CAS 701 Fall 2002

04 Simple Type Theory

Instructor: W. M. Farmer

Revised: 11 November 2002

What is Higher-Order Logic?

- Higher-order functions (or predicates) can be represented by terms
  - Note: A function $f : A \rightarrow B$ is **higher-order** if A or B contains functions
- Quantified variables can range over functions
- Types or sorts are used to:
  - Organize the functions of the logic
  - Control the formation of expressions
  - Classify expressions by value

Type Theory

- A higher-order logic can be viewed as a theory of types

- Russell introduced a logic called the **Theory of Types (TT)** in 1908 to serve as a foundation for mathematics
  - Included a hierarchy of types to avoid set-theoretic paradoxes like Russell's Paradox
  - Employed as the logic of Whitehead and Russell's **Principia Mathematica**
  - Not used today due to its high complexity

- Carnap, Chwistek, Ramsey and others suggested a simplified version of TT called **Simple Type Theory (STT)** in the 1920s
  - A formulation of STT with lambda-notation was introduced by Church in 1940

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

Syntax of STT: Types

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

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

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

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

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

Syntax of STT: Symbols

- The **logical symbols** are present in every STT language:
  - Propositional connectives: $\neg$, $\Rightarrow$
  - Function application and abstraction: @ (hidden), λ
  - Equality: $=$
- The **nonlogical symbols** characterize an STT language
- A **language** of STT is a tuple $L = (\mathcal{V}, \mathcal{C}, \tau)$ where:
  - $\mathcal{V}$ is an infinite set of symbols called **variables**
  - $\mathcal{C}$ is a set of symbols called **constants**
  - $\mathcal{V}$ and $\mathcal{C}$ are disjoint
  - $\tau : \mathcal{C} \to \mathcal{T}$ is a total function

Syntax of STT: Expressions

- An **expression** e of **type** α of an STT language $L = (\mathcal{V}, \mathcal{C}, \tau)$ is defined by the following rules:

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

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

E3 $$\frac{\mathbf{expr}_L[a, \alpha],\ \mathbf{expr}_L[f, (\alpha \to \beta)]}{\mathbf{expr}_L[f(a), \beta]}$$ **(Application)**

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

E5 $$\frac{\mathbf{expr}_L[e_1, \alpha],\ \mathbf{expr}_L[e_2, \alpha]}{\mathbf{expr}_L[(e_1 = e_2), *]}$$ **(Equality)**

Semantics of STT: Models

- A **model** of a language $L = (\mathcal{V}, \mathcal{C}, \tau)$ of STT is a pair $M = (D, I)$ where:
  - $D = \{D_\alpha : \alpha \in \mathcal{T}\}$
  - D_ι is nonempty
  - $D_* = \{\mathsf{T}, \mathsf{F}\}$
  - $D_{(\alpha\to\beta)}$ is the set of functions from D_α to D_β
  - I maps each $c \in \mathcal{C}$ to an element of $D_{\tau(c)}$
- A **variable assignment** into M is a function that maps each expression $(x : \alpha)$ with $x \in \mathcal{V}$ to an element of D_α
- Given a variable assignment A into M, $(x : \alpha)$ with $x \in \mathcal{V}$, and $d \in D_\alpha$, let $A[(x : \alpha) \mapsto d]$ be the variable assignment A' into M such $A'((x : \alpha)) = d$ and $A'(v) = A(v)$ for all $v \neq (x : \alpha)$

Semantics of STT: Valuation

- Let $M = (D, I)$ be a model for a language $L = (\mathcal{V}, \mathcal{C}, \tau)$ of STT

- The **valuation function** of STT is the binary function V^M that satisfies the following conditions for all variable assignments A into M and all expressions e of L:

  1. If e is $(x : \alpha)$, then $V_A^M(e) = A((x : \alpha))$
  2. If $e \in \mathcal{C}$, then $V_A^M(e) = I(e)$
  3. If e is $f(a)$, then $V_A^M(e) = V_A^M(f)(V_A^M(a))$
  4. If e is $(\lambda\, x : \alpha\ .\ b)$ and b is of type β, then $V_A^M(e)$ is the function $F : D_\alpha \to D_\beta$ such that, for each $d \in D_\alpha$, $F(d) = V^M_{A[(x:\alpha)\mapsto d]}(b)$
  5. If e is $(e_1 = e_2)$, then $V_A^M(e) = \begin{cases} \mathsf{T} & \text{if } V_A^M(e_1) = V_A^M(e_2) \\ \mathsf{F} & \text{otherwise} \end{cases}$

Alternate Semantics for STT

- General models semantics
  - A **general model** for a language of STT has function domains that are not necessarily fully inhabited
  - Henkin showed that STT is complete respect to the general models semantics
  - Reference: L. Henkin, "Completeness in the theory of types", *Journal of Symbolic Logic*, 15:81–91, 1950
- Partial functions semantics
  - Terms may be undefined, but formulas are always true or false
  - Two versions: standard models and generals model
  - The IMPS logic is an extension of STT with a partial functions semantics
  - Reference: W. Farmer, "A partial functions version of Church's simple theory of types," *Journal of Symbolic Logic*, 55:1269–1291, 1990

Extensions to STT

- Types
  - Additional type constants and constructors
  - Type variables
  - Subtypes
- Expressions
  - Additional expression constants and constructors
  - Multivariate functions
- Examples:
  - HOL logic
  - PVS logic
  - IMPS logic
  - BESTT, a Basic Extended Simple Type Theory