CAS 760

Simple Type Theory

Winter 2026

2 Introduction to Simple Type Theory

William M. Farmer

Department of Computing and Software
McMaster University

December 29, 2025

Outline

1. Simple type theory.
2. Church's type theory.
3. Model theory vs. proof theory.

1. Simple Type Theory

What is Simple Type Theory?

- Simple type theory (STT) is a classical higher-order version of predicate logic and a 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 sets and functions.
  - Can be viewed as a weak set theory.
- Natural extension of first-order logic.
  - Based on the same principles as first-order logic.
  - Includes *n*th-order logic for all $n \geq 1$.
- Simple type theory is simpler than dependent type theory, a constructive form of type theory.
  - Types are simple types (i.e., nondependent) types).

History

1908	Russell
	Ramified theory of types.
1910-12	Russell, Whitehead
	Principia Mathematica.
1920s	Chwistek, Ramsey
	Simple theory of types (simple type theory).
1930s	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, HOL Light, IMPS, Isabelle, ProofPower, PVS, TPS
	Higher-order theorem proving systems.

Seven Virtues of Simple Type Theory

1. Simple type theory has a simple and highly uniform syntax.
2. The semantics of simple type theory is based on a small collection of well-established ideas.
3. Simple type theory is a highly expressive logic.
4. Simple type theory admits categorical theories of infinite structures.
5. There is a proof system for simple type theory that is simple, elegant, and powerful.
6. Henkin's general models semantics enables the techniques of first-order model theory to be applied to simple type theory and illuminates the distinction between standard and nonstandard models.
7. There are practical extensions of simple type theory that can be effectively implemented.

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

2. Church's Type Theory

Church's Type Theory

- Church's type theory (CTT) is a very practical version of simple type theory introduced by Alonzo Church in 1940.
  - Tailored for reasoning with functions.
  - Includes lambda notation, definite description, a type of Boolean values.
- Has been highly influential in computing.
  - Inspired the use of types in programming languages.
  - Inspired dependent type theories such as Martin-Löf type theory and the calculus of constructions.
  - Underlying logic for several proof assistants.
- Improved by Church's students.
  - Axiomatized using just equality, function application, and function abstraction (Leon Henkin).
  - Given an elegant proof system (Peter Andrews).
- Is now the dominate form of simple type theory.

Proof Assistants Based on Church's Type Theory

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

Alternatives to Church's Type Theory

- First-order logic.
  - Comparable theoretical expressivity as STT.
  - Much less practical expressivity than STT.
- Set theory.
  - Focuses on reasoning using sets.
  - Much greater theoretical expressivity than STT.
  - Much less practical expressivity than STT.
  - Lacks types and support for reasoning with functions.
- Dependent type theory.
  - Focuses on reasoning using dependent types.
  - Has rich type system, exploits the Curry-Howard isomorphism, and adheres to constructive reasoning.
  - Ideal for exploring where programming and proving meet.
  - A "bridge too far" for the vast majority of mathematics practitioners.

Alonzo

- Version of Church's type theory.
- Named in honor of Church.
- Inspired by $\mathcal{Q}_0$, Andrews' version of Church's type theory.
- How Alonzo differs from $\mathcal{Q}_0$:
  - Has seven expression constructors including equality.
  - Has no logical constants except expression constructors.
  - Includes support for ordered pairs.
  - Admits undefined expressions (in accordance with the traditional approach to undefinedness).
- Exceptionally well suited for expressing and reasoning about mathematical ideas.
  - Ideally suited for mathematical structures.
  - Has a rich set of notational definitions and conventions.
  - Has quasitypes for representing subtypes.
  - Has two semantics, one for logic and one for math.

3. Model Theory vs. Proof Theory

Model Theory vs. Proof Theory

- Ways to prove B is a logical consequence of $\{A_1, \ldots, A_n\}$:
  1. Give a model-theoretic argument that B is a semantic consequence of $\{A_1, \ldots, A_n\}$.
  2. Give a proof-theoretic argument that B is a syntactic consequence of $\{A_1, \ldots, A_n\}$ in a sound proof system.
  3. Construct a derivation of B from $\{A_1, \ldots, A_n\}$ in a sound proof system.
- 1 is usually easier to understand than 2 and 3, and thus easier to read and write.
- 2 is challenging since it requires an understanding of both the semantics and proof system.
- 3 is usually impossible to do without software support, but 3 can be mechanically checked, unlike 1 and 2.
- Simple Type Theory focuses on proofs of form 1.

The End.