CAS 760
Simple Type Theory
Winter 2026

11 Alonzo Variants

William M. Farmer

Department of Computing and Software
McMaster University

December 30, 2025

Outline

1. AlonzoID: Alonzo with Indefinite Description.
2. AlonzoS: Alonzo with Sorts.
3. AlonzoQE: Alonzo with Quotation and Evaluation

1. AlonzoID: Alonzo with Indefinite Description

AlonzoID: Definition [1/2]

We add indefinite description to Alonzo to obtain AlonzoID by making the following changes to the definition of Alonzo:

1. A new formation rule is added to the definition of an expression of type α:

   E8. Indefinite description: $\mathsf{IndefDes}(\mathsf{Var}(\mathbf{x}, \alpha), \mathbf{A}_{\mathsf{BoolTy}})$ is an expression of type α where $\alpha \neq \mathsf{BoolTy}$.

   $$(\epsilon\, \mathbf{x} : \alpha \,.\, \mathbf{A}_o) \quad \text{stands for} \quad \mathsf{IndefDes}(\mathsf{Var}(\mathbf{x}, \alpha), \mathbf{A}_o).$$

2. An interpretation of $L = (\mathcal{B}, \mathcal{C})$ is a tuple $M = (\mathcal{D}, I, \mathcal{F})$ where:
   a. $\mathcal{D} = \{D_\alpha \mid \alpha \in \mathcal{T}(L)\}$ is a frame for L.
   b. I is an interpretation function that maps each constant in $\mathcal{C}$ of type α to an element of D_α.
   c. $\mathcal{F} = \{f_\alpha \mid f_\alpha$ is a choice function on $D_{\{\alpha\}}$ for $\alpha \in \mathcal{T}(L)$ with $\alpha \neq o\}$.

AlonzoID: Definition [2/2]

3. A new condition is added to the definition of a general model:

   V8. $V^M_\varphi(\epsilon\,\mathbf{x} : \alpha \,.\, \mathbf{A}_o) = f_\alpha(D)$ where $D = \{d \in D_\alpha \mid V^M_{\varphi[(\mathbf{x}:\alpha)\mapsto d]}(\mathbf{A}_o) = \mathrm{T}\}$, provided D is nonempty. Otherwise, $V^M_\varphi(\epsilon\,\mathbf{x} : \alpha \,.\, \mathbf{A}_o)$ is undefined.

4. $\mathfrak{A}$ can be extended to a sound and complete proof system for AlonzoID by simply adding the following axioms to it:
   a. $(\exists\, x : \alpha \,.\, \mathbf{A}_o) \Rightarrow (\epsilon\, x : \alpha \,.\, \mathbf{A}_o) \in \{x : \alpha \mid \mathbf{A}_o\}$ $(\alpha \neq o)$.
   b. $\neg(\exists\, x : \alpha \,.\, \mathbf{A}_o) \Rightarrow (\epsilon\, x : \alpha \,.\, \mathbf{A}_o)\!\uparrow$ $(\alpha \neq o)$.

AlonzoID: Theorems

- Let AlonzoID be the logic.
- Proposition. The formula

  $$\mathrm{I}\,\mathbf{x} : \alpha \,.\, \mathbf{A}_o \simeq \epsilon\,\mathbf{x} : \alpha \,.\, (\mathbf{A}_o \wedge \exists!\,\mathbf{x} : \alpha \,.\, \mathbf{A}_o)$$

  is valid for all $\alpha \in \mathcal{T}(L)$ with $\alpha \neq o$.
- Lemma. The expression

  $$\lambda\, s : \{\alpha\} \,.\, \epsilon\, x : \alpha \,.\, x \in s.$$

  is a choice function on α for all $\alpha \in \mathcal{T}(L)$ with $\alpha \neq o$.
- Theorem (Axiom of Choice). The sentence

  $$\exists\, f : \{\alpha\} \to \alpha \,.\, \forall\, s : \{\alpha\} \,.\, s \neq \emptyset_{\{\alpha\}} \mapsto f\, s \in s \mid (f\, s)\!\uparrow,$$

  which expresses the axiom of choice for α, is valid for all $\alpha \in \mathcal{T}(L)$ with $\alpha \neq o$.

2. AlonzoS: Alonzo with Sorts

Sorts [1/2]

- Benefits of quasitypes:
  1. Restrict the value of a bound variable: $\lambda\, \mathbf{x} : \mathbf{Q}_{\{\alpha\}} \,.\, \mathbf{B}_{\beta}$.
  2. Sharpen a definedness statement: $\mathbf{A}_{\alpha} \downarrow \mathbf{Q}_{\{\alpha\}}$.
- However, quasitypes are not interchangeable with types:
  1. A constant symbol cannot be bound to a quasitype to obtain a constants $\mathbf{c}_{\mathbf{Q}_{\{\alpha\}}}$.
  2. The value $\forall\, \mathbf{x} : \alpha \,.\, F_o$ is always F, but the value of $\forall\, \mathbf{x} : \mathbf{Q}_{\{\alpha\}} \,.\, F_o$ is T if $\mathbf{Q}_{\{\alpha\}}$ denotes the empty set.
- Is there a mechanism for subtypes without the limitations of quasitypes?
  - Yes, IMPS-style sorts.
  - A sort is either a type or a nonempty subtype of a type.

Sorts [2/2]

- If α and β are sorts that are subtypes of the types α' and β', then $\alpha \to \beta$ is a sort that is a subtype of the type $\alpha' \to \beta'$.
  1. $\alpha \to \beta$ denotes a set F of functions f in $D_{\alpha'} \to D_{\beta'}$ such that $\mathrm{dom}(f) \subseteq D_\alpha$ and $\mathrm{ran}(f) \subseteq D_\beta$.
  2. $\alpha' \to \beta'$ denotes a set G of functions in $D_{\alpha'} \to D_{\beta'}$.
  3. $F \subseteq G$.
- This works since functions are allowed to be partial: if $D_\alpha \subset D_{\alpha'}$, then each (partial or total) function in $D_\alpha \to D_\beta$ is represented by a partial function in $D_{\alpha'} \to D_{\beta'}$.

AlonzoS: Symbols and Metasymbols

- New symbols.
  - $\mathcal{S}_{\mathsf{as}}$ is a set of atomic sort symbols.
  - $\mathcal{S}_{\mathsf{bt}}$ and $\mathcal{S}_{\mathsf{as}}$ are disjoint.
- New metasymbols.
  - $\mathbf{s}, \mathbf{t}$, etc. range over members of $\mathcal{S}_{\mathsf{as}}$.
  - σ, τ, etc. range over sorts of AlonzoS.
  - α, β, etc. range over types of AlonzoS.

AlonzoS: Sorts and Types

- A sort of AlonzoS is a string of symbols defined inductively by the following formation rules:
  - U1. Type of truth values: BoolTy is a sort.
  - U2. Base type: BaseTy(**a**) is a sort.
  - U3. Atomic sort: AtSort(**s**) is a sort.
  - U4. Function sort: FunSort(σ, τ) is a sort.
  - U5. Product sort: ProdSort(σ, τ) is a sort.
- A type of AlonzoS is a sort of Alonzo that is constructed without using U3.
- Compact notation for sorts:

o	stands for	BoolTy.
a	stands for	BaseTy(**a**).
s	stands for	AtSort(**s**).
$(\sigma \to \tau)$	stands for	FunSort(σ, τ).
$(\sigma \times \tau)$	stands for	ProdSort(σ, τ).

AlonzoS: Sort Systems [1/2]

- A sort system of AlonzoS is a tuple $S = (\mathcal{B}, \mathcal{A}, \xi)$ where:
  1. $\mathcal{B}$ is a finite set of base types.
  2. $\mathcal{A}$ is a finite set of atomic sorts.
  3. ξ is a function that maps each member of $\mathcal{A}$ to a type α such that all the base types occurring in α are members of $\mathcal{B}$.
- A sort σ is a sort of S if all the base types occurring in σ are members of $\mathcal{B}$ and all the atomic sorts occurring in σ are members of $\mathcal{A}$.
- A type α is a type of S if α is a sort of S.

AlonzoS: Sort Systems [1/2]

- Let $\overline{\xi}$ be the canonical extension of ξ to the sorts of S defined by:
  1. $\overline{\xi}(o) = o$.
  2. $\overline{\xi}(\mathbf{a}) = \mathbf{a}$.
  3. $\overline{\xi}(\mathbf{s}) \equiv \xi(\mathbf{s})$.
  4. $\overline{\xi}(\sigma \to \tau) \equiv \overline{\xi}(\sigma) \to \overline{\xi}(\tau)$.
  5. $\overline{\xi}(\sigma \times \tau) \equiv \overline{\xi}(\sigma) \times \overline{\xi}(\tau)$.
- $\overline{\xi}(\sigma)$ is the type of σ in S.
- If σ is a type of S, then $\overline{\xi}(\sigma) = \sigma$.

AlonzoS: Expressions and Languages [1/2]

- Let $S = (\mathcal{B}, \mathcal{A}, \xi)$ be a sort system of AlonzoS and
- Let $\mathbf{A}_\sigma, \mathbf{B}_\sigma$, etc. be syntactic variables that range over expressions of S of sort σ.
- An expression of S of sort σ of AlonzoS is a string of symbols defined inductively by the following formation rules:
  - E1. Variable: $\mathsf{Var}(\mathbf{x}, \sigma)$ is an expression of S of sort σ if σ is a sort of S.
  - E2. Constant: $\mathsf{Con}(\mathbf{c}, \sigma)$ is an expression of S of sort σ if σ is a sort of S.
  - E3. Equality: $\mathsf{Eq}(\mathbf{A}_\sigma, \mathbf{B}_\tau)$ is an expression of S of sort BoolTy if $\overline{\xi}(\sigma) = \overline{\xi}(\tau)$.
  - E4. Function application: $\mathsf{FunApp}(\mathbf{F}_{\sigma \to \tau}, \mathbf{A}_{\sigma'})$ is an expression of S of sort τ if $\overline{\xi}(\sigma) = \overline{\xi}(\sigma')$.

AlonzoS: Expressions and Languages [2/2]

E5. Function abstraction: $\mathsf{FunAbs}(\mathsf{Var}(\mathbf{x}, \sigma), \mathbf{B}_\tau)$ is an expression of S of sort $\mathsf{FunSort}(\sigma, \tau)$ if σ is a sort of S.

E6. Definite description: $\mathsf{DefDes}(\mathsf{Var}(\mathbf{x}, \sigma), \mathbf{A}_{\mathsf{BoolTy}})$ is an expression of S of sort σ if σ is a sort of S with $\sigma \neq \mathsf{BoolTy}$.

E7. Ordered pair: $\mathsf{OrdPair}(\mathbf{A}_\sigma, \mathbf{B}_\tau)$ is an expression of S of sort $\mathsf{ProdSort}(\sigma, \tau)$.

- A language (or signature) of AlonzoS is a pair $L = (S, \mathcal{C})$ where S is sort system of AlonzoS and $\mathcal{C}$ is a set of constants $\mathbf{c}_\sigma$ of S.
  - A type α is a type of L if it is a type of S.
  - A sort σ is a sort of L if it is a sort of S.
  - $\mathbf{A}_\sigma$ is an expression of L if $\mathbf{A}_\sigma$ is an expression of S and all the constants occurring in $\mathbf{A}_\sigma$ are members of $\mathcal{C}$.

AlonzoS: Interpretations [1/2]

- Let $L = (S, \mathcal{C})$ be a language of AlonzoS.
- A frame for L is a collection $\mathcal{D} = \{D_\sigma \mid \sigma \in \mathcal{U}(L)\}$ of nonempty domains (sets) of values such that:
  - F1. Domain of truth values: $D_o = \mathbb{B} = \{\mathrm{F}, \mathrm{T}\}$.
  - F2. Predicate domain: $D_{\sigma \to o}$ is a set of some total functions f from $D_{\overline{\xi}(\sigma)}$ to D_o such that $f(d) = \mathrm{F}$ for all $d \in D_{\overline{\xi}(\sigma)} \setminus D_\sigma$ for $\sigma \in \mathcal{U}(L)$.
  - F3. Function domain: $D_{\sigma \to \tau}$ is a set of some partial and total functions f from $D_{\overline{\xi}(\sigma)}$ to $D_{\overline{\xi}(\tau)}$ such that $\mathrm{dom}(f) \subseteq D_\sigma$ and $\mathrm{ran}(f) \subseteq D_\tau$ for $\sigma, \tau \in \mathcal{U}(L)$ with $\tau \neq o$.
  - F4. Product domain: $D_{\sigma \times \tau} = D_\sigma \times D_\tau$ for $\sigma, \tau \in \mathcal{U}(L)$.
  - F5. Type domain: $D_\sigma \subseteq D_{\overline{\xi}(\sigma)}$ for $\sigma \in \mathcal{U}(L)$.

AlonzoS: Interpretations [2/2]

- An interpretation of L is a pair $M = (\mathcal{D}, I)$ where:
  1. $\mathcal{D} = \{D_\sigma \mid \sigma \in \mathcal{U}(L)\}$ is a frame for L.
  2. I is an interpretation function that maps each constant in $\mathcal{C}$ of sort σ to an element of D_σ.

AlonzoS: General Models [1/2]

- Let $\mathcal{D} = \{D_\sigma \mid \sigma \in \mathcal{U}(L)\}$ be a frame for L and $M = (\mathcal{D}, I)$ be an interpretation of L.
- M is a general model of L if there is a partial binary valuation function V^M such that, for all assignments $\varphi \in \text{assign}(M)$ and expressions $\mathbf{C}_\sigma$ of L, (1) either $V^M_\varphi(\mathbf{C}_\sigma) \in D_\sigma$ or $V^M_\varphi(\mathbf{C}_\sigma)$ is undefined and (2) each of the following conditions is satisfied:
  - V1. $V^M_\varphi((\mathbf{x} : \sigma)) = \varphi((\mathbf{x} : \sigma))$.
  - V2. $V^M_\varphi(\mathbf{c}_\sigma) = I(\mathbf{c}_\sigma)$.
  - V3. $V^M_\varphi(\mathbf{A}_\sigma = \mathbf{B}_\tau) = \mathrm{T}$ if $V^M_\varphi(\mathbf{A}_\sigma)$ is defined, $V^M_\varphi(\mathbf{B}_\tau)$ is defined, and $V^M_\varphi(\mathbf{A}_\sigma) = V^M_\varphi(\mathbf{B}_\tau)$. Otherwise, $V^M_\varphi(\mathbf{A}_\sigma = \mathbf{B}_\tau) = \mathrm{F}$.

AlonzoS: General Models [2/2]

V4. $V_\varphi^M(\mathbf{F}_{\sigma\to\tau}\,\mathbf{A}_{\sigma'}) = V_\varphi^M(\mathbf{F}_{\sigma\to\tau})(V_\varphi^M(\mathbf{A}_{\sigma'}))$ if $V_\varphi^M(\mathbf{F}_{\sigma\to\tau})$ is defined, $V_\varphi^M(\mathbf{A}_{\sigma'})$ is defined, and $V_\varphi^M(\mathbf{F}_{\sigma\to\tau})$ is defined at $V_\varphi^M(\mathbf{A}_{\sigma'})$. Otherwise, $V_\varphi^M(\mathbf{F}_{\sigma\to\tau}\,\mathbf{A}_{\sigma'}) = \mathrm{F}$ if $\tau = o$ and $V_\varphi^M(\mathbf{F}_{\sigma\to\tau}\,\mathbf{A}_{\sigma'})$ is undefined if $\tau \neq o$.

V5. $V_\varphi^M(\lambda\,\mathbf{x} : \sigma\,.\,\mathbf{B}_\tau)$ is the (partial or total) function $f \in D_{\sigma\to\tau}$ such that, for each $d \in D_\sigma$, $f(d) = V^M_{\varphi[(\mathbf{x}:\sigma)\mapsto d]}(\mathbf{B}_\tau)$ if $V^M_{\varphi[(\mathbf{x}:\sigma)\mapsto d]}(\mathbf{B}_\tau)$ is defined and $f(d)$ is undefined if $V^M_{\varphi[(\mathbf{x}:\sigma)\mapsto d]}(\mathbf{B}_\tau)$ is undefined.

V6. $V_\varphi^M(\mathrm{I}\,\mathbf{x} : \sigma\,.\,\mathbf{A}_o)$ is the $d \in D_\sigma$ such that $V^M_{\varphi[(\mathbf{x}:\sigma)\mapsto d]}(\mathbf{A}_o) = \mathrm{T}$ if there is exactly one such d. Otherwise, $V_\varphi^M(\mathrm{I}\,\mathbf{x} : \sigma\,.\,\mathbf{A}_o)$ is undefined.

V7. $V_\varphi^M((\mathbf{A}_\sigma, \mathbf{B}_\tau)) = (V_\varphi^M(\mathbf{A}_\sigma), V_\varphi^M(\mathbf{B}_\tau))$ if $V_\varphi^M(\mathbf{A}_\sigma)$ and $V_\varphi^M(\mathbf{B}_\tau)$ are defined. Otherwise, $V_\varphi^M((\mathbf{A}_\sigma, \mathbf{B}_\tau))$ is undefined.

AlonzoS: Notational Definitions

Let $\alpha = \overline{\xi}(\sigma) = \overline{\xi}(\tau)$.

$(\mathbf{A}_\sigma \downarrow \tau)$	stands for	$(\lambda\, x : \tau \,.\, T_o)\, \mathbf{A}_\sigma$.
$(\mathbf{A}_\sigma \uparrow \tau)$	stands for	$\neg(\mathbf{A}_\sigma \downarrow \tau)$.
$(\sigma \subseteq \tau)$	stands for	$\forall\, x : \sigma \,.\, x \downarrow \tau$.
$(\sigma = \tau)$	stands for	$\forall\, x : \alpha \,.\, x \downarrow \sigma \Leftrightarrow x \downarrow \tau$.

AlonzoS: Proof System

- PF^* is a version of Church's type theory with undefined expressions and sorts.
- For any language L of PF^*, there is a proof system for L that is sound and complete.
- Following the design of the proof system for L, we can modify $\mathfrak{A}$ to obtain a proof system for a language of AlonzoS that is sound and complete.
- See W. M. Farmer, "A simple type theory with partial functions and subtypes", *Annals of Pure and Applied Logic*, 64:211–240, 1993 for details.

AlonzoS: Theorems

- Let AlonzoS be the logic, and $L = (S, \mathcal{C})$ be a language of AlonzoS where $S = (\mathcal{B}, \mathcal{A}, \xi)$.
- Proposition. Each of the following formulas of L are valid:
  1. $\mathbf{A}_\sigma\!\downarrow \Rightarrow \mathbf{A}_\sigma \downarrow \sigma$.
  2. $\mathbf{A}_\sigma \downarrow \tau \Rightarrow \mathbf{A}_\sigma\!\downarrow$.
  3. $(\mathbf{A}_\sigma \downarrow \tau \wedge \tau \subseteq \tau') \Rightarrow \mathbf{A}_\sigma \downarrow \tau'$.
  4. $\sigma \subseteq \overline{\xi}(\sigma)$.
  5. $\sigma \subseteq \sigma$.
  6. $(\sigma_1 \subseteq \sigma_2 \wedge \sigma_2 \subseteq \sigma_1) \Rightarrow \sigma_1 = \sigma_2$.
  7. $(\sigma_1 \subseteq \sigma_2 \wedge \sigma_2 \subseteq \sigma_3) \Rightarrow \sigma_1 \subseteq \sigma_3$.
  8. $(\sigma \subseteq \sigma' \wedge \tau \subseteq \tau') \Rightarrow \sigma \to \tau \subseteq \sigma' \to \tau'$.
  9. $(\sigma \subseteq \sigma' \wedge \tau \subseteq \tau') \Rightarrow \sigma \times \tau \subseteq \sigma' \times \tau'$.
- Theorem (Sort Beta-Reduction). Let $\mathbf{C}_o$ be a formula of L of the form

  $$\mathbf{A}_{\sigma'} \downarrow \sigma \Rightarrow (\lambda\, \mathbf{x} : \sigma \,.\, \mathbf{B}_\tau)\, \mathbf{A}_{\sigma'} \simeq \mathbf{B}_\tau[(\mathbf{x} : \sigma) \mapsto \mathbf{A}_{\sigma'}].$$

  If $\mathbf{A}_{\sigma'}$ is free for $(\mathbf{x} : \sigma)$ in $\mathbf{B}_\tau$, then $\mathbf{C}_o$ is valid.

3. AlonzoQE: Alonzo with Quotation and Evaluation

Interplay of Syntax and Semantics [1/2]

- It is often useful to be able to reason about the interplay of the syntax and semantics of expressions.
- A syntax-based mathematical algorithm (SBMA) manipulates mathematical expressions in a mathematically meaningful way.
  - Example. A symbolic differentiation algorithm.
- Reasoning about a SBMA requires reasoning about the relationship between how the expressions are manipulated and what the manipulations mean mathematically.
- In standard predicate logics, it is not possible to directly refer to the syntax of an expression and thus reason directly about the interplay of syntax and semantics.

Interplay of Syntax and Semantics [2/2]

- Quotation is a mechanism for referring to a syntactic value (e.g., a syntax tree) that represents the syntactic structure of an expression.
- Evaluation is a mechanism for referring to the value of the expression that a syntactic value represents.
- Incorporating quotation and evaluation into a traditional logic is tricky due to the following problems:
  1. Evaluation problem.
  2. Variable problem.
  3. Double substitution problem.
- We present in W. M. Farmer, "Incorporating Quotation and Evaluation into Church's Type Theory", *Information and Computation*, 260:9–50, 2018, a version of Church's type theory called $\mathrm{CTT}_{\mathrm{qe}}$ with quotation and evaluation operators that overcomes these problems.

AlonzoQE: Alonzo with Quote and Eval [1/5]

- A new formation rule is added to the definition of a type:

  T5. Type of expressions: ExprTy is a type.

- Two new formation rules are added to the definition of an expression:

  E8. Quotation: Quote($\mathbf{A}_\alpha$) is an expression of type ExprTy if $\mathbf{A}_\alpha$ is eval-free.

  E9. Evaluation: Eval($\mathbf{A}_{\mathsf{ExprTy}}, \alpha$) is an expression of type α.

- An expression is eval-free if it is constructed using only the constructors presented in E1–E8.

- Notational definitions for AlonzoQE:

ϵ	stands for	ExprTy.
$\ulcorner \mathbf{A}_\alpha \urcorner$	stands for	Quote($\mathbf{A}_\alpha$).
$[\![\mathbf{A}_\epsilon]\!]_\alpha$	stands for	Eval($\mathbf{A}_\epsilon, \alpha$).

AlonzoQE: Alonzo with Quote and Eval [2/5]

- AlonzoQE includes the following logical constants:

$\mathsf{q\text{-}eq}_{\epsilon\to\epsilon\to\epsilon}$
$\mathsf{q\text{-}app}_{\epsilon\to\epsilon\to\epsilon}$
$\mathsf{q\text{-}abs}_{\epsilon\to\epsilon\to\epsilon}$
$\mathsf{q\text{-}dd}_{\epsilon\to\epsilon\to\epsilon}$
$\mathsf{q\text{-}op}_{\epsilon\to\epsilon\to\epsilon}$
$\mathsf{q\text{-}quo}_{\epsilon\to\epsilon}$

- A construction 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

$\mathsf{q\text{-}eq}_{\epsilon\to\epsilon\to\epsilon}\,\mathbf{A}_\epsilon\,\mathbf{B}_\epsilon$,
$\mathsf{q\text{-}app}_{\epsilon\to\epsilon\to\epsilon}\,\mathbf{A}_\epsilon\,\mathbf{B}_\epsilon$,
$\mathsf{q\text{-}abs}_{\epsilon\to\epsilon\to\epsilon}\,\mathbf{A}_\epsilon\,\mathbf{B}_\epsilon$,
$\mathsf{q\text{-}dd}_{\epsilon\to\epsilon\to\epsilon}\,\mathbf{A}_\epsilon\,\mathbf{B}_\epsilon$,
$\mathsf{q\text{-}op}_{\epsilon\to\epsilon\to\epsilon}\,\mathbf{A}_\epsilon\,\mathbf{B}_\epsilon$, and
$\mathsf{q\text{-}quo}_{\epsilon\to\epsilon}\,\mathbf{A}_\epsilon$

  are constructions.

AlonzoQE: Alonzo with Quote and Eval [3/5]

- Let Ω be the function mapping eval-free expressions to constructions that is defined by:
  1. $\Omega((\mathbf{x} : \alpha)) = \ulcorner(\mathbf{x} : \alpha)\urcorner$.
  2. $\Omega(\mathbf{c}_\alpha) = \ulcorner \mathbf{c}_\alpha \urcorner$.
  3. $\Omega(\mathbf{A}_\alpha = \mathbf{B}_\alpha) = \mathsf{q\text{-}eq}_{\epsilon\to\epsilon\to\epsilon}\, \Omega(\mathbf{A}_\alpha)\, \Omega(\mathbf{B}_\alpha)$.
  4. $\Omega(\mathbf{F}_{\alpha\to\beta}\, \mathbf{A}_\alpha) = \mathsf{q\text{-}app}_{\epsilon\to\epsilon\to\epsilon}\, \Omega(\mathbf{F}_{\alpha\to\beta})\, \Omega(\mathbf{A}_\alpha)$.
  5. $\Omega(\lambda\, \mathbf{x} : \alpha\, .\, \mathbf{B}_\beta) = \mathsf{q\text{-}abs}_{\epsilon\to\epsilon\to\epsilon}\, \Omega((\mathbf{x} : \alpha))\, \Omega(\mathbf{B}_\beta)$.
  6. $\Omega(\mathrm{I}\, \mathbf{x} : \alpha\, .\, \mathbf{A}_o) = \mathsf{q\text{-}dd}_{\epsilon\to\epsilon\to\epsilon}\, \Omega((\mathbf{x} : \alpha))\, \Omega(\mathbf{A}_o)$.
  7. $\Omega((\mathbf{A}_\alpha, \mathbf{B}_\alpha)) = \mathsf{q\text{-}op}_{\epsilon\to\epsilon\to\epsilon}\, \Omega(\mathbf{A}_\alpha)\, \Omega(\mathbf{B}_\alpha)$.
  8. $\Omega(\ulcorner \mathbf{A}_\alpha \urcorner) = \mathsf{q\text{-}quo}_{\epsilon\to\epsilon}\, \Omega(\mathbf{A}_\alpha)$.
- When $\mathbf{A}_\alpha$ is eval-free, $\Omega(\mathbf{A}_\alpha)$ is the unique construction that represents the syntax tree of $\mathbf{A}_\alpha$.
- A new clause is added to the definition of a frame:

  F5. D_ϵ is the set of eval-free expressions in $\mathcal{E}(L)$.

AlonzoQE: Alonzo with Quote and Eval [4/5]

- An interpretation function maps each constant in $\mathcal{C}$ of type α and each logical constant of type α to an element of D_α such that:
  1. $I(\text{q-eq}_{\epsilon\to\epsilon\to\epsilon})(\mathbf{A}_\alpha)(\mathbf{B}_\beta) = (\mathbf{A}_\alpha = \mathbf{B}_\beta)$ (i.e., $I(\text{q-eq}_{\epsilon\to\epsilon\to\epsilon})$ applied to the expressions $\mathbf{A}_\alpha$ and $\mathbf{B}_\beta$ is the expression $(\mathbf{A}_\alpha = \mathbf{B}_\beta)$) if $\alpha = \beta$ and is undefined otherwise.
  2. $I(\text{q-app}_{\epsilon\to\epsilon\to\epsilon})(\mathbf{A}_\alpha)(\mathbf{B}_\beta) = (\mathbf{A}_\alpha\, \mathbf{B}_\beta)$ if $\alpha = \beta \to \gamma$ for some type γ and is undefined otherwise.
  3. $I(\text{q-abs}_{\epsilon\to\epsilon\to\epsilon})(\mathbf{A}_\alpha)(\mathbf{B}_\beta) = (\lambda\, \mathbf{x} : \alpha\, .\, \mathbf{B}_\beta)$ if $\mathbf{A}_\alpha$ has the form $(\mathbf{x} : \alpha)$ and is undefined otherwise.
  4. $I(\text{q-dd}_{\epsilon\to\epsilon\to\epsilon})(\mathbf{A}_\alpha)(\mathbf{B}_\beta) = (\mathrm{I}\, \mathbf{x} : \alpha\, .\, \mathbf{B}_\beta)$ if $\mathbf{A}_\alpha$ has the form $(\mathbf{x} : \alpha)$ with $\alpha \neq o$ and $\beta = o$ and is undefined otherwise.
  5. $I(\text{q-op}_{\epsilon\to\epsilon\to\epsilon})(\mathbf{A}_\alpha)(\mathbf{B}_\beta) = (\mathbf{A}_\alpha, \mathbf{B}_\beta)$ if $\alpha = \beta$ and is undefined otherwise.
  6. $I(\text{q-quo}_{\epsilon\to\epsilon})(\mathbf{A}_\alpha) = \ulcorner \mathbf{A}_\alpha \urcorner$.

AlonzoQE: Alonzo with Quote and Eval [5/5]

- Finally, two new conditions are added to the definition of a general model:

  V8. $V_\varphi^M(\ulcorner \mathbf{A}_\alpha \urcorner) = \mathbf{A}_\alpha$ (i.e., $V_\varphi^M(\ulcorner \mathbf{A}_\alpha \urcorner)$ is the expression $\mathbf{A}_\alpha$).

  V9. $V_\varphi^M([\![\mathbf{A}_\epsilon]\!]_\alpha) = V_\varphi^M(V_\varphi^M(\mathbf{A}_\epsilon))$ if $V_\varphi^M(\mathbf{A}_\epsilon)$ is an expression $\mathbf{E}_\alpha$ such that $V_\varphi^M(\mathbf{E}_\alpha)$ is defined. Otherwise, $V_\varphi^M([\![\mathbf{A}_\epsilon]\!]_\alpha) = \mathrm{F}$ if $\alpha = o$ and is undefined if $\alpha \neq o$.

- It is not clear how to extend $\mathfrak{A}$ to a sound and complete proof system for AlonzoQE.
- However, $\mathfrak{A}$ can be extended to a sound proof system for AlonzoQE that is complete for eval-free formulas by following the design of the proof system for $\mathrm{CTT}_{\mathrm{qe}}$.

AlonzoQE: Theorems

- Let AlonzoQE be the logic.
- Theorem. Let $\mathbf{A}_\alpha$ be an eval-free expression of a language L, M be a general model of L, and $\varphi \in \text{assign}(M)$. Then $V_\varphi^M(\Omega(\mathbf{A}_\alpha)) = \mathbf{A}_\alpha$.
- Theorem (Law of Quotation). $\ulcorner \mathbf{A}_\alpha \urcorner = \Omega(\mathbf{A}_\alpha)$ is valid.
- Theorem (Law of Disquotation). $[\![\ulcorner \mathbf{A}_\alpha \urcorner]\!]_\alpha \simeq \mathbf{A}_\alpha$ is valid.

The End.