CAS 760 Winter 2010

02 Three Traditional Logics

William M. Farmer

Department of Computing and Software
McMaster University

15 March 2010

Outline

- Part 1: First-order logic
- Part 2: Simple type theory
- Part 3: Zermelo-Fraenkel (ZF) set theory

Part 1

First-Order Logic

What is First-Order Logic?

- First-order logic is the study of statements about individuals using functions, predicates, and quantification.
  - 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.
- First-order logic is "first-order" because quantification is over individuals but not over higher-order objects such as functions and predicates.
- There are many versions of first-order logic.
- We will define and employ a version of first-order logic named FOL.

Syntax of FOL: Languages

- Let $\mathcal{V}$ be a fixed infinite set of symbols called variables.
- A language of FOL is a triple $L = (\mathcal{C}, \mathcal{F}, \mathcal{P})$ where:
  - $\mathcal{C}$ is a set of symbols called individual constants.
  - $\mathcal{F}$ is a set of symbols called function symbols, each with an assigned arity ≥ 1.
  - $\mathcal{P}$ is a set of symbols called predicate symbols, each with an assigned arity ≥ 1. $\mathcal{P}$ contains the binary predicate symbol $=$.
  - $\mathcal{V}$, $\mathcal{C}$, $\mathcal{F}$, and $\mathcal{P}$ are pairwise disjoint.

Syntax of FOL: Terms and Formulas

- Let $L = (\mathcal{C}, \mathcal{F}, \mathcal{P})$ be a language of FOL.
- A term of L is a string of symbols inductively defined by the following formation rules:
  - Each $x \in \mathcal{V}$ and $a \in \mathcal{C}$ is a term of L.
  - If $f \in \mathcal{F}$ is n-ary and $t_1, \ldots, t_n$ are terms of L, then $f(t_1, \ldots, t_n)$ is a term of L.
- A formula of L is a string of symbols inductively defined by the following formation rules:
  - If $p \in \mathcal{P}$ is n-ary and $t_1, \ldots, t_n$ are terms of L, then $p(t_1, \ldots, t_n)$ is a formula of L.
  - If A and B are formulas of L and $x \in \mathcal{V}$, then $(\neg A)$ and $(A \Rightarrow B)$, and $(\forall x \,.\, A)$ are formulas of L.
- $=$, $\neg$, $\Rightarrow$, and $\forall$ are the logical constants of FOL.

Syntax of FOL: Notational Definitions

$(s = t)$	denotes	$= (s, t)$.
$(s \neq t)$	denotes	$(\neg(s = t))$.
T	denotes	$(\forall\, x \,.\, (x = x))$.
F	denotes	$(\neg(\mathsf{T}))$.
$(A \vee B)$	denotes	$((\neg A) \Rightarrow B)$.
$(A \wedge B)$	denotes	$(\neg((\neg A) \vee (\neg B)))$.
$(A \Leftrightarrow B)$	denotes	$((A \Rightarrow B) \wedge (B \Rightarrow A))$.
$(\exists\, x \,.\, A)$	denotes	$(\neg(\forall\, x \,.\, (\neg A))$.
$(\square\, x_1, \ldots, x_n \,.\, A)$	denotes	$(\square\, x_1 \,.\, (\square\, x_2, \ldots, x_n \,.\, A))$ where $n \geq 2$ and $\square \in \{\forall, \exists\}$.

Free and Bound Variables

- The scope of a quantifier $\forall x$ or $\exists x$ in a formula $\forall x \,.\, B$ or $\exists x \,.\, B$, respectively, is the part of B that is not in a subformula of B of the form $\forall x \,.\, C$ or $\exists x \,.\, C$.
- An occurrence of a variable x in a formula A is free if it is not in the scope of a quantifier $\forall x$ or $\exists x$; otherwise the occurrence of x in A is bound.
  - An occurrence of a variable in a formula is either free or bound but never both.
  - A variable can be both bound and free in a formula.
- A formula is closed if it contains no free variables.
- A sentence is a closed formula.

Substitution

- Let x be a variable, t a term, and A a formula.
- The substitution of t for x in A, written

  $$A[x \mapsto t] \quad \text{or} \quad A[t/x],$$

  is the result of replacing each free occurrence of x in A with t.
- Suppose A is $\forall\, y \,.\, x = y$ and t is $f(y)$. Then the substitution $A[x \mapsto t]$ is said to capture y.
  - Variable captures often produce unsound results.
- t is free for x in A if no free occurrence of x in A is in the scope of $\forall\, y$ or $\exists\, y$ for any variable y occurring t.
  - Hence, t is free for x in A if the substitution $A[x \mapsto t]$ does not result in any variable captures.

Semantics of FOL: Models

- A model for a language $L = (\mathcal{C}, \mathcal{F}, \mathcal{P})$ of FOL is a pair $M = (D, I)$ where D is a nonempty domain (set) and I is a total function on $\mathcal{C} \cup \mathcal{F} \cup \mathcal{P}$ such that:
  - If $a \in \mathcal{C}$, $I(a) \in D$.
  - If $f \in \mathcal{F}$ is n-ary, $I(f) : D^n \to D$ and $I(f)$ is total.
  - If $p \in \mathcal{P}$ is n-ary, $I(p) : D^n \to \{\mathrm{T}, \mathrm{F}\}$ and $I(p)$ is total.
  - $I(=)$ is id_D, the identity predicate on D.
- A variable assignment into M is a function that maps each $x \in \mathcal{V}$ to an element of D.
- Given a variable assignment φ into M, $x \in \mathcal{V}$, and $d \in D$, let $\varphi[x \mapsto d]$ be the variable assignment φ' into M such $\varphi'(x) = d$ and $\varphi'(y) = \varphi(y)$ for all $y \neq x$.

Semantics of FOL: Valuation Function

The valuation function for a model M for a language $L = (\mathcal{C}, \mathcal{F}, \mathcal{P})$ of FOL is the binary function V^M that satisfies the following conditions for all variable assignments φ into M and all terms t and formulas A of L:

1. Let $t \in \mathcal{V}$. Then $V^M_\varphi(t) = \varphi(t)$.
2. Let $t \in \mathcal{C}$. Then $V^M_\varphi(t) = I(t)$.
3. Let $t = f(t_1, \ldots, t_n)$. Then $V^M_\varphi(t) = I(f)(V^M_\varphi(t_1), \ldots, V^M_\varphi(t_n))$.
4. Let $A = p(t_1, \ldots, t_n)$. Then $V^M_\varphi(A) = I(p)(V^M_\varphi(t_1), \ldots, V^M_\varphi(t_n))$.
5. Let $A = (\neg A')$. If $V^M_\varphi(A') = \mathrm{F}$, then $V^M_\varphi(A) = \mathrm{T}$; otherwise $V^M_\varphi(A) = \mathrm{F}$.
6. Let $A = (A_1 \Rightarrow A_2)$. If $V^M_\varphi(A_1) = \mathrm{T}$ and $V^M_\varphi(A_2) = \mathrm{F}$, then $V^M_\varphi(A) = \mathrm{F}$; otherwise $V^M_\varphi(A) = \mathrm{T}$.
7. Let $A = (\forall\, x \,.\, A')$. If $V^M_{\varphi[x \mapsto d]}(A') = \mathrm{T}$ for all $d \in D$, then $V^M_\varphi(A) = \mathrm{T}$; otherwise $V^M_\varphi(A) = \mathrm{F}$.

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, $\forall x \,.\, \exists y \,.\, A \not\Leftrightarrow \exists y \,.\, \forall x \,.\, A$.
- 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 rodents are mammals" is formalized as $\forall x \,.\, \mathsf{rodent}(x) \Rightarrow \mathsf{mammal}(x)$.
- An existential statement like "Some mammals are rodents" is formalized as $\exists x \,.\, \mathsf{mammal}(x) \wedge \mathsf{rodent}(x)$.

Algebras as Models

- If $L = (\mathcal{C}, \mathcal{F}, \mathcal{P})$ is a finite language of FOL, we may present the language as

  $$L = (c_1, \ldots, c_k, f_1, \ldots, f_m, p_1, \ldots, p_n)$$

  where $\mathcal{C} = \{c_1, \ldots, c_k\}$, $\mathcal{F} = \{f_1, \ldots, f_m\}$, and $\mathcal{P} = \{p_1, \ldots, p_n\}$.
- An algebra

  $$(D, d_1, \ldots, d_k, g_1, \ldots, g_m, q_1, \ldots . q_n)$$

  can then be considered a model for L if $M = (D, I)$ is a model for L where:

  1. $I(c_i) = d_i$ for $1 \leq i \leq k$.
  2. $I(f_i) = g_i$ for $1 \leq i \leq m$.
  3. $I(p_i) = q_i$ for $1 \leq i \leq n$.

Metatheorems of FOL

- Completeness Theorem (Gödel 1930). There is a sound and complete proof system for FOL.
- Compactness Theorem. Let Σ be a set of formulas of a language of FOL. If Σ is finitely satisfiable, then Σ is satisfiable.
- Undecidability Theorem (Church 1936). First-order logic is undecidable. That is, for some language L of FOL, the problem of whether or not a given formula of L is valid is undecidable.

A Hilbert-Style Proof System (1/2)

Let **H** be the following Hilbert-style proof system for a language L of FOL:

- The logical axioms of **H** are all formulas of L that are instances of the following schemas:
  - For propositional logic:
    A1: $A \Rightarrow (B \Rightarrow A)$.
    A2: $(A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))$.
    A3: $(\neg A \Rightarrow \neg B) \Rightarrow (B \Rightarrow A)$.
  - For quantification:
    A4: $(\forall\, x\ .\ (A \Rightarrow B)) \Rightarrow (A \Rightarrow (\forall\, x\ .\ B))$
    provided x is not free in A.
    A5: $(\forall\, x\ .\ A) \Rightarrow A[x \mapsto t]$
    provided t is free for x in A.

A Hilbert-Style Proof System (2/2)

- For equality:

  A6: $\forall x \,.\, x = x$.

  A7: $\forall x, y \,.\, x = y \Rightarrow y = x$.

  A8: $\forall x, y, z \,.\, (x = y \wedge y = z) \Rightarrow x = z$.

  A9: $\forall x_1, \ldots, x_n, y_1, \ldots, y_n \,.$
  $(x_1 = y_1 \wedge \cdots \wedge x_n = y_n) \Rightarrow$
  $f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$
  where $f \in \mathcal{F}$ is n-ary.

  A10: $\forall x_1, \ldots, x_n, y_1, \ldots, y_n \,.$
  $(x_1 = y_1 \wedge \cdots \wedge x_n = y_n) \Rightarrow$
  $(p(x_1, \ldots, x_n) \Leftrightarrow p(y_1, \ldots, y_n))$
  where $p \in \mathcal{P}$ is n-ary.

- The rules of inference of **H** are:

  MP: From A and $(A \Rightarrow B)$, infer B.

  GEN: From A, infer $(\forall x \,.\, A)$, for any $x \in \mathcal{V}$.

More Metatheorems of FOL

- Deduction Theorem. $\Sigma \cup \{A\} \vdash_{\mathbf{H}} B$ implies $\Sigma \vdash_{\mathbf{H}} A \Rightarrow B$.
- Soundness Theorem. $\Sigma \vdash_{\mathbf{H}} A$ implies $\Sigma \models A$.
- Completeness Theorem. $\Sigma \models A$ implies $\Sigma \vdash_{\mathbf{H}} A$.
- Soundness and Completeness Theorem (second form).
  Σ is consistent in $\mathbf{H}$ iff Σ is satisfiable.

Theories

- A theory in FOL is a pair $T = (L, \Gamma)$ where:
  1. L is a language of FOL.
  2. Γ is a set of sentences of L.
- Examples:
  - Theories of orders, lattices, and boolean algebras.
  - Theories of monoids and groups.
  - Presburger arithmetic.
  - First-order Peano arithmetic.
  - Theory of real closed fields.

The Theory of Boolean Algebras

- Let **BA** $= (L, \Gamma)$ be the theory of FOL where L is defined below and Γ is the set of sentences of L on the next page.
- $L = (+, *, ^-, 0, 1, =)$ is a language of FOL such that $+$ and $*$ are binary function symbols, $^-$ is a unary function symbol, and 0 and 1 are individual constants.
- A boolean algebra is a model of **BA**.
  - Named after the logician George Boole (1815-1864).
  - There are infinitely many nonisomorphic models of **BA**.
  - If $(B, +, *, ^-, 0, 1)$ is a boolean algebra, then $(B, \leq)$ is a complemented distributive lattice with a top and bottom where $a \leq b$ means $a = a * b \wedge a + b = b$.
- Examples:
  - $M_1 = (\{\mathsf{T}, \mathsf{F}\}, \vee, \wedge, \neg, \mathsf{F}, \mathsf{T}, \Leftrightarrow)$.
  - $M_2 = (\{S \mid S \subseteq U\}, \cup, \cap, ^-, \emptyset, U, =)$ where U is any set.
- **BA** is used to model electronic circuits.

The Axioms of **BA**

Associativity Laws

$$\forall x, y, z \,.\, (x + y) + z = x + (y + z)$$
$$\forall x, y, z \,.\, (x * y) * z = x * (y * z)$$

Commutativity Laws

$$\forall x, y \,.\, x + y = y + x \qquad \forall x, y \,.\, x * y = y * x$$

Distributive Laws

$$\forall x, y, z \,.\, x + (y * z) = (x + y) * (x + z)$$
$$\forall x, y, z \,.\, x * (y + z) = (x * y) + (x * z)$$

Identity Laws

$$\forall x \,.\, x + 0 = x \qquad \forall x \,.\, x * 1 = x$$

Complement Laws

$$\forall x \,.\, x + \overline{x} = 1 \qquad \forall x \,.\, x * \overline{x} = 0$$

Theorems of **BA**

Idempotent Laws

$$\forall x \,.\, x + x = x \qquad \forall x \,.\, x * x = x$$

Absorption Laws

$$\forall x, y \,.\, x + (x * y) = x \qquad \forall x, y \,.\, x * (x + y) = x$$

De Morgan Laws

$$\forall x, y \,.\, \overline{x + y} = \overline{x} * \overline{y}$$
$$\forall x, y \,.\, \overline{x * y} = \overline{x} + \overline{y}$$

Laws of Zero and One

$$\forall x \,.\, x + 1 = 1 \qquad \forall x, y \,.\, x * 0 = 0$$
$$\overline{0} = 1 \qquad \overline{1} = 0$$

Law of Double Complement

$$\forall x \,.\, \overline{\overline{x}} = x$$

Peano Arithmetic

- **PA** $= (L, \Gamma)$ is (second-order) Peano arithmetic (devised by G. Peano, 1889).
- L is a language of second-order logic with an individual constant symbol 0 and a unary function symbol S.
  - 0 is intended to represent the number zero.
  - S is intended to represent the successor function, i.e., $S(a)$ means $a + 1$.
- Γ is the following set of axioms:
  - 0 has no predecessor. $\forall x \,.\, \neg(0 = S(x))$.
  - S is injective. $\forall x, y \,.\, S(x) = S(y) \Rightarrow x = y$.
  - Induction principle.
    $\forall P \,.\, (P(0) \wedge \forall x \,.\, P(x) \Rightarrow P(S(x))) \Rightarrow \forall x \,.\, P(x)$.
- $+$ and $*$ can be defined in **PA**.
- **PA** is categorical, i.e, it has exactly one model up to isomorphism (Dedekind, 1888).

First-Order Peano Arithmetic

- $\mathbf{PA}' = (L', \Gamma')$ is first-order Peano arithmetic.
- L' is a language of FOL with an individual constant symbol 0, a unary function symbol S, and binary function symbols $+$ and $*$.
- Γ' is the following set of axioms:
  - $\forall x \,.\, \neg(S(x) = 0)$.
  - $\forall x, y \,.\, S(x) = S(y) \Rightarrow x = y$.
  - $\forall x \,.\, x + 0 = x$.
  - $\forall x, y \,.\, x + S(y) = S(x + y)$.
  - $\forall x \,.\, x * 0 = 0$.
  - $\forall x, y \,.\, x * S(y) = (x * y) + x$.
  - 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'.
- $\mathbf{PA}'$ is a noncategorical approximation of Peano arithmetic with infinitely many "nonstandard" models.

Language and Theory Extensions

- Let $L_i = (\mathcal{C}_i, \mathcal{F}_i, \mathcal{P}_i)$ be a language of FOL and let $T_i = (L_i, \Gamma_i)$ be a theory of FOL for $i = 1, 2$.
- L_1 is a sublanguage of L_2, and L_2 is a super language or an extension of L_1, written $L_1 \leq L_2$, if $\mathcal{C}_1 \subseteq \mathcal{C}_2$, $\mathcal{F}_1 \subseteq \mathcal{F}_2$, and $\mathcal{P}_1 \subseteq \mathcal{P}_2$.
- T_1 is a subtheory of T_2, and T_2 is a super theory or an extension of T_1, written $T_1 \leq T_2$, if $L_1 \leq L_2$ and $\Gamma_1 \subseteq \Gamma_2$.

Conservative Theory Extension

- Let $T = (L, \Gamma)$ and $T' = (L', \Gamma')$ be theories of FOL.
- T' is a conservative extension of T if $T \leq T'$ and, for every formula A of L, $T' \models A$ implies $T \models A$.
  - A conservative extension of a theory adds new machinery to the theory without compromising the theory's original machinery.
- The obligation of a purported conservative extension is a formula that implies that the extension is conservative.
- There are two important kinds of conservative extensions that add new vocabulary to a theory:
  1. Definitions.
  2. Profiles.

Definitions

- A definition is a conservative extension that adds a new symbol s and a defining axiom $A(s)$ to a theory T.
  - In some logics, the defining axiom can have the form $s = D$ (where s does not occur in D).
- The obligation of the definition is

  $$\exists\,!\,x\,.\,A(x).$$

- The symbol s can usually be eliminated from any new expression of involving s.

Profiles

- A profile is a conservative extension that adds a set $\{s_1, \ldots, s_n\}$ of symbols and a profiling axiom $A(s_1, \ldots, s_n)$ to a theory T.
- The obligation of the profile is

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

- The symbols $s_1, \ldots, s_n$ cannot usually be eliminated from expressions involving $s_1, \ldots, s_n$.
- Profiles can be used for introducing:
  - Underspecified objects.
  - Recursively defined functions.
  - Algebras.

Attributes of a Practical Logic: FOL

1. Formal Syntax. Yes.
2. Precise Semantics. Yes.
3. Familiarity. Yes.
4. Faithfulness. Low.
5. Theoretical Expressivity. Low.
6. Practical Expressivity. Very low.
7. Multiparadigm Reasoning. No.
8. Metalogical Reasoning. No.
9. Axiomatizability. Yes.
10. Implementability. Yes.

Ways of Making FOL More Practical

- Make the logic many-sorted by allowing several sort of individuals.
- Add definite and indefinite description.
- Modify the semantics of FOL to admit undefined expressions and partial functions.

Part 2

Simple Type Theory

Type Theory

- Russell introduced a logic now known as the ramified theory of types in 1908 to serve as a foundation for mathematics.
  - Included a hierarchy of types to avoid set-theoretic paradoxes such Russell's paradox and semantic paradoxes such as Richard's paradox.
  - Employed as the logic of Whitehead and Russell's *Principia Mathematica*.
  - Not used today due to its high complexity.
- Chwistek and Ramsey suggested in the 1920s a simplified version of the ramified theory of types called the simple theory of types or, more briefly, simple theory theory.
- Church published in 1940 a formulation of simple theory theory with lambda-notation and lambda-conversion.

Intuitionistic Type Theory

- Several intuitionistic or constructive type theories have been developed.
- Examples:
  - Martin-Löf's Intuitionistic Type Theory (1980).
  - Coquand and Huet's Calculus of Constructions (1984).
- Many intuitionistic type theories exploit the Curry-Howard Formulas-as-Types Isomorphism.
  - Formulas serve as types or specifications.
  - Terms serve as proofs or programs.

What is Simple Type Theory?

- A simple, elegant, highly expressive, and practical logic.
  - Familiar to some computer scientists but not to many mathematicians, engineers, and other scientists.
- Most popular form of type theory.
  - Types are used to classify expressions by value and control the formation of expressions.
  - Classical: nonconstructive, 2-valued.
  - Higher order: quantification over functions.
  - Can be viewed as a "function theory".
- Natural extension of first-order logic.
  - Based on the same principles as first-order logic.
  - Includes nth-order logic for all $n \geq 1$.

Who needs Simple Type Theory?

An understanding of simple type would be beneficial to anyone who needs to work with or apply mathematical logic. This is particularly true for:

- Engineers who need to write (and read) precise specifications.
- Computer scientists who employ functional programming languages such as Lisp, ML, and Haskell.
- Software engineers who use higher-order theorem proving systems to model and analyze software systems.
- Mathematics students who are studying the foundations of mathematics or model theory.

Purpose of this Presentation

- Present a pure form of simple type theory named STT.
- Show the virtues of simple type theory using STT.
- Argue that simple type theory is an attractive alternative to first-order logic for practical-minded scientists, engineers, and mathematicians.

History

1908	Russell
	Ramified theory of types.
1910	Russell, Whitehead
	Principia Mathematica.
1920s	Chwistek, Ramsey
	Simple theory of types (simple type theory).
1920–30s	Carnap, Gödel, Tarski, Quine
	Detailed formulations of simple type theory.
1940	Church
	Simple type theory with lambda-notation.
1950	Henkin
	General models and completeness theorem.
1963	Henkin, Andrews
	Concise formulation based on equality.
1980-90s	HOL, IMPS, Isabelle, ProofPower, PVS, TPS
	Higher-order theorem proving systems.

Syntax of STT: Types

- A type of STT is defined by the following rules:

$$\text{T1}\ \frac{}{\mathbf{type}[\iota]}\ \text{(Type of individuals)}$$

$$\text{T2}\ \frac{}{\mathbf{type}[*]}\ \text{(Type of truth values)}$$

$$\text{T3}\ \frac{\mathbf{type}[\alpha],\ \mathbf{type}[\beta]}{\mathbf{type}[(\alpha \to \beta)]}\ \text{(Function type)}$$

- Let $\mathcal{T}$ denote the set of types of STT.

Syntax of STT: Symbols

- The logical symbols of STT are:
  - Function application: @ (hidden).
  - Function abstraction: λ.
  - Equality: $=$.
  - Definite description: I (capital iota).
  - An infinite set $\mathcal{V}$ of symbols called variables.
- A language of STT is a pair $L = (\mathcal{C}, \tau)$ where:
  - $\mathcal{C}$ is a set of symbols called constants.
  - $\tau : \mathcal{C} \to \mathcal{T}$ is a total function.

Syntax of STT: Expressions

An expression E of type α of a STT language $L = (\mathcal{C}, \tau)$ is defined by the following rules:

$$\text{E1}\quad \frac{x \in \mathcal{V},\ \mathbf{type}[\alpha]}{\mathbf{expr}_L[(x : \alpha), \alpha]}\quad \text{(Variable)}$$

$$\text{E2}\quad \frac{c \in \mathcal{C}}{\mathbf{expr}_L[c, \tau(c)]}\quad \text{(Constant)}$$

$$\text{E3}\quad \frac{\mathbf{expr}_L[A, \alpha],\ \mathbf{expr}_L[F, (\alpha \to \beta)]}{\mathbf{expr}_L[F(A), \beta]}\quad \text{(Application)}$$

$$\text{E4}\quad \frac{x \in \mathcal{V},\ \mathbf{type}[\alpha],\ \mathbf{expr}_L[B, \beta]}{\mathbf{expr}_L[(\lambda\, x : \alpha\ .\ B), (\alpha \to \beta)]}\quad \text{(Abstraction)}$$

$$\text{E5}\quad \frac{\mathbf{expr}_L[E_1, \alpha],\ \mathbf{expr}_L[E_2, \alpha]}{\mathbf{expr}_L[(E_1 = E_2), *]}\quad \text{(Equality)}$$

$$\text{E6}\quad \frac{x \in \mathcal{V},\ \mathbf{type}[\alpha],\ \mathbf{expr}_L[A, *]}{\mathbf{expr}_L[(\mathrm{I}\, x : \alpha\ .\ A), \alpha]}\quad \text{(Definite description)}$$

Syntax of STT: Conventions

- E_α denotes an expression E of type α.
- Parentheses and the types of variables may be dropped when meaning is not lost.

Semantics of STT: Standard Models

- A standard model for a language $L = (\mathcal{C}, \tau)$ of STT is a triple $M = (\mathcal{D}, I, e)$ where:
  - $\mathcal{D} = \{D_\alpha : \alpha \in \mathcal{T}\}$ is a set of nonempty domains (sets).
  - $D_* = \{\mathrm{T}, \mathrm{F}\}$, the domain of truth values.
  - $D_{\alpha\to\beta}$ is the set of all functions from D_α to D_β.
  - I maps each $c \in \mathcal{C}$ to an element of $D_{\tau(c)}$.
  - e maps each $\alpha \in \mathcal{T}$ to a member of D_α.
- A variable assignment into M is a function that maps each expression $(x : \alpha)$ to an element of D_α.
- Given a variable assignment φ into M, an expression $(x : \alpha)$, and $d \in D_\alpha$, let $\varphi[(x : \alpha) \mapsto d]$ be the variable assignment φ' into M such that $\varphi'((x : \alpha)) = d$ and $\varphi'(v) = \varphi(v)$ for all $v \neq (x : \alpha)$.

Semantics of STT: Valuation Function

The valuation function for a standard model $M = (\mathcal{D}, I, e)$ for a language $L = (\mathcal{C}, \tau)$ of STT is the binary function V^M that satisfies the following conditions for all variable assignments φ into M and all expressions E of L:

1. Let E is $(x : \alpha)$. Then $V^M_\varphi(E) = \varphi((x : \alpha))$.
2. Let $E \in \mathcal{C}$. Then $V^M_\varphi(E) = I(E)$.
3. Let E be $F(A)$. Then $V^M_\varphi(E) = V^M_\varphi(F)(V^M_\varphi(A))$.
4. Let E be $(\lambda\, x : \alpha\ .\ B_\beta)$. Then $V^M_\varphi(E)$ is the $f : D_\alpha \longrightarrow D_\beta$ such that, for each $d \in D_\alpha$, $f(d) = V^M_{\varphi[(x:\alpha)\mapsto d]}(B_\beta)$.
5. Let E be $(E_1 = E_2)$. If $V^M_\varphi(E_1) = V^M_\varphi(E_2)$, then $V^M_\varphi(E) = \mathrm{T}$; otherwise $V^M_\varphi(E) = \mathrm{F}$.
6. Let E be $(\mathrm{I}\, x : \alpha\ .\ A)$. If there is a unique $d \in D_\alpha$ such that $V^M_{\varphi[(x:\alpha)\mapsto d]}(A) = \mathrm{T}$, then $V^M_\varphi(E) = d$; otherwise $V^M_\varphi(E) = e(\alpha)$.

Notational Definitions

T	means	$(\lambda\, x : * \,.\, x) = (\lambda\, x : * \,.\, x)$.
F	means	$(\lambda\, x : * \,.\, \mathsf{T}) = (\lambda\, x : * \,.\, x)$.
$(\neg A_*)$	means	$A_* = \mathsf{F}$.
$(A_\alpha \neq B_\alpha)$	means	$\neg(A_\alpha = B_\alpha)$.
$(A_* \wedge B_*)$	means	$(\lambda\, f : * \to (* \to *) \,.\, f(\mathsf{T})(\mathsf{T})) =$ $(\lambda\, f : * \to (* \to *) \,.\, f(A_*)(B_*))$.
$(A_* \vee B_*)$	means	$\neg(\neg A_* \wedge \neg B_*)$.
$(A_* \Rightarrow B_*)$	means	$\neg A_* \vee B_*$.
$(A_* \Leftrightarrow B_*)$	means	$A_* = B_*$.
$(\forall\, x : \alpha \,.\, A_*)$	means	$(\lambda\, x : \alpha \,.\, A_*) = (\lambda\, x : \alpha \,.\, \mathsf{T})$.
$(\exists\, x : \alpha \,.\, A_*)$	means	$\neg(\forall\, x : \alpha \,.\, \neg A_*)$.
$\bot_\alpha$	means	$\mathrm{I}\, x : \alpha \,.\, x \neq x$.
$\mathsf{if}(A_*, B_\alpha, C_\alpha)$	means	$\mathrm{I}\, x : \alpha \,.\, (A_* \Rightarrow x = B_\alpha) \wedge (\neg A_* \Rightarrow x = C_\alpha)$ where x does not occur in A_*, B_α, or C_α

Expressivity

- Theorem. *There is a faithful interpretation of nth-order logic in STT for all $n \geq 1$.*
- Most mathematical notions can be directly and naturally expressed in STT.
- Examples:

$$
\begin{aligned}
&\text{equiv-rel} = \\
&\quad \lambda\, p : (\iota \to (\iota \to *)) \,. \\
&\qquad \forall\, x : \iota \,.\, p(x)(x) \wedge \\
&\qquad \forall\, x, y : \iota \,.\, p(x)(y) \Rightarrow p(y)(x) \wedge \\
&\qquad \forall\, x, y, z : \iota \,.\, (p(x)(y) \wedge p(y)(z)) \Rightarrow p(x)(z) \\
&\text{compose} = \\
&\quad \lambda\, f : (\iota \to \iota) \,.\, \lambda\, g : (\iota \to \iota) \,.\, \lambda\, x : \iota \,.\, f(g(x)) \\
&\text{inv-image} = \\
&\quad \lambda\, f : (\iota \to \iota) \,.\, \lambda\, s : (\iota \to *) \,. \\
&\qquad \mathrm{I}\, s' : (\iota \to *) \,.\, \forall\, x : \iota \,.\, s'(x) \Leftrightarrow s(f(x))
\end{aligned}
$$

Peano Arithmetic

- Let **PA** $= (L, \Gamma)$ be the theory of STT such that:
  $L = (\{0, S\}, \tau)$ where $\tau(0) = \iota$ and $\tau(S) = \iota \to \iota$.
  Γ is the set of the following three formulas:
  1. 0 has no predecessor: $\forall x : \iota \,.\, 0 \neq S(x)$.
  2. S is injective: $\forall x, y : \iota \,.\, S(x) = S(y) \Rightarrow x = y$.
  3. Induction principle:
     $\forall P : \iota \to * \,.$
     $P(0) \wedge (\forall x : \iota \,.\, P(x) \Rightarrow P(S(x))) \Rightarrow \forall x : \iota \,.\, P(x)$.
- Theorem (Dedekind, 1888). **PA** *has (up to isomorphism) a unique standard model* $M = (\mathcal{D}, I, e)$ *where* $D_\iota = \{0, 1, 2, \ldots\}$.

Complete Ordered Field (1/3)

- Let **COF** $= (L, \Gamma)$ be the theory of STT such that: $L = (\{+, 0, -, \cdot, 1, {}^{-1}, \mathsf{pos}, <, \leq, \mathsf{ub}, \mathsf{lub}\}, \tau)$ where

Constant c	Type $\tau(c)$
0,1	ι
$-, {}^{-1}$	$\iota \to \iota$
pos	$\iota \to *$
$+, \cdot$	$\iota \to (\iota \to \iota)$
$<, \leq$	$\iota \to (\iota \to *)$
ub, lub	$\iota \to ((\iota \to *) \to *)$

Complete Ordered Field (2/3)

Γ is the set of the following eighteen formulas:

1. $\forall x, y, z : \iota \,.\, (x + y) + z = x + (y + z)$.
2. $\forall x, y : \iota \,.\, x + y = y + x$.
3. $\forall x : \iota \,.\, x + 0 = x$.
4. $\forall x : \iota \,.\, x + (-x) = 0$.
5. $\forall x, y, z : \iota \,.\, (x \cdot y) \cdot z = x \cdot (y \cdot z)$.
6. $\forall x, y : \iota \,.\, x \cdot y = y \cdot x$.
7. $\forall x : \iota \,.\, x \cdot 1 = x$.
8. $\forall x : \iota \,.\, x \neq 0 \Rightarrow x \cdot x^{-1} = 1$.
9. $0 \neq 1$.
10. $\forall x, y, z : \iota \,.\, x \cdot (y + z) = (x \cdot y) + (x \cdot z)$.
11. $\forall x : \iota \,.\, (x = 0 \wedge \neg \mathsf{pos}(x) \wedge \neg \mathsf{pos}(-x)) \vee$
    $(x \neq 0 \wedge \mathsf{pos}(x) \wedge \neg \mathsf{pos}(-x)) \vee$
    $(x \neq 0 \wedge \neg \mathsf{pos}(x) \wedge \mathsf{pos}(-x))$.

Complete Ordered Field (3/3)

12. $\forall x, y : \iota \,.\, (\mathsf{pos}(x) \wedge \mathsf{pos}(y)) \Rightarrow \mathsf{pos}(x + y)$.
13. $\forall x, y : \iota \,.\, (\mathsf{pos}(x) \wedge \mathsf{pos}(y)) \Rightarrow \mathsf{pos}(x \cdot y)$.
14. $\forall x, y : \iota \,.\, x < y \Leftrightarrow \mathsf{pos}(y - x)$.
15. $\forall x, y : \iota \,.\, x \leq y \Leftrightarrow (x < y \vee x = y)$.
16. $\forall x : \iota \,.\, \forall s : \iota \to * \,.\, \mathsf{ub}(x)(s) = \forall y : \iota \,.\, s(y) \Rightarrow y \leq x$.
17. $\forall x : \iota \,.\, \forall s : \iota \to * \,.$
    $\mathsf{lub}(x)(s) = (\mathsf{ub}(x)(s) \wedge (\forall y : \iota \,.\, \mathsf{ub}(y)(s) \Rightarrow x \leq y))$.
18. $\forall s : \iota \to * \,.$
    $\exists x : \iota \,.\, s(x) \wedge \exists x : \iota \,.\, \mathsf{ub}(x)(s) \Rightarrow \exists x : \iota \,.\, \mathsf{lub}(x)(s)$.

- Theorem. **COF** *has (up to isomorphism) a unique standard model* $M = (\mathcal{D}, I, e)$ *where* $D_\iota = \mathbf{R}$, *the set of real numbers.*

Incompleteness of STT

Theorem. *There is no sound and complete proof system for STT.*

Proof. Suppose **P** is a sound and complete proof system for STT. By the soundness of **P** and Gödel's Incompleteness Theorem, there is a sentence A such that (1) $M \models A$, where M is the unique standard model for **PA** (up to isomorphism), and (2) $\mathbf{PA} \nvdash_{\mathbf{P}} A$. By the completeness of **P**, (2) implies $\mathbf{PA} \not\models A$ and hence $M \not\models A$ since M is the only standard model of **PA**, which contradicts (1). □

A Hilbert-Style Proof System for STT (1/2)

- Axioms:

  A1 (Truth Values)

  $\forall f : * \to * \,.\, (f(\mathsf{T}_*) \wedge f(\mathsf{F}_*)) \Leftrightarrow (\forall x : * \,.\, f(x))$.

  A2 (Leibniz' Law)

  $\forall x, y : \alpha \,.\, (x = y) \Rightarrow (\forall p : \alpha \to * \,.\, p(x) \Leftrightarrow p(y))$.

  A3 (Extensionality)

  $\forall f, g : \alpha \to \beta \,.\, (f = g) = (\forall x : \alpha \,.\, f(x) = g(x))$.

  A4 (Beta-Reduction)

  $(\lambda x : \alpha \,.\, B_\beta)(A_\alpha) = B_\beta[(x : \alpha) \mapsto A_\alpha]$

  provided A_α is free for x in B_β.

  A5 (Proper Definite Description)

  $(\exists ! x : \alpha \,.\, A) \Rightarrow A[(x : \alpha) \mapsto (\mathrm{I}\, x : \alpha \,.\, A)]$.

  A6 (Improper Definite Description)

  $\neg(\exists ! x : \alpha \,.\, A) \Rightarrow (\mathrm{I}\, x : \alpha \,.\, A) = \bot_\alpha$.

A Hilbert-Style Proof System for STT (2/2)

- Rule of inference:

  R (Equality Substitution)
  From $A_\alpha = B_\alpha$ and C_* infer the result of replacing one occurrence of A_α in C_* by an occurrence of B_α.

- Call this proof system **A**.
  - Due to Andrews, 1963.
- Theorem (Jensen, 1969). **A** *plus an axiom of infinity is equiconsistent with bounded Zermelo set theory.*

General Models

- A general structure for a language $L = (\mathcal{C}, \tau)$ of STT is a triple $M = (\mathcal{D}, I, e)$ where:
  - $\mathcal{D} = \{D_\alpha : \alpha \in \mathcal{T}\}$ is a set of nonempty domains (sets).
  - $D_* = \{\mathrm{T}, \mathrm{F}\}$, the domain of truth values.
  - $D_{\alpha\to\beta}$ is some set of functions from D_α to D_β.
  - I maps each $c \in \mathcal{C}$ to an element of $D_{\tau(c)}$.
  - e maps each $\alpha \in \mathcal{T}$ to a member of D_α.
- M is a general model for L if there is a binary function V^M that satisfies the same conditions as the valuation function for a standard model.
- A general model is a nonstandard model if it is not a standard model.

Completeness of STT

Theorem (Henkin, 1950). *There is a sound and complete proof system for STT with respect to general models.*

Corollary. *STT is compact with respect to general models.*

Theorem (Andrews, 1963). **A** *is a sound and complete proof system for STT with respect to general models.*

Attributes of a Practical Logic: STT

1. Formal Syntax. Yes.
2. Precise Semantics. Yes.
3. Familiarity. Yes.
4. Faithfulness. Moderate.
5. Theoretical Expressivity. High.
6. Practical Expressivity. Moderate.
7. Multiparadigm Reasoning. Functions and sets.
8. Metalogical Reasoning. No.
9. Axiomatizability. Yes.
10. Implementability. Yes.

Ways of Making STT More Practical

- Make the logic many-sorted by allowing several types of individuals, e.g., $\iota_1, \ldots, \iota_n$.
- Add machinery for basic mathematical objects such as sets, tuples, and lists.
- Add indefinite description.
- Modify the semantics of STT to admit undefined expressions and partial functions.
- Admit polymorphic operators like $(\lambda\, x : t \;.\; x)$ and user-defined type constructors by introducing type variables.
- Extend the type system of STT to support subtypes and dependent types.

Theorem Proving Systems Based on Variants of STT

- HOL (Gordon).
- IMPS (Farmer, Guttman, Thayer).
- Isabelle (Paulson).
- ProofPower (Lemma 1).
- PVS (Owre, Rushby, Shankar).
- TPS (Andrews).

Conclusion

- Simple type theory is a logic that is effective for practice as well as theory—unlike first-order logic.
  - More expressive and more convenient.
  - Closer to mathematical practice.
  - Based on the same principles as first-order logic.
  - Includes the full machinery of first-order logic.
  - Integrates predicate logic, function theory, and type theory.
- We recommend that simple type theory be incorporated into:
  - Logic courses offered by mathematics departments.
  - The undergraduate curriculum for computer science and software engineering students.

The Seven Virtues

Virtue 1: STT has a simple and highly uniform syntax.

Virtue 2: The semantics of STT is based on a small collection of well-established ideas.

Virtue 3: STT is a highly expressive logic.

Virtue 4: STT admits categorical theories of infinite structures.

Virtue 5: There is a proof system for STT that is simple, elegant, and powerful.

Virtue 6: Henkin's general models semantics enables the techniques of first-order model theory to be applied to STT and illuminates the distinction between standard and nonstandard models.

Virtue 7: There are practical variants of STT that can be effectively implemented.

References: Key Historical Publications

1. B. Russell, "Mathematical Logic as Based on the Theory of types", *American Journal of Mathematics*, 30:222–262, 1908.
2. A. N. Whitehead and B. Russell, *Principia Mathematica*, Cambridge University Press, 1910–13.
3. A. Church, "A Formulation of the Simple Theory of Types", *Journal of Symbolic Logic*, 5:56–68, 1940.
4. L. Henkin, "Completeness in the Theory of Types", *Journal of Symbolic Logic*, 15:81–91, 1950.
5. L. Henkin, "A Theory of Propositional Types", *Fundamenta Mathematicae*, 52:323–344, 1963.
6. P. B. Andrews, "A Reduction of the Axioms for the Theory of Propositional Types", *Fundamenta Mathematicae* 52:345–350, 1963.

References: Introductions

1. P. B. Andrews, *An Introduction to Mathematical Logic and Type Theory: To Truth through Proof*, Second Edition, Kluwer, 2002.
2. W. M. Farmer, "A basic extended simple type theory", SQRL Report No. 14, 12 pp., McMaster University, 2003 (revised 2004).
3. W. M. Farmer, "The Seven Virtues of Simple Type Theory", *Journal of Applied Logic*, forthcoming.

Part 3

Zermelo-Fraenkel (ZF) Set Theory

Set Theory as a Logic

- A set theory is a good candidate for a logic since almost all mathematical objects and concepts can be defined in terms of sets.
- Informal set theories (so-called naive set theories) capture the basic concepts of a set but admit the set-theoretic paradoxes like Russell's paradox.
- Some formal set theories (such as **BA**) capture only certain aspects about sets (algebraic concepts) and are thus not suitable to serve as general purpose logics.
- A formal set theory suitable for use as a formal logic is relatively complicated because it needs:
  1. A mechanism to avoid the set-theoretic paradoxes.
  2. Powerful tools for building sets.

Formalizations of Set Theory

- The standard formalization of set theory is known as Zermelo-Fraenkel (ZF) set theory [Zermelo, 1908].
- Other major formalizations:
  - von-Neumann-Bernays-Gödel (NBG) set theory [von Neumann, 1925].
  - Morse-Kelley (MK) set theory [Kelley, 1955].
  - Tarski-Grothendieck set theory [Tarski, 1938].
  - New Foundations (NF) [Quine, 1937].

ZF

- Proposed by Zermelo in 1908.
  - Developed to avoid the set-theoretic paradoxes.
  - Improvements made by Fraenkel (1922) and Skolem (1923).
- ZF is formalized as a theory in first-order logic.
  - Language contains two predicate symbols $=$ and $\in$.
  - Not finitely axiomatizable.
- Proper classes (e.g., the collection of all sets) are not first-class objects.
  - They cannot be denoted by terms.
  - They are used in the metatheory.
  - They can be denoted by predicate symbols.
- ZF is an exceedingly rich theory.
  - Total functions are effectively polymorphic.

NBG

- Proposed by von Neumann in 1925.
  - Improvements made by R. Robinson (1937), Bernays (1937–54), and Gödel (1940).
- NBG is formalized as a theory in first-order logic.
  - Has the same language as ZF.
  - Finitely axiomatizable!
- Proper classes are first-class objects.
- NBG is closely related to ZF.
  - NBG is a conservative extension of ZF.
  - NBG is consistent iff ZF is consistent.
  - NBG and ZF share the same intuitive model of the iterated hierarchy of sets.
  - NBG and ZF have very similar axioms.

ZF Axioms (1/2)

1. Extensionality.

$$\forall x, y \,.\, ((\forall u \,.\, u \in x \Leftrightarrow u \in y) \Rightarrow x = y).$$

2. Foundation.

$$\forall x \,.\, (x \neq \emptyset \Rightarrow \exists y \,.\, (y \in x \wedge x \cap y = \emptyset)).$$

3. Empty Set.

$$\exists x \,.\, \neg(\exists y \,.\, y \in x).$$

4. Pairs.

$$\forall x, y \,.\, \exists z \,.\, \forall u \,.\, (u \in z \Leftrightarrow (u = x \vee u = y)).$$

5. Sum Set.

$$\forall x \,.\, \exists y \,.\, \forall u \,.\, (u \in y \Leftrightarrow \exists z \,.\, (u \in z \wedge z \in x)).$$

6. Power Set.

$$\forall x \,.\, \exists y \,.\, \forall u \,.\, (u \in y \Leftrightarrow u \subseteq x).$$

ZF Axioms (2/2)

7. Infinity.

$$\exists x \,.\, (\emptyset \in x \wedge \forall u \,.\, (u \in x \Rightarrow u \cup \{u\} \in x)).$$

8. Restricted Comprehension Schema.
   Each universal closure of

$$\forall x \,.\, \exists y \,.\, \forall u \,.\, (u \in y \Leftrightarrow (u \in x \wedge A))$$

   where A is any formula in which y does not occur.

9. Replacement Schema.
   Each universal closure of

$$(\forall w \,.\, (\forall x \,.\, \exists ! \, y \,.\, A) \Rightarrow \exists z \,.\, \forall y \,.\, (y \in z \Leftrightarrow \exists x \,.\, (x \in w \wedge A))).$$

   where A is any formula in which z does not occur..

10. Choice.

$$\forall x \,.\, \exists y \,.\, \mathsf{fun}(y) \wedge \mathsf{dom}(y) = x \wedge \forall u \,.\, ((u \in x \wedge u \neq \emptyset) \Rightarrow y(u) \in u).$$

Remarks on the ZF Axioms

- The Restricted Comprehension Schema is restricted to avoid the set-theoretic paradoxes like Russell's paradox.
- Zermelo-Frankel set theory (ZF) has axioms 1–9.
- Zermelo set theory (Z) has axioms 1-8.
- ZF with the Axiom of Choice (ZFC) has axioms 1-10.
- The Axiom of Foundation is dispensable and could be replaced with an "antifoundation" axiom.
- The Axiom of Empty Set can be proved from the other axioms.

Axioms for NBG Set Theory

1. Definition of V, the universal class.
2. Axioms 1 and 2.
3. Relativizations to V of axioms 3–7.
4. NBG Class Comprehension Schema.
   Each universal closure of

   $$\exists x \,.\, \forall u \,.\, (u \in x \Leftrightarrow (u \in V \wedge A^V)).$$

   where A is any formula in which x does not occur and A^V is A relativized to V.
5. A single formula version of the Axiom of Replacement.
6. Various versions of the Axiom of Choice.

Axioms for MK Set Theory

- Same axioms as NBG except NBG Comprehension is replaced with the stronger:

  MK Class Comprehension Schema.
  Each universal closure of

  $$\exists\, x \,.\, \forall\, u \,.\, (u \in x \Leftrightarrow (u \in V \wedge A)).$$

  where A is any formula in which x does not occur.

- MK is a nonconservative extension of NBG.
- MK is not finitely axiomatizable.

General Set Theory (GST)

- GST has the following axioms:
  1. Extensionality.
  2. Restricted Comprehension Schema.
  3. Adjunction.

$$\forall x, y \,.\, \exists z \,.\, \forall u \,.\, (u \in z \Leftrightarrow (u \in x \vee x = y)).$$

- GST is mutually interpretable with First-Order Peano Arithmetic.
- GST is not finitely axiomatizable.

Attributes of a Practical Logic: ZF

1. Formal Syntax. Yes.
2. Precise Semantics. Yes.
3. Familiarity. Yes.
4. Faithfulness. Moderate.
5. Theoretical Expressivity. Extremely high.
6. Practical Expressivity. Low.
7. Multiparadigm Reasoning. Indirect.
8. Metalogical Reasoning. No.
9. Axiomatizability. Yes.
10. Implementability. Yes.

Ways of Making ZF More Practical

- Add machinery for functions.
- Add a type system.
- Add definite description and indefinite description.
- Modify the semantics of ZF to admit undefined expressions and partial functions.