# CAS 760
# Simple Type Theory
# Winter 2026

# 1 Review of First-Order Logic

William M. Farmer

Department of Computing and Software
McMaster University

December 30, 2025

# Outline

1. What is First-Order Logic?
2. First-Order Structures.
3. FOL: Syntax.
4. FOL: Semantics.
5. FOL: Theories.
6. Observations.

# 1. What is First-Order Logic?

# What is Logic?

1. The study of the principles underlying sound reasoning.
   - Central ideas: truth and logical consequence.
2. The branch of mathematics underlying mathematical reasoning and computation.
3. Main application areas:
   a. Computing.
   b. Mathematics.
   c. Philosophy.

# Fundamental Distinctions

Logic makes several fundamental distinctions:

- Syntax vs. semantics.
- Language vs. metalanguage.
- Theory vs. model.
- Truth vs. proof.

# What is a Logic?

- Informally, a logic is a system of reasoning.
- Formally, a logic is a family of formal languages with:
  1. A precise common syntax.
  2. A precise common semantics with a notion of logical consequence.
- A logic may include one or more formal proof systems for mechanically deriving that a given formula is a logical consequence of a given set of formulas.
- Two most common examples:
  - Propositional logic.
  - First-order logic.

# What is Predicate Logic?

- Predicate logic is the study of statements about individuals constructed using predicates, functions, Boolean operators, and quantifiers.
- The foundations of predicate logic were developed independently by Gottlob Frege (1848-1925) and Charles Sanders Peirce (1839–1914).
- Examples of predicate logics:
  - First-order logic.
  - Many-sorted first-order logic.
  - Second-order logic.
  - Simple type theory (classical higher-order logic).
  - Dependent type theory (constructive higher-order logic).

# What is First-Order Logic?

- First-order logic is the leading form of predicate logic.
  - It is "first-order" because quantification is over individuals but not over "higher-order objects" such as sets, relations, functions, and predicates.
  - It is suited more for theory than practice.
  - First-order logic is also called first-order predicate logic and first-order quantificational logic.
- First-order logic is propositional logic plus:
  - Terms that denote individuals.
  - Predicates that are applied to terms.
  - Quantifiers applied to individual variables and formulas.
- There are many formulations of first-order logic.
- We will develop a version of first-order logic called FOL

# 2. First-Order Structures

# Mathematical Structures

- Loosely speaking, a "mathematical structure" is a set of mathematical values on which a structure is placed using a set of distinguished values.
- More precisely, a mathematical structure (structure for short) is a pair $S = (\mathcal{D}, \mathcal{A})$ where:
  1. $\mathcal{D}$ is a nonempty finite set of base domains that are nonempty sets of values.
  2. $\mathcal{A}$ is a set of distinguished values that are members of the domains in $\{\mathbb{B}\} \cup \mathcal{D}$ (where $\mathbb{B} = \{\mathrm{F}, \mathrm{T}\}$) or domains constructed from these domains by the function space, power set, Cartesian product, and Kleene star operations.
- If $\mathcal{D} = \{D_1, \ldots, D_m\}$ and $\mathcal{A} = \{a_1, \ldots, a_n\}$, $S$ may be written as the tuple:

  $$(D_1, \ldots, D_m, a_1, \ldots, a_n).$$

# Examples of Mathematical Structures

- Number systems.
- Orders.
- Equivalence relations.
- Algebraic structures.
- Lattices.
- Graphs.
- Trees.
- Vector spaces.
- Metric spaces.
- Topological spaces.
- Measure spaces.
- Abstract data types (ADTs) used in computing (e.g, ADTs for strings, lists, streams, arrays, records, stacks, and queues).

# First-Order Structures

- A first-order structure is a structure $(\{D\}, \mathcal{A})$ where each $a \in \mathcal{A}$ is
  1. A member of $D$,
  2. A function over $D$, or
  3. A relation or predicate over $D$.
- $(\{D\}, \mathcal{A})$ is usually written as the tuple $(D, a_1, \ldots, a_n)$ when $\mathcal{A}$ is the finite set $\{a_1, \ldots, a_n\}$.
- Examples:
  - The well-order of natural numbers $(\mathbb{N}, \leq)$.
  - The semiring of natural numbers $(\mathbb{N}, 0, 1, +, *)$.
  - The ring of integers $(\mathbb{Z}, 0, 1, +, *, -)$.
  - The field of rational numbers $(\mathbb{Q}, 0, 1, +, *, -, \cdot^{-1}))$.
  - The Boolean algebra of truth values $(\mathbb{B}, \mathrm{F}, \mathrm{T}, \vee, \wedge, \neg)$.
  - The Boolean algebra of sets $(\mathcal{P}(A), \emptyset, A, \cup, \cap, \overline{\cdot})$ for any nonempty set $A$.

# 3. FOL: Syntax

# Symbols of FOL

- Let $\mathcal{S}_{\text{var}}$ be a set of variable symbols.
- Let $\mathcal{S}_{\text{con}}$ be a set of (individual) constant symbols.
- For each $n \geq 1$, let $\mathcal{S}^n_{\text{fun}}$ be a set of $n$-ary function symbols.
- For each $n \geq 1$, let $\mathcal{S}^n_{\text{pred}}$ be a set of $n$-ary predicate symbols.
- Assume that these sets above are countably infinite.
- The members of these sets above are the nonlogical symbols of FOL.
- Let $\{=, \neg, \Rightarrow, \forall, \text{“(”}, \text{“)”}, \text{“,”}, \text{“.”}\}$ be the set of logical symbols of FOL.
- Assume that these sets above are pairwise disjoint.

# FOL: Terms

- A term of FOL is a string of symbols defined inductively by the following formation rules:
  1. Variable. If $x \in \mathcal{S}_{\text{var}}$, then $x$ is a term of FOL.
  2. Constant. If $c \in \mathcal{S}_{\text{con}}$, then $c$ is a term of FOL.
  3. Function Application. If $f \in \mathcal{S}^n_{\text{fun}}$ and $t_1, \ldots, t_n$ are terms of FOL, then $f(t_1, \ldots, t_n)$ is a term of FOL.
- A term denotes an individual in a first-order structure, i.e., a member of the base domain of a first-order structure.
- Let $\mathbb{T}$ be the set of terms of FOL.

# Formulas

- A formula of FOL is a string of symbols defined inductively by the following formation rules:
  1. Equality. If $t_1$ and $t_2$ are terms of FOL, then $(t_1 = t_2)$ is a formula of FOL.
  2. Predicate Application. If $p \in \mathcal{S}^n_{\text{pred}}$ and $t_1, \ldots, t_n$ are terms of FOL, then $p(t_1, \ldots, t_n)$ is a formula of FOL.
  3. Negation. If $A$ is a formula of FOL, then $\neg A$ is a formula of FOL.
  4. Implication. If $A$ and $B$ are formulas of FOL, then $(A \Rightarrow B)$ is a formula of FOL.
  5. Universal Quantification. If $x \in \mathcal{S}_{\text{var}}$ and $A$ is a formula of FOL, then $(\forall x \,.\, A)$ is a formula of FOL.
- A formula denotes a Boolean value (i.e., true or false).
- Let $\mathbb{F}$ be the set of formulas of FOL.

# Notational Definitions

| | | |
|---|---|---|
| $(s \neq t)$ | stands for | $\neg(s = t)$. |
| $\mathsf{T}$ | stands for | $(\forall\, x \,.\, (x = x))$. |
| $\mathsf{F}$ | stands for | $\neg\mathsf{T}$. |
| $(A \vee B)$ | stands for | $(\neg A \Rightarrow B)$. |
| $(A \wedge B)$ | stands for | $\neg(\neg A \vee \neg B)$. |
| $(A \Leftrightarrow B)$ | stands for | $((A \Rightarrow B) \wedge (B \Rightarrow A))$. |
| $(\exists\, x \,.\, A)$ | stands for | $\neg(\forall\, x \,.\, \neg A)$. |

# Notational Conventions

- A pair of matching parentheses in a term or a formula may be dropped if there is no loss of meaning.
- An application of a binary function or predicate symbol $s(t_1, t_2)$ may be written as $t_1\ s\ t_2$ using infix notation.
- Let $\Box$ be $\forall$ or $\exists$. $(\Box\, x_1\ .\ \cdots (\Box\, x_n\ .\ A)\cdots)$ may be written as:

  $$(\Box\, x_1, \ldots, x_n\ .\ A).$$

# Bound and Free Variables [1/3]

- A variable $x$ is free in a term $t$ if it occurs in $t$.
- An occurrence of a variable $x$ in a formula $A$ is bound [free] if it is [not] in a subformula of $A$ of the form $\forall\, x\, .\, B$.
- A variable $x$ is bound [free] in $A$ if there is a bound [free] occurrence of $x$ in $B$.
- A term or formula is open [closed] if there are [no] variables free in it.
- A sentence is a closed formula.
- $t$ is free for $x$ in $A$ if no free occurrence of $x$ in $A$ is within a subformula of $A$ of the form $\forall\, y\, .\, B$ such that $y$ is free in $t$.
  - That is, if no variable captures occurs if $t$ is substituted for the free occurrences of $x$ in $A$..

# Bound and Variables [2/3]

- The set of free variables of a term $t$, written $\mathrm{fvar}(t)$, is defined by recursion and pattern matching by:
  1. $\mathrm{fvar}(x) = \{x\}$.
  2. $\mathrm{fvar}(c) = \emptyset$.
  3. $\mathrm{fvar}(f(t_1, \ldots, t_n)) = \mathrm{fvar}(t_1) \cup \cdots \cup \mathrm{fvar}(t_n)$.
- The set of free variables of a formula $A$, written $\mathrm{fvar}(A)$, is defined by recursion and pattern matching by:
  1. $\mathrm{fvar}(s = t) = \mathrm{fvar}(s) \cup \mathrm{fvar}(t)$.
  2. $\mathrm{fvar}(p(t_1, \ldots, t_n)) = \mathrm{fvar}(t_1) \cup \cdots \cup \mathrm{fvar}(t_n)$.
  3. $\mathrm{fvar}(\neg A) = \mathrm{fvar}(A)$.
  4. $\mathrm{fvar}(A \Rightarrow B) = \mathrm{fvar}(A) \cup \mathrm{fvar}(B)$.
  5. $\mathrm{fvar}(\forall x \,.\, A) = \mathrm{fvar}(A) \setminus \{x\}$.

# Bound and Free Variables [3/3]

- The set of bound variables of a term $t$ is the empty set.
- The set of bound variables of a formula $A$, written $\mathrm{bvar}(A)$, is defined by recursion and pattern matching by:
  1. $\mathrm{bvar}(s = t) = \emptyset$.
  2. $\mathrm{bvar}(p(t_1, \ldots, t_n)) = \emptyset$.
  3. $\mathrm{bvar}(\neg A) = \mathrm{bvar}(A)$.
  4. $\mathrm{bvar}(A \Rightarrow B) = \mathrm{bvar}(A) \cup \mathrm{bvar}(B)$.
  5. $\mathrm{bvar}(\forall x \,.\, A) = \mathrm{bvar}(A) \cup \{x\}$.

# Substitutions [1/2]

- The result of substituting a term $t$ for the variable $x$ in a term $s$, written $s[x \mapsto t]$, is defined by recursion and pattern matching by:
  1. $x[x \mapsto t] = t$.
  2. $y[x \mapsto t] = y$ where $x$ and $y$ differ.
  3. $c[x \mapsto t] = c$.
  4. $f(t_1, \ldots, t_n)[x \mapsto t] = f(t_1[x \mapsto t], \ldots, t_n[x \mapsto t])$.

# Substitutions [2/2]

- The result of substituting a term $t$ for the free occurrences of the variable $x$ in a formula $A$, written $A[x \mapsto t]$, is defined by recursion and pattern matching by:
  1. $(s_1 = s_2)[x \mapsto t] = (s_1[x \mapsto t] = s_2[x \mapsto t])$.
  2. $p(t_1, \ldots, t_n)[x \mapsto t] = p(t_1[x \mapsto t], \ldots, t_n[x \mapsto t])$.
  3. $(\neg A)[x \mapsto t] = \neg A[x \mapsto t]$.
  4. $(A \Rightarrow B)[x \mapsto t] = (A[x \mapsto t] \Rightarrow B[x \mapsto t])$.
  5. $(\forall x \,.\, A)[x \mapsto t] = (\forall x \,.\, A)$.
  6. $(\forall y \,.\, A)[x \mapsto t] = (\forall y \,.\, A[x \mapsto t])$ where $x$ and $y$ differ.
- When a variable is captured in a substitution, the formula produced by the substitution may not be a valid result.

# Languages

- A language (or signature) of FOL is a triple $L = (\mathcal{C}, \mathcal{F}, \mathcal{P})$ where:
  1. $\mathcal{C} \subseteq \mathcal{S}_{\mathsf{con}}$.
  2. $\mathcal{F} \subseteq \bigcup_{n \geq 1} \mathcal{S}^n_{\mathsf{fun}}$.
  3. $\mathcal{P} \subseteq \bigcup_{n \geq 1} \mathcal{S}^n_{\mathsf{pred}}$.
- A term $t$ is a term of $L$ if all the constant and function symbols in $t$ are members of $\mathcal{C}$ and $\mathcal{F}$, respectively.
- A formula $A$ is a formula of $L$ if all the terms in $A$ are terms of $L$ and all the predicate symbols in $A$ are members of $\mathcal{P}$.
- Let $\mathbb{T}(L)$ and $\mathbb{F}(L)$ denote the sets of terms and formulas of $L$, respectively.
- $L$ is finite if $\mathcal{C}$, $\mathcal{F}$, and $\mathcal{P}$ are finite.
- We use the constant, function, and predicate symbols to denote the distinguished values of a first-order structure.

# 4. FOL : Semantics

# Semantics [1/4]

- Let $L = (\mathcal{C}, \mathcal{F}, \mathcal{P})$ be a language of FOL.
- A model of $L$ is a pair $M = (D, I)$ where $D$ is a nonempty domain and $I$ is a function on $\mathcal{C} \cup \mathcal{F} \cup \mathcal{P}$ such that:
  1. $I(c) \in D$ for each $c \in \mathcal{C}$.
  2. $I(f) : D, \ldots, D \to D$ is an $n$-ary function for each $n$-ary $f \in \mathcal{F}$.
  3. $I(p) : D, \ldots, D \to \{\mathrm{F}, \mathrm{T}\}$ is an $n$-ary predicate for each $n$-ary $p \in \mathcal{P}$.

  $I$ is called an interpretation function.
- Thus a model of $L$ is an interpretation of the symbols of $L$ as the distinguished values in a first-order structure.
- If $L = (\{c_1, \ldots, c_k\}, \{f_1, \ldots, f_m\}, \{p_1, \ldots, p_n\})$ and $M = (D, I)$ is a model of $L$, then
$$(D, I(c_1), \ldots, I(c_k), I(f_1), \ldots, I(f_m), I(p_1), \ldots, I(p_n))$$
is the first-order structure defined by $M$.

# Semantics [2/4]

- Let $M = (D, I)$ be a model of $L$.
- We need to define the value of the terms and formulas of $L$ with respect to $M$.
  - The value of a term is a member of $D$.
  - The value of a formula is a member of $\mathbb{B}$.
- A variable assignment into $M$ is a function that maps each $x \in \mathcal{V}$ to a member of $D$.
- Let $\text{assign}(M)$ be the set of variable assignments into $M$.
- Given $\varphi \in \text{assign}(M)$, $x \in \mathcal{V}$, and $d \in D$, let $\varphi[x \mapsto d]$ be the $\varphi' \in \text{assign}(M)$ such that $\varphi'(x) = d$ and $\varphi'(y) = \varphi(y)$ for all $y \in \mathcal{V}$ that differ from $x$.

# Semantics [3/4]

- The valuation function for $M$ is the binary function $V^M$ that satisfies the following statements conditions for all $\varphi \in \text{assign}(M)$ and all terms $t$ and formulas $A$ of $L$:
  1. $V^M_\varphi(x) = \varphi(x)$.
  2. $V^M_\varphi(c) = I(c)$.
  3. $V^M_\varphi(f(t_1, \ldots, t_n)) = I(f)(V^M_\varphi(t_1), \ldots, V^M_\varphi(t_n))$.
  4. $V^M_\varphi(s = t) = \mathrm{T}$ if $V^M_\varphi(s) = V^M_\varphi(t)$ and $V^M_\varphi(s = t) = \mathrm{F}$ otherwise.
  5. $V^M_\varphi(p(t_1, \ldots, t_n)) = I(p)(V^M_\varphi(t_1), \ldots, V^M_\varphi(t_n))$.
  6. $V^M_\varphi(\neg A) = \mathrm{F}$ if $V^M_\varphi(A) = \mathrm{T}$ and $V^M_\varphi(\neg A) = \mathrm{T}$ otherwise.
  7. $V^M_\varphi(A \Rightarrow B) = \mathrm{F}$ if $V^M_\varphi(A) = \mathrm{T}$ and $V^M_\varphi(B) = \mathrm{F}$ and $V^M_\varphi(A \Rightarrow B) = \mathrm{T}$ otherwise.
  8. $V^M_\varphi(\forall x \,.\, A) = \mathrm{T}$ if $V^M_{\varphi[x \mapsto d]}(A) = \mathrm{T}$ for all $d \in D$ and $V^M_\varphi(\forall x \,.\, A) = \mathrm{F}$ otherwise.

# Semantics [4/4]

- Let $A \in \mathbb{F}(L)$, $\Gamma \subseteq \mathbb{F}(L)$, and $M$ be a model of $L$.
- $A$ is satisfiable if $V_\varphi^M(A) = \mathrm{T}$ for some model $M$ of $L$ and $\varphi \in \mathrm{assign}(M)$.
- $A$ is valid in M or M is a model for $A$, written $M \models A$, if $V_\varphi^M(A) = \mathrm{T}$ for all $\varphi \in \mathrm{assign}(M)$.
- $A$ is (universally) valid, written $\models A$, if $A$ is valid in every model of $L$.
- M is a model for $\Gamma$, written $M \models \Gamma$, if $M \models B$ for all $B \in \Gamma$.
- A formula $A$ is a semantic consequence of a set $\Gamma$ of sentences, written $\Gamma \models A$, if $M \models \Gamma$ implies $M \models A$ for all models $M$ of $L$.
  - A somewhat more complicated definition is needed when $\Gamma$ contains open formulas.
  - Semantic consequence is a form of logical consequence.

# Notes on Quantifiers

- The universal and existential quantifiers are duals of each other:

$$\neg(\forall x \,.\, A) \;\Leftrightarrow\; \exists x \,.\, \neg A, \quad \neg(\exists x \,.\, A) \;\Leftrightarrow\; \forall x \,.\, \neg A.$$

- Changing the order of quantifiers in a formula usually changes the meaning of the formula!
  - As a rule, $\forall x \,.\, \exists y \,.\, A \;\Leftrightarrow\; \exists y \,.\, \forall x \,.\, A$ is not valid.
- In a formula of the form $\forall x \,.\, \exists y \,.\, A$, the value of the existentially quantified variable $y$ depends on the value of the universally quantified variable $x$.
- A universal statement like "All mice are rodents" is formalized as $\forall x \,.\, \text{mouse}(x) \Rightarrow \text{rodent}(x)$.
- An existential statement like "Some rodents are mice" is formalized as $\exists x \,.\, \text{rodent}(x) \wedge \text{mouse}(x)$.

# 5. FOL : Theories

# Theories

- A theory of FOL is a pair $T = (L, \Gamma)$ where:
  1. $L$ is a language of FOL.
  2. $\Gamma$ is a set of sentences of $L$ called the axioms of $T$.

  The axioms serve as the assumptions of $T$.
- $M$ is a model of $T$, written $M \models T$, if $M \models \Gamma$.
- $A$ is valid in $T$ or a theorem of $T$, written $T \models A$, if $\Gamma \models A$.
- $T$ is satisfiable if there is a model for $\Gamma$.
- Examples:
  - Theories of orders, lattices, and boolean algebras.
  - Theories of monoids, groups, rings, and fields.
  - First-order Peano arithmetic (natural number arithmetic).
  - Theory of real closed fields (real number arithmetic).

# Example: Theory of Strong Partial Orders

- Let $L = (\emptyset, \emptyset, \{<\})$ where $< \in \mathcal{S}^2_{\text{pred}}$.
- Let $\Gamma$ be the following set of sentences of $L$:
  1. $\forall x \,.\, \neg(x < x)$.
  2. $\forall x, y \,.\, x < y \Rightarrow \neg(y < x)$.
  3. $\forall x, y, z \,.\, (x < y \wedge y < z) \Rightarrow x < z$.
- Then $T_{\text{spo}} = (L, \Gamma)$ is a theory of FOL called the theory of strong partial orders.
- The models of $T_{\text{spo}}$ are all possible strong partial orders.
- If we can show $T_{\text{spo}} \models A$, then we know that $A$ is valid in every strong partial order!

# Example: Theory of Monoids

- Let $L = (\{e\}, \{\cdot\}, \emptyset)$ where $\cdot \in \mathcal{S}^2_{\mathsf{fun}}$.
- Let $\Gamma$ be the following set of sentences of $L$.
  1. $\forall x, y, z \,.\, (x \cdot y) \cdot z = x \cdot (y \cdot z)$.
  2. $\forall x \,.\, e \cdot x = x$.
  3. $\forall x \,.\, x \cdot e = x$.
- Then $T_{\mathrm{mon}} = (L, \Gamma)$ is a theory of FOL called the theory of monoids.
- Examples of models of $T_{\mathrm{mon}}$:
  1. $(\mathbb{N}, 0, +)$, $(\mathbb{N}, 1, *)$, $(\mathbb{Z}, 0, +)$, etc.
  2. (String, "", concatenation).
  3. (IntegerList, [ ], list-concatenation).
  4. $(U \to U,$ identity-function, function-composition$)$.

# First-Order Peano Arithmetic

- Let $L = (\{\{0\}, \{S, +, *\}, \emptyset\})$ where $S \in \mathcal{S}^1_{\text{fun}}$ and $+, * \in \mathcal{S}^2_{\text{fun}}$.
- Let $\Gamma$ be the following set of formulas of $L$:
  1. $\forall x \,.\, 0 \neq S(x)$.
  2. $\forall x, y \,.\, (S(x) = S(y) \Rightarrow x = y)$.
  3. $\forall x \,.\, x + 0 = x$.
  4. $\forall x, y \,.\, x + S(y) = S(x + y)$.
  5. $\forall x \,.\, x * 0 = 0$.
  6. $\forall x, y \,.\, x * S(y) = (x * y) + x$.
  7. Each universal closure $A$ of a formula of the form

$$(B[x \mapsto 0] \wedge (\forall x \,.\, B \Rightarrow B[x \mapsto S(x)])) \Rightarrow \forall x \,.\, B$$

     where $B$ is a formula of $L$.
- Then $T_{\text{pa}} = (L, \Gamma)$ is a theory of FOL called first-order Peano arithmetic.

# Theory vs. Model

- A model in logic is a concrete mathematical model consisting of an interpretation of the expressions of a language as values of a structure.
- A theory in logic is an abstract mathematical model consisting of a language $L$ and a set of sentences of $L$.
- A theory can be viewed as a specification of the structures defined by its models.
  - A theory is to a structure as a specification is to an implementation.
- Theories fall into two categories:
  - Those intended to specify a collection of structures (e.g., theory of monoids).
  - Those intended to specify a single structure (e.g., first-order Peano arithmetic).

# 6. Observations

# Variants of FOL

- First-order logic without function symbols (FOLwoFS).
  - Same theoretical expressivity, but much lower practical expressivity.
  - Less complex with same logical principles.
  - Usually used only for theoretical purposes.
- Many-sorted first-order logic (MSFOL).
  - Same theoretical expressivity, but much higher practical expressivity.
  - More complex with same logical principles.
  - Often used for practical purposes.
- First-order logic with undefined terms (FOLwUT).
  - Same theoretical expressivity, but much higher practical expressivity.
  - More complex with slightly different logical principles.
  - Rarely used.

# Shortcomings of FOL

- Only one base type of individuals.
- Functions and predicates are only applied to individuals.
- Quantifiers are only applied to variables over individuals.
- Does not admit undefined terms.
- No lambda notation.
- No definite description (or indefinite description).
- Can only specify structures that are first-order.
- Cannot uniquely specify infinite first-order structures.

# Transforming FOL into Alonzo

1. Allow higher-order functions and predicates (STT).
2. Use types to organize the different values (STT).
3. Combine terms and formulas into expressions (CTT).
4. Introduce lambda notation (CTT).
5. Define Boolean operators and quantifiers using a Boolean type and lambda notation (CTT).
6. Introduce definite description (CTT).
7. Provide two semantics, one for mathematics and one for logic (CTT).
8. Allow multiple base types (Alonzo).
9. Eliminate logical constants (Alonzo).
10. Allow ordered pairs (Alonzo).
11. Allow expressions to be undefined (Alonzo).

# The End.