tag{1}$$

$$\vdash \mathbf{A}_\alpha = \mathbf{A}_\alpha \tag{2}$$

(1) is an instance of Axiom A4.2 and (2) follows from (1) and (1) by Rule R. □

Lemma 6.1.2 (Equality Rules)

1. *If* $T, \mathcal{H} \vdash \boldsymbol{A}_\alpha = \boldsymbol{B}_\alpha$, *then* $T, \mathcal{H} \vdash \boldsymbol{B}_\alpha = \boldsymbol{A}_\alpha$.
2. *If* $T, \mathcal{H} \vdash \boldsymbol{A}_\alpha = \boldsymbol{B}_\alpha$ *and* $T, \mathcal{H} \vdash \boldsymbol{B}_\alpha = \boldsymbol{C}_\alpha$, *then* $T, \mathcal{H} \vdash \boldsymbol{A}_\alpha = \boldsymbol{C}_\alpha$.
3. *If* $T, \mathcal{H} \vdash \boldsymbol{A}_{\alpha\to\beta} = \boldsymbol{B}_{\alpha\to\beta}$ *and* $T, \mathcal{H} \vdash \boldsymbol{C}_\alpha = \boldsymbol{D}_\alpha$, *then* $T, \mathcal{H} \vdash \boldsymbol{A}_{\alpha\to\beta}\, \boldsymbol{C}_\alpha = \boldsymbol{B}_{\alpha\to\beta}\, \boldsymbol{D}_\alpha$.
4. *If* $T, \mathcal{H} \vdash \boldsymbol{A}_\epsilon = \boldsymbol{B}_\epsilon$, *then* $T, \mathcal{H} \vdash [\![\boldsymbol{A}_\epsilon]\!]_\alpha = [\![\boldsymbol{B}_\epsilon]\!]_\alpha$.
5. *If* $T, \mathcal{H} \vdash \boldsymbol{A}_o$ *and* $T, \mathcal{H} \vdash \boldsymbol{A}_o = \boldsymbol{B}_o$, *then* $T, \mathcal{H} \vdash \boldsymbol{B}_o$.

Proof By Lemma 6.1.1 and Rule R′. □

6.2 Substitution

It will be convenient to define in the metalogic of CTT_{qe} a substitution operator named SUB. Roughly speaking, $\mathsf{SUB}(\mathbf{A}_\alpha, \mathbf{x}_\alpha, \mathbf{B}_\beta)$ denotes the expression obtained by replacing each free occurrence of $\mathbf{x}_\alpha$ in $\mathbf{B}_\beta$ with $\mathbf{A}_\alpha$ except that the substitution is curtained (1) within a function abstraction when a variable capture will occur and (2) within an evaluation when $\mathbf{A}_\alpha$ is not $\mathbf{x}_\alpha$.

The *substitution of an expression* $\boldsymbol{A}_\alpha$ *for a variable* $\boldsymbol{x}_\alpha$ *in an expression* $\boldsymbol{B}_\beta$, written $\mathsf{SUB}(\mathbf{A}_\alpha, \mathbf{x}_\alpha, \mathbf{B}_\beta)$, is defined recursively as follows:

1. $\mathsf{SUB}(\mathbf{A}_\alpha, \mathbf{x}_\alpha, \mathbf{y}_\beta) = \mathbf{y}_\beta$ where $\mathbf{x}_\alpha$ and $\mathbf{y}_\beta$ are distinct.
2. $\mathsf{SUB}(\mathbf{A}_\alpha, \mathbf{x}_\alpha, \mathbf{x}_\alpha) = \mathbf{A}_\alpha$.
3. $\mathsf{SUB}(\mathbf{A}_\alpha, \mathbf{x}_\alpha, \mathbf{c}_\beta) = \mathbf{c}_\beta$.
4. $\mathsf{SUB}(\mathbf{A}_\alpha, \mathbf{x}_\alpha, \mathbf{B}_{\beta\to\gamma}\, \mathbf{C}_\beta) = \mathsf{SUB}(\mathbf{A}_\alpha, \mathbf{x}_\alpha, \mathbf{B}_{\beta\to\gamma})\, (\mathsf{SUB}(\mathbf{A}_\alpha, \mathbf{x}_\alpha, \mathbf{C}_\beta))$.
5. $\mathsf{SUB}(\mathbf{A}_\alpha, \mathbf{x}_\alpha, \lambda\, \mathbf{y}_\beta\, .\, \mathbf{B}_\gamma) = \lambda\, \mathbf{y}_\beta\, .\, \mathsf{SUB}(\mathbf{A}_\alpha, \mathbf{x}_\alpha, \mathbf{B}_\gamma)$ where $\mathbf{x}_\alpha$ and $\mathbf{y}_\beta$ are distinct and either (1) $\mathbf{A}_\alpha$ is eval-free and $\mathbf{y}_\beta$ is not free in $\mathbf{A}_\alpha$ or (2) $\mathbf{B}_\gamma$ is eval-free and $\mathbf{x}_\alpha$ is not free in $\mathbf{B}_\gamma$.
6. $\mathsf{SUB}(\mathbf{A}_\alpha, \mathbf{x}_\alpha, \lambda\, \mathbf{y}_\beta\, .\, \mathbf{B}_\gamma) = (\lambda\, \mathbf{x}_\alpha\, .\, \lambda\, \mathbf{y}_\beta\, .\, \mathbf{B}_\gamma)\, \mathbf{A}_\alpha$ where $\mathbf{x}_\alpha$ and $\mathbf{y}_\beta$ are distinct and not (1) $\mathbf{A}_\alpha$ is eval-free and $\mathbf{y}_\beta$ is not free in $\mathbf{A}_\alpha$ and not (2) $\mathbf{B}_\gamma$ is eval-free and $\mathbf{x}_\alpha$ is not free in $\mathbf{B}_\gamma$.
7. $\mathsf{SUB}(\mathbf{A}_\alpha, \mathbf{x}_\alpha, \lambda\, \mathbf{x}_\alpha\, .\, \mathbf{B}_\beta) = \lambda\, \mathbf{x}_\alpha\, .\, \mathbf{B}_\beta$.
8. $\mathsf{SUB}(\mathbf{A}_\alpha, \mathbf{x}_\alpha, \ulcorner \mathbf{B}_\beta \urcorner) = \ulcorner \mathbf{B}_\beta \urcorner$.
9. $\mathsf{SUB}(\mathbf{x}_\alpha, \mathbf{x}_\alpha, [\![\mathbf{B}_\epsilon]\!]_\beta) = [\![\mathbf{B}_\epsilon]\!]_\beta$.
10. $\mathsf{SUB}(\mathbf{A}_\alpha, \mathbf{x}_\alpha, [\![\mathbf{B}_\epsilon]\!]_\beta) = (\lambda\, \mathbf{x}_\alpha\, .\, [\![\mathbf{B}_\epsilon]\!]_\beta)\, \mathbf{A}_\alpha$ where $\mathbf{A}_\alpha$ is not $\mathbf{x}_\alpha$.

Notice that in clauses 6 and 10, the substitution is not performed and the intent of the substitution is recorded as an appropriate lambda application.

The following theorem shows that beta-reduction can be performed (in part) by substitution via SUB.

Theorem 6.2.1 (Beta-Reduction by Substitution)

$$\vdash (\lambda\, \boldsymbol{x}_\alpha\ .\ \boldsymbol{B}_\beta)\, \boldsymbol{A}_\alpha = \mathsf{SUB}(\boldsymbol{A}_\alpha, \boldsymbol{x}_\alpha, \boldsymbol{B}_\beta).$$

Proof The proof is by induction on the structure of $\mathbf{B}_\beta$. There are six cases corresponding to the six formation rules for expressions.

Case 1: $\mathbf{B}_\beta$ is a variable.

Subcase 1a: $\mathbf{B}_\beta$ is $\mathbf{y}_\beta$ and $\mathbf{x}_\alpha$ and $\mathbf{y}_\beta$ are distinct. The theorem follows immediately from Axiom A4.1 and the definition of SUB.

Subcase 1b: $\mathbf{B}_\beta$ is $\mathbf{x}_\alpha$. The theorem follows immediately from Axiom A4.2 and the definition of SUB.

Case 2: $\mathbf{B}_\beta$ is a constant. The theorem follows immediately from Axiom A4.3 and the definition of SUB.

Case 3: $\mathbf{B}_\beta$ is a function application $\mathbf{B}_{\beta\to\gamma}\, \mathbf{C}_\beta$.

$$\vdash (\lambda\, \mathbf{x}_\alpha\ .\ (\mathbf{B}_{\beta\to\gamma}\, \mathbf{C}_\beta))\, \mathbf{A}_\alpha = ((\lambda\, \mathbf{x}_\alpha\ .\ \mathbf{B}_{\beta\to\gamma})\, \mathbf{A}_\alpha)\, ((\lambda\, \mathbf{x}_\alpha\ .\ \mathbf{C}_\beta)\, \mathbf{A}_\alpha) \quad (1)$$

$$\vdash (\lambda\, \mathbf{x}_\alpha\ .\ \mathbf{B}_{\beta\to\gamma})\, \mathbf{A}_\alpha = \mathsf{SUB}(\mathbf{A}_\alpha, \mathbf{x}_\alpha, \mathbf{B}_{\beta\to\gamma}) \quad (2)$$

$$\vdash (\lambda\, \mathbf{x}_\alpha\ .\ \mathbf{C}_\beta)\, \mathbf{A}_\alpha = \mathsf{SUB}(\mathbf{A}_\alpha, \mathbf{x}_\alpha, \mathbf{C}_\beta) \quad (3)$$

$$\vdash (\lambda\, \mathbf{x}_\alpha\ .\ (\mathbf{B}_{\beta\to\gamma}\, \mathbf{C}_\beta))\, \mathbf{A}_\alpha = \mathsf{SUB}(\mathbf{A}_\alpha, \mathbf{x}_\alpha, \mathbf{B}_{\beta\to\gamma})(\mathsf{SUB}(\mathbf{A}_\alpha, \mathbf{x}_\alpha, \mathbf{C}_\beta)) \quad (4)$$

$$\vdash (\lambda\, \mathbf{x}_\alpha\ .\ (\mathbf{B}_{\beta\to\gamma}\, \mathbf{C}_\beta))\, \mathbf{A}_\alpha = \mathsf{SUB}(\mathbf{A}_\alpha, \mathbf{x}_\alpha, \mathbf{B}_{\beta\to\gamma}\, \mathbf{C}_\beta) \quad (5)$$

(1) is by Axiom A4.4; (2) and (3) are by the induction hypothesis; (4) follows from (1), (2), and (3) by the Equality Rules; and (5), the theorem, follows from (4) by the definition of SUB.

Case 4: $\mathbf{B}_\beta$ is a function abstraction.

Subcase 4a: $\mathbf{B}_\beta$ is $\lambda\, \mathbf{y}_\beta\ .\ \mathbf{B}_\gamma$, $\mathbf{x}_\alpha$ and $\mathbf{y}_\beta$ are distinct, and either (1) $\mathbf{A}_\alpha$ is eval-free and $\mathbf{y}_\beta$ is not free in $\mathbf{A}_\alpha$ or (2) $\mathbf{B}_\gamma$ is eval-free and $\mathbf{x}_\alpha$ is not free in $\mathbf{B}_\gamma$.

$$\vdash (\lambda\, \mathbf{x}_\alpha\ .\ \lambda\, \mathbf{y}_\beta\ .\ \mathbf{B}_\gamma)\, \mathbf{A}_\alpha = \lambda\, \mathbf{y}_\beta\ .\ ((\lambda\, \mathbf{x}_\alpha\ .\ \mathbf{B}_\gamma)\, \mathbf{A}_\alpha) \quad (1)$$

$$\vdash (\lambda\, \mathbf{x}_\alpha\ .\ \mathbf{B}_\gamma)\, \mathbf{A}_\alpha = \mathsf{SUB}(\mathbf{A}_\alpha, \mathbf{x}_\alpha, \mathbf{B}_\gamma) \quad (2)$$

$$\vdash (\lambda\, \mathbf{x}_\alpha\ .\ \lambda\, \mathbf{y}_\beta\ .\ \mathbf{B}_\gamma)\, \mathbf{A}_\alpha = \lambda\, \mathbf{y}_\beta\ .\ \mathsf{SUB}(\mathbf{A}_\alpha, \mathbf{x}_\alpha, \mathbf{B}_\gamma) \quad (3)$$

$$\vdash (\lambda\, \mathbf{x}_\alpha\ .\ \lambda\, \mathbf{y}_\beta\ .\ \mathbf{B}_\gamma)\, \mathbf{A}_\alpha = \mathsf{SUB}(\mathbf{A}_\alpha, \mathbf{x}_\alpha, \lambda\, \mathbf{y}_\beta\ .\ \mathbf{B}_\gamma) \quad (4)$$

(1) is by Axiom A4.5; (2) is by the induction hypothesis; (3) follows from (1) and (2) by the Equality Rules; and (4), the theorem, follows from (3) by the definition of SUB.

Subcase 4b: $\mathbf{B}_\beta$ is $\lambda\,\mathbf{y}_\beta\,.\,\mathbf{B}_\gamma$, $\mathbf{x}_\alpha$ and $\mathbf{y}_\beta$ are distinct, not (1) $\mathbf{A}_\alpha$ is eval-free and $\mathbf{y}_\beta$ is not free in $\mathbf{A}_\alpha$, and not (2) $\mathbf{B}_\gamma$ is eval-free and $\mathbf{x}_\alpha$ is not free in $\mathbf{B}_\gamma$. The theorem follows immediately from the definition of SUB.

Subcase 4c: $\mathbf{B}_\beta$ is $\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_\beta$. The theorem follows immediately from Axiom A4.6 and the definition of SUB.

Case 5: $\mathbf{B}_\beta$ is a quotation. The theorem follows immediately from Axiom B9 and the definition of SUB.

Case 6: $\mathbf{B}_\beta$ is an evaluation.

Subcase 6a: $\mathbf{A}_\alpha$ is $\mathbf{x}_\alpha$. The theorem follows immediately from Axiom B11.1 and the definition of SUB.

Subcase 6b: $\mathbf{A}_\alpha$ is not $\mathbf{x}_\alpha$. The theorem follows immediately from the definition of SUB.

□

Theorem 6.2.2 (Universal Instantiation) *If* $T, \mathcal{H} \vdash \forall\,\boldsymbol{x}_\alpha\,.\,\boldsymbol{B}_o$, *then* $T, \mathcal{H} \vdash \mathsf{SUB}(\boldsymbol{A}_\alpha, \boldsymbol{x}_\alpha, \boldsymbol{B}_o)$.

Proof Similar to the proof of 5215 in [3, p. 221]. □

Lemma 6.2.3 *If* $\vdash \boldsymbol{B}_\beta = \boldsymbol{C}_\beta$, *then* $\vdash \mathsf{SUB}(\boldsymbol{A}_\alpha, \boldsymbol{x}_\alpha, \boldsymbol{B}_\beta = \boldsymbol{C}_\beta)$.

Proof Similar to the proof of 5204 in [3, p. 217]. □

Lemma 6.2.4 $\vdash f_{\alpha\to\beta} = \lambda\,\boldsymbol{y}_\alpha\,.\,f_{\alpha\to\beta}\,\boldsymbol{y}_\alpha$.

Proof Similar to the proof of 5205 in [3, p. 217]. □

Lemma 6.2.5 *If* $\boldsymbol{B}_\beta$ *is eval-free and* $\boldsymbol{y}_\alpha$ *is not free in* $\boldsymbol{B}_\beta$, *then* $\vdash \lambda\,\boldsymbol{x}_\alpha\,.\,\boldsymbol{B}_\beta = \lambda\,\boldsymbol{y}_\alpha\,.\,\mathsf{SUB}(\boldsymbol{y}_\alpha, \boldsymbol{x}_\alpha, \boldsymbol{B}_\beta)$.

Proof

$$\vdash f_{\alpha\to\beta} = \lambda\,\mathbf{y}_\alpha\,.\,f_{\alpha\to\beta}\,\mathbf{y}_\alpha \tag{1}$$

$$\vdash \mathsf{SUB}(\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_\beta,\ f_{\alpha\to\beta},\ f_{\alpha\to\beta} = \lambda\,\mathbf{y}_\alpha\,.\,f_{\alpha\to\beta}\,\mathbf{y}_\alpha) \tag{2}$$

$$\vdash \lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_\beta = \lambda\,\mathbf{y}_\alpha\,.\,((\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_\beta)\,\mathbf{y}_\alpha) \tag{3}$$

$$\vdash \lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_\beta = \lambda\,\mathbf{y}_\alpha\,.\,\mathsf{SUB}(\mathbf{y}_\alpha, \mathbf{x}_\alpha, \mathbf{B}_\beta) \tag{4}$$

(1) is Lemma 6.2.4; (2) follows from (1) by Lemma 6.2.3; (3) follows from (2) and the hypothesis by the definition of SUB; and (4) follows from (3) by Beta-Reduction by Substitution and the Equality Rules. □

Corollary 6.2.6 (Alpha-Equivalence) *If* $\boldsymbol{B}_\beta$ *is eval-free,* $\boldsymbol{y}_\alpha$ *is not be free in* $\boldsymbol{B}_\beta$, *and* $\boldsymbol{y}_\alpha$ *is free for* $\boldsymbol{x}_\alpha$ *in* $\boldsymbol{B}_\beta$, *then* $\lambda\,\boldsymbol{x}_\alpha\,.\,\boldsymbol{B}_\beta$ *and* $\lambda\,\boldsymbol{y}_\alpha\,.\,\mathsf{SUB}(\boldsymbol{y}_\alpha, \boldsymbol{x}_\alpha, \boldsymbol{B}_\beta)$ *are alpha-equivalent.*

Proof By Lemma 6.2.5 and the definition of SUB. □

Remark 6.2.7 (Nominal Data Types) Since alpha-conversion is not universally valid in CTT_{qe}, it is not clear whether techniques for managing variable naming and binding — such as *higher-order abstract syntax* [91, 101] and *nominal techniques* [62, 102] — are applicable to CTT_{qe}. However, the paper [94] does combine quotation/evaluation techniques with nominal techniques.

Lemma 6.2.8 $\vdash \forall \boldsymbol{x}_\alpha \,.\, \boldsymbol{B}_o \supset \mathsf{SUB}(\boldsymbol{A}_\alpha, \boldsymbol{x}_\alpha, \boldsymbol{B}_o)$.

Proof Similar to the proof of 5226 in [3, p. 224]. □

Theorem 6.2.9 (Universal Generalization) *If* $T, \mathcal{H} \vdash \boldsymbol{A}_o$, *then* $T, \mathcal{H} \vdash \forall \boldsymbol{x}_\alpha \,.\, \boldsymbol{A}_o$ *provided* $T, \mathcal{H} \vdash \neg\mathsf{IS\text{-}EFFECTIVE\text{-}IN}(\boldsymbol{x}_\alpha, \boldsymbol{H}_o)$ *for all* $\boldsymbol{H}_o \in \mathcal{H}$.

Proof Similar to the proof of 5220 in [3, p. 222]. The notion of "is effective in" is used in place of the notion of "is free in". □

Theorem 6.2.10 (Existential Generalization) *If* $T, \mathcal{H} \vdash \mathsf{SUB}(\boldsymbol{A}_\alpha, \boldsymbol{x}_\alpha, \boldsymbol{B}_o)$, *then* $T, \mathcal{H} \vdash \exists \boldsymbol{x}_\alpha \,.\, \boldsymbol{B}_o$.

Proof Similar to the proof of 5242 in [3, p. 229]. □

6.3 Propositional Reasoning

Theorem 6.3.1 (Modus Ponens) *If* $T, \mathcal{H} \vdash \boldsymbol{A}_o$ *and* $T, \mathcal{H} \vdash \boldsymbol{A}_o \supset \boldsymbol{B}_o$, *then* $T, \mathcal{H} \vdash \boldsymbol{B}_o$.

Proof Similar to the proof of 5224 in [3, p. 226]. □

Theorem 6.3.2 (Tautology Theorem) *If* $\boldsymbol{A}_o$ *is a substitution instance of a tautology, then* $\vdash \boldsymbol{A}_o$.

Proof Similar to the proof of 5234 in [3, p. 227]. □

Lemma 6.3.3 *If* $T, \mathcal{H} \vdash \neg\mathsf{IS\text{-}EFFECTIVE\text{-}IN}(\boldsymbol{x}_\alpha, \boldsymbol{A}_o)$, *then*

$$T, \mathcal{H} \vdash \forall \boldsymbol{x}_\alpha \,.\, (\boldsymbol{A}_o \vee \boldsymbol{B}_o) \supset (\boldsymbol{A}_o \vee \forall \boldsymbol{x}_\alpha \,.\, \boldsymbol{B}_o).$$

Proof Similar to the proof of 5235 in [3, p. 227] except Axiom B13 is needed in the last step of the proof. □

Lemma 6.3.4 *If* $T, \mathcal{H} \vdash \neg\mathsf{IS\text{-}EFFECTIVE\text{-}IN}(\boldsymbol{x}_\alpha, \boldsymbol{A}_o)$, *then*

$$T, \mathcal{H} \vdash (\forall \boldsymbol{x}_\alpha \,.\, \boldsymbol{A}_o) \equiv \boldsymbol{A}_o.$$

Proof

$$\vdash \neg \mathbf{A}_o \vee \mathbf{A}_o \tag{1}$$
$$\vdash \forall \mathbf{x}_\alpha \,.\, (\neg \mathbf{A}_o \vee \mathbf{A}_o) \tag{2}$$
$$T, \mathcal{H} \vdash \neg \mathbf{A}_o \vee \forall \mathbf{x}_\alpha \,.\, \mathbf{A}_o \tag{3}$$
$$T, \mathcal{H} \vdash \mathbf{A}_o \supset \forall \mathbf{x}_\alpha \,.\, \mathbf{A}_o \tag{4}$$
$$T, \mathcal{H} \vdash (\forall \mathbf{x}_\alpha \,.\, \mathbf{A}_o) \supset (\lambda \mathbf{x}_\alpha \,.\, \mathbf{A}_o)\, \mathbf{x}_\alpha \tag{5}$$
$$T, \mathcal{H} \vdash (\forall \mathbf{x}_\alpha \,.\, \mathbf{A}_o) \supset \mathbf{A}_o \tag{6}$$
$$T, \mathcal{H} \vdash (\forall \mathbf{x}_\alpha \,.\, \mathbf{A}_o) \equiv \mathbf{A}_o \tag{7}$$

(1) is an instance of the Tautology Theorem; (2) follows from (1) by Universal Generalization; (3) follows from (2), the hypothesis, and Lemma 6.3.3 by Modus Ponens; (4) follows from (3) by the Tautology Theorem; (5) follows from Lemma 6.2.8 and Beta-Reduction by Substitution by the Equality Rules; (6) follows from Beta-Reduction by Substitution and the Equality Rules if $\mathbf{A}_o$ is eval-free and by Axiom B11.1 and the Equality Rules if $\mathbf{A}_o$ is not eval-free; and (7) follows from (4) and (6) by the Tautology Theorem. □

Remark 6.3.5 Lemma 6.3.4 shows that a variable that is not effective in an expression has the same behavior with respect to universal quantification as a variable that is not free in an eval-free expression.

Theorem 6.3.6 (Deduction Theorem) *If* $T, \mathcal{H} \cup \{\boldsymbol{H}_o\} \vdash \boldsymbol{A}_o$*, then* $T, \mathcal{H} \vdash \boldsymbol{H}_o \supset \boldsymbol{A}_o$*.*

Proof Similar to the proof of 5240 in [3, p. 228]. The notion of "is effective in" is used in place of the notion of "is free in". □

Lemma 6.3.7 *If* $T, \mathcal{H} \cup \{\boldsymbol{A}_o\} \vdash \boldsymbol{B}_o$*, then* $T, \mathcal{H} \cup \{\exists\, \boldsymbol{x}_\alpha \,.\, \boldsymbol{A}_o\} \vdash \boldsymbol{B}_o$ *provided* $T, \mathcal{H} \vdash \neg\mathsf{IS\text{-}EFFECTIVE\text{-}IN}(\boldsymbol{x}_\alpha, \boldsymbol{B}_o)$ *and* $T, \mathcal{H} \vdash \neg\mathsf{IS\text{-}EFFECTIVE\text{-}IN}(\boldsymbol{x}_\alpha, \boldsymbol{H}_o)$ *for all* $\boldsymbol{H}_o \in \mathcal{H}$*.*

Proof

$$T, \mathcal{H} \cup \{\mathbf{A}_o\} \vdash \mathbf{B}_o \tag{1}$$
$$T, \mathcal{H} \cup \{\neg \mathbf{B}_o\} \vdash \neg \mathbf{A}_o \tag{2}$$
$$T, \mathcal{H} \cup \{\neg \mathbf{B}_o\} \vdash \forall \mathbf{x}_\alpha \,.\, \neg \mathbf{A}_o \tag{3}$$
$$T, \mathcal{H} \cup \{\exists \mathbf{x}_\alpha \,.\, \mathbf{A}_o\} \vdash \mathbf{B}_o \tag{4}$$

(1) is the hypothesis; (2) follows from (1) by the Deduction Theorem and propositional logic; (3) follows from (2) by Universal Generalization and the condition placed on $\mathbf{x}_\alpha$; and (4) follows from (3) by the Deduction Theorem, the definition of $\exists$, and propositional logic. □

Theorem 6.3.8 (Weakening) *Let* $\mathcal{H}'$ *be a set of formulas of* T *such that* $\mathcal{H} \subseteq \mathcal{H}'$*. If* $T, \mathcal{H} \vdash \boldsymbol{A}_o$*, then* $T, \mathcal{H}' \vdash \boldsymbol{A}_o$*.*

Proof Follows immediately from the definition of $T, \mathcal{H} \vdash \mathbf{A}_o$. □

6.4 Quotations

Theorem 6.4.1 (Syntactic Law of Quotation) $\vdash \ulcorner \boldsymbol{A}_\alpha \urcorner = \mathcal{E}(\boldsymbol{A}_\alpha)$.

Proof Let $\mathbf{A}_\alpha$ be eval-free. Our proof is by induction on the structure of $\mathbf{A}_\alpha$. There are five cases corresponding to the five formation rules for eval-free expressions.

Case 1: $\mathbf{A}_\alpha$ is a variable $\mathbf{x}_\alpha$. $\vdash \ulcorner \mathbf{x}_\alpha \urcorner = \ulcorner \mathbf{x}_\alpha \urcorner$ by Lemma 6.1.1. So $\vdash \ulcorner \mathbf{x}_\alpha \urcorner = \mathcal{E}(\mathbf{x}_\alpha)$ since $\mathcal{E}(\mathbf{x}_\alpha) = \ulcorner \mathbf{x}_\alpha \urcorner$.

Case 2: $\mathbf{A}_\alpha$ is a constant $\mathbf{c}_\alpha$. Similar to Case 1.

Case 3: $\mathbf{A}_\alpha$ is an function application $\mathbf{F}_{\beta\to\gamma}\,\mathbf{B}_\beta$.

$$\vdash \ulcorner \mathbf{F}_{\beta\to\gamma}\,\mathbf{B}_\beta \urcorner = \mathsf{app}_{\epsilon\to\epsilon\to\epsilon} \ulcorner \mathbf{F}_{\beta\to\gamma} \urcorner \ulcorner \mathbf{B}_\beta \urcorner \tag{1}$$
$$\vdash \ulcorner \mathbf{F}_{\beta\to\gamma} \urcorner = \mathcal{E}(\mathbf{F}_{\beta\to\gamma}) \tag{2}$$
$$\vdash \ulcorner \mathbf{B}_\beta \urcorner = \mathcal{E}(\mathbf{B}_\beta) \tag{3}$$
$$\vdash \ulcorner \mathbf{F}_{\beta\to\gamma}\,\mathbf{B}_\beta \urcorner = \mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\,\mathcal{E}(\mathbf{F}_{\beta\to\gamma})\,\mathcal{E}(\mathbf{B}_\beta) \tag{4}$$
$$\vdash \ulcorner \mathbf{F}_{\beta\to\gamma}\,\mathbf{B}_\beta \urcorner = \mathcal{E}(\mathbf{F}_{\beta\to\gamma}\,\mathbf{B}_\beta) \tag{5}$$

(1) is an instance of Axiom B8.1; (2) and (3) are by the induction hypothesis; (4) follows from (1), (2), and (3) by Rule R used twice; and (5) follows from (4) since $\mathcal{E}(\mathbf{F}_{\beta\to\gamma}\,\mathbf{B}_\beta) = \mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\,\mathcal{E}(\mathbf{F}_{\beta\to\gamma})\,\mathcal{E}(\mathbf{B}_\beta)$.

Case 4: $\mathbf{A}_\alpha$ is an function abstraction $(\lambda\,\mathbf{x}_\beta\,.\,\mathbf{B}_\gamma)$.

$$\vdash \ulcorner \lambda\,\mathbf{x}_\beta\,.\,\mathbf{B}_\gamma \urcorner = \mathsf{abs}_{\epsilon\to\epsilon\to\epsilon} \ulcorner \mathbf{x}_\beta \urcorner \ulcorner \mathbf{B}_\gamma \urcorner \tag{1}$$
$$\vdash \ulcorner \mathbf{B}_\gamma \urcorner = \mathcal{E}(\mathbf{B}_\gamma) \tag{2}$$
$$\vdash \ulcorner \lambda\,\mathbf{x}_\beta\,.\,\mathbf{B}_\gamma \urcorner = \mathsf{abs}_{\epsilon\to\epsilon\to\epsilon} \ulcorner \mathbf{x}_\beta \urcorner \mathcal{E}(\mathbf{B}_\gamma) \tag{3}$$
$$\vdash \ulcorner \lambda\,\mathbf{x}_\beta\,.\,\mathbf{B}_\gamma \urcorner = \mathcal{E}(\lambda\,\mathbf{x}_\beta\,.\,\mathbf{B}_\gamma) \tag{4}$$

(1) is an instance of Axiom B8.2; (2) is by the induction hypothesis; (3) follows from (2) and (1) by Rule R; and (4) follows from (3) since $\mathcal{E}(\lambda\,\mathbf{x}_\beta\,.\,\mathbf{B}_\gamma) = \mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}\,\mathcal{E}(\mathbf{x}_\beta)\,\mathcal{E}(\mathbf{B}_\gamma) = \mathsf{abs}_{\epsilon\to\epsilon\to\epsilon} \ulcorner \mathbf{x}_\beta \urcorner \mathcal{E}(\mathbf{B}_\gamma)$.

Case 5: $\mathbf{A}_\alpha$ is a quotation $\ulcorner \mathbf{B}_\beta \urcorner$.

$$\vdash \ulcorner \ulcorner \mathbf{B}_\beta \urcorner \urcorner = \mathsf{quo}_{\epsilon\to\epsilon} \ulcorner \mathbf{B}_\beta \urcorner \tag{1}$$
$$\vdash \ulcorner \mathbf{B}_\beta \urcorner = \mathcal{E}(\mathbf{B}_\beta) \tag{2}$$
$$\vdash \ulcorner \ulcorner \mathbf{B}_\beta \urcorner \urcorner = \mathsf{quo}_{\epsilon\to\epsilon}\,\mathcal{E}(\mathbf{B}_\beta) \tag{3}$$
$$\vdash \ulcorner \ulcorner \mathbf{B}_\beta \urcorner \urcorner = \mathcal{E}(\ulcorner \mathbf{B}_\beta \urcorner) \tag{4}$$

(1) is an instance of Axiom B8.3; (2) is by the induction hypothesis; (3) follows from (2) and (1) by Rule R; and (4) follows from (3) since $\mathcal{E}(\ulcorner \mathbf{B}_\beta \urcorner) = \mathsf{quo}_{\epsilon\to\epsilon}\,\mathcal{E}(\mathbf{B}_\beta)$.

□

6.5 Constructions

Lemma 6.5.1 $\vdash \mathsf{is\text{-}expr}^{\alpha}_{\epsilon\to o}\ulcorner \boldsymbol{A}_{\alpha}\urcorner$.

Proof By induction on the structure of $\mathbf{A}_{\alpha}$ using Axioms B1.1–4, B2.1–4, and B3.1–9, Universal Generalization, Universal Instantiation, the Syntactic Law of Quotation, and propositional logic. □

Lemma 6.5.2 $\vdash \ulcorner \boldsymbol{A}_{\alpha}\urcorner \sqsubset_{\epsilon\to\epsilon\to o} \ulcorner \boldsymbol{B}_{\beta}\urcorner$ *iff* $\boldsymbol{A}_{\alpha}$ *is a proper subexpression of* $\boldsymbol{B}_{\beta}$.

Proof By induction on the structure of $\mathbf{B}_{\beta}$ using Axioms B5.1–6, Universal Generalization, Universal Instantiation, the Syntactic Law of Quotation, and propositional logic. □

Lemma 6.5.3 $\vdash \mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}\ulcorner \boldsymbol{x}_{\alpha}\urcorner\ulcorner \boldsymbol{B}_{\beta}\urcorner$ *iff* $\boldsymbol{x}_{\alpha}$ *is free in* $\boldsymbol{B}_{\beta}$.

Proof By induction on the structure of $\mathbf{B}_{\beta}$ using Axioms B1.1–4, B2.1–4, B7.1–8, Lemma 6.5.1, Universal Generalization, Universal Instantiation, the Syntactic Law of Quotation, and propositional logic. □

6.6 Evaluations

Theorem 6.6.1 (Syntactic Law of Disquotation) $\vdash [\![\ulcorner \boldsymbol{A}_{\alpha}\urcorner]\!]_{\alpha} = \boldsymbol{A}_{\alpha}$.

Proof Let $\mathbf{A}_{\alpha}$ be eval-free. Our proof is by induction on the structure of $\mathbf{A}_{\alpha}$. There are five cases corresponding to the five formation rules for eval-free expressions.

Case 1: $\mathbf{A}_{\alpha}$ is a variable $\mathbf{x}_{\alpha}$. $\vdash [\![\ulcorner \mathbf{x}_{\alpha}\urcorner]\!]_{\alpha} = \mathbf{x}_{\alpha}$ by Axiom B10.1.

Case 2: $\mathbf{A}_{\alpha}$ is a constant $\mathbf{c}_{\alpha}$. $\vdash [\![\ulcorner \mathbf{c}_{\alpha}\urcorner]\!]_{\alpha} = \mathbf{c}_{\alpha}$ by Axiom B10.2.

Case 3: $\mathbf{A}_{\alpha}$ is a function application $\mathbf{F}_{\beta\to\gamma}\,\mathbf{B}_{\beta}$.

$$\vdash \mathsf{is\text{-}expr}^{\beta\to\gamma}_{\epsilon\to o}\ulcorner \mathbf{F}_{\beta\to\gamma}\urcorner \tag{1}$$

$$\vdash \mathsf{is\text{-}expr}^{\beta}_{\epsilon\to o}\ulcorner \mathbf{B}_{\beta}\urcorner \tag{2}$$

$$\vdash [\![\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\ulcorner \mathbf{F}_{\beta\to\gamma}\urcorner\ulcorner \mathbf{B}_{\beta}\urcorner]\!]_{\gamma} = [\![\ulcorner \mathbf{F}_{\beta\to\gamma}\urcorner]\!]_{\beta\to\gamma}\,[\![\ulcorner \mathbf{B}_{\beta}\urcorner]\!]_{\beta} \tag{3}$$

$$\vdash [\![\ulcorner \mathbf{F}_{\beta\to\gamma}\urcorner]\!]_{\beta\to\gamma} = \mathbf{F}_{\beta\to\gamma} \tag{4}$$

$$\vdash [\![\ulcorner \mathbf{B}_{\beta}\urcorner]\!]_{\beta} = \mathbf{B}_{\beta} \tag{5}$$

$$\vdash \ulcorner \mathbf{F}_{\beta\to\gamma}\,\mathbf{B}_{\beta}\urcorner = \mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\ulcorner \mathbf{F}_{\beta\to\gamma}\urcorner\ulcorner \mathbf{B}_{\beta}\urcorner \tag{6}$$

$$\vdash [\![\ulcorner \mathbf{F}_{\beta\to\gamma}\,\mathbf{B}_{\beta}\urcorner]\!]_{\gamma} = \mathbf{F}_{\beta\to\gamma}\,\mathbf{B}_{\beta} \tag{7}$$

(1) and (2) are by Lemma 6.5.1; (3) follows from (1) and (2) by Axiom B10.3; (4) and (5) are by the induction hypothesis; (6) is an instance of Axiom B8.1; and (7) follows from (3)–(6) by the Equality Rules.

Case 4: $\mathbf{A}_\alpha$ is an function abstraction $\lambda\,\mathbf{x}_\beta\,.\,\mathbf{B}_\gamma$.

$$\vdash \mathsf{is\text{-}expr}^{\gamma}_{\epsilon\to o}\ulcorner\mathbf{B}_\gamma\urcorner \tag{1}$$
$$\vdash \neg(\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}\ulcorner\mathbf{x}_\beta\urcorner\ulcorner\ulcorner\mathbf{B}_\gamma\urcorner\urcorner) \tag{2}$$
$$\vdash [\![\mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}\ulcorner\mathbf{x}_\beta\urcorner\ulcorner\mathbf{B}_\gamma\urcorner]\!]_{\beta\to\gamma} = \lambda\,\mathbf{x}_\beta\,.\,[\![\ulcorner\mathbf{B}_\gamma\urcorner]\!]_\gamma \tag{3}$$
$$\vdash [\![\ulcorner\mathbf{B}_\gamma\urcorner]\!]_\gamma = \mathbf{B}_\gamma \tag{4}$$
$$\vdash \ulcorner\lambda\,\mathbf{x}_\beta\,.\,\mathbf{B}_\gamma\urcorner = \mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}\ulcorner\mathbf{x}_\beta\urcorner\ulcorner\mathbf{B}_\gamma\urcorner \tag{5}$$
$$\vdash [\![\ulcorner\lambda\,\mathbf{x}_\beta\,.\,\mathbf{B}_\gamma\urcorner]\!]_{\beta\to\gamma} = \lambda\,\mathbf{x}_\beta\,.\,\mathbf{B}_\gamma \tag{6}$$

(1) is by Lemma 6.5.1; (2) is by Lemma 6.5.3; (3) follows from (1) and (2) by Axiom B10.4; (4) is by the induction hypothesis; (5) is an instance of Axiom B8.2; and (6) follows from (3)–(5) by the Equality Rules.

Case 5: $\mathbf{A}_\alpha$ is a quotation $\ulcorner\mathbf{B}_\beta\urcorner$.

$$\vdash \mathsf{is\text{-}expr}^{\beta}_{\epsilon\to o}\ulcorner\mathbf{B}_\beta\urcorner \tag{1}$$
$$\vdash [\![\mathsf{quo}_{\epsilon\to\epsilon}\ulcorner\mathbf{B}_\beta\urcorner]\!]_\epsilon = \ulcorner\mathbf{B}_\beta\urcorner \tag{2}$$
$$\vdash \ulcorner\ulcorner\mathbf{B}_\beta\urcorner\urcorner = \mathsf{quo}_{\epsilon\to\epsilon}\ulcorner\mathbf{B}_\beta\urcorner \tag{3}$$
$$\vdash [\![\ulcorner\ulcorner\mathbf{B}_\beta\urcorner\urcorner]\!]_\epsilon = \ulcorner\mathbf{B}_\beta\urcorner \tag{4}$$

(1) is by Lemma 6.5.1; (2) follows from (1) by Axiom B10.5; (3) is an instance of Axiom B8.3; and (4) follows from (2) and (3) by the Equality Rules.

□

The next lemma, which illustrates the application of Axiom B11.2, will be employed in section 9.

Lemma 6.6.2

1. *If* $\boldsymbol{A}_\alpha$ *is eval-free and* $\boldsymbol{x}_\epsilon$ *is not free in* $\boldsymbol{A}_\alpha$*, then*

$$\vdash (\lambda\,\boldsymbol{x}_\epsilon\,.\,[\![\boldsymbol{x}_\epsilon]\!]_\alpha)\ulcorner\boldsymbol{A}_\alpha\urcorner = \boldsymbol{A}_\alpha.$$

2. *If* $T,\mathcal{H}\vdash \mathsf{is\text{-}expr}^{\alpha}_{\epsilon\to o}\,\boldsymbol{B}_\epsilon$ *and* $T,\mathcal{H}\vdash \neg\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}\ulcorner\boldsymbol{x}_\epsilon\urcorner\,\boldsymbol{B}_\epsilon$*, then*

$$T,\mathcal{H}\vdash (\lambda\,\boldsymbol{x}_\epsilon\,.\,[\![\boldsymbol{x}_\epsilon]\!]_\alpha)\,\boldsymbol{B}_\epsilon = [\![\boldsymbol{B}_\epsilon]\!]_\alpha.$$

3. *If* $T,\mathcal{H}\vdash \mathsf{is\text{-}expr}^{\alpha}_{\epsilon\to o}\,\boldsymbol{y}_\epsilon$*,* $T,\mathcal{H}\vdash \neg\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}\ulcorner\boldsymbol{x}_\epsilon\urcorner\,\boldsymbol{y}_\epsilon$*, and* $\boldsymbol{x}_\epsilon$ *and* $\boldsymbol{y}_\epsilon$ *are different variables, then*

$$T,\mathcal{H}\vdash (\lambda\,\boldsymbol{x}_\epsilon\,.\,[\![\boldsymbol{y}_\epsilon]\!]_\alpha)\,\boldsymbol{B}_\epsilon = [\![\boldsymbol{y}_\epsilon]\!]_\alpha.$$

4. *Let* $\boldsymbol{B}_\epsilon$ *be eval-free;* $T, \mathcal{H} \vdash \neg\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}\ulcorner\boldsymbol{x}_\epsilon\urcorner\ulcorner\boldsymbol{B}_\epsilon\urcorner$*;* $T, \mathcal{H} \vdash \mathsf{is\text{-}expr}^{\alpha}_{\epsilon\to o}\,\boldsymbol{B}_\epsilon$*; and* $T, \mathcal{H} \vdash \neg\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}\ulcorner\boldsymbol{x}_\epsilon\urcorner\,\boldsymbol{B}_\epsilon$*. Also assume that there is some variable* $\boldsymbol{y}_\epsilon$ *different from* $\boldsymbol{x}_\epsilon$ *such that*

$$T, \mathcal{H} \vdash \neg\mathsf{IS\text{-}EFFECTIVE\text{-}IN}(\boldsymbol{y}_\epsilon, \boldsymbol{H}_o)$$

for all $\boldsymbol{H}_o \in \mathcal{H}$*. Then*

$$T, \mathcal{H} \vdash \neg\mathsf{IS\text{-}EFFECTIVE\text{-}IN}(\boldsymbol{x}_\epsilon, [\![\boldsymbol{B}_\epsilon]\!]_\alpha).$$

Proof

Part 1

$$\vdash (\lambda\,\mathbf{x}_\epsilon\,.\,\mathbf{x}_\epsilon)\ulcorner\mathbf{A}_\alpha\urcorner = \ulcorner\mathbf{A}_\alpha\urcorner \tag{1}$$
$$\vdash \mathsf{is\text{-}expr}^{\alpha}_{\epsilon\to o}\,((\lambda\,\mathbf{x}_\epsilon\,.\,\mathbf{x}_\epsilon)\ulcorner\mathbf{A}_\alpha\urcorner) \tag{2}$$
$$\vdash \neg\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}\ulcorner\mathbf{x}_\epsilon\urcorner((\lambda\,\mathbf{x}_\epsilon\,.\,\mathbf{x}_\epsilon)\ulcorner\mathbf{A}_\alpha\urcorner) \tag{3}$$
$$\vdash (\lambda\,\mathbf{x}_\epsilon\,.\,[\![\mathbf{x}_\epsilon]\!]_\alpha)\ulcorner\mathbf{A}_\alpha\urcorner = [\![(\lambda\,\mathbf{x}_\epsilon\,.\,\mathbf{x}_\epsilon)\ulcorner\mathbf{A}_\alpha\urcorner]\!]_\alpha \tag{4}$$
$$\vdash (\lambda\,\mathbf{x}_\epsilon\,.\,[\![\mathbf{x}_\epsilon]\!]_\alpha)\ulcorner\mathbf{A}_\alpha\urcorner = \mathbf{A}_\alpha \tag{5}$$

(1) is an instance of Axiom A4.2; (2) follows from (1) by Lemma 6.5.1 and the Equality Rules; (3) follows from (1) by Lemma 6.5.3, the hypothesis, and the Equality Rules; (4) follows from (2), (3), and Axiom B11.2 by Modus Ponens; (5) follows from (1) and (4) by the Equality Rules and the Syntactic Law of Disquotation.

Part 2

$$\vdash (\lambda\,\mathbf{x}_\epsilon\,.\,\mathbf{x}_\epsilon)\,\mathbf{B}_\epsilon = \mathbf{B}_\epsilon \tag{1}$$
$$T, \mathcal{H} \vdash \mathsf{is\text{-}expr}^{\alpha}_{\epsilon\to o}\,((\lambda\,\mathbf{x}_\epsilon\,.\,\mathbf{x}_\epsilon)\,\mathbf{B}_\epsilon) \tag{2}$$
$$T, \mathcal{H} \vdash \neg\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}\ulcorner\mathbf{x}_\epsilon\urcorner((\lambda\,\mathbf{x}_\epsilon\,.\,\mathbf{x}_\epsilon)\,\mathbf{B}_\epsilon) \tag{3}$$
$$T, \mathcal{H} \vdash (\lambda\,\mathbf{x}_\epsilon\,.\,[\![\mathbf{x}_\epsilon]\!]_\alpha)\,\mathbf{B}_\epsilon = [\![(\lambda\,\mathbf{x}_\epsilon\,.\,\mathbf{x}_\epsilon)\,\mathbf{B}_\epsilon]\!]_\alpha \tag{4}$$
$$T, \mathcal{H} \vdash (\lambda\,\mathbf{x}_\epsilon\,.\,[\![\mathbf{x}_\epsilon]\!]_\alpha)\,\mathbf{B}_\epsilon = [\![\mathbf{B}_\epsilon]\!]_\alpha \tag{5}$$

(1) is an instance of Axiom A4.2; (2) and (3) follow from (1) and the two hypotheses by the Equality Rules; (4) follows from (2), (3), and Axiom B11.2 by Modus Ponens; (5) follows from (1) and (4) by the Equality Rules.

Part 3 Similar to the proof of part 2.

Part 4 By the definitions of IS-EFFECTIVE-IN and $\exists$ and propositional logic, we must show

$$T, \mathcal{H} \vdash \forall\,\mathbf{y}_\epsilon\,.\,((\lambda\,\mathbf{x}_\epsilon\,.\,[\![\mathbf{B}_\epsilon]\!]_\alpha)\,\mathbf{y}_\epsilon = [\![\mathbf{B}_\epsilon]\!]_\alpha).$$

$$T, \mathcal{H} \vdash (\lambda\, \mathbf{x}_\epsilon \,.\, \mathbf{B}_\epsilon)\, \mathbf{y}_\epsilon = \mathbf{B}_\epsilon \tag{1}$$

$$T, \mathcal{H} \vdash \mathsf{is\text{-}expr}^{\alpha}_{\epsilon\to o}\,((\lambda\, \mathbf{x}_\epsilon \,.\, \mathbf{B}_\epsilon)\, \mathbf{y}_\epsilon) \tag{2}$$

$$T, \mathcal{H} \vdash \neg\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}\ulcorner \mathbf{x}_\epsilon \urcorner((\lambda\, \mathbf{x}_\epsilon \,.\, \mathbf{B}_\epsilon)\, \mathbf{y}_\epsilon) \tag{3}$$

$$T, \mathcal{H} \vdash (\lambda\, \mathbf{x}_\epsilon \,.\, [\![\mathbf{B}_\epsilon]\!]_\alpha)\, \mathbf{y}_\epsilon = [\![(\lambda\, \mathbf{x}_\epsilon \,.\, \mathbf{B}_\epsilon)\, \mathbf{y}_\epsilon]\!]_\alpha \tag{4}$$

$$T, \mathcal{H} \vdash (\lambda\, \mathbf{x}_\epsilon \,.\, [\![\mathbf{B}_\epsilon]\!]_\alpha)\, \mathbf{y}_\epsilon = [\![\mathbf{B}_\epsilon]\!]_\alpha \tag{5}$$

$$T, \mathcal{H} \vdash \forall\, \mathbf{y}_\epsilon \,.\, ((\lambda\, \mathbf{x}_\epsilon \,.\, [\![\mathbf{B}_\epsilon]\!]_\alpha)\, \mathbf{y}_\epsilon = [\![\mathbf{B}_\epsilon]\!]_\alpha) \tag{6}$$

(1) follows from the second hypothesis by Beta-Reduction by Substitution and Lemma 6.5.3; (2) and (3) follow from (1) and the third and fourth hypotheses by the Equality Rules; (4) follows from (2), (3), and Axiom B11.2 by Modus Ponens; (5) follows from (1) and (4) by the Equality Rules; and (6) follows from (5) and the hypothesis about $\mathbf{y}_\epsilon$ by Universal Generalization. □

7 Soundness

The proof system for CTT_{qe} is *sound* if $T \vdash \mathbf{A}_o$ implies $T \models \mathbf{A}_o$ whenever T is a theory of CTT_{qe} and $\mathbf{A}_o$ is a formula of T. We will prove that the proof system for CTT_{qe} is sound by showing that its axioms are valid in all general models for CTT_{qe} and its single rule of inference preserves validity in all general models for CTT_{qe}.

7.1 Lemmas Concerning Semantics

Lemma 7.1.1 *Let $\mathcal{M} = (\{D_\alpha \mid \alpha \in \mathcal{T}\}, I)$ be a general model for CTT_{qe} and $\varphi \in \mathsf{assign}(\mathcal{M})$.*

1. $V^{\mathcal{M}}_{\varphi}(\boldsymbol{A}_\alpha = \boldsymbol{B}_\alpha) = \text{T}$ *iff* $V^{\mathcal{M}}_{\varphi}(\boldsymbol{A}_\alpha) = V^{\mathcal{M}}_{\varphi}(\boldsymbol{B}_\alpha)$.
2. $V^{\mathcal{M}}_{\varphi}((\lambda\, \boldsymbol{x}_\alpha \,.\, \boldsymbol{B}_\beta)\, \boldsymbol{A}_\alpha) = V^{\mathcal{M}}_{\varphi[\mathbf{x}_\alpha \mapsto V^{\mathcal{M}}_{\varphi}(\mathbf{A}_\alpha)]}(\boldsymbol{B}_\beta)$.
3. $V^{\mathcal{M}}_{\varphi}(\lambda\, \boldsymbol{x}_\alpha \,.\, \boldsymbol{B}_\beta) = V^{\mathcal{M}}_{\varphi[\mathbf{x}_\alpha \mapsto d]}(\lambda\, \boldsymbol{x}_\alpha \,.\, \boldsymbol{B}_\beta)$ *for all* $d \in D_\alpha$.
4. $V^{\mathcal{M}}_{\varphi}(\forall\, \boldsymbol{x}_\alpha \,.\, \boldsymbol{A}_o) = \text{T}$ *iff* $V^{\mathcal{M}}_{\varphi[\mathbf{x}_\alpha \mapsto d]}(\boldsymbol{A}_o) = \text{T}$ *for all* $d \in D_\alpha$.

Proof Part 1 is by the semantics of equality; part 2 is by the semantics of function application and abstraction; part 3 is by the semantics of function abstraction; and part 4 is by parts 1 and 3 and the definition of $\forall$. □

Lemma 7.1.2 *Let $\mathcal{M}$ be a general model for CTT_{qe}, $\boldsymbol{A}_\alpha$ be eval-free, and $\varphi, \psi \in \mathsf{assign}(\mathcal{M})$ agree on all free variables of $\boldsymbol{A}_\alpha$. Then $V^{\mathcal{M}}_{\varphi}(\boldsymbol{A}_\alpha) = V^{\mathcal{M}}_{\psi}(\boldsymbol{A}_\alpha)$.*

Proof By induction on the structure of $\mathbf{A}_\alpha$. □

Lemma 7.1.3 *Let $\mathcal{M} = (\{D_\alpha \mid \alpha \in \mathcal{T}\}, I)$ be a general model for CTT_{qe}. $\mathcal{M} \models \neg\mathsf{IS\text{-}EFFECTIVE\text{-}IN}(\boldsymbol{x}_\alpha, \boldsymbol{B}_\beta)$ iff $V^{\mathcal{M}}_{\varphi[\mathbf{x}_\alpha \mapsto d_1]}(\boldsymbol{B}_\beta) = V^{\mathcal{M}}_{\varphi[\mathbf{x}_\alpha \mapsto d_2]}(\boldsymbol{B}_\beta)$ for all $\varphi \in \mathsf{assign}(\mathcal{M})$ and all $d_1, d_2 \in D_\alpha$ with $d_1 \neq d_2$.*

Proof Assume $\mathbf{x}_\alpha$ and $\mathbf{y}_\alpha$ are different variables.

$$\mathcal{M} \models \neg\mathsf{IS\text{-}EFFECTIVE\text{-}IN}(\mathbf{x}_\alpha, \mathbf{B}_\beta) \tag{1}$$
$$\mathcal{M} \models \neg(\exists\, \mathbf{y}_\alpha \,.\, ((\lambda\, \mathbf{x}_\alpha \,.\, \mathbf{B}_\beta)\, \mathbf{y}_\alpha \neq \mathbf{B}_\beta)) \tag{2}$$
$$\mathcal{M} \models \forall\, \mathbf{y}_\alpha \,.\, ((\lambda\, \mathbf{x}_\alpha \,.\, \mathbf{B}_\beta)\, \mathbf{y}_\alpha = \mathbf{B}_\beta) \tag{3}$$
$$V_\varphi^{\mathcal{M}}(\forall\, \mathbf{y}_\alpha \,.\, ((\lambda\, \mathbf{x}_\alpha \,.\, \mathbf{B}_\beta)\, \mathbf{y}_\alpha = \mathbf{B}_\beta)) = \mathrm{T} \text{ for all } \varphi \in \mathsf{assign}(\mathcal{M}) \tag{4}$$
$$V_\varphi^{\mathcal{M}}((\lambda\, \mathbf{x}_\alpha \,.\, \mathbf{B}_\beta)\, \mathbf{y}_\alpha) = V_\varphi^{\mathcal{M}}(\mathbf{B}_\beta) \text{ for all } \varphi \in \mathsf{assign}(\mathcal{M}) \tag{5}$$
$$V_{\varphi[\mathbf{x}_\alpha \mapsto V_\varphi^{\mathcal{M}}(\mathbf{y}_\alpha)]}^{\mathcal{M}}(\mathbf{B}_\beta) = V_\varphi^{\mathcal{M}}(\mathbf{B}_\beta) \text{ for all } \varphi \in \mathsf{assign}(\mathcal{M}) \tag{6}$$

(1) is the left side of the iff statement of the lemma; (2) holds iff (1) holds by the definition of IS-EFFECTIVE-IN; (3) holds iff (2) holds by the definition of $\exists$; (4) holds iff (3) holds by the definition of a valid formula; (5) holds iff (4) holds by parts 1 and 4 of Lemma 7.1.1; and (6) holds iff (5) holds by part 2 of Lemma 7.1.1.

Now let $\psi \in \mathsf{assign}(\mathcal{M})$ and $d_1, d_2 \in D_\alpha$ with $d_1 \neq d_1$.

$$V_{\psi[\mathbf{x}_\alpha \mapsto d_1]}^{\mathcal{M}}(\mathbf{B}_\beta) \tag{7}$$
$$= V_{\psi[\mathbf{x}_\alpha \mapsto d_1][\mathbf{x}_\alpha \mapsto V_\psi^{\mathcal{M}}(\mathbf{y}_\alpha)]}^{\mathcal{M}}(\mathbf{B}_\beta) \tag{8}$$
$$= V_{\psi[\mathbf{x}_\alpha \mapsto d_2][\mathbf{x}_\alpha \mapsto V_\psi^{\mathcal{M}}(\mathbf{y}_\alpha)]}^{\mathcal{M}}(\mathbf{B}_\beta) \tag{9}$$
$$= V_{\psi[\mathbf{x}_\alpha \mapsto d_2]}^{\mathcal{M}}(\mathbf{B}_\beta) \tag{10}$$

(8) and (10) are by (6) and (9) holds since

$$\psi[\mathbf{x}_\alpha \mapsto d][\mathbf{x}_\alpha \mapsto V_\psi^{\mathcal{M}}(\mathbf{y}_\alpha)] = \psi[\mathbf{x}_\alpha \mapsto V_\psi^{\mathcal{M}}(\mathbf{y}_\alpha)]$$

for any $d \in D_\alpha$. Hence (6) implies

$$V_{\varphi[\mathbf{x}_\alpha \mapsto d_1]}^{\mathcal{M}}(\mathbf{B}_\beta) = V_{\varphi[\mathbf{x}_\alpha \mapsto d_2]}^{\mathcal{M}}(\mathbf{B}_\beta) \tag{11}$$

for all $\varphi \in \mathsf{assign}(\mathcal{M})$ and $d_1, d_2 \in D_\alpha$ with $d_1 \neq d_1$, and obviously (11) implies (6). Therefore, (1) holds iff (11) holds. □

Lemma 7.1.4 *Let $\mathcal{M}$ be a general model for* $\mathrm{CTT}_{\mathrm{qe}}$ *and $\varphi \in \mathsf{assign}(\mathcal{M})$.*

1. $V_\varphi^{\mathcal{M}}(\mathsf{is\text{-}expr}_{\epsilon\to o}^{\alpha}\, \mathcal{E}(\boldsymbol{A}_\alpha)) = \mathrm{T}$.
2. $V_\varphi^{\mathcal{M}}(\mathsf{is\text{-}expr}_{\epsilon\to o}^{\alpha}\, \ulcorner \boldsymbol{A}_\alpha \urcorner) = \mathrm{T}$.
3. $V_\varphi^{\mathcal{M}}(\mathsf{is\text{-}expr}_{\epsilon\to o}^{\alpha}\, \boldsymbol{B}_\epsilon) = \mathrm{T}$ *implies* $V_\varphi^{\mathcal{M}}(\boldsymbol{B}_\epsilon) = V_\varphi^{\mathcal{M}}(\mathcal{E}(\boldsymbol{A}_\alpha))$ *for some (eval-free)* $\boldsymbol{A}_\alpha$.

Proof

Part 1 $V_\varphi^{\mathcal{M}}(\mathcal{E}(\mathbf{A}_\alpha)) = \mathcal{E}(\mathbf{A}_\alpha)$ by Proposition 3.3.4. This implies $V_\varphi^{\mathcal{M}}(\mathsf{is\text{-}expr}_{\epsilon\to o}^{\alpha}\, \mathcal{E}(\mathbf{A}_\alpha)) = \mathrm{T}$ by the semantics of $\mathsf{is\text{-}expr}_{\epsilon\to o}^{\alpha}$.

Part 2 $V_\varphi^{\mathcal{M}}(\ulcorner \mathbf{A}_\alpha \urcorner) = \mathcal{E}(\mathbf{A}_\alpha)$ by condition 5 of the definition of a general model. This implies $V_\varphi^{\mathcal{M}}(\mathsf{is\text{-}expr}^{\alpha}_{\epsilon\to o} \ulcorner \mathbf{A}_\alpha \urcorner) = \mathrm{T}$ by the semantics of $\mathsf{is\text{-}expr}^{\alpha}_{\epsilon\to o}$.

Part 3 By the hypothesis and the semantics of $\mathsf{is\text{-}expr}^{\alpha}_{\epsilon\to o}$, $V_\varphi^{\mathcal{M}}(\mathbf{B}_\epsilon) = \mathcal{E}(\mathbf{A}_\alpha)$ for some $\mathbf{A}_\alpha$. This implies the conclusion of the lemma by Proposition 3.3.4. □

Lemma 7.1.5 *Let $\mathcal{M}$ be a general model for* $\mathrm{CTT}_{\mathrm{qe}}$, $\varphi \in \mathsf{assign}(\mathcal{M})$. *Then* $V_\varphi^{\mathcal{M}}(\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o} \ulcorner \boldsymbol{x}_\alpha \urcorner \ulcorner \boldsymbol{B}_\beta \urcorner) = \mathrm{T}$ *iff* $\boldsymbol{x}_\alpha$ *is free in* $\boldsymbol{B}_\beta$.

Proof By induction on the structure of $\mathbf{B}_\beta$. □

Lemma 7.1.6 *Let $\mathcal{M}$ be a general model for* $\mathrm{CTT}_{\mathrm{qe}}$ *and* $\varphi \in \mathsf{assign}(\mathcal{M})$. *Then* $V_\varphi^{\mathcal{M}}([\![\mathcal{E}(\boldsymbol{A}_\alpha)]\!]_\alpha) = V_\varphi^{\mathcal{M}}(\boldsymbol{A}_\alpha)$.

Proof

$$
\begin{aligned}
& V_\varphi^{\mathcal{M}}([\![\mathcal{E}(\mathbf{A}_\alpha)]\!]_\alpha) && (1)\\
&= V_\varphi^{\mathcal{M}}(\mathcal{E}^{-1}(V_\varphi^{\mathcal{M}}(\mathcal{E}(\mathbf{A}_\alpha)))) && (2)\\
&= V_\varphi^{\mathcal{M}}(\mathcal{E}^{-1}(\mathcal{E}(\mathbf{A}_\alpha))) && (3)\\
&= V_\varphi^{\mathcal{M}}(\mathbf{A}_\alpha) && (4)
\end{aligned}
$$

(2) is by part 1 of Lemma 7.1.4 and condition 6 of the definition of a general model; (3) is by Proposition 3.3.4; and (4) is immediate. □

7.2 Soundness of Axioms and Rule of Inference

Lemma 7.2.1 (Axioms are Valid) *Each axiom of the proof system for* $\mathrm{CTT}_{\mathrm{qe}}$ *is valid in* $\mathrm{CTT}_{\mathrm{qe}}$.

Proof Let $\mathcal{M} = (\{D_\alpha \mid \alpha \in \mathcal{T}\}, I)$ be a general model for $\mathrm{CTT}_{\mathrm{qe}}$ and $\varphi \in \mathsf{assign}(\mathcal{M})$. We must prove that each of the 64 axioms is valid in $\mathcal{M}$.

Axiom A1 The proof is the same as the proof of 5402 for Axiom 1 in [3, p. 241].

Axiom A2 The proof is the same as the proof of 5402 for Axiom 2 in [3, p. 242].

Axiom A3 The proof is the same as the proof of 5402 for Axiom 3 in [3, p. 242].

Axiom Group A4

Axiom A4.1 Let $\mathbf{x}_\alpha$ and $\mathbf{y}_\beta$ be distinct. We must show

$$V_\varphi^{\mathcal{M}}((\lambda\, \mathbf{x}_\alpha\, .\, \mathbf{y}_\beta)\, \mathbf{A}_\alpha) = V_\varphi^{\mathcal{M}}(\mathbf{y}_\beta)$$

to prove Axiom A4.1 is valid in $\mathcal{M}$.

$$
\begin{aligned}
& V_\varphi^{\mathcal{M}}((\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{y}_\beta)\,\mathbf{A}_\alpha) && (1)\\
& = V_{\varphi[\mathbf{x}_\alpha \mapsto V_\varphi^{\mathcal{M}}(\mathbf{A}_\alpha)]}^{\mathcal{M}}(\mathbf{y}_\beta) && (2)\\
& = V_\varphi^{\mathcal{M}}(\mathbf{y}_\beta) && (3)
\end{aligned}
$$

(2) is by part 2 of Lemma 7.1.1 and (3) is by Lemma 7.1.2.

Axiom A4.2 We must show

$$V_\varphi^{\mathcal{M}}((\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{x}_\alpha)\,\mathbf{A}_\alpha) = V_\varphi^{\mathcal{M}}(\mathbf{A}_\alpha)$$

to prove Axiom A4.2 is valid in $\mathcal{M}$.

$$
\begin{aligned}
& V_\varphi^{\mathcal{M}}((\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{x}_\alpha)\,\mathbf{A}_\alpha) && (1)\\
& = V_{\varphi[\mathbf{x}_\alpha \mapsto V_\varphi^{\mathcal{M}}(\mathbf{A}_\alpha)]}^{\mathcal{M}}(\mathbf{x}_\alpha) && (2)\\
& = V_\varphi^{\mathcal{M}}(\mathbf{A}_\alpha) && (3)
\end{aligned}
$$

(2) is by part 2 of Lemma 7.1.1 and (3) is by the semantics of variables.

Axiom A4.3 Similar to Axiom A4.1.

Axiom A4.4 We must show

$$
\begin{aligned}
& V_\varphi^{\mathcal{M}}((\lambda\,\mathbf{x}_\alpha\,.\,(\mathbf{B}_{\beta\to\gamma}\,\mathbf{C}_\beta))\,\mathbf{A}_\alpha)\\
& = V_\varphi^{\mathcal{M}}(((\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_{\beta\to\gamma})\,\mathbf{A}_\alpha)\,((\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{C}_\beta)\,\mathbf{A}_\alpha))
\end{aligned}
$$

to prove Axiom A4.4 is valid in $\mathcal{M}$.

$$
\begin{aligned}
& V_\varphi^{\mathcal{M}}((\lambda\,\mathbf{x}_\alpha\,.\,(\mathbf{B}_{\beta\to\gamma}\,\mathbf{C}_\beta))\,\mathbf{A}_\alpha) && (1)\\
& = V_{\varphi[\mathbf{x}_\alpha \mapsto V_\varphi^{\mathcal{M}}(\mathbf{A}_\alpha)]}^{\mathcal{M}}(\mathbf{B}_{\beta\to\gamma}\,\mathbf{C}_\beta) && (2)\\
& = V_{\varphi[\mathbf{x}_\alpha \mapsto V_\varphi^{\mathcal{M}}(\mathbf{A}_\alpha)]}^{\mathcal{M}}(\mathbf{B}_{\beta\to\gamma})(V_{\varphi[\mathbf{x}_\alpha \mapsto V_\varphi^{\mathcal{M}}(\mathbf{A}_\alpha)]}^{\mathcal{M}}(\mathbf{C}_\beta)) && (3)\\
& = V_\varphi^{\mathcal{M}}((\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_{\beta\to\gamma})\,\mathbf{A}_\alpha)(V_\varphi^{\mathcal{M}}((\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{C}_\beta)\,\mathbf{A}_\alpha)) && (4)\\
& = V_\varphi^{\mathcal{M}}(((\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_{\beta\to\gamma})\,\mathbf{A}_\alpha)\,((\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{C}_\beta)\,\mathbf{A}_\alpha)) && (5)
\end{aligned}
$$

(2) and (4) are by part 2 of Lemma 7.1.1 and (3) and (5) are by the semantics of function application.

Axiom A4.5 Let $\mathbf{x}_\alpha$ and $\mathbf{y}_\beta$ be distinct and either (a) $\mathbf{A}_\alpha$ is eval-free and $\mathbf{y}_\beta$ is not free in $\mathbf{A}_\alpha$ or (b) $\mathbf{B}_\gamma$ is eval-free and $\mathbf{x}_\alpha$ is not free in $\mathbf{B}_\gamma$. We must show

$$V_\varphi^{\mathcal{M}}((\lambda\,\mathbf{x}_\alpha\,.\,\lambda\,\mathbf{y}_\beta\,.\,\mathbf{B}_\gamma)\,\mathbf{A}_\alpha)(d) = V_\varphi^{\mathcal{M}}(\lambda\,\mathbf{y}_\beta\,.\,((\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_\gamma)\,\mathbf{A}_\alpha))(d),$$

where $d \in D_\beta$, to prove Axiom 4.5 is valid in $\mathcal{M}$.

$$
\begin{aligned}
& V^{\mathcal{M}}_{\varphi}((\lambda\,\mathbf{x}_\alpha\,.\,\lambda\,\mathbf{y}_\beta\,.\,\mathbf{B}_\gamma)\,\mathbf{A}_\alpha)(d) && (1)\\
&= V^{\mathcal{M}}_{\varphi[\mathbf{x}_\alpha \mapsto V^{\mathcal{M}}_{\varphi}(\mathbf{A}_\alpha)]}(\lambda\,\mathbf{y}_\beta\,.\,\mathbf{B}_\gamma)(d) && (2)\\
&= V^{\mathcal{M}}_{\varphi[\mathbf{x}_\alpha \mapsto V^{\mathcal{M}}_{\varphi}(\mathbf{A}_\alpha)][\mathbf{y}_\beta \mapsto d]}(\mathbf{B}_\gamma) && (3)\\
&= V^{\mathcal{M}}_{\varphi[\mathbf{y}_\beta \mapsto d][\mathbf{x}_\alpha \mapsto V^{\mathcal{M}}_{\varphi}(\mathbf{A}_\alpha)]}(\mathbf{B}_\gamma) && (4)\\
&= V^{\mathcal{M}}_{\varphi[\mathbf{y}_\beta \mapsto d][\mathbf{x}_\alpha \mapsto V^{\mathcal{M}}_{\varphi[\mathbf{y}_\beta \mapsto d]}(\mathbf{A}_\alpha)]}(\mathbf{B}_\gamma) && (5)\\
&= V^{\mathcal{M}}_{\varphi[\mathbf{y}_\beta \mapsto d]}((\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_\gamma)\,\mathbf{A}_\alpha) && (6)\\
&= V^{\mathcal{M}}_{\varphi}(\lambda\,\mathbf{y}_\beta\,.\,((\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_\gamma)\,\mathbf{A}_\alpha))(d) && (7)
\end{aligned}
$$

(2) and (6) are by part 2 of Lemma 7.1.1; (3) and (7) are by the semantics of function abstraction; (4) is by $\mathbf{x}_\alpha$ and $\mathbf{y}_\beta$ being distinct; and (5) is by the hypothesis and Lemma 7.1.2.

Axiom A4.6 We must show

$$V^{\mathcal{M}}_{\varphi}((\lambda\,\mathbf{x}_\alpha\,.\,\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_\beta)\,\mathbf{A}_\alpha) = V^{\mathcal{M}}_{\varphi}(\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_\beta)$$

to prove Axiom A4.6 is valid in $\mathcal{M}$.

$$
\begin{aligned}
& V^{\mathcal{M}}_{\varphi}((\lambda\,\mathbf{x}_\alpha\,.\,\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_\beta)\,\mathbf{A}_\alpha) && (1)\\
&= V^{\mathcal{M}}_{\varphi[\mathbf{x}_\alpha \mapsto V^{\mathcal{M}}_{\varphi}(\mathbf{A}_\alpha)]}(\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_\beta) && (2)\\
&= V^{\mathcal{M}}_{\varphi}(\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_\beta) && (3)
\end{aligned}
$$

(2) and (3) are by parts 2 and 3 of Lemma 7.1.1, respectively.

Axiom Group B1 By clauses 2 and 3 of the definition of an interpretation.

Axiom Group B2 By clauses 4 and 5 of the definition of an interpretation.

Axiom Group B3 By clauses 9 and 10 of the definition of an interpretation.

Axiom Group B4 By clauses 2–8 of the definition of an interpretation.

Axiom Group B5 By clause 11 of the definition of an interpretation.

Axiom B6 By the definition of the domain of constructions in a frame.

Axiom Group B7 By clause 12 of the definition of an interpretation.

Axiom Group B8

Axiom B8.1 We must show

$$V^{\mathcal{M}}_{\varphi}(\ulcorner \mathbf{F}_{\alpha\to\beta}\,\mathbf{A}_\alpha \urcorner) = V^{\mathcal{M}}_{\varphi}(\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\ulcorner \mathbf{F}_{\alpha\to\beta}\urcorner\ulcorner \mathbf{A}_\alpha\urcorner)$$

to prove Axiom B8.1 is valid in $\mathcal{M}$.

$$
\begin{aligned}
& V_{\varphi}^{\mathcal{M}}(\ulcorner \mathbf{F}_{\alpha\to\beta}\, \mathbf{A}_{\alpha} \urcorner) && (1)\\
&= \mathcal{E}(\ulcorner \mathbf{F}_{\alpha\to\beta}\, \mathbf{A}_{\alpha} \urcorner) && (2)\\
&= \mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\, \mathcal{E}(\mathbf{F}_{\alpha\to\beta})\, \mathcal{E}(\mathbf{A}_{\alpha}) && (3)\\
&= \mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\, V_{\varphi}^{\mathcal{M}}(\ulcorner \mathbf{F}_{\alpha\to\beta} \urcorner)\, V_{\varphi}^{\mathcal{M}}(\ulcorner \mathbf{A}_{\alpha} \urcorner) && (4)\\
&= V_{\varphi}^{\mathcal{M}}(\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\, \ulcorner \mathbf{F}_{\alpha\to\beta} \urcorner \ulcorner \mathbf{A}_{\alpha} \urcorner) && (5)
\end{aligned}
$$

(2) and (4) are by condition 5 of the definition of a general model; (3) is by the definition of $\mathcal{E}$; and (5) is by the semantics of $\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}$ and clause 6 of the definition of an interpretation.

Axiom B8.2 Similar to Axiom B8.1.

Axiom B8.3 Similar to Axiom B8.1.

Axiom B9 We must show

$$V_{\varphi}^{\mathcal{M}}((\lambda\, \mathbf{x}_{\alpha}\, .\, \ulcorner \mathbf{B}_{\beta} \urcorner)\, \mathbf{A}_{\alpha}) = V_{\varphi}^{\mathcal{M}}(\ulcorner \mathbf{B}_{\beta} \urcorner)$$

to prove Axiom B9 is valid in $\mathcal{M}$.

$$
\begin{aligned}
& V_{\varphi}^{\mathcal{M}}((\lambda\, \mathbf{x}_{\alpha}\, .\, \ulcorner \mathbf{B}_{\beta} \urcorner)\, \mathbf{A}_{\alpha}) && (1)\\
&= V_{\varphi[\mathbf{x}_{\alpha}\mapsto V_{\varphi}^{\mathcal{M}}(\mathbf{A}_{\alpha})]}^{\mathcal{M}}(\ulcorner \mathbf{B}_{\beta} \urcorner) && (2)\\
&= V_{\varphi}^{\mathcal{M}}(\ulcorner \mathbf{B}_{\beta} \urcorner) && (3)
\end{aligned}
$$

(2) is by part 2 of Lemma 7.1.1 and (3) is by Lemma 7.1.2.

Axiom Group B10

Axiom B10.1 We must show

$$V_{\varphi}^{\mathcal{M}}([\![\ulcorner \mathbf{x}_{\alpha} \urcorner]\!]_{\alpha}) = V_{\varphi}^{\mathcal{M}}(\mathbf{x}_{\alpha})$$

to prove Axiom B10.1 is valid in $\mathcal{M}$. This equation follows from the Law of Disquotation and part 1 of Lemma 7.1.1.

Axiom B10.2 Same proof as for Axiom B10.1.

Axiom B10.3 Let (a) $V_{\varphi}^{\mathcal{M}}(\mathsf{is\text{-}expr}_{\epsilon\to o}^{\alpha\to\beta}\, \mathbf{A}_{\epsilon}) = \mathrm{T}$ and (b) $V_{\varphi}^{\mathcal{M}}(\mathsf{is\text{-}expr}_{\epsilon\to o}^{\alpha}\, \mathbf{B}_{\epsilon}) = \mathrm{T}$. We must show

$$V_{\varphi}^{\mathcal{M}}([\![\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\, \mathbf{A}_{\epsilon}\, \mathbf{B}_{\epsilon}]\!]_{\beta}) = V_{\varphi}^{\mathcal{M}}([\![\mathbf{A}_{\epsilon}]\!]_{\alpha\to\beta}\, [\![\mathbf{B}_{\epsilon}]\!]_{\alpha})$$

to prove Axiom B10.3 is valid in $\mathcal{M}$. By part 3 of Lemma 7.1.4, (a) and (b) imply (c) $V_{\varphi}^{\mathcal{M}}(\mathbf{A}_{\epsilon}) = V_{\varphi}^{\mathcal{M}}(\mathcal{E}(\mathbf{C}_{\alpha\to\beta}))$ for some $\mathbf{C}_{\alpha\to\beta}$ and (d) $V_{\varphi}^{\mathcal{M}}(\mathbf{B}_{\epsilon}) =$

$V_\varphi^{\mathcal{M}}(\mathcal{E}(\mathbf{D}_\alpha))$ for some $\mathbf{D}_\alpha$, respectively.

$$
\begin{aligned}
& V_\varphi^{\mathcal{M}}([\![\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\,\mathbf{A}_\epsilon\,\mathbf{B}_\epsilon]\!]_\beta) && (1)\\
&= V_\varphi^{\mathcal{M}}([\![\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\,\mathcal{E}(\mathbf{C}_{\alpha\to\beta})\,\mathcal{E}(\mathbf{D}_\alpha)]\!]_\beta) && (2)\\
&= V_\varphi^{\mathcal{M}}([\![\mathcal{E}(\mathbf{C}_{\alpha\to\beta}\,\mathbf{D}_\alpha)]\!]_\beta) && (3)\\
&= V_\varphi^{\mathcal{M}}(\mathbf{C}_{\alpha\to\beta}\,\mathbf{D}_\alpha) && (4)\\
&= V_\varphi^{\mathcal{M}}(\mathbf{C}_{\alpha\to\beta})(V_\varphi^{\mathcal{M}}(\mathbf{D}_\alpha)) && (5)\\
&= V_\varphi^{\mathcal{M}}([\![\mathcal{E}(\mathbf{C}_{\alpha\to\beta})]\!]_{\alpha\to\beta})(V_\varphi^{\mathcal{M}}([\![\mathcal{E}(\mathbf{D}_\alpha)]\!]_\alpha)) && (6)\\
&= V_\varphi^{\mathcal{M}}([\![\mathbf{A}_\epsilon]\!]_{\alpha\to\beta})(V_\varphi^{\mathcal{M}}([\![\mathbf{B}_\epsilon]\!]_\alpha)) && (7)\\
&= V_\varphi^{\mathcal{M}}([\![\mathbf{A}_\epsilon]\!]_{\alpha\to\beta}\,[\![\mathbf{B}_\epsilon]\!]_\alpha) && (8)
\end{aligned}
$$

(2) and (7) are by (c) and (d); (3) is by the definition of $\mathcal{E}$; (4) and (6) are by Lemma 7.1.6; and (5) and (8) are by the semantics of function application.

Axiom B10.4 Let (a) $V_\varphi^{\mathcal{M}}(\mathsf{is\text{-}expr}^{\beta}_{\epsilon\to o}\,\mathbf{A}_\epsilon) = \mathrm{T}$ and
(b) $V_\varphi^{\mathcal{M}}(\neg(\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}\ulcorner\mathbf{x}_\alpha\urcorner\ulcorner\mathbf{A}_\epsilon\urcorner)) = \mathrm{T}$. We must show

$$V_\varphi^{\mathcal{M}}([\![\mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}\ulcorner\mathbf{x}_\alpha\urcorner\,\mathbf{A}_\epsilon]\!]_{\alpha\to\beta})(d) = V_\varphi^{\mathcal{M}}(\lambda\,\mathbf{x}_\alpha\,.\,[\![\mathbf{A}_\epsilon]\!]_\beta)(d),$$

where $d \in D_\alpha$, to prove Axiom B10.4 is valid in $\mathcal{M}$. By part 3 of Lemma 7.1.4, (a) implies (c) $V_\varphi^{\mathcal{M}}(\mathbf{A}_\epsilon) = V_\varphi^{\mathcal{M}}(\mathcal{E}(\mathbf{B}_\beta))$ for some $\mathbf{B}_\beta$. By Lemma 7.1.5, (b) implies (d) $\mathbf{x}_\alpha$ is not free in $\mathbf{A}_\epsilon$.

$$
\begin{aligned}
& V_\varphi^{\mathcal{M}}([\![\mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}\ulcorner\mathbf{x}_\alpha\urcorner\,\mathbf{A}_\epsilon]\!]_{\alpha\to\beta})(d) && (1)\\
&= V_\varphi^{\mathcal{M}}([\![\mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}\ulcorner\mathbf{x}_\alpha\urcorner\,\mathcal{E}(\mathbf{B}_\beta)]\!]_{\alpha\to\beta})(d) && (2)\\
&= V_\varphi^{\mathcal{M}}([\![\mathcal{E}(\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_\beta)]\!]_{\alpha\to\beta})(d) && (3)\\
&= V_\varphi^{\mathcal{M}}(\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_\beta)(d) && (4)\\
&= V_{\varphi[\mathbf{x}_\alpha\mapsto d]}^{\mathcal{M}}(\mathbf{B}_\beta) && (5)\\
&= V_{\varphi[\mathbf{x}_\alpha\mapsto d]}^{\mathcal{M}}([\![\mathcal{E}(\mathbf{B}_\beta)]\!]_\beta) && (6)\\
&= V_{\varphi[\mathbf{x}_\alpha\mapsto d]}^{\mathcal{M}}([\![\mathbf{A}_\epsilon]\!]_\beta) && (7)\\
&= V_\varphi^{\mathcal{M}}(\lambda\,\mathbf{x}_\alpha\,.\,[\![\mathbf{A}_\epsilon]\!]_\beta)(d) && (8)
\end{aligned}
$$

(2) is by (c); (3) is by the definition of $\mathcal{E}$; (4) and (6) are by Lemma 7.1.6; (5) and (8) are by the semantics of function abstraction; and (7) is by (c), (d), Lemma 7.1.2, and the fact that constructions are closed expressions.

Axiom B10.5 Let (a) $V_\varphi^{\mathcal{M}}(\mathsf{is\text{-}expr}^{\epsilon}_{\epsilon\to o}\,\mathbf{A}_\epsilon) = \mathrm{T}$. We must show

$$V_\varphi^{\mathcal{M}}([\![\mathsf{quo}_{\epsilon\to\epsilon}\,\mathbf{A}_\epsilon]\!]_\epsilon) = V_\varphi^{\mathcal{M}}(\mathbf{A}_\epsilon)$$

to prove Axiom B10.5 is valid in $\mathcal{M}$. By part 3 of Lemma 7.1.4, (a) implies (b) $V_\varphi^{\mathcal{M}}(\mathbf{A}_\epsilon) = V_\varphi^{\mathcal{M}}(\mathcal{E}(\mathbf{B}_\epsilon))$ for some $\mathbf{B}_\epsilon$.

$$
\begin{aligned}
&V_\varphi^{\mathcal{M}}([\![\mathsf{quo}_{\epsilon\to\epsilon}\,\mathbf{A}_\epsilon]\!]_\epsilon) && (1)\\
&= V_\varphi^{\mathcal{M}}([\![\mathsf{quo}_{\epsilon\to\epsilon}\,\mathcal{E}(\mathbf{B}_\epsilon)]\!]_\epsilon) && (2)\\
&= V_\varphi^{\mathcal{M}}([\![\mathcal{E}(\ulcorner\mathbf{B}_\epsilon\urcorner)]\!]_\epsilon) && (3)\\
&= V_\varphi^{\mathcal{M}}(\ulcorner\mathbf{B}_\epsilon\urcorner) && (4)\\
&= V_\varphi^{\mathcal{M}}(\mathcal{E}(\mathbf{B}_\epsilon)) && (5)\\
&= V_\varphi^{\mathcal{M}}(\mathbf{A}_\epsilon) && (6)
\end{aligned}
$$

(2) and (6) are by (b); (3) is by the definition of $\mathcal{E}$; (4) is by Lemma 7.1.6; and (5) is by Law of Quotation.

Axiom Group B11

Axiom B11.1

We must show

$$V_\varphi^{\mathcal{M}}((\lambda\,\mathbf{x}_\alpha\,.\,[\![\mathbf{B}_\epsilon]\!]_\beta)\,\mathbf{x}_\alpha) = V_\varphi^{\mathcal{M}}([\![\mathbf{B}_\epsilon]\!]_\beta)$$

to prove Axiom B11.1 is valid in $\mathcal{M}$.

$$
\begin{aligned}
&V_\varphi^{\mathcal{M}}((\lambda\,\mathbf{x}_\alpha\,.\,[\![\mathbf{B}_\epsilon]\!]_\beta)\,\mathbf{x}_\alpha) && (1)\\
&= V_{\varphi[\mathbf{x}_\alpha\mapsto V_\varphi^{\mathcal{M}}(\mathbf{x}_\alpha)]}^{\mathcal{M}}([\![\mathbf{B}_\epsilon]\!]_\beta) && (2)\\
&= V_\varphi^{\mathcal{M}}([\![\mathbf{B}_\epsilon]\!]_\beta) && (3)
\end{aligned}
$$

(2) is by part 2 of Lemma 7.1.1 and (3) is by

$$\varphi[\mathbf{x}_\alpha \mapsto V_\varphi^{\mathcal{M}}(\mathbf{x}_\alpha)] = \varphi[\mathbf{x}_\alpha \mapsto \varphi(\mathbf{x}_\alpha)] = \varphi.$$

Axiom B11.2 Let (a) $V_\varphi^{\mathcal{M}}(\mathsf{is\text{-}expr}_{\epsilon\to o}^{\beta}\,((\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_\epsilon)\,\mathbf{A}_\alpha)) = \mathrm{T}$ and (b) $V_\varphi^{\mathcal{M}}(\neg(\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}\,\ulcorner\mathbf{x}_\alpha\urcorner\,((\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_\epsilon)\,\mathbf{A}_\alpha))) = \mathrm{T}$. We must show

$$V_\varphi^{\mathcal{M}}((\lambda\,\mathbf{x}_\alpha\,.\,[\![\mathbf{B}_\epsilon]\!]_\beta)\,\mathbf{A}_\alpha) = V_\varphi^{\mathcal{M}}([\![(\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_\epsilon)\,\mathbf{A}_\alpha]\!]_\beta)$$

to prove Axiom B11.2 is valid in $\mathcal{M}$. By part 3 of Lemma 7.1.4, (a) implies (c) $V_\varphi^{\mathcal{M}}((\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_\epsilon)\,\mathbf{A}_\alpha) = V_\varphi^{\mathcal{M}}(\mathcal{E}(\mathbf{C}_\beta))$ for some (eval-free) $\mathbf{C}_\beta$. By (a) and part 2 of Lemma 7.1.1, (d) $V_{\varphi[\mathbf{x}_\alpha\mapsto V_\varphi^{\mathcal{M}}(\mathbf{A}_\alpha)]}^{\mathcal{M}}(\mathsf{is\text{-}expr}_{\epsilon\to o}^{\beta}\,\mathbf{B}_\epsilon) = \mathrm{T}$. By

(b), (c), Lemma 7.1.5, and the Law of Quotation, (e) $\mathbf{x}_\alpha$ is not free in $\mathbf{C}_\beta$.

$$
\begin{aligned}
& V_\varphi^{\mathcal{M}}((\lambda\,\mathbf{x}_\alpha\,.\,[\![\mathbf{B}_\epsilon]\!]_\beta)\,\mathbf{A}_\alpha) && (1)\\
&= V^{\mathcal{M}}_{\varphi[\mathbf{x}_\alpha \mapsto V_\varphi^{\mathcal{M}}(\mathbf{A}_\alpha)]}([\![\mathbf{B}_\epsilon]\!]_\beta) && (2)\\
&= V^{\mathcal{M}}_{\varphi[\mathbf{x}_\alpha \mapsto V_\varphi^{\mathcal{M}}(\mathbf{A}_\alpha)]}(\mathcal{E}^{-1}(V^{\mathcal{M}}_{\varphi[\mathbf{x}_\alpha \mapsto V_\varphi^{\mathcal{M}}(\mathbf{A}_\alpha)]}(\mathbf{B}_\epsilon))) && (3)\\
&= V^{\mathcal{M}}_{\varphi[\mathbf{x}_\alpha \mapsto V_\varphi^{\mathcal{M}}(\mathbf{A}_\alpha)]}(\mathcal{E}^{-1}(V_\varphi^{\mathcal{M}}((\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_\epsilon)\,\mathbf{A}_\alpha))) && (4)\\
&= V^{\mathcal{M}}_{\varphi[\mathbf{x}_\alpha \mapsto V_\varphi^{\mathcal{M}}(\mathbf{A}_\alpha)]}(\mathcal{E}^{-1}(V_\varphi^{\mathcal{M}}(\mathcal{E}(\mathbf{C}_\beta)))) && (5)\\
&= V^{\mathcal{M}}_{\varphi[\mathbf{x}_\alpha \mapsto V_\varphi^{\mathcal{M}}(\mathbf{A}_\alpha)]}(\mathcal{E}^{-1}(\mathcal{E}(\mathbf{C}_\beta))) && (6)\\
&= V^{\mathcal{M}}_{\varphi[\mathbf{x}_\alpha \mapsto V_\varphi^{\mathcal{M}}(\mathbf{A}_\alpha)]}(\mathbf{C}_\beta) && (7)\\
&= V_\varphi^{\mathcal{M}}(\mathbf{C}_\beta) && (8)\\
&= V_\varphi^{\mathcal{M}}([\![\mathcal{E}(\mathbf{C}_\beta)]\!]_\beta) && (9)\\
&= V_\varphi^{\mathcal{M}}([\![(\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_\epsilon)\,\mathbf{A}_\alpha]\!]_\beta) && (10)
\end{aligned}
$$

(2) and (4) are by part 2 of Lemma 7.1.1; (3) is by (d) and condition 6 of the definition of a general model; (5) is by (c); (6) is by Proposition 3.3.4; (7) is immediate; (8) is by (e) and Lemma 7.1.5; (9) is by Lemma 7.1.6; and (10) is by (c).

Axiom B12 By Lemmas 7.1.2 and 7.1.3.

Axiom B13 By Lemma 7.1.3 and the proof for Axiom A4.5. □

Lemma 7.2.2 (Rule R Preserves Validity) *Rule R preserves validity in all general models for* CTT_{qe}.

Proof Let $\mathcal{M}$ be a general model for CTT_{qe}. Suppose $\mathbf{C}_o$ and $\mathbf{C}'_o$ are formulas such that $\mathbf{C}'_o$ is the result of replacing one occurrence of $\mathbf{A}_\alpha$ in $\mathbf{C}_o$ by an occurrence of $\mathbf{B}_\alpha$, provided that the occurrence of $\mathbf{A}_\alpha$ in $\mathbf{C}_o$ is not within a quotation, not the first argument of a function abstraction, and not the second argument of an evaluation. Then it easily follows that $V_\varphi^{\mathcal{M}}(\mathbf{A}_\alpha) = V_\varphi^{\mathcal{M}}(\mathbf{B}_\alpha)$ for all $\varphi \in \mathsf{assign}(\mathcal{M})$ implies $V_\varphi^{\mathcal{M}}(\mathbf{C}_o) = V_\varphi^{\mathcal{M}}(\mathbf{C}'_o)$ for all $\varphi \in \mathsf{assign}(\mathcal{M})$ by induction on the structure of $\mathbf{C}_o$ (condition 7 of the definition of a general model is needed for the argument). $\mathcal{M} \vDash \mathbf{A}_\alpha = \mathbf{B}_\alpha$ implies $V_\varphi^{\mathcal{M}}(\mathbf{A}_\alpha) = V_\varphi^{\mathcal{M}}(\mathbf{B}_\alpha)$ for all $\varphi \in \mathsf{assign}(\mathcal{M})$ by part 1 of Lemma 7.1.1, and hence $\mathcal{M} \vDash \mathbf{A}_\alpha = \mathbf{B}_\alpha$ and $\mathcal{M} \vDash \mathbf{C}_o$ imply $V_\varphi^{\mathcal{M}}(\mathbf{C}_o) = V_\varphi^{\mathcal{M}}(\mathbf{C}'_o) = \mathrm{T}$ for all $\varphi \in \mathsf{assign}(\mathcal{M})$. Therefore, $\mathcal{M} \vDash \mathbf{A}_\alpha = \mathbf{B}_\alpha$ and $\mathcal{M} \vDash \mathbf{C}_o$ imply $\mathcal{M} \vDash \mathbf{C}'_o$, and so Rule R preserves validity in $\mathcal{M}$. □

7.3 Soundness and Consistency Theorems

Theorem 7.3.1 (Soundness Theorem) *Let T be a theory of* CTT_{qe}, $\boldsymbol{A}_o$ *be a formula of T, and $\mathcal{H}$ be a set of formulas of T.*

1. *$T \vdash \boldsymbol{A}_o$ implies $T \vDash \boldsymbol{A}_o$ (i.e., the proof system for CTT_{qe} is sound).*

2. *$T, \mathcal{H} \vdash \boldsymbol{A}_o$ implies $T, \mathcal{H} \vDash \boldsymbol{A}_o$.*

Proof

Part 1 Assume $T \vdash \mathbf{A}_o$ and $\mathcal{M}$ is a general model for T. We must show that $\mathcal{M} \vDash \mathbf{A}_o$. By assumption, each member of Γ is valid in $\mathcal{M}$. By Lemma 7.2.1, each axiom of CTT_{qe} is valid in $\mathcal{M}$. And by Lemma 7.2.2, Rule R preserves validity in $\mathcal{M}$. Therefore, $T \vdash \mathbf{A}_o$ implies $\mathcal{M} \vDash \mathbf{A}_o$.

Part 2 Assume $T, \mathcal{H} \vdash \mathbf{A}_o$, $\mathcal{M}$ is a general model for T, $\varphi \in \mathsf{assign}(\mathcal{M})$, and $V_\varphi^{\mathcal{M}}(\mathbf{H}_o) = \text{T}$ for all $\mathbf{H}_o \in \mathcal{H}$. We need to show $V_\varphi^{\mathcal{M}}(\mathbf{A}_o) = \text{T}$. There is a finite subset $\{\mathbf{H}_o^1, \ldots, \mathbf{H}_o^n\}$ of $\mathcal{H}$ such that $T, \{\mathbf{H}_o^1, \ldots, \mathbf{H}_o^n\} \vdash \mathbf{A}_o$. Then

$$T \vdash \mathbf{H}_o^1 \supset (\ldots(\mathbf{H}_o^n \supset \mathbf{A}_o)\ldots)$$

by the Deduction Theorem, and so by part 1 of the theorem,

$$\mathcal{M} \vDash \mathbf{H}_o^1 \supset (\ldots(\mathbf{H}_o^n \supset \mathbf{A}_o)\ldots).$$

By hypothesis, $V_\varphi^{\mathcal{M}}(\mathbf{H}_o^i) = \text{T}$ for each i with $1 \leq i \leq n$, and so $V_\varphi^{\mathcal{M}}(\mathbf{A}_o) = \text{T}$. □

Corollary 7.3.2 *The proof system for CTT_{qe} satisfies Requirement R1.*

Theorem 7.3.3 (Consistency Theorem) *Let T be a theory of CTT_{qe}. If T has a general model, then T is consistent.*

Proof Assume $\mathcal{M}$ is a general model for T and T is inconsistent, i.e., $T \vdash F_o$. By the Soundness Theorem, $T \vDash F_o$ and hence $\mathcal{M} \vDash F_o$. This means $V_\varphi^{\mathcal{M}}(F_o) = \text{T}$ and hence $V_\varphi^{\mathcal{M}}(F_o) \neq \text{F}$ (for any assignment φ), which contracts the definition of a general model. □

8 Completeness

We will now show that the proof system for CTT_{qe} is complete with respect to the (general models) semantics for CTT_{qe} for eval-free formulas. More precisely, we will show that $T \vDash \mathbf{A}_o$ implies $T \vdash \mathbf{A}_o$ whenever T is an eval-free theory of CTT_{qe} and $\mathbf{A}_o$ is an eval-free formula of T. We will also show that the proof system for CTT_{qe} is not complete with respect to the semantics for CTT_{qe} for non-eval-free formulas.

8.1 Eval-Free Completeness

Let CTT_ϵ be the logic obtained from CTT_{qe} as follows:

1. Replace the set $\mathcal{C}$ of constants by the expanded set $\mathcal{C} \cup \mathcal{C}'$ where:

$$\mathcal{C}' = \{d_\epsilon^{\ulcorner \mathbf{x}_\alpha \urcorner} \mid \mathbf{x}_\alpha \in \mathcal{V}\} \cup \{d_\epsilon^{\ulcorner \mathbf{c}_\alpha \urcorner} \mid \mathbf{c}_\alpha \in \mathcal{C}\}.$$

2. Remove the quotation operator $\ulcorner\cdot\urcorner$ from the set of expression constructors so that there are no (primitive) quotations in CTT_ϵ, but let $\ulcorner\mathbf{A}_\alpha\urcorner$ be an abbreviation defined by:

   a. $\ulcorner\mathbf{x}_\alpha\urcorner$ stands for $d_\epsilon^{\ulcorner\mathbf{x}_\alpha\urcorner}$.

   b. $\ulcorner\mathbf{c}_\alpha\urcorner$ stands for $d_\epsilon^{\ulcorner\mathbf{c}_\alpha\urcorner}$.

   c. $\ulcorner\mathbf{F}_{\alpha\to\beta}\,\mathbf{A}_\alpha\urcorner$ stands for $\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\ulcorner\mathbf{F}_{\alpha\to\beta}\urcorner\ulcorner\mathbf{A}_\alpha\urcorner$.

   d. $\ulcorner\lambda\,\mathbf{x}_\alpha\,.\,\mathbf{B}_\beta\urcorner$ stands for $\mathsf{abs}_{\epsilon\to\epsilon\to\epsilon}\ulcorner\mathbf{x}_\alpha\urcorner\ulcorner\mathbf{B}_\beta\urcorner$.

   e. $\ulcorner\ulcorner\mathbf{A}_\alpha\urcorner\urcorner$ stands for $\mathsf{quo}_{\epsilon\to\epsilon}\ulcorner\mathbf{A}_\alpha\urcorner$.

3. Remove the evaluation operator $[\![\cdot]\!]$. from the set of expression constructors so that there are no evaluations in CTT_ϵ.

4. A general model for CTT_ϵ is the same as a general model for CTT_{qe} except that (1) an interpretation maps the constants of the form $d_\epsilon^{\ulcorner\mathbf{x}_\alpha\urcorner}$ and $d_\epsilon^{\ulcorner\mathbf{c}_\alpha\urcorner}$ to the constructions $\ulcorner\mathbf{x}_\alpha\urcorner$ and $\ulcorner\mathbf{c}_\alpha\urcorner$, respectively, and (2) a valuation function is not applicable to quotations and evaluations.

5. The proof system for CTT_ϵ is the same as the proof system for CTT_{qe} except the axioms involving quotation, evaluation, and IS-EFFECTIVE-IN are removed: B8.1–3, B9, B10.1–5, B11.1–2, B12, and B13.

CTT_ϵ is essentially the same as $\mathcal{Q}_0$ with the inductive type ϵ added to it.

Lemma 8.1.1 *Let $T = (L_\mathcal{D}, \Gamma)$ be an eval-free theory of CTT_{qe}, $\boldsymbol{A}_o$ be an eval-free formula of T, and $T' = (L_{\mathcal{D}\cup\mathcal{C}'}, \Gamma)$.*

1. *If $\boldsymbol{A}_o$ is valid in T in CTT_{qe}, then $\boldsymbol{A}_o$ is valid in T' in CTT_ϵ.*

2. *If $\boldsymbol{A}_o$ is valid in T' in CTT_ϵ, then $\boldsymbol{A}_o$ is a theorem of T' in CTT_ϵ.*

3. *If $\boldsymbol{A}_o$ is a theorem of T' in CTT_ϵ, then $\boldsymbol{A}_o$ is a theorem of T in CTT_{qe}.*

Proof Let (a) $T = (L_\mathcal{D}, \Gamma)$ be an eval-free theory of CTT_{qe}, (b) $\mathbf{A}_o$ be an eval-free formula of T, and $T' = (L_{\mathcal{D}\cup\mathcal{C}'}, \Gamma)$. (a) implies T' is a theory of CTT_ϵ, and (b) implies $\mathbf{A}_o$ is a formula of T'.

Part 1 Let (c) $T \models \mathbf{A}_o$ in CTT_{qe}. Let $\mathcal{M}$ be a model for T' in CTT_ϵ. Then (c) and clauses 2 and 4 of the definition of CTT_ϵ implies $\mathcal{M} \models \mathbf{A}_o$. Therefore $T' \models \mathbf{A}_o$ in CTT_ϵ.

Part 2 Let $T' \models \mathbf{A}_o$ in CTT_ϵ. Then $T' \vdash \mathbf{A}_o$ in CTT_ϵ by a proof that is essentially the same as the proof of the completeness of the proof system for $\mathcal{Q}_0$ (see 5502 in [3, p. 253]).

Part 3 Let P be a proof of $\mathbf{A}_o$ in T' in CTT_ϵ. Define $\mathbf{A}'_o$ to be result of replacing each $\ulcorner\mathbf{B}_\beta\urcorner$ in $\mathbf{A}_o$, where $\mathbf{B}_\beta \notin \mathcal{V}\cup\mathcal{C}$, with $\mathcal{E}(\mathbf{B}_\beta)$, and define P' to be the result of replacing each $d_\epsilon^{\ulcorner\mathbf{x}_\alpha\urcorner}$ in P' with $\ulcorner\mathbf{x}_\alpha\urcorner$ and each $d_\epsilon^{\ulcorner\mathbf{c}_\alpha\urcorner}$ in P' with $\ulcorner\mathbf{c}_\alpha\urcorner$. Since the axioms of the proof system for CTT_ϵ are a subset of the

axioms of the proof system for CTT_{qe} and both proof systems share the same rule of inference, (d) P' is a proof of $\mathbf{A}'_o$ in T in CTT_{qe}. By the Syntactic Law of Quotation and Rule R, (e) $T \vdash \mathbf{A}_o = \mathbf{A}'_o$ in CTT_{qe}. Therefore, $T \vdash \mathbf{A}_o$ in CTT_{qe} follows from (d) and (e) by the Equality Rules.

□

Theorem 8.1.2 (Completeness for Eval-Free Formulas) *Let T be an eval-free theory of CTT_{qe} and $\boldsymbol{A}_o$ be an eval-free formula of T. If $T \vDash \boldsymbol{A}_o$, then $T \vdash \boldsymbol{A}_o$.*

Proof Let $T = (L_{\mathcal{D}}, \Gamma)$ be an eval-free theory of CTT_{qe}, $\mathbf{A}_o$ be an eval-free formula of T, and $T' = (L_{\mathcal{D}\cup\mathcal{C}'}, \Gamma)$. Assume $T \vDash \mathbf{A}_o$ in CTT_{qe}. This implies $T' \vDash \mathbf{A}_o$ in CTT_{ϵ}, which implies $T' \vdash \mathbf{A}_o$ in CTT_{ϵ}, which implies $T \vdash \mathbf{A}_o$ in CTT_{qe} by parts 1, 2, and 3, respectively, of Lemma 8.1.1. □

Corollary 8.1.3 *The proof system for CTT_{qe} satisfies Requirement R2.*

8.2 Non-Eval-Free Incompleteness

We will show that the example mentioned in the paragraph about the Double Substitution problem in section 1 is valid in CTT_{qe} but not provable in the proof system for CTT_{qe}.

Proposition 8.2.1 $\vDash (\lambda\, x_\epsilon \,.\, [\![x_\epsilon]\!]_\epsilon)\ulcorner x_\epsilon \urcorner = \ulcorner x_\epsilon \urcorner$.

Proof Let $\mathcal{M}$ be a general model for CTT_{qe} and $\varphi \in \mathsf{assign}(\mathcal{M})$. By part 1 of Lemma 7.1.1, we need to show that

$$V^{\mathcal{M}}_{\varphi}((\lambda\, x_\epsilon \,.\, [\![x_\epsilon]\!]_\epsilon)\ulcorner x_\epsilon \urcorner) = V^{\mathcal{M}}_{\varphi}(\ulcorner x_\epsilon \urcorner)$$

to prove the proposition.

$$\begin{aligned}
&V^{\mathcal{M}}_{\varphi}((\lambda\, x_\epsilon \,.\, [\![x_\epsilon]\!]_\epsilon)\ulcorner x_\epsilon \urcorner) && (1)\\
&= V^{\mathcal{M}}_{\varphi[x_\epsilon \mapsto V^{\mathcal{M}}_{\varphi}(\ulcorner x_\epsilon \urcorner)]}([\![x_\epsilon]\!]_\epsilon) && (2)\\
&= V^{\mathcal{M}}_{\varphi[x_\epsilon \mapsto V^{\mathcal{M}}_{\varphi}(\ulcorner x_\epsilon \urcorner)]}(\mathcal{E}^{-1}(V^{\mathcal{M}}_{\varphi[x_\epsilon \mapsto V^{\mathcal{M}}_{\varphi}(\ulcorner x_\epsilon \urcorner)]}(x_\epsilon))) && (3)\\
&= V^{\mathcal{M}}_{\varphi[x_\epsilon \mapsto V^{\mathcal{M}}_{\varphi}(\ulcorner x_\epsilon \urcorner)]}(\mathcal{E}^{-1}(V^{\mathcal{M}}_{\varphi}(\ulcorner x_\epsilon \urcorner)))) && (4)\\
&= V^{\mathcal{M}}_{\varphi[x_\epsilon \mapsto V^{\mathcal{M}}_{\varphi}(\ulcorner x_\epsilon \urcorner)]}(\mathcal{E}^{-1}(\mathcal{E}(x_\epsilon))) && (5)\\
&= V^{\mathcal{M}}_{\varphi[x_\epsilon \mapsto V^{\mathcal{M}}_{\varphi}(\ulcorner x_\epsilon \urcorner)]}(x_\epsilon) && (6)\\
&= V^{\mathcal{M}}_{\varphi}(\ulcorner x_\epsilon \urcorner) && (7)
\end{aligned}$$

(2) by part 2 of Lemma 7.1.2; (3) is by condition 6 of the definition of a general model since

$$V^{\mathcal{M}}_{\varphi[x_\epsilon \mapsto V^{\mathcal{M}}_{\varphi}(\ulcorner x_\epsilon \urcorner)]}(\mathsf{is\text{-}expr}^{\epsilon}_{\epsilon\to o}(x_\epsilon)) = V^{\mathcal{M}}_{\varphi[x_\epsilon \mapsto V^{\mathcal{M}}_{\varphi}(\ulcorner x_\epsilon \urcorner)]}(\mathsf{is\text{-}expr}^{\epsilon}_{\epsilon\to o}(\ulcorner x_\epsilon \urcorner)) = \text{T}$$

by the semantics of variables and part 2 of Lemma 7.1.4; (4) and (7) are by the semantics of variables; (5) is by condition 5 of the definition of a general model; and (6) is immediate. □

Proposition 8.2.2 $\not\vdash (\lambda\, x_\epsilon \,.\, [\![x_\epsilon]\!]_\epsilon)\ulcorner x_\epsilon \urcorner = \ulcorner x_\epsilon \urcorner$.

Proof It is necessary to use Axiom B11.2 in order to prove $(\lambda\, x_\epsilon \,.\, [\![x_\epsilon]\!]_\epsilon)\ulcorner x_\epsilon \urcorner = \ulcorner x_\epsilon \urcorner$, but Axiom B11.2 requires that

$$\vdash \neg \mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}\ulcorner x_\epsilon \urcorner((\lambda\, x_\epsilon \,.\, x_\epsilon)\ulcorner x_\epsilon \urcorner).$$

However, $\vdash (\lambda\, x_\epsilon \,.\, x_\epsilon)\ulcorner x_\epsilon \urcorner = \ulcorner x_\epsilon \urcorner$ holds by Axiom A4.2 and $\vdash \mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}\ulcorner x_\epsilon \urcorner\ulcorner x_\epsilon \urcorner$ holds by Lemma 6.5.3, which implies

$$\vdash \mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}\ulcorner x_\epsilon \urcorner((\lambda\, x_\epsilon \,.\, x_\epsilon)\ulcorner x_\epsilon \urcorner)$$

by the Equality Rules. Therefore, Axiom B11.2 is not applicable to $(\lambda\, x_\epsilon \,.\, [\![x_\epsilon]\!]_\epsilon)\ulcorner x_\epsilon \urcorner$ by the Consistency Theorem, and thus $(\lambda\, x_\epsilon \,.\, [\![x_\epsilon]\!]_\epsilon)\ulcorner x_\epsilon \urcorner = \ulcorner x_\epsilon \urcorner$ cannot be proved in the proof system for $\mathrm{CTT}_{\mathrm{qe}}$. □

Theorem 8.2.3 (Incompleteness Theorem) *The proof system for* $\mathrm{CTT}_{\mathrm{qe}}$ *is incomplete.*

Proof This theorem follows directly from the previous two propositions. □

9 Examples Revisited

We prove in this section within the proof system for $\mathrm{CTT}_{\mathrm{qe}}$ the results that were stated in section 4. These proofs show the efficacy of the proof system for $\mathrm{CTT}_{\mathrm{qe}}$ for reasoning about syntax, instantiating formulas containing evaluations, and proving schemas and meaning formulas.

9.1 Reasoning about Syntax

Let $T = (L_{\mathcal{D}}, \Gamma)$ be a theory of $\mathrm{CTT}_{\mathrm{qe}}$ such that $\mathsf{make\text{-}implication}_{\epsilon\to\epsilon\to\epsilon} \in \mathcal{D}$ and Γ contains the definition

$$\mathsf{make\text{-}implication}_{\epsilon\to\epsilon\to\epsilon} = \lambda\, x_\epsilon \,.\, \lambda\, y_\epsilon \,.\, (\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}(\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\ulcorner \supset_{o\to o\to o}\urcorner\, x_\epsilon)\, y_\epsilon).$$

Proposition 9.1.1 $T \vdash \mathsf{make\text{-}implication}_{\epsilon\to\epsilon\to\epsilon}\ulcorner \boldsymbol{A}_o\urcorner\ulcorner \boldsymbol{B}_o\urcorner = \ulcorner \boldsymbol{A}_o \supset \boldsymbol{B}_o\urcorner$.

Proof

$$T \vdash \mathsf{make\text{-}implication}_{\epsilon\to\epsilon\to\epsilon} = \lambda\, x_\epsilon \,.\, \lambda\, y_\epsilon \,.\, (\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}(\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\ulcorner \supset_{o\to o\to o}\urcorner\, x_\epsilon)\, y_\epsilon) \tag{1}$$

$$T \vdash (\lambda\, x_\epsilon \,.\, \lambda\, y_\epsilon \,.\, (\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}(\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\ulcorner \supset_{o\to o\to o}\urcorner\, x_\epsilon)\, y_\epsilon))\ulcorner \mathbf{A}_o\urcorner\ulcorner \mathbf{B}_o\urcorner = \mathsf{app}_{\epsilon\to\epsilon\to\epsilon}(\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\ulcorner \supset_{o\to o\to o}\urcorner\ulcorner \mathbf{A}_o\urcorner)\ulcorner \mathbf{B}_o\urcorner \tag{2}$$

$$T \vdash \mathcal{E}(\mathbf{A}_o \supset \mathbf{B}_o) = \mathsf{app}_{\epsilon\to\epsilon\to\epsilon}(\mathsf{app}_{\epsilon\to\epsilon\to\epsilon}\ulcorner \supset_{o\to o\to o}\urcorner\ulcorner \mathbf{A}_o\urcorner)\ulcorner \mathbf{B}_o\urcorner \tag{3}$$

$$T \vdash \ulcorner \mathbf{A}_o \supset \mathbf{B}_o\urcorner = \mathcal{E}(\mathbf{A}_o \supset \mathbf{B}_o) \tag{4}$$

$$T \vdash \mathsf{make\text{-}implication}_{\epsilon\to\epsilon\to\epsilon}\ulcorner \mathbf{A}_o\urcorner\ulcorner \mathbf{B}_o\urcorner = \ulcorner \mathbf{A}_o \supset \mathbf{B}_o\urcorner \tag{5}$$

(1) follows from T containing the definition of $\mathsf{make\text{-}implication}_{\epsilon\to\epsilon\to\epsilon}$; (2) is by Beta-Reduction by Substitution; (3) is by the definition of $\mathcal{E}$; (4) is by the Syntactic Law of Quotation; and (5) follows from (1), (2), (3), and (4) by the Equality Rules. □

Proposition 9.1.2 $T \vdash [\![\mathsf{make\text{-}implication}_{\epsilon\to\epsilon\to\epsilon} \ulcorner \boldsymbol{A}_o \urcorner \ulcorner \boldsymbol{B}_o \urcorner]\!]_o = \boldsymbol{A}_o \supset \boldsymbol{B}_o$.

Proof

$$T \vdash \mathsf{make\text{-}implication}_{\epsilon\to\epsilon\to\epsilon} \ulcorner \mathbf{A}_o \urcorner \ulcorner \mathbf{B}_o \urcorner = \ulcorner \mathbf{A}_o \supset \mathbf{B}_o \urcorner \tag{1}$$

$$T \vdash [\![\mathsf{make\text{-}implication}_{\epsilon\to\epsilon\to\epsilon} \ulcorner \mathbf{A}_o \urcorner \ulcorner \mathbf{B}_o \urcorner]\!]_o = [\![\ulcorner \mathbf{A}_o \supset \mathbf{B}_o \urcorner]\!]_o \tag{2}$$

$$T \vdash [\![\mathsf{make\text{-}implication}_{\epsilon\to\epsilon\to\epsilon} \ulcorner \mathbf{A}_o \urcorner \ulcorner \mathbf{B}_o \urcorner]\!]_o = \mathbf{A}_o \supset \mathbf{B}_o \tag{3}$$

(1) is Proposition 9.1.1; (2) follows from (1) by the Equality Rules; and (3) follows from (2) and the Syntactic Law of Disquotation by the Equality Rules. □

9.2 Schemas

Theorem 9.2.1 (Law of Excluded Middle)

$$\vdash \forall\, x_\epsilon \,.\, \mathsf{is\text{-}expr}^o_{\epsilon\to o}\, x_\epsilon \supset ([\![x_\epsilon]\!]_o \vee \neg[\![x_\epsilon]\!]_o).$$

Proof

$$\vdash \mathsf{is\text{-}expr}^o_{\epsilon\to o}\, x_\epsilon \supset ([\![x_\epsilon]\!]_o \vee \neg[\![x_\epsilon]\!]_o) \tag{1}$$

$$\vdash \forall\, x_\epsilon \,.\, \mathsf{is\text{-}expr}^o_{\epsilon\to o}\, x_\epsilon \supset ([\![x_\epsilon]\!]_o \vee \neg[\![x_\epsilon]\!]_o) \tag{2}$$

(1) is by the Tautology Theorem and (2) follows from (1) by Universal Generalization. □

Proposition 9.2.2 *If $\boldsymbol{A}_o$ is eval-free and x_ϵ is not free in $\boldsymbol{A}_o$, then $\boldsymbol{A}_o \vee \neg \boldsymbol{A}_o$ can be derived from the Law of Excluded Middle within the proof system for* $\mathrm{CTT}_{\mathrm{qe}}$.

Proof

$$\vdash \forall\, x_\epsilon \,.\, \mathsf{is\text{-}expr}^o_{\epsilon\to o}\, x_\epsilon \supset ([\![x_\epsilon]\!]_o \vee \neg[\![x_\epsilon]\!]_o) \tag{1}$$

$$\vdash \mathsf{is\text{-}expr}^o_{\epsilon\to o} \ulcorner \mathbf{A}_o \urcorner \supset ((\lambda\, x_\epsilon \,.\, [\![x_\epsilon]\!]_o) \ulcorner \mathbf{A}_o \urcorner \vee \neg((\lambda\, x_\epsilon \,.\, [\![x_\epsilon]\!]_o) \ulcorner \mathbf{A}_o \urcorner)) \tag{2}$$

$$\vdash (\lambda\, x_\epsilon \,.\, [\![x_\epsilon]\!]_o) \ulcorner \mathbf{A}_o \urcorner \vee \neg((\lambda\, x_\epsilon \,.\, [\![x_\epsilon]\!]_o) \ulcorner \mathbf{A}_o \urcorner) \tag{3}$$

$$\vdash (\lambda\, x_\epsilon \,.\, [\![x_\epsilon]\!]_o) \ulcorner \mathbf{A}_o \urcorner = \mathbf{A}_o \tag{4}$$

$$\vdash \mathbf{A}_o \vee \neg \mathbf{A}_o \tag{5}$$

(1) is the Law of Excluded Middle; (2) follows from (1) by Universal Instantiation; (3) follows from (2) by Lemma 6.5.1 and Modus Ponens; (4) follows from part 1 of Lemma 6.6.2 and the hypothesis; and (5) follows from (3) and (4) by the Equality Rules. □

9.3 Meaning Formulas

We show now that the meaning formula for $\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}$ and applications of it are provable in the proof system for CTT_{qe}. These results illustrate the power of CTT_{qe}'s facility for reasoning about the interplay of syntax and semantics.

Theorem 9.3.1 (Derivatives of Polynomial Functions)

1. $T_{\mathbb{R}} \vdash \mathsf{is\text{-}diff}_{(\iota\to\iota)\to o}\,(\lambda\,\boldsymbol{x}_\iota\,.\,\boldsymbol{x}_\iota) \wedge \mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}(\lambda\,\boldsymbol{x}_\iota\,.\,\boldsymbol{x}_\iota) = \lambda\,\boldsymbol{x}_\iota\,.\,1_\iota$.
2. $T_{\mathbb{R}} \vdash \mathsf{is\text{-}diff}_{(\iota\to\iota)\to o}\,(\lambda\,\boldsymbol{x}_\iota\,.\,\boldsymbol{y}_\iota) \wedge \mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}(\lambda\,\boldsymbol{x}_\iota\,.\,\boldsymbol{y}_\iota) = \lambda\,\boldsymbol{x}_\iota\,.\,0_\iota$
   where $\boldsymbol{x}_\iota$ *and* $\boldsymbol{y}_\iota$ *are distinct.*
3. $T_{\mathbb{R}} \vdash \mathsf{is\text{-}diff}_{(\iota\to\iota)\to o}\,(\lambda\,\boldsymbol{x}_\iota\,.\,\boldsymbol{c}_\iota) \wedge \mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}(\lambda\,\boldsymbol{x}_\iota\,.\,\boldsymbol{c}_\iota) = \lambda\,\boldsymbol{x}_\iota\,.\,0_\iota$.
4. $T_{\mathbb{R}} \vdash \mathsf{is\text{-}diff}_{(\iota\to\iota)\to o}\,\boldsymbol{F}_{\iota\to\iota} \supset$
   $(\mathsf{is\text{-}diff}_{(\iota\to\iota)\to o}\,(-_{(\iota\to\iota)\to(\iota\to\iota)}\,\boldsymbol{F}_{\iota\to\iota}) \wedge$
   $\mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}\,(-_{(\iota\to\iota)\to(\iota\to\iota)}\,\boldsymbol{F}_{\iota\to\iota}) =$
   $-_{(\iota\to\iota)\to(\iota\to\iota)}\,(\mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}\,\boldsymbol{F}_{\iota\to\iota}))$.
5. $T_{\mathbb{R}} \vdash \mathsf{is\text{-}diff}_{(\iota\to\iota)\to o}\,\boldsymbol{F}_{\iota\to\iota} \wedge \mathsf{is\text{-}diff}_{(\iota\to\iota)\to o}\,\boldsymbol{G}_{\iota\to\iota} \supset$
   $(\mathsf{is\text{-}diff}_{(\iota\to\iota)\to o}\,(\boldsymbol{F}_{\iota\to\iota} +_{(\iota\to\iota)\to(\iota\to\iota)\to(\iota\to\iota)}\,\boldsymbol{G}_{\iota\to\iota}) \wedge$
   $\mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}\,(\boldsymbol{F}_{\iota\to\iota} +_{(\iota\to\iota)\to(\iota\to\iota)\to(\iota\to\iota)}\,\boldsymbol{G}_{\iota\to\iota}) =$
   $(\mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}\,\boldsymbol{F}_{\iota\to\iota})$
   $+_{(\iota\to\iota)\to(\iota\to\iota)\to(\iota\to\iota)}$
   $(\mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}\,\boldsymbol{G}_{\iota\to\iota}))$.
6. $T_{\mathbb{R}} \vdash \mathsf{is\text{-}diff}_{(\iota\to\iota)\to o}\,\boldsymbol{F}_{\iota\to\iota} \wedge \mathsf{is\text{-}diff}_{(\iota\to\iota)\to o}\,\boldsymbol{G}_{\iota\to\iota} \supset$
   $(\mathsf{is\text{-}diff}_{(\iota\to\iota)\to o}\,(\boldsymbol{F}_{\iota\to\iota} *_{(\iota\to\iota)\to(\iota\to\iota)\to(\iota\to\iota)}\,\boldsymbol{G}_{\iota\to\iota}) \wedge$
   $\mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}\,(\boldsymbol{F}_{\iota\to\iota} *_{(\iota\to\iota)\to(\iota\to\iota)\to(\iota\to\iota)}\,\boldsymbol{G}_{\iota\to\iota}) =$
   $((\mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}\,\boldsymbol{F}_{\iota\to\iota}) *_{(\iota\to\iota)\to(\iota\to\iota)\to(\iota\to\iota)}\,\boldsymbol{G}_{\iota\to\iota})$
   $+_{(\iota\to\iota)\to(\iota\to\iota)\to(\iota\to\iota)}$
   $(\boldsymbol{F}_{\iota\to\iota} *_{(\iota\to\iota)\to(\iota\to\iota)\to(\iota\to\iota)}\,(\mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}\,\boldsymbol{G}_{\iota\to\iota})))$.

Proof This theorem is proved in the standard way from the definition of $\mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}$. See [112] for details. □

Lemma 9.3.2

1. $T_{\mathbb{R}} \vdash \mathsf{is\text{-}poly}_{\epsilon\to o}\,\boldsymbol{B}_\epsilon \supset \mathsf{is\text{-}expr}^{\iota}_{\epsilon\to o}\,\boldsymbol{B}_\epsilon$.
2. $T_{\mathbb{R}} \vdash \mathsf{is\text{-}poly}_{\epsilon\to o}\,\boldsymbol{B}_\epsilon \supset (\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}\,\boldsymbol{A}_\epsilon\,\boldsymbol{B}_\epsilon \supset (\boldsymbol{A}_\epsilon = \ulcorner x_\iota \urcorner \vee \boldsymbol{A}_\epsilon = \ulcorner y_\iota \urcorner))$.
3. $T_{\mathbb{R}} \vdash \mathsf{is\text{-}poly}_{\epsilon\to o}\,\boldsymbol{B}_\epsilon \supset \mathsf{is\text{-}poly}_{\epsilon\to o}\,(\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\,\boldsymbol{B}_\epsilon\,\ulcorner \boldsymbol{x}_\iota \urcorner)$.
4. $T_{\mathbb{R}} \vdash \mathsf{is\text{-}poly}_{\epsilon\to o}\,\boldsymbol{B}_\epsilon \supset (\lambda\,u_\epsilon\,.\,[\![u_\epsilon]\!]_\iota)\,\boldsymbol{B}_\epsilon = [\![\boldsymbol{B}_\epsilon]\!]_\iota$.
5. $T_{\mathbb{R}} \vdash \mathsf{is\text{-}poly}_{\epsilon\to o}\,u_\epsilon \supset (\lambda\,z_\epsilon\,.\,[\![u_\epsilon]\!]_\iota)\,\boldsymbol{B}_\epsilon = [\![u_\epsilon]\!]_\iota$.
6. $T_{\mathbb{R}} \vdash \mathsf{is\text{-}poly}_{\epsilon\to o}\,\boldsymbol{B}_\epsilon \supset$
   $(\lambda\,u_\epsilon\,.\,[\![\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\,u_\epsilon\,\ulcorner \boldsymbol{x}_\iota \urcorner]\!]_\iota)\,\boldsymbol{B}_\epsilon = [\![\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\,\boldsymbol{B}_\epsilon\,\ulcorner \boldsymbol{x}_\iota \urcorner]\!]_\iota$.

7. $T_{\mathbb{R}} \vdash \mathsf{is\text{-}poly}_{\epsilon\to o}\, u_\epsilon \supset$
$(\lambda\, z_\epsilon \,.\, [\![\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, u_\epsilon \ulcorner \boldsymbol{x}_\iota \urcorner]\!]_\iota)\, \boldsymbol{B}_\epsilon = [\![\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, u_\epsilon \ulcorner \boldsymbol{x}_\iota \urcorner]\!]_\iota$.

Proof

Parts 1–3 Follow straightforwardly from the definitions of $\mathsf{is\text{-}poly}_{\epsilon\to o}$ and $\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}$ and the axioms for $\mathsf{is\text{-}var}^\iota_{\epsilon\to o}$, $\mathsf{is\text{-}con}^\iota_{\epsilon\to o}$, $\mathsf{is\text{-}expr}^\iota_{\epsilon\to o}$, $\sqsubset_{\epsilon\to\epsilon\to o}$, and $\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o}$ (Axioms B1.1–4, B2.1–4, B3.1–9, B5.1–6, and B7.1–8) by induction using the Induction Principle for Constructions (Axiom B6).

Part 4 Follows from parts 1 and 2 of the lemma and part 2 of Lemma 6.6.2.

Part 5 Follows from parts 1 and 2 of the lemma and part 3 of Lemma 6.6.2.

Part 6 Let $\mathbf{A}_o$ be $\mathsf{is\text{-}poly}_{\epsilon\to o}\, \mathbf{B}_\epsilon$.

$$T_{\mathbb{R}} \vdash (\lambda\, u_\epsilon \,.\, \mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, u_\epsilon \ulcorner \mathbf{x}_\iota \urcorner)\, \mathbf{B}_\epsilon = \mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, \mathbf{B}_\epsilon \ulcorner \mathbf{x}_\iota \urcorner \tag{1}$$

$$T_{\mathbb{R}}, \{\mathbf{A}_o\} \vdash \mathsf{is\text{-}expr}^\iota_{\epsilon\to o}\, ((\lambda\, u_\epsilon \,.\, \mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, u_\epsilon \ulcorner \mathbf{x}_\iota \urcorner)\, \mathbf{B}_\epsilon) \tag{2}$$

$$T_{\mathbb{R}}, \{\mathbf{A}_o\} \vdash \neg(\mathsf{is\text{-}free\text{-}in}_{\epsilon\to\epsilon\to o} \ulcorner u_\epsilon \urcorner\, ((\lambda\, u_\epsilon \,.\, \mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, u_\epsilon \ulcorner \mathbf{x}_\iota \urcorner)\, \mathbf{B}_\epsilon)) \tag{3}$$

$$T_{\mathbb{R}}, \{\mathbf{A}_o\} \vdash (\lambda\, u_\epsilon \,.\, [\![\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, u_\epsilon \ulcorner \mathbf{x}_\iota \urcorner]\!]_\iota)\, \mathbf{B}_\epsilon = [\![(\lambda\, u_\epsilon \,.\, \mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, u_\epsilon \ulcorner \mathbf{x}_\iota \urcorner)\, \mathbf{B}_\epsilon]\!]_\iota \tag{4}$$

$$T_{\mathbb{R}}, \{\mathbf{A}_o\} \vdash (\lambda\, u_\epsilon \,.\, [\![\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, u_\epsilon \ulcorner \mathbf{x}_\iota \urcorner]\!]_\iota)\, \mathbf{B}_\epsilon = [\![\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, \mathbf{B}_\epsilon \ulcorner \mathbf{x}_\iota \urcorner]\!]_\iota \tag{5}$$

$$T_{\mathbb{R}} \vdash \mathbf{A}_o \supset (\lambda\, u_\epsilon \,.\, [\![\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, u_\epsilon \ulcorner \mathbf{x}_\iota \urcorner]\!]_\iota)\, \mathbf{B}_\epsilon = [\![\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, \mathbf{B}_\epsilon \ulcorner \mathbf{x}_\iota \urcorner]\!]_\iota \tag{6}$$

(1) is by Beta-Reduction by Substitution; (2) and (3) follow from (1), the hypothesis $\mathbf{A}_o$, and parts 1–3 of the lemma by the Equality Rules and propositional logic; (4) follows from (2), (3), and Axiom B11.2 by Modus Ponens; (5) follows from (1) and (4) by the Equality Rules; and (6), the lemma to be proved, follows from (5) by the Deduction Theorem.

Part 7 Similar to the proof of part 6. □

Theorem 9.3.3 (Meaning Formula for $\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}$)

$$T_{\mathbb{R}} \vdash \forall\, u_\epsilon \,.\, (\mathsf{is\text{-}poly}_{\epsilon\to o}\, u_\epsilon \supset \mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}(\lambda\, \boldsymbol{x}_\iota \,.\, [\![u_\epsilon]\!]_\iota) = \lambda\, \boldsymbol{x}_\iota \,.\, [\![\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, u_\epsilon \ulcorner \boldsymbol{x}_\iota \urcorner]\!]_\iota),$$

where $\boldsymbol{x}_\iota$ is either x_ι or y_ι.

Proof Without loss of generality, we may assume that $\mathbf{x}_\iota$ is x_ι; the proof is exactly the same when $\mathbf{x}_\iota$ is y_ι. For this proof we make the following notational definitions:

1. $\mathbf{C}_o$ is

$$\mathsf{is\text{-}diff}_{(\iota\to\iota)\to o}\, (\lambda\, x_\iota \,.\, [\![u_\epsilon]\!]_\iota) \wedge \mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}(\lambda\, x_\iota \,.\, [\![u_\epsilon]\!]_\iota) = \lambda\, x_\iota \,.\, [\![\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, u_\epsilon \ulcorner x_\iota \urcorner]\!]_\iota.$$

2. $\mathbf{P}_{\epsilon\to o}$ is $\lambda\, u_\epsilon\, .\, (\mathsf{is\text{-}poly}_{\epsilon\to o}\, u_\epsilon \supset \mathbf{C}_o)$.

3. $\mathbf{D}_o$ is $\forall\, z_\epsilon\, .\, (z_\epsilon \sqsubset_{\epsilon\to\epsilon\to o} u_\epsilon \supset \mathbf{P}_{\epsilon\to o}\, z_\epsilon)$.

4. $\mathbf{E}^1_o$ is $\mathsf{is\text{-}poly}_{\epsilon\to o}\, v_\epsilon \wedge u_\epsilon = \ulcorner -_{\iota\to\iota}\, \lfloor v_\epsilon \rfloor \urcorner$.

5. $\mathbf{E}^2_o$ is $\mathsf{is\text{-}poly}_{\epsilon\to o}\, v_\epsilon \wedge \mathsf{is\text{-}poly}_{\epsilon\to o}\, w_\epsilon \wedge u_\epsilon = \ulcorner \lfloor v_\epsilon \rfloor +_{\iota\to\iota\to\iota} \lfloor w_\epsilon \rfloor \urcorner$.

6. $\mathbf{E}^3_o$ is $\mathsf{is\text{-}poly}_{\epsilon\to o}\, v_\epsilon \wedge \mathsf{is\text{-}poly}_{\epsilon\to o}\, w_\epsilon \wedge u_\epsilon = \ulcorner \lfloor v_\epsilon \rfloor *_{\iota\to\iota\to\iota} \lfloor w_\epsilon \rfloor \urcorner$.

We will prove the following statement, a stronger result that immediately implies the theorem:

$$T_{\mathbb{R}} \vdash \forall\, u_\epsilon\, .\, \mathbf{P}_{\epsilon\to o}\, u_\epsilon.$$

Our proof is given by the following derivation:

$$T_{\mathbb{R}} \vdash \neg\mathsf{IS\text{-}EFFECTIVE\text{-}IN}(u_\iota, \mathbf{P}_{\epsilon\to o}) \tag{1}$$
$$T_{\mathbb{R}} \vdash \neg\mathsf{IS\text{-}EFFECTIVE\text{-}IN}(z_\iota, \mathbf{P}_{\epsilon\to o}) \tag{2}$$
$$T_{\mathbb{R}} \vdash \forall\, u_\epsilon\, .\, (\mathbf{D}_o \supset \mathbf{P}_{\epsilon\to o}\, u_\epsilon) \supset \forall\, u_\epsilon\, .\, \mathbf{P}_{\epsilon\to o}\, u_\epsilon \tag{3}$$
$$T_{\mathbb{R}}, \{\mathbf{D}_o, u_\epsilon = \ulcorner x_\iota \urcorner\} \vdash \mathbf{C}_o \tag{4}$$
$$T_{\mathbb{R}}, \{\mathbf{D}_o, u_\epsilon = \ulcorner y_\iota \urcorner\} \vdash \mathbf{C}_o \tag{5}$$
$$T_{\mathbb{R}}, \{\mathbf{D}_o, u_\epsilon = \ulcorner 0_\iota \urcorner\} \vdash \mathbf{C}_o \tag{6}$$
$$T_{\mathbb{R}}, \{\mathbf{D}_o, u_\epsilon = \ulcorner 1_\iota \urcorner\} \vdash \mathbf{C}_o \tag{7}$$
$$T_{\mathbb{R}}, \{\mathbf{D}_o, \exists\, v_\epsilon\, .\, \mathbf{E}^1_o\} \vdash \mathbf{C}_o \tag{8}$$
$$T_{\mathbb{R}}, \{\mathbf{D}_o, \exists\, v_\epsilon\, .\, \exists\, w_\epsilon\, .\, \mathbf{E}^2_o\} \vdash \mathbf{C}_o \tag{9}$$
$$T_{\mathbb{R}}, \{\mathbf{D}_o, \exists\, v_\epsilon\, .\, \exists\, w_\epsilon\, .\, \mathbf{E}^3_o\} \vdash \mathbf{C}_o \tag{10}$$
$$T_{\mathbb{R}}, \{\mathbf{D}_o, \mathsf{is\text{-}poly}_{\epsilon\to o}\, u_\epsilon\} \vdash \mathbf{C}_o \tag{11}$$
$$T_{\mathbb{R}} \vdash \mathbf{D}_o \supset (\mathsf{is\text{-}poly}_{\epsilon\to o}\, u_\epsilon \supset \mathbf{C}_o) \tag{12}$$
$$T_{\mathbb{R}} \vdash \mathbf{D}_o \supset \mathbf{P}_{\epsilon\to o}\, u_\epsilon \tag{13}$$
$$T_{\mathbb{R}} \vdash \forall\, u_\epsilon\, .\, (\mathbf{D}_o \supset \mathbf{P}_{\epsilon\to o}\, u_\epsilon) \tag{14}$$
$$T_{\mathbb{R}} \vdash \forall\, u_\epsilon\, .\, \mathbf{P}_{\epsilon\to o}\, u_\epsilon \tag{15}$$

(1) follows from the definition of IS-EFFECTIVE-IN using Axiom A4.6; (2) follows the definition of IS-EFFECTIVE-IN using part 5 of Lemma 9.3.2; (3) follows from (1), (2), Axiom B6, the Induction Principle for Constructions, Alpha-Equivalence, Universal Generalization, and Universal Instantiation; (4)–(10) are proved below; (11) follows from (4)–(10) by the definition of $\mathsf{is\text{-}poly}_{\epsilon\to o}$ and propositional logic (proof by cases); (12) follows from (11) by the Deduction Theorem; (13) follows from (12) by Beta-Reduction by Substitution and the Equality Rules; (14) follows from (13) by Universal Generalization; and (15) follows from (14) and (3) by Modus Ponens.

Proof of (4)

$$T_{\mathbb{R}} \vdash \mathsf{is\text{-}diff}_{(\iota\to\iota)\to o}\,(\lambda\, x_\iota \,.\, x_\iota) \wedge \mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}(\lambda\, x_\iota \,.\, x_\iota) = \lambda\, x_\iota \,.\, 1_\iota \tag{16}$$

$$T_{\mathbb{R}} \vdash [\![\ulcorner x_\iota \urcorner]\!]_\iota = x_\iota \tag{17}$$

$$T_{\mathbb{R}} \vdash [\![\ulcorner 1_\iota \urcorner]\!]_\iota = 1_\iota \tag{18}$$

$$T_{\mathbb{R}} \vdash \mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon} \ulcorner x_\iota \urcorner \ulcorner x_\iota \urcorner = \ulcorner 1_\iota \urcorner \tag{19}$$

$$\begin{aligned} T_{\mathbb{R}} \vdash\ & \mathsf{is\text{-}diff}_{(\iota\to\iota)\to o}\,(\lambda\, x_\iota \,.\, [\![\ulcorner x_\iota \urcorner]\!]_\iota) \wedge \\ & \mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}(\lambda\, x_\iota \,.\, [\![\ulcorner x_\iota \urcorner]\!]_\iota) = \lambda\, x_\iota \,.\, [\![\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon} \ulcorner x_\iota \urcorner \ulcorner x_\iota \urcorner]\!]_\iota \end{aligned} \tag{20}$$

$$\begin{aligned} T_{\mathbb{R}}, \{\mathbf{D}_o, u_\epsilon = \ulcorner x_\iota \urcorner\} \vdash\ & \\ & \mathsf{is\text{-}diff}_{(\iota\to\iota)\to o}\,(\lambda\, x_\iota \,.\, [\![\ulcorner x_\iota \urcorner]\!]_\iota) \wedge \\ & \mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}(\lambda\, x_\iota \,.\, [\![\ulcorner x_\iota \urcorner]\!]_\iota) = \lambda\, x_\iota \,.\, [\![\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon} \ulcorner x_\iota \urcorner \ulcorner x_\iota \urcorner]\!]_\iota \end{aligned} \tag{21}$$

$$T_{\mathbb{R}} \vdash \neg \mathsf{IS\text{-}EFFECTIVE\text{-}IN}(x_\iota, u_\epsilon = \ulcorner x_\iota \urcorner) \tag{22}$$

$$T_{\mathbb{R}}, \{\mathbf{D}_o, u_\epsilon = \ulcorner x_\iota \urcorner\} \vdash \mathbf{C}_o \tag{23}$$

(16) is an instance of part 1 of Theorem 9.3.1; (17) and (18) are by the Syntactic Law of Disquotation; (19) follows from the definition of $\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}$ by Universal Generalization, Universal Instantiation, Axiom B1.2, and Modus Ponens; (20) follows from (16)–(19) by the Equality Rules; (21) follows from (20) by Weakening; (22) is by Axiom B12; and (23) follows from (21), (22), and the hypothesis $u_\epsilon = \ulcorner x_\iota \urcorner$ by Rule R′.

Proof of (5) Similar to the proof of (4).

Proof of (6) Similar to the proof of (4).

Proof of (7) Similar to the proof of (4).

Proof of (8) Similar to the proof of (9).

Proof of (9)

$$\begin{aligned} T_{\mathbb{R}} \vdash\ & \mathsf{is\text{-}diff}_{(\iota\to\iota)\to o}\,(\lambda\, x_\iota \,.\, [\![v_\epsilon]\!]_\iota) \wedge \mathsf{is\text{-}diff}_{(\iota\to\iota)\to o}\,(\lambda\, x_\iota \,.\, [\![w_\epsilon]\!]_\iota) \supset \\ & (\mathsf{is\text{-}diff}_{(\iota\to\iota)\to o}\,(\lambda\, x_\iota \,.\, [\![v_\epsilon]\!]_\iota +_{(\iota\to\iota)\to(\iota\to\iota)\to(\iota\to\iota)} \lambda\, x_\iota \,.\, [\![w_\epsilon]\!]_\iota) \wedge \\ & \mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}\,(\lambda\, x_\iota \,.\, [\![v_\epsilon]\!]_\iota +_{(\iota\to\iota)\to(\iota\to\iota)\to(\iota\to\iota)} \lambda\, x_\iota \,.\, [\![w_\epsilon]\!]_\iota) = \\ & (\mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}\,\lambda\, x_\iota \,.\, [\![v_\epsilon]\!]_\iota) \\ & \quad +_{(\iota\to\iota)\to(\iota\to\iota)\to(\iota\to\iota)} \\ & (\mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}\,\lambda\, x_\iota \,.\, [\![w_\epsilon]\!]_\iota)) \end{aligned} \tag{24}$$

$$
\begin{aligned}
&T_{\mathbb{R}}, \{\mathbf{D}_o, \mathbf{E}^2_o\} \vdash \\
&\quad \mathsf{is\text{-}diff}_{(\iota\to\iota)\to o}\,(\lambda\, x_\iota \,.\, [\![v_\epsilon]\!]_\iota) \wedge \mathsf{is\text{-}diff}_{(\iota\to\iota)\to o}\,(\lambda\, x_\iota \,.\, [\![w_\epsilon]\!]_\iota) \supset \\
&\qquad (\mathsf{is\text{-}diff}_{(\iota\to\iota)\to o}\,(\lambda\, x_\iota \,.\, [\![v_\epsilon]\!]_\iota +_{(\iota\to\iota)\to(\iota\to\iota)\to(\iota\to\iota)} \lambda\, x_\iota \,.\, [\![w_\epsilon]\!]_\iota) \wedge \\
&\qquad\quad \mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}\,(\lambda\, x_\iota \,.\, [\![v_\epsilon]\!]_\iota +_{(\iota\to\iota)\to(\iota\to\iota)\to(\iota\to\iota)} \lambda\, x_\iota \,.\, [\![w_\epsilon]\!]_\iota) = \\
&\qquad\quad (\mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}\,\lambda\, x_\iota \,.\, [\![v_\epsilon]\!]_\iota) \\
&\qquad\qquad +_{(\iota\to\iota)\to(\iota\to\iota)\to(\iota\to\iota)} \\
&\qquad\quad (\mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}\,\lambda\, x_\iota \,.\, [\![w_\epsilon]\!]_\iota))
\end{aligned} \tag{25}
$$

$$
T_{\mathbb{R}}, \{\mathbf{D}_o, \mathbf{E}^2_o\} \vdash v_\epsilon \sqsubset u_\epsilon \tag{26}
$$

$$
T_{\mathbb{R}}, \{\mathbf{D}_o, \mathbf{E}^2_o\} \vdash w_\epsilon \sqsubset u_\epsilon \tag{27}
$$

$$
T_{\mathbb{R}}, \{\mathbf{D}_o, \mathbf{E}^2_o\} \vdash \mathbf{P}_{\epsilon\to o}\, v_\epsilon \tag{28}
$$

$$
T_{\mathbb{R}}, \{\mathbf{D}_o, \mathbf{E}^2_o\} \vdash \mathbf{P}_{\epsilon\to o}\, w_\epsilon \tag{29}
$$

$$
\begin{aligned}
&T_{\mathbb{R}}, \{\mathbf{D}_o, \mathbf{E}^2_o\} \vdash \\
&\quad \mathsf{is\text{-}diff}_{(\iota\to\iota)\to o}\,(\lambda\, x_\iota \,.\, [\![v_\epsilon]\!]_\iota) \wedge \\
&\quad \mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}(\lambda\, x_\iota \,.\, [\![v_\epsilon]\!]_\iota) = \lambda\, x_\iota \,.\, [\![\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, v_\epsilon\, {}^\ulcorner x_\iota {}^\urcorner]\!]_\iota
\end{aligned} \tag{30}
$$

$$
\begin{aligned}
&T_{\mathbb{R}}, \{\mathbf{D}_o, \mathbf{E}^2_o\} \vdash \\
&\quad \mathsf{is\text{-}diff}_{(\iota\to\iota)\to o}\,(\lambda\, x_\iota \,.\, [\![w_\epsilon]\!]_\iota) \wedge \\
&\quad \mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}(\lambda\, x_\iota \,.\, [\![w_\epsilon]\!]_\iota) = \lambda\, x_\iota \,.\, [\![\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, w_\epsilon\, {}^\ulcorner x_\iota {}^\urcorner]\!]_\iota
\end{aligned} \tag{31}
$$

$$
\begin{aligned}
&T_{\mathbb{R}}, \{\mathbf{D}_o, \mathbf{E}^2_o\} \vdash \\
&\quad \mathsf{is\text{-}diff}_{(\iota\to\iota)\to o}\,(\lambda\, x_\iota \,.\, [\![v_\epsilon]\!]_\iota +_{(\iota\to\iota)\to(\iota\to\iota)\to(\iota\to\iota)} \lambda\, x_\iota \,.\, [\![w_\epsilon]\!]_\iota) \wedge \\
&\quad \mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}\,(\lambda\, x_\iota \,.\, [\![v_\epsilon]\!]_\iota +_{(\iota\to\iota)\to(\iota\to\iota)\to(\iota\to\iota)} \lambda\, x_\iota \,.\, [\![w_\epsilon]\!]_\iota) = \\
&\quad (\mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}\,\lambda\, x_\iota \,.\, [\![v_\epsilon]\!]_\iota) \\
&\qquad +_{(\iota\to\iota)\to(\iota\to\iota)\to(\iota\to\iota)} \\
&\quad (\mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}\,\lambda\, x_\iota \,.\, [\![w_\epsilon]\!]_\iota)
\end{aligned} \tag{32}
$$

$$
\begin{aligned}
&T_{\mathbb{R}}, \{\mathbf{D}_o, \mathbf{E}^2_o\} \vdash \\
&\quad \mathsf{is\text{-}diff}_{(\iota\to\iota)\to o}\,(\lambda\, x_\iota \,.\, [\![v_\epsilon]\!]_\iota +_{(\iota\to\iota)\to(\iota\to\iota)\to(\iota\to\iota)} \lambda\, x_\iota \,.\, [\![w_\epsilon]\!]_\iota) \wedge \\
&\quad \mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}\,(\lambda\, x_\iota \,.\, [\![v_\epsilon]\!]_\iota +_{(\iota\to\iota)\to(\iota\to\iota)\to(\iota\to\iota)} \lambda\, x_\iota \,.\, [\![w_\epsilon]\!]_\iota) = \\
&\quad \lambda\, x_\iota \,.\, [\![\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, v_\epsilon\, {}^\ulcorner x_\iota {}^\urcorner]\!]_\iota \\
&\qquad +_{(\iota\to\iota)\to(\iota\to\iota)\to(\iota\to\iota)} \\
&\quad \lambda\, x_\iota \,.\, [\![\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, w_\epsilon\, {}^\ulcorner x_\iota {}^\urcorner]\!]_\iota
\end{aligned} \tag{33}
$$

$$T_{\mathbb{R}}, \{\mathbf{D}_o, \mathbf{E}_o^2\} \vdash$$
$$\quad \mathsf{is\text{-}diff}_{(\iota\to\iota)\to o}\,(\lambda\, x_\iota\,.\,([\![v_\epsilon]\!]_\iota +_{\iota\to\iota\to\iota} [\![w_\epsilon]\!]_\iota)) \wedge$$
$$\quad \mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}\,(\lambda\, x_\iota\,.\,([\![v_\epsilon]\!]_\iota +_{\iota\to\iota\to\iota} [\![w_\epsilon]\!]_\iota)) =$$
$$\quad \lambda\, x_\iota\,.\,([\![\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, v_\epsilon\, \ulcorner x_\iota \urcorner]\!]_\iota +_{\iota\to\iota\to\iota} [\![\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, w_\epsilon\, \ulcorner x_\iota \urcorner]\!]_\iota) \tag{34}$$

$$T_{\mathbb{R}}, \{\mathbf{D}_o, \mathbf{E}_o^2\} \vdash$$
$$\quad \mathsf{is\text{-}diff}_{(\iota\to\iota)\to o}\,(\lambda\, x_\iota\,.\,([\![\ulcorner \lfloor v_\epsilon \rfloor +_{\iota\to\iota\to\iota} \lfloor w_\epsilon \rfloor \urcorner]\!]_\iota)) \wedge$$
$$\quad \mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}\,(\lambda\, x_\iota\,.\,([\![\ulcorner \lfloor v_\epsilon \rfloor +_{\iota\to\iota\to\iota} \lfloor w_\epsilon \rfloor \urcorner]\!]_\iota)) =$$
$$\quad \lambda\, x_\iota\,.\,([\![\ulcorner \lfloor \mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, v_\epsilon\, \ulcorner x_\iota \urcorner \rfloor +_{\iota\to\iota\to\iota} \lfloor \mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, w_\epsilon\, \ulcorner x_\iota \urcorner \rfloor \urcorner]\!]_\iota) \tag{35}$$

$$T_{\mathbb{R}}, \{\mathbf{D}_o, \mathbf{E}_o^2\} \vdash$$
$$\quad \mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, \ulcorner \lfloor v_\epsilon \rfloor +_{\iota\to\iota\to\iota} \lfloor w_\epsilon \rfloor \urcorner\, \ulcorner x_\iota \urcorner =$$
$$\quad \ulcorner \lfloor \mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, v_\epsilon\, \ulcorner x_\iota \urcorner \rfloor +_{\iota\to\iota\to\iota} \lfloor \mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, w_\epsilon\, \ulcorner x_\iota \urcorner \rfloor \urcorner \tag{36}$$

$$T_{\mathbb{R}}, \{\mathbf{D}_o, \mathbf{E}_o^2\} \vdash$$
$$\quad \mathsf{is\text{-}diff}_{(\iota\to\iota)\to o}\,(\lambda\, x_\iota\,.\,([\![\ulcorner \lfloor v_\epsilon \rfloor +_{\iota\to\iota\to\iota} \lfloor w_\epsilon \rfloor \urcorner]\!]_\iota)) \wedge$$
$$\quad \mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}\,(\lambda\, x_\iota\,.\,([\![\ulcorner \lfloor v_\epsilon \rfloor +_{\iota\to\iota\to\iota} \lfloor w_\epsilon \rfloor \urcorner]\!]_\iota)) =$$
$$\quad \lambda\, x_\iota\,.\,([\![\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, \ulcorner \lfloor v_\epsilon \rfloor +_{\iota\to\iota\to\iota} \lfloor w_\epsilon \rfloor \urcorner\, \ulcorner x_\iota \urcorner]\!]_\iota) \tag{37}$$

$$T_{\mathbb{R}}, \{\mathbf{D}_o, \mathbf{E}_o^2\} \vdash \neg \mathsf{IS\text{-}EFFECTIVE\text{-}IN}(x_\iota, u_\epsilon = \ulcorner \lfloor v_\epsilon \rfloor +_{\iota\to\iota\to\iota} \lfloor w_\epsilon \rfloor \urcorner) \tag{38}$$

$$T_{\mathbb{R}}, \{\mathbf{D}_o, \mathbf{E}_o^2\} \vdash \mathbf{C}_o \tag{39}$$

$$T_{\mathbb{R}}, \{\mathbf{D}_o, \mathbf{E}_o^2\} \vdash \neg \mathsf{IS\text{-}EFFECTIVE\text{-}IN}(v_\epsilon, \mathbf{C}_o) \tag{40}$$

$$T_{\mathbb{R}}, \{\mathbf{D}_o, \mathbf{E}_o^2\} \vdash \neg \mathsf{IS\text{-}EFFECTIVE\text{-}IN}(w_\epsilon, \mathbf{C}_o) \tag{41}$$

$$T_{\mathbb{R}}, \{\mathbf{D}_o, \mathbf{E}_o^2\} \vdash \neg \mathsf{IS\text{-}EFFECTIVE\text{-}IN}(v_\epsilon, \mathbf{D}_o) \tag{42}$$

$$T_{\mathbb{R}}, \{\mathbf{D}_o, \mathbf{E}_o^2\} \vdash \neg \mathsf{IS\text{-}EFFECTIVE\text{-}IN}(w_\epsilon, \mathbf{D}_o) \tag{43}$$

$$T_{\mathbb{R}}, \{\mathbf{D}_o, \exists\, v_\epsilon\,.\,\exists\, w_\epsilon\,.\,\mathbf{E}_o^2\} \vdash \mathbf{C}_o \tag{44}$$

(24) is an instance of part 5 of Theorem 9.3.1; (25) follows from (24) by Weakening; (26) and (27) follow from the hypothesis $\mathbf{E}_o^2$ and Axioms B5.1–6 by Universal Generalization, Universal Instantiation, the Equality Rules and propositional logic; (28) and (29) follow from (26), (27), and the hypothesis $\mathbf{D}_o$ by Universal Instantiation, parts 5 and 7 of Lemma 9.3.2, the Equality Rules, and propositional logic; (30) and (31) follow from (28), (29), and the hypothesis $\mathbf{E}_o^2$ by Universal Instantiation, parts 4 and 6 of Lemma 9.3.2, the Equality Rules, and propositional logic; (32) follows from (25), (30), and (31) by propositional logic; (33) follows from (30)–(32) by the Equality Rules and propositional logic; (34) follows from (33) by the definition of $+_{(\iota\to\iota)\to(\iota\to\iota)\to(\iota\to\iota)}$; (35) follows from (34) and the hypothesis $\mathbf{E}_o^2$ by Axiom B10.3, Lemma 9.3.2, quasiquotation, and propositional logic; (36) follows from the definition of $\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}$ and the hypothesis $\mathbf{E}_o^2$ by Universal Generalization, Universal Instantiation, Axiom B1.2, and propositional logic; (37) follows from (35) and (36) by the Equality Rules; (38) is by Axiom B12; (39) follows from the hypothesis $\mathbf{E}_o^2$, (37), and (38) by Rule R′ and propositional logic; (40)–(43) follow from the hypothesis $\mathbf{E}_o^2$ and the definition of IS-EFFECTIVE by Beta-Reduction by Substitution, part 4 of Lemma 6.6.2 and Lemma 9.3.2; and (44) follows from (41)–(43) by Lemma 6.3.7.

Proof of (10) Similar to the proof of (9).

This finally completes the proof of Theorem 9.3.3. □

Proposition 9.3.4

$$T_{\mathbb{R}} \vdash \mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}(\lambda\, x_\iota \,.\, x_\iota^2) = \lambda\, x_\iota \,.\, 2_\iota * x_\iota.$$

Proof

$$T_{\mathbb{R}} \vdash \forall\, u_\epsilon \,.\, (\mathsf{is\text{-}poly}_{\epsilon\to o}\, u_\epsilon \supset \mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}(\lambda\, x_\iota \,.\, [\![u_\epsilon]\!]_\iota) = \lambda\, x_\iota \,.\, [\![\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, u_\epsilon\, \ulcorner x_\iota \urcorner]\!]_\iota) \quad (1)$$

$$T_{\mathbb{R}} \vdash \mathsf{is\text{-}poly}_{\epsilon\to o}\, \ulcorner x_\iota^2 \urcorner \supset \mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}(\lambda\, x_\iota \,.\, ((\lambda\, u_\epsilon \,.\, [\![u_\epsilon]\!]_\iota)\, \ulcorner x_\iota^2 \urcorner)) = \lambda\, x_\iota \,.\, ((\lambda\, u_\epsilon \,.\, [\![\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, u_\epsilon\, \ulcorner x_\iota \urcorner]\!]_\iota)\, \ulcorner x_\iota^2 \urcorner)) \quad (2)$$

$$T_{\mathbb{R}} \vdash \mathsf{is\text{-}poly}_{\epsilon\to o}\, \ulcorner x_\iota^2 \urcorner \quad (3)$$

$$T_{\mathbb{R}} \vdash \mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}(\lambda\, x_\iota \,.\, ((\lambda\, u_\epsilon \,.\, [\![u_\epsilon]\!]_\iota)\, \ulcorner x_\iota^2 \urcorner)) = \lambda\, x_\iota \,.\, ((\lambda\, u_\epsilon \,.\, [\![\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, u_\epsilon\, \ulcorner x_\iota \urcorner]\!]_\iota)\, \ulcorner x_\iota^2 \urcorner) \quad (4)$$

$$T_{\mathbb{R}} \vdash (\lambda\, u_\epsilon \,.\, [\![u_\epsilon]\!]_\iota)\, \ulcorner x_\iota^2 \urcorner = [\![\ulcorner x_\iota^2 \urcorner]\!]_\iota \quad (5)$$

$$T_{\mathbb{R}} \vdash (\lambda\, u_\epsilon \,.\, [\![\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, u_\epsilon\, \ulcorner x_\iota \urcorner]\!]_\iota)\, \ulcorner x_\iota^2 \urcorner = [\![\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, \ulcorner x_\iota^2 \urcorner\, \ulcorner x_\iota \urcorner]\!]_\iota \quad (6)$$

$$T_{\mathbb{R}} \vdash \mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}\, \ulcorner x_\iota^2 \urcorner\, \ulcorner x_\iota \urcorner = \ulcorner 2_\iota * x_\iota \urcorner \quad (7)$$

$$T_{\mathbb{R}} \vdash [\![\ulcorner x_\iota^2 \urcorner]\!]_\iota = x_\iota^2 \quad (8)$$

$$T_{\mathbb{R}} \vdash [\![\ulcorner 2_\iota * x_\iota \urcorner]\!]_\iota = 2_\iota * x_\iota \quad (9)$$

$$T_{\mathbb{R}} \vdash \mathsf{deriv}_{(\iota\to\iota)\to(\iota\to\iota)}(\lambda\, x_\iota \,.\, x_\iota^2) = \lambda\, x_\iota \,.\, 2_\iota * x_\iota \quad (10)$$

(1) is Theorem 9.3.3; (2) follows from (1) by Universal Instantiation; (3) follows from the definition of $\mathsf{is\text{-}poly}_{\epsilon\to o}$ by Beta-Reduction by Substitution, Existential Generalization, and propositional logic; (4) follows from (3) and (2) by Modus Ponens; (5) follows from (3) by part 4 of Lemma 9.3.2 by Modus Ponens; (6) follows from (3) and part 6 of Lemma 9.3.2 by Modus Ponens; (7) follows from the definitions of $\mathsf{is\text{-}poly}_{\epsilon\to o}$ and $\mathsf{poly\text{-}diff}_{\epsilon\to\epsilon\to\epsilon}$ by Universal Generalization, Universal Instantiation, Axiom B1.2, and propositional logic; (8) and (9) are by the Syntactic Law of Disquotation; and (10) follows from (4)–(9) by the Equality Rules. □

10 Related Work

10.1 Metareasoning with Reflection

Metareasoning is reasoning about the behavior of a reasoning system such as a proof system for a logic. Metareasoning about the proof system for a *logic* L is performed in a proof system for a *metalogic* M where M may be L itself. Since a proof system involves manipulating expressions as syntactic objects, metareasoning starts with reasoning about syntax. This can be done in a number of ways. Kurt Gödel's famously used *Gödel numbers* in [66] to encode expressions and thereby reduce reasoning about expressions to reasoning about natural numbers. The technique of a *deep embedding* — in which a particular language of expressions is represented by an inductive type of values — is the most common means used today to reason about the syntax of a language [13, 39, 124].

Metareasoning is the most interesting when the metalogic M is the same as the logic L and reasoning about L's proof system is integrated into reasoning within L's proof system. This is commonly called *reflection*. The integration of meta-level reasoning in object-level reasoning requires some form of *quotation* and some form of *evaluation*. Stanfania Costantini presents a general survey of metareasoning and reflection in [42], and John Harrison in his excellent paper [75] surveys the applications of reflection to computer theorem proving while arguing that LCF-style proof assistants do not have an inherent need for reflection.

Harrison identifies two kinds of reflection: logical and computational. *Logical reflection* employs metareasoning about L's proof system within itself to reveal logical properties about L. Gödel, Tarski, and others have used reflection in this form to show the limits of formal logic [66, 118] and to explore the logical impact of various *reflection principles* [75, 86]. *Computational reflection* incorporates algorithms that manipulate expressions and other meta-level objects into the logic's proof system. Examples of such algorithms are the differentiation algorithm for polynomials defined in section 4 and the ring tactic in the Coq proof assistant [14, 71].

Computational reflection has been explored and exploited in several computer theorem systems. In the seminal paper [15], Robert Boyer and J Moore developed a global infrastructure for incorporating symbolic algorithms into the

Nqthm [16] theorem prover. This approach is also used in ACL2 [83], the successor to Nqthm; see [80]. Over the last 30 years, the Nuprl group lead by Robert Constable has produced a large body of work on metareasoning and reflection for theorem proving [2, 7, 37, 79, 85, 96, 125] that has been implemented in the Nuprl [38] and MetaPRL [78] systems. Proof by reflection has become a mainstream technique in the Coq [40] proof assistant with the development of tactics based on symbolic computations like the Coq ring tactic [14, 71] and the formalizations in Coq of the *four color theorem* [67] and the *Feit-Thompson odd-order theorem* [68] led by Georges Gonthier. See [14, 17, 31, 69, 71, 81, 100] for a selection of the work done on using reflection in Coq. Agda [97, 98] supports reflection in both programming and proving; see [120, 121]. Martin Giese and Bruno Buchberger present in [63] the design for a global infrastructure for employing reflection in the Theorema [22] theorem prover. See the following references for research on using metareasoning and reflection in other systems: Idris [32, 33, 34], Isabelle/HOL [30], Lean [45], Maude [36], PVS [122], and reFLect [89].

The programming language community has likewise looked at reflection. Its use for metaprogramming will be covered in the next section, but trying to come to grasp with these ideas produced some interesting papers on reflective theories [90] and reification [61]. Of particular note is that, even in a pure programming context, unrestricted reflection leads to problems [123]. Kavvos' recent D.Phil thesis [84] has a very interesting overview of the impossibility of building a quotation operator (with certain properties) and other dangers. His literature review is also quite extensive.

The Nqthm/ACL2 [15, 80] and Theorema [63] approaches to computational reflection are the approaches in the literature that are closest to CTT_{qe}. Like CTT_{qe}, these approaches utilize a global reflection infrastructure for a traditional logic.

10.2 Metaprogramming with Reflection

Metaprogramming is writing computer programs to manipulate and generate computer programs in some programming language L. Metaprogramming is especially useful when the "metaprograms" can be written in L itself. This is facilitated by implementing in L metaprogramming techniques for L that involve the manipulation of program code. See [44] for a survey of how this kind of "reflection" can be done for the major programming paradigms.

We listed in section 1 several programming languages that support metaprogramming with reflection: Lisp, Agda [97, 98, 120], Elixir [103], F# [119], Idris [32, 33, 34], MetaML [115], MetaOCaml [109], reFLect [72], Scala [99, 110], and Template Haskell [111]. These languages represent fragments of computer code as values in an inductive type and include quotation, quasiquotation, and evaluation operations. For example, these operations are called *quote*, *backquote*, and *eval* in the Lisp programming language. The metaprogramming language Archon [113] developed by Aaron Stump offers an interesting alternate approach

in which program code is manipulated directly instead of manipulating representations of computer code.

The reflection infrastructure in a programming language provides the basis for *multistage programming* [114] in which code generation and code execution are interleaved to produce programs that are both general and efficient. The code generation and execution can take place at compile-time or run-time. See [10, 23, 93] for research on developing models for multistage programming and [43, 95] for research on type systems that support multistage programming.

10.3 Theories of Quotation

The semantics of the quotation operator $\ulcorner \cdot \urcorner$ is based on the *disquotational theory of quotation* [24]. According to this theory, a quotation of an expression e is an expression that denotes e itself. In CTT_{qe}, $\ulcorner \mathbf{A}_\alpha \urcorner$ denotes a value that represents the syntactic structure of $\mathbf{A}_\alpha$. Andrew Polonsky presents in [104] a set of axioms for quotation operators of this kind. There are several other theories of quotation that have been proposed [24]. For instance, quotation can be viewed as an operation that constructs literals for syntactic values. Florian Rabe explores in [106] this approach to quotation.

10.4 Theories of Truth

Truth is a major subject in philosophy [65]. A theory of truth seeks to explain what truth is and how the liar and other related paradoxes can be resolved. A *truth predicate* [65] is the face of a *theory of truth*: the properties of a truth predicate characterize a theory of truth [87]. A *semantics theory of truth* defines a truth predicate for a formal language, while an *axiomatic theory of truth* [73, 74] specifies a truth predicate for a formal language by means of an axiomatic theory.

In CTT_{qe}, $[\![\mathbf{A}_\epsilon]\!]_o$ asserts the truth of the formula represented by $\mathbf{A}_\epsilon$, and thus the evaluation operator $[\![\cdot]\!]_o$ is a truth predicate. Hence CTT_{qe} provides a semantic theory of truth via it semantics and an axiomatic theory of truth via its proof system. Since our goal is not to explicate the nature of truth, it is not surprising that the semantic and axiomatic theories of truth provided by CTT_{qe} are not very innovative. Theories of truth — starting with Tarski's work [116, 117, 118] in the 1930s — have traditionally been restricted to the truth of sentences, i.e., formulas with no free variables. However, the CTT_{qe} semantic and axiomatic theories of truth admit formulas with free variables.

10.5 Reasoning in the Lambda Calculus about Syntax

Corrado Böhm and Alessandro Berarducci present in [11] a method for representing an inductive type of values as a collection of lambda-terms. Then functions defined on the members of the inductive type can also be represented as lambda terms. Both the lambda terms representing the values and those

representing the functions defined on the values can be typed in the second-order lambda calculus (System F) [64, 108] as shown in [11]. Böhm and his collaborators present in [9, 12] a second, more powerful method for representing inductive types as collections of lambda-terms in which the lambda terms are not as easily typeable as in the first method. These two methods provide the means to efficiently formalize syntax-based mathematical algorithms in the lambda calculus.

Using the fact that inductive types can be directly represented in the lambda calculus, Torben Æ. Mogensen in [92] represents the inductive type of lambda terms in lambda calculus itself as well as defines a global evaluation operator in the lambda calculus. (See Henk Barendregt's survey paper [6] on the impact of the lambda calculus for a nice description of this work.) Nevertheless these representations were only partially typed. The *finally tagless* approach to embedded representations [29] kicked off a series of papers on typed self-representation [4, 5, 18, 19, 20, 21, 82, 107] which eventually succeeded at providing elegant solutions.

10.6 Undefinedness

Undefinedness naturally occurs in two places in CTT_{qe}. It occurs when a syntax constructor is applied to inappropriate arguments and when the evaluation operator $[\![\cdot]\!]_\alpha$ is applied to an expression $\ulcorner \mathbf{B}_\beta \urcorner$ where $\alpha \neq \beta$. We would prefer a cleaner version of CTT_{qe} that formalizes the traditional approach to undefinedness [49]. Then improper constructions would not be needed and checking whether an expression $\mathbf{A}_\epsilon$ denotes a construction or an evaluation $[\![\mathbf{A}_\epsilon]\!]_\alpha$ is meaningful would be reduced to checking for definedness. We argue in [49] that a logic that supports the traditional approach to undefinedness is much closer to mathematical practice than traditional logics and can be effectively implemented.

We show in [51] how to formalize the traditional approach to undefinedness in a traditional logic. The paper [51] presents $\mathcal{Q}_0^{\text{u}}$, a version of Andrews' $\mathcal{Q}_0$ that takes this approach to undefinedness. $\mathcal{Q}_0^{\text{u}}$ is a simplified version of LUTINS [46, 47, 48], the logic of the IMPS theorem proving system [58, 59]. Roughly speaking, $\mathcal{Q}_0^{\text{uqe}}$ is $\mathcal{Q}_0^{\text{u}}$ plus quotation and evaluation. CTT_{uqe} [57] is a variant of CTT_{qe} in which undefinedness is incorporated in CTT_{qe} in the same way that it is incorporated in $\mathcal{Q}_0^{\text{u}}$ and $\mathcal{Q}_0^{\text{uqe}}$.

11 Conclusion

Quotation and evaluation provide a basis for metaprogramming as seen in Lisp and other programming languages. We believe that these mechanisms can also provide a basis for metareasoning in traditional logics like first-order logic or simple type theory [52]. However, incorporating quotation and evaluation into a traditional logic is much more challenging than incorporating them into a programming language due to the Evaluation, Variable, and Double Substitution

Problems we described in the Introduction.

In this paper we have introduced CTT_{qe}, a logic based on $\mathcal{Q}_0$ [3], Andrews' version of Church's type theory, that includes quotation and evaluation. We have presented the syntax and semantics of CTT_{qe} as well as a proof system for CTT_{qe}. The syntax of CTT_{qe} has the machinery of Church's type theory plus an inductive type ϵ of syntactic values, a partial quotation operator, and a typed evaluation operator. The semantics of CTT_{qe} is based on Henkin-style general models [77]. Constructions — certain expressions of type ϵ — represent the syntactic structures of eval-free expressions (i.e., expressions that do not contain the evaluation operator); they serve as the syntactic values in the semantics. The proof system for CTT_{qe} is an extension of the proof system for $\mathcal{Q}_0$. We proved that it is sound for all formulas (Requirement R1) and complete for eval-free formulas (R2). We also showed it can be used to reason about constructions (R3), can instantiate free variables occurring within evaluations (R4), and can prove formulas containing evaluations such as schemas and meaning formulas for syntax-based mathematical algorithms (R5).

The Evaluation Problem is completely avoided in CTT_{qe} by restricting the quotation operator to eval-free expressions. The Variable Problem is solved by (1) using the more restrictive semantic notion of "a variable is effective in an expression" in place of the syntactic notion of "a variable is free in an expression" and (2) adding beta-reduction axioms for quotations and evaluations to the beta-reduction axioms used by Andrews in the proof system for $\mathcal{Q}_0$ [3, p. 213]. The Double Substitution Problem is solved by not allowing beta-reductions that embody a double substitution.

Using examples, we have shown that CTT_{qe} is suitable for reasoning about the interplay of syntax and semantics, expressing quasiquotations, and stating and proving schemas and meaning formulas. In particular, we proved within the proof system for CTT_{qe} the meaning formula for an symbolic differentiation algorithm for polynomials. The proof of this result (Theorem 9.3.3) is an comprehensive test of the efficacy of CTT_{qe}'s proof system.

CTT_{qe} is much simpler than $\mathcal{Q}_0^{\text{uqe}}$ [55], a richer, but more complicated, version of $\mathcal{Q}_0$ with undefinedness, quotation, and evaluation. In $\mathcal{Q}_0^{\text{uqe}}$, quotation may be applied to expressions containing evaluations, expressions may be undefined and functions may be partial, and substitution is implemented explicitly as a logical constant. Allowing quotation to be applied to all expressions makes $\mathcal{Q}_0^{\text{uqe}}$ much more expressive than CTT_{qe} but also much more difficult to implement since substitution in the presence of evaluations is highly complex. The decision to represent "a variable is free in an expression" in the logic but represent substitution only in the metalogic gives the proof system for CTT_{qe} much greater fluency that the proof system for $\mathcal{Q}_0^{\text{uqe}}$.

We believe that CTT_{qe} is the first version of simple type theory with global quotation and evaluation that has a practicable proof system. We also believe that our approach for incorporating quotation and evaluation into Church's type theory — introducing an inductive type of constructions, a partial quotation operator, and a typed evaluation operator — can be applied to other logics including many-sorted first-order logic. We have shown that developing the

needed syntax and semantics is relatively straightforward, while developing a proof system for the logic is fraught with difficulties.

In our future research we will seek to answer the following three questions:

1. Can CTT_{qe} (or a logic like CTT_{qe} with global quotation and evaluation) be effectively implemented as a software system?

2. Is CTT_{qe} an effective logic for developing defining, applying, proving properties about syntax-based mathematical algorithms.

3. Is CTT_{qe} an effective logic for formalizing graphs of biform theories?

Since CTT_{qe} is a version of Church's type theory, the most promising approach to answering the first question is to implement CTT_{qe} by extending HOL Light [76], a simple implementation of the HOL proof assistant [70]. We have developed a system called HOL Light QE [27] by modifying HOL Light to include global quotation and evaluation operators. As future work, we intend to continue the development of HOL Light QE and to show that HOL Light QE can be effectively used to develop syntax-based mathematical algorithms.

A *biform theory* [25, 50] is a basic unit of mathematical knowledge that consists of a set of *concepts* that denote mathematical values, *transformers* that denote symbolic algorithms, and *facts* about the concepts and transformers. Since transformers manipulate the syntax of expressions, biform theories are difficult to formalize in a traditional logic. The notion of a biform theory is a key component of a framework for integrating axiomatic and algorithmic mathematics that is being developed under the MathScheme project [28] at McMaster University, led by Jacques Carette and the author. One of the main goals of the MathScheme is to see if a logic like CTT_{qe} can be used to develop a library of biform theories connected by meaning preserving theory morphisms. As part of a case study [26], we have formalized a graph of biform theories encoding natural number arithmetic in CTT_{uqe} [57], a variant of CTT_{qe} with undefinedness and theory morphisms. Our next step in this direction will be to formalize this same graph of biform theories in the HOL Light QE system we mentioned above.

Acknowledgments

The author is grateful to Marc Bender, Jacques Carette, Michael Kohlhase, Pouya Larjani, and Florian Rabe for many valuable discussions on the use of quotation and evaluation in logic. Peter Andrews deserves special thanks for writing *An Introduction to Mathematical Logic and Type Theory: To Truth through Proof* [3]. The ideas embodied in CTT_{qe} heavily depend on the presentation of $\mathcal{Q}_0$ given in this superb textbook. This research was supported by NSERC. Finally, the author would like to thank the referees for their insightful comments and suggestions.

References

[1] M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit substitution. *Journal of Functional Programming*, 1:375–416, 1991.

[2] S. F. Allen, R. L. Constable, D. J. Howe, and W. E. Aitken. The semantics of reflected proof. In *Proceedings of the Fifth Annual Symposium on Logic in Computer Science (LICS '90)*, pages 95–105. IEEE Computer Society, 1990.

[3] P. B. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth through Proof, Second Edition.* Kluwer, 2002.

[4] R. Atkey. Syntax for free: Representing syntax with binding using parametricity. In *International Conference on Typed Lambda Calculi and Applications*, pages 35–49. Springer, 2009.

[5] R. Atkey, S. Lindley, and J. Yallop. Unembedding domain-specific languages. In *Proceedings of the 2nd ACM SIGPLAN symposium on Haskell*, pages 37–48. ACM, 2009.

[6] H. Barendregt. The impact of the lambda calculus in logic and computer science. *Bulletin of Symbolic Logic*, 3:181–215, 1997.

[7] E. Barzilay. *Implementing Reflection in Nuprl.* PhD thesis, Cornell University, 2005.

[8] A. Bawden. Quasiquotation in Lisp. In O. Danvy, editor, *Proceedings of the 1999 ACM SIGPLAN Symposium on Partial Evaluation and Semantics-Based Program Manipulation*, pages 4–12, 1999. Technical report BRICS-NS-99-1, University of Aarhus, 1999.

[9] A. Berarducci and C. Böhm. A self-interpreter of lambda calculus having a normal form. In E. Börger, G. Jäger, H. Kleine Büning, S. Martini, and M. M. Richter, editors, *Computer Science Logic*, volume 702 of *Lecture Notes in Computer Science*, pages 85–99. Springer, 1993.

[10] M. Berger, L. Tratt, and C. Urban. Modelling homogeneous generative meta-programming. *Computing Research Repository (CoRR)*, abs/1602.06568, 2016.

[11] C. Böhm and A. Berarducci. Automatic synthesis of typed lambda-programs on term algebras. *Theoretical Computer Science*, 39:135–154, 1985.

[12] C. Böhm, A. Piperno, and S. Guerrini. Lambda-definition of function(al)s by normal forms. In D. Sannella, editor, *Programming Languages and Systems — ESOP'94*, volume 788 of *Lecture Notes in Computer Science*, pages 135–149. Springer, 1994.

[13] R. Boulton, A. Gordon, M. Gordon, J. Harrison, J. Herbert, and J. Van Tassel. Experience with embedding hardware description languages in HOL. In V. Stavridou, T. F. Melham, and R. T. Boute, editors, *Proceedings of the IFIP TC10/WG 10.2 International Conference on Theorem Provers in Circuit Design: Theory, Practice and Experience*, volume A-10 of *IFIP Transactions A: Computer Science and Technology*, pages 129–156. North-Holland, 1993.

[14] S. Boutin. Using reflection to build efficient and certified decision procedures. In M. Abadi and T. Ito, editors, *Theoretical Aspects of Computer Software*, volume 1281 of *Lecture Notes in Computer Science*, pages 515–529. Springer, 1997.

[15] R. Boyer and J Moore. Metafunctions: Proving them correct and using them efficiently as new proof procedures. In R. Boyer and J Moore, editors, *The Correctness Problem in Computer Science*, pages 103–185. Academic Press, 1981.

[16] R. Boyer and J Moore. *A Computational Logic Handbook*. Academic Press, 1988.

[17] T. Braibant and D. Pous. Tactics for reasoning modulo AC in Coq. In J.-P. Jouannaud and Z. Shao, editors, *Certified Programs and Proofs (CPP 2011)*, volume 7086 of *Lecture Notes in Computer Science*, pages 167–182. Springer, 2011.

[18] M. Brown and J. Palsberg. Self-representation in girard's system u. *ACM SIGPLAN Notices*, 50(1):471–484, 2015.

[19] M. Brown and J. Palsberg. Breaking through the normalization barrier: A self-interpreter for F-omega. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2016)*, pages 5–17. ACM, 2016.

[20] M. Brown and J. Palsberg. Typed self-evaluation via intensional type functions. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017*, pages 415–428, 2017.

[21] M. Brown and J. Palsberg. Jones-optimal partial evaluation by specialization-safe normalization. *PACMPL*, 2(POPL):14:1–14:28, 2018.

[22] B. Buchberger, A. Craciun, T. Jebelean, L. Kovacs, T. Kutsia, K. Nakagawa, F. Piroi, N. Popov, J. Robu, M. Rosenkranz, and W. Windsteiger. Theorema: Towards computer-aided mathematical theory exploration. *Journal of Applied Logic*, 4:470–504, 2006.

[23] C. Calcagno, W. Taha, L. Huang, and X. Leroy. Implementing multi-stage languages using ASTs, gensym, and reflection. In F. Pfenning and

Y. Smaragdakis, editors, *Generative Programming and Component Engineering (GPCE 2003)*, volume 2830 of *Lecture Notes in Computer Science*, pages 57–76. Springer, 2003.

[24] H. Cappelen and E. LePore. Quotation. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Spring 2012 edition, 2012.

[25] J. Carette and W. M. Farmer. High-level theories. In A. Autexier, J. Campbell, J. Rubio, M. Suzuki, and F. Wiedijk, editors, *Intelligent Computer Mathematics*, volume 5144 of *Lecture Notes in Computer Science*, pages 232–245. Springer, 2008.

[26] J. Carette and W. M. Farmer. Formalizing mathematical knowledge as a biform theory graph: A case study. In H. Geuvers, M. England, O. Hasan, F. Rabe, and O. Teschke, editors, *Intelligent Computer Mathematics*, volume 10383 of *Lecture Notes in Computer Science*, pages 9–24. Springer, 2017.

[27] J. Carette, W. M. Farmer, and P. Laskowski. HOL Light QE. *Computing Research Repository (CoRR)*, abs/1802.00405 (19 pp.), 2018.

[28] J. Carette, W. M. Farmer, and R. O'Connor. Mathscheme: Project description. In J. H. Davenport, W. M. Farmer, F. Rabe, and J. Urban, editors, *Intelligent Computer Mathematics*, volume 6824 of *Lecture Notes in Computer Science*, pages 287–288. Springer, 2011.

[29] J. Carette, O. Kiselyov, and C. Shan. Finally tagless, partially evaluated: Tagless staged interpreters for simpler typed languages. *J. Funct. Program.*, 19(5):509–543, 2009.

[30] A. Chaieb and T. Nipkow. Proof synthesis and reflection for linear arithmetic. *Journal of Automated Reasoning*, 41:33–59, 2008.

[31] A. Chlipala. *Certified Programming with Dependent Types: A Pragmatic Introduction to the Coq Proof Assistant*. MIT Press, 2013.

[32] D. Christiansen and E. Brady. Elaborator reflection: Extending Idris in Idris. *SIGPLAN Not.*, 51(9):284–297, September 2016.

[33] D. R. Christiansen. Type-directed elaboration of quasiquotations: A high-level syntax for low-level reflection. In *Proceedings of the 26Nd 2014 International Symposium on Implementation and Application of Functional Languages*, IFL '14, pages 1:1–1:9, New York, NY, USA, 2014. ACM.

[34] D. R. Christiansen. *Practical Reflection and Metaprogramming for Dependent Types*. PhD thesis, IT University of Copenhagen, 2016.

[35] A. Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[36] M. Clavel and J. Meseguer. Reflection in conditional rewriting logic. *Theoretical Computer Science*, 285:245–288, 2002.

[37] R. L. Constable. Using reflection to explain and enhance type theory. In H. Schwichtenberg, editor, *Proof and Computation*, volume 139 of *NATO ASI Series*, pages 109–144. Springer, 1995.

[38] R. L. Constable, S. F. Allen, H. M. Bromley, W. R. Cleaveland, J. F. Cremer, R. W. Harper, D. J. Howe, T. B. Knoblock, N. P. Mendler, P. Panangaden, J. T. Sasaki, and S. F. Smith. *Implementing Mathematics with the Nuprl Proof Development System*. Prentice-Hall, Englewood Cliffs, New Jersey, 1986.

[39] E. Contejean, P. Courtieu, J. Forest, O. Pons, and X. Urbain. Certification of automated termination proofs. In *Frontiers of Combining Systems*, volume 4720 of *Lecture Notes in Computer Science*, pages 148–162. Springer, 2007.

[40] Coq Development Team. *The Coq Proof Assistant Reference Manual, Version 8.7.2*, 2018. Available at `https://coq.inria.fr/distrib/current/refman/`.

[41] T. Coquand and G. Huet. The calculus of constructions. *Information and Computation*, 76:95–120, 1988.

[42] S. Costantini. Meta-reasoning: A survey. In A. C. Kakas and F. Sadri, editors, *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part II*, volume 2408 of *Lecture Notes in Computer Science*, pages 253–288, 2002.

[43] R. Davies and F. Pfenning. A modal analysis of staged computation. *Journal of the ACM*, 48:555–604, 2001.

[44] F.-N. Demers and J. Malenfant. Reflection in logic, functional and object-oriented programming: A short comparative study. In *IJCAI '95 Workshop on Reflection and Metalevel Architectures and their Applications in AI*, pages 29–38, 1995.

[45] G. Ebner, S. Ullrich, J. Roesch, J. Avigad, and L. de Moura. A metaprogramming framework for formal verification. *Proceedings of the ACM on Programming Languages*, 1(ICFP):34, 2017.

[46] W. M. Farmer. A partial functions version of Church's simple theory of types. *Journal of Symbolic Logic*, 55:1269–91, 1990.

[47] W. M. Farmer. A simple type theory with partial functions and subtypes. *Annals of Pure and Applied Logic*, 64:211–240, 1993.

[48] W. M. Farmer. Theory interpretation in simple type theory. In J. Heering, K. Meinke, B. Möller, and T. Nipkow, editors, *Higher-Order Algebra, Logic, and Term Rewriting*, volume 816 of *Lecture Notes in Computer Science*, pages 96–123. Springer, 1994.

[49] W. M. Farmer. Formalizing undefinedness arising in calculus. In D. Basin and M. Rusinowitch, editors, *Automated Reasoning—IJCAR 2004*, volume 3097 of *Lecture Notes in Computer Science*, pages 475–489. Springer, 2004.

[50] W. M. Farmer. Biform theories in Chiron. In M. Kauers, M. Kerber, R. R. Miner, and W. Windsteiger, editors, *Towards Mechanized Mathematical Assistants*, volume 4573 of *Lecture Notes in Computer Science*, pages 66–79. Springer, 2007.

[51] W. M. Farmer. Andrews' type system with undefinedness. In C. Benzmüller, C. Brown, J. Siekmann, and R. Statman, editors, *Reasoning in Simple Type Theory: Festschrift in Honor of Peter B. Andrews on his 70th Birthday*, Studies in Logic, pages 223–242. College Publications, 2008.

[52] W. M. Farmer. The seven virtues of simple type theory. *Journal of Applied Logic*, 6:267–286, 2008.

[53] W. M. Farmer. Chiron: A set theory with types, undefinedness, quotation, and evaluation. *Computing Research Repository (CoRR)*, abs/1305.6206 (154 pp.), 2013.

[54] W. M. Farmer. The formalization of syntax-based mathematical algorithms using quotation and evaluation. In J. Carette, D. Aspinall, C. Lange, P. Sojka, and W. Windsteiger, editors, *Intelligent Computer Mathematics*, volume 7961 of *Lecture Notes in Computer Science*, pages 35–50. Springer, 2013.

[55] W. M. Farmer. Simple type theory with undefinedness, quotation, and evaluation. *Computing Research Repository (CoRR)*, abs/1406.6706 (87 pp.), 2014.

[56] W. M. Farmer. Incorporating quotation and evaluation into Church's type theory: Syntax and semantics. In M. Kohlhase, M. Johansson, B. Miller, L. de Moura, and F. Tompa, editors, *Intelligent Computer Mathematics*, volume 9791 of *Lecture Notes in Computer Science*, pages 83–98. Springer, 2016.

[57] W. M. Farmer. Theory morphisms in Church's type theory with quotation and evaluation. In H. Geuvers, M. England, O. Hasan, F. Rabe, and O. Teschke, editors, *Intelligent Computer Mathematics*, volume 10383 of *Lecture Notes in Computer Science*, pages 147–162. Springer, 2017.

[58] W. M. Farmer, J. D. Guttman, and F. J. Thayer. IMPS: An Interactive Mathematical Proof System. *Journal of Automated Reasoning*, 11:213–248, 1993.

[59] W. M. Farmer, J. D. Guttman, and F. J. Thayer Fábrega. IMPS: An updated system description. In M. McRobbie and J. Slaney, editors, *Automated Deduction—CADE-13*, volume 1104 of *Lecture Notes in Computer Science*, pages 298–302. Springer, 1996.

[60] W. M. Farmer and P. Larjani. Frameworks for reasoning about syntax that utilize quotation and evaluation. *Computing Research Repository (CoRR)*, abs/1308.2149 (38 pp.), 2013.

[61] D. P. Friedman and M. Wand. Reification: Reflection without metaphysics. In *LISP and Functional Programming*, pages 348–355, 1984.

[62] M. J. Gabbay and A. M. Pitts. A new approach to abstract syntax involving binders. *Formal Aspects of Computing*, 13:341–363, 2002.

[63] M. Giese and B. Buchberger. Towards practical reflection for formal mathematics. RISC Report Series 07-05, Research Institute for Symbolic Computation (RISC), Johannes Kepler University, 2007.

[64] J.-Y. Girard. *Interprétation fonctionelle et élimination des coupures de l'arithmtique d'ordre supérieur*. PhD thesis, Université Paris 7, 1972.

[65] M. Glanzberg. Truth. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Spring 2013 edition, 2013.

[66] K. Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für Mathematik und Physik*, 38:173–198, 1931.

[67] G. Gonthier. The four colour theorem: Engineering of a formal proof. In D. Kapur, editor, *Computer Mathematics (ASCM 2007)*, volume 5081 of *Lecture Notes in Computer Science*, page 333. Springer, 2008.

[68] G. Gonthier, A. Asperti, J. Avigad, Y. Bertot, C. Cohen, F. Garillot, S. Le Roux, A. Mahboubi, R. O'Connor, S. Ould Biha, I. Pasca, L. Rideau, A. Solovyev, E. Tassi, and L. Théry. A machine-checked proof of the odd order theorem. In *Interactive Theorem Proving (ITP 2013)*, volume 7998 of *Lecture Notes in Computer Science*, pages 163–179. Springer, 2013.

[69] G. Gonthier, A. Mahboubi, and E. Tassi. A Small Scale Reflection Extension for the Coq system. Research Report RR-6455, Inria Saclay Ile de France, 2015.

[70] M. J. C. Gordon and T. F. Melham. *Introduction to HOL: A Theorem Proving Environment for Higher Order Logic*. Cambridge University Press, 1993.

[71] B. Grégoire and A. Mahboubi. Proving equalities in a commutative ring done right in Coq. In J. Hurd and T. F. Melham, editors, *Theorem Proving in Higher Order Logics (TPHOLs 2005)*, volume 3603 of *Lecture Notes in Computer Science*, pages 98–113. Springer, 2005.

[72] J. Grundy, T. Melham, and J. O'Leary. A reflective functional language for hardware design and theorem proving. *Journal of Functional Programming*, 16, 2006.

[73] V. Halbach. *Axiomatic Theories of Truth*. Cambridge University Press, 2011.

[74] V. Halbach and G. E. Leigh. Axiomatic theories of truth. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Winter 2013 edition, 2013.

[75] J. Harrison. Metatheory and reflection in theorem proving: A survey and critique. Technical Report CRC-053, SRI Cambridge, 1995. Available at `http://www.cl.cam.ac.uk/~jrh13/papers/reflect.ps.gz`.

[76] J. Harrison. HOL Light: An overview. In S. Berghofer, T. Nipkow, C. Urban, and M. Wenzel, editors, *Theorem Proving in Higher Order Logics*, volume 5674 of *Lecture Notes in Computer Science*, pages 60–66. Springer, 2009.

[77] L. Henkin. Completeness in the theory of types. *Journal of Symbolic Logic*, 15:81–91, 1950.

[78] J. Hickey, A. Nogin, R. L. Constable, B. E. Aydemir, E. Barzilay, Y. Bryukhov, R. Eaton, A. Granicz, A. Kopylov, C. Kreitz, V. Krupski, L. Lorigo, S. Schmitt, C. Witty, and X. Yu. MetaPRL — A modular logical environment. In D. Basin and B. Wolff, editors, *Theorem Proving in Higher Order Logics (TPHOLs 2003)*, volume 2758 of *Lecture Notes in Computer Science*, pages 287–303, 2003.

[79] D. Howe. Reflecting the semantics of reflected proof. In P. Aczel, H. Simmons, and S. Wainer, editors, *Proof Theory*, pages 229–250. Cambridge University Press, 1992.

[80] W. A. Hunt Jr., M. Kaufmann, R. B. Krug, J S. Moore, and E. W. Smith. Meta reasoning in ACL2. In J. Hurd and T. F. Melham, editors, *Theorem Proving in Higher Order Logics (TPHOLs 2005)*, volume 3603 of *Lecture Notes in Computer Science*, pages 163–178. Springer, 2005.

[81] D. W. H. James and R. Hinze. A reflection-based proof tactic for lattices in Coq. In Z. Horváth, V. Zsók, P. Achten, and P. W. M. Koopman, editors, *Proceedings of the Tenth Symposium on Trends in Functional Programming (TFP 2009)*, volume 10 of *Trends in Functional Programming*, pages 97–112. Intellect, 2009.

[82] C. B. Jay and J. Palsberg. Typed self-interpretation by pattern matching. In *Proceeding of the 16th ACM SIGPLAN international conference on Functional Programming, ICFP 2011, Tokyo, Japan, September 19-21, 2011*, pages 247–258, 2011.

[83] M. Kaufmann and J S. Moore. An industrial strength theorem prover for a logic based on Common Lisp. *IEEE Transactions on Software Engineering*, 23:203–213, 1997.

[84] G. A. Kavvos. On the Semantics of Intensionality and Intensional Recursion. Available from `http://arxiv.org/abs/1712.09302`, December 2017.

[85] T. B. Knoblock and R. L. Constable. Formalized metareasoning in type theory. In *Proceedings of the Symposium on Logic in Computer Science (LICS '86)*, pages 237–248. IEEE Computer Society, 1986.

[86] P. Koellner. On reflection principles. *Annals of Pure and Applied Logic*, 157:206–219, 2009.

[87] H. Leitgeb. What theories of truth should be like (but cannot be). *Philosophy Compass*, 2:276–290, 2007.

[88] P. Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, 1984.

[89] T. Melham, R. Cohn, and I. Childs. On the semantics of ReFLect as a basis for a reflective theorem prover. *Computing Research Repository (CoRR)*, abs/1309.5742, 2013.

[90] A. Mendhekar and D. P. Friedman. An exploration of relationships between reflective theories. In *Proceedings, Reflection 96*, pages 233–250, April 1996.

[91] D. Miller. Abstract syntax for variable binders: An overview. In J. Lloyd et al., editor, *Computational Logic — CL 2000*, volume 1861 of *Lecture Notes in Computer Science*, pages 239–253. Springer, 2000.

[92] T. Æ. Mogensen. Efficient self-interpretation in lambda calculus. *Journal of Functional Programming*, 2:345–364, 1994.

[93] E. Moggi and S. Fagorzi. A monadic multi-stage metalanguage. In A. D. Gordon, editor, *Foundations of Software Science and Computational Structures (FOSSACS 2003)*, volume 2620 of *Lecture Notes in Computer Science*, pages 358–374. Springer, 2003.

[94] A. Nanevski and F. Pfenning. Staged computation with names and necessity. *Journal of Functional Programmming*, 15:893–939, 2005.

[95] A. Nanevski, F. Pfenning, and B. Pientka. Contextual modal type theory. *ACM Transactions on Computational Logic*, 9, 2008.

[96] A. Nogin, A. Kopylov, X. Yu, and J. Hickey. A computational approach to reflective meta-reasoning about languages with bindings. In R. Pollack, editor, *ACM SIGPLAN International Conference on Functional Programming, Workshop on Mechanized Reasoning about Languages with Variable Binding, (MERLIN 2005)*, pages 2–12. ACM, 2005.

[97] U. Norell. *Towards a Practical Programming Language based on Dependent Type Theory*. PhD thesis, Chalmers University of Technology, 2007.

[98] U. Norell. Dependently typed programming in Agda. In A. Kennedy and A. Ahmed, editors, *Proceedings of TLDI'09*, pages 1–2. ACM, 2009.

[99] M. Odersky, L. Spoon, and B. Venners. *Programming in Scala*. Artima, third edition, 2016.

[100] M. Oostdijk and H. Geuvers. Proof by computation in the Coq system. *Theoretical Computer Science*, 272, 2002.

[101] F. Pfenning and C. Elliot. Higher-order abstract syntax. In *Proceedings of the ACM SIGPLAN 1988 conference on Programming Language design and Implementation*, pages 199–208. ACM Press, 1988.

[102] A. M. Pitts. Nominal Logic, a first order theory of names and binding. *Information and Computation*, 186:165–193, 2003.

[103] Plataformatec. Elixir. `http://elixir-lang.org/`, 2018.

[104] A. Polonsky. Axiomatizing the quote. In M. Bezem, editor, *Computer Science Logic (CSL'11) — 25th International Workshop/20th Annual Conference of the EACSL*, volume 12 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 458–469. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2011.

[105] W. V. O. Quine. *Mathematical Logic: Revised Edition*. Harvard University Press, 2003.

[106] F. Rabe. Generic literals. In M. Kerber, J. Carette, C. Kaliszyk, F. Rabe, and V. Sorge, editors, *Intelligent Computer Mathematics*, volume 9150 of *Lecture Notes in Computer Science*, pages 102–117. Springer, 2015.

[107] T. Rendel, K. Ostermann, and C. Hofer. Typed self-representation. *SIGPLAN Not.*, 44(6):293–303, June 2009.

[108] J. C. Reynolds. Towards a theory of type structure. In B. Robinet, editor, *Programming Symposium*, volume 19 of *Lecture Notes in Computer Science*, pages 408–425. Springer, 1974.

[109] Rice University Programming Languages Team. MetaOCaml: A compiled, type-safe, multi-stage programming language. `http://www.metaocaml.org/`, 2011.

[110] Scalameta Project. Scalameta. `http://scalameta.org/`, 2018.

[111] T. Sheard and S. P. Jones. Template meta-programming for Haskell. *ACM SIGPLAN Notices*, 37:60–75, 2002.

[112] M. Spivak. *Calculus*. Publish or Perish, fourth edition, 2008.

[113] A. Stump. Directly reflective meta-programming. *Higher-Order and Symbolic Computation*, 22:115–144, 2009.

[114] W. Taha. A gentle introduction to multi-stage programming. In C. Lengauer, D. Batory, C. Consel, and M. Odersky, editors, *Domain-Specific Program Generation*, volume 3016 of *Lecture Notes in Computer Science*, pages 30–50. Springer, 2004.

[115] W. Taha and T. Sheard. MetaML and multi-stage programming with explicit annotations. *Theoretical Computer Science*, 248:211–242, 2000.

[116] A. Tarski. Pojęcie prawdy w językach nauk dedukcyjnych (The concept of truth in the languages of the deductive sciences). *Prace Towarzystwa Naukowego Warszawskiego*, 3(34), 1933.

[117] A. Tarski. Der Wahrheitsbegriff in den formalisierten Sprachen. *Studia Philosophica*, 1:261–405, 1935.

[118] A. Tarski. The concept of truth in formalized languages. In J. Corcoran, editor, *Logic, Semantics, Meta-Mathematics*, pages 152–278. Hackett, second edition, 1983.

[119] The F# Software Foundation. F#. `http://fsharp.org/`, 2018.

[120] P. van der Walt. Reflection in Agda. Master's thesis, Universiteit Utrecht, 2012.

[121] P. van der Walt and W. Swierstra. Engineering proof by reflection in Agda. In R. Hinze, editor, *Implementation and Application of Functional Languages*, volume 8241 of *Lecture Notes in Computer Science*, pages 157–173. Springer, 2012.

[122] F. W. von Henke, S. Pfab, H. Pfeifer, and H. Rueß. Case studies in meta-level theorem proving. In J. Grundy and M. Newey, editors, *Theorem Proving in Higher Order Logics (TPHOLs'98*, volume 1479, pages 461–478. Springer, 1998.

[123] M. Wand. The theory of fexprs is trivial. *Lisp and Symbolic Computation*, 10(3):189–199, 1998.

[124] M. Wildmoser and T. Nipkow. Certifying machine code safety: Shallow versus deep embedding. In K. Slind, A. Bunker, and G. Gopalakrishnan, editors, *Theorem Proving in Higher Order Logics (TPHOLs 2004)*, volume 3223 of *Lecture Notes in Computer Science*, pages 305–320. Springer, 2004.

[125] X. Yu. *Reflection and Its Application to Mechanized Metareasoning about Programming Languages*. PhD thesis, California Institute of Technology, 2007.