CAS 734 Winter 2014

07 Practice-Oriented Logics

William M. Farmer

Department of Computing and Software
McMaster University

22 February 2014

Theory-Oriented vs. Practice-Oriented Logics

- Most traditional logics are theory oriented: they are designed to be studied and used only for theoretical purposes.
  - Examples: First-order logic, Zermelo-Fraenkel set theory.
- A practice-oriented logic is intended for actual use in practice by engineers, scientists, mathematicians, and students.
  - Often are modifications of traditional logics.
  - Examples: Versions of Church's type theory used in the HOL, IMPS, PVS, and TPS systems.
- The logic of a proof assistant needs to be practice oriented.

Expressivity of a Logic

- The theoretical expressivity of a logic is the measure of what ideas can be expressed in the logic without regard to how the ideas are expressed.
- The practical expressivity of a logic is the measure of how readily ideas can be expressed in the logic.
- A good practice-oriented logic should have both high theoretical and practical expressivity.

Issues concerning Practicality in a Logic

1. Basic mathematical values.
2. Types.
3. Undefinedness.
4. Polymorphism.
5. Definite and indefinite description.
6. Syntactic values.

Issue 1: Basic Mathematical Objects

A practice-oriented logic needs strong support for the basic mathematical objects:

- Truth values.
- Strings.
- Numbers:
  - Natural numbers.
  - Integers.
  - Rational numbers.
  - Real numbers.
  - Complex numbers.
- Compound values:
  - Sets.
  - Functions.
  - Relations.
  - Tuples.
  - Sequences.

Support for Basic Mathematical Objects

- Literals for strings and numbers.
- Types.
- Quantification.
- Abstraction mechanisms (e.g., lambda-abstraction).
- Subtyping for numbers.
- Way of handling improper function application.

Issue 2: Types

- The use of types can make a logic much more practical than it would otherwise be.
- Types can be used in a logic to:
  1. Restrict the scope of variables.
  2. Restrict the scope of operators.
  3. Control the formation of expressions.
  4. Classify expressions by their values.
- In mathematical practice, types are informal and used mainly for restricting the scope of variables.
- Important issues:
  - How is type checking performed?
  - Can types be hidden?
  - Can types be inferred?
  - Is type checking/inference decidable?

Type Systems

- Possible components of a type system:
  - Type constants including base types.
  - Type constructors like $\alpha \to \beta$ and $\alpha \times \beta$.
  - Dependent function (product) type constructor: $\prod x : \alpha \,.\, \beta$.
  - Dependent pair (sum) type constructor: $\sum x : \alpha \,.\, \beta$.
  - A universal type.
  - Possibly empty types.
  - Type variables.
  - Subtypes.
- Implementation approaches:
  - The type system is built into the logic.
  - The type system is simulated using predicates.

Issue 3: Undefinedness

- A mathematical term is undefined if it has no prescribed meaning or if it denotes a value that does not exist.
  - Undefined terms are commonplace in mathematics.
- Sources of undefinedness:
  1. Improper function applications: $\sqrt{-4}$.
  2. Improper definite descriptions:

     "the x such that $x^2 = 4$".
  3. Improper indefinite descriptions:

     "some x such $x^2 = -4$".
- A practice-oriented logic needs a way of handling undefinedness:
  - Ill-formed terms
  - Unspecified values
  - Error values
  - Traditional approach to undefinedness

Issue 4: Polymorphism

- An operator is polymorphic if it can be applied to expressions of different types.
- Polymorphic operators are not usually needed in mathematical practice since, by convention, operators can be applied to all expressions (but the applications may be undefined).
- A practice-oriented logic needs polymorphic operators in some form:
  - Type variables.
  - Macro-abbreviations.
  - All values are members of a universal class.

Issue 5: Definite and Indefinite Description

- A definite description is an expression of the form "the x such that A" written formally as $\iota\, x\ .\ A$.
- An indefinite description is an expression of the form "some x such that A" written formally as $\epsilon\, x\ .\ A$.
- Definite descriptions, and to a less extent indefinite descriptions, are quite common in mathematical practice, but they often occur in a disguised form.
- Improper definite and indefinite descriptions are undefined.
- A practice-oriented logic needs either definite description or indefinite description.

Issue 6: Syntactic Values

- An expression has two meanings:
  1. The value it denotes.
  2. Its syntactic structure.
- Both meanings are important in mathematics, but the distinction between them is often confused.
- A practice-oriented logic needs to be able to reason about both meanings of an expression.
- Important issues:
  - How is the liar paradox avoided?
  - How are expression fragments handled?
  - How are the notions of a free variable, substitution for a variable, etc. defined?

Syntax Framework

- A syntax framework is an abstract model of a system for reasoning about the syntax of an interpreted language L.
- A syntax framework contains:
  1. A syntax representation that maps each expression e in L to a syntactic value that represents the syntactic structures of e.
  2. A language L_{syn} called a syntax language whose expressions denote syntactic values.
  3. A quotation function that maps an expression e in L to an expression $\ulcorner e \urcorner$ in the syntax language L_{syn} that denotes the syntactic value of e.
  4. An evaluation function that maps an expression e in L_{syn} to an expression $[\![e]\!]$ in L whose semantic value is the same as that of the expression in L whose syntactic value is denoted by e.

Candidates for a Practice-Oriented Logic

- Higher-order logics.
  - Simple type theory.
  - Extensions of simple type theory.
  - Constructive type theories.
- Set theories.
  - Zermelo-Fraenkel (ZF) set theory.
  - Von-Neumann-Bernays-Gödel (NBG) set theory.
- Note: First-order logic is not good candidate for a practice-oriented logic.

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.

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.

Axioms of ZF

1. Extensionality.
2. Foundation.
3. Comprehension scheme.
4. Pairing.
5. Union.
6. Replacement scheme.
7. Powerset.
8. Infinity.
9. Choice.

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 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.

Some Proposed Practice-Oriented Logics

- LUTINS, the IMPS logic.
  - Version of Church's type theory with undefinedness and a subtype system.
  - Addresses issues 1,2,3,5.
- BESTT, a Basic Extended Simple Type Theory.
  - Version of Church's type theory with undefinedness, tuples, lists, and sets.
  - Addresses issues 1–5.
- STMM, a Set Theory for Mechanized Mathematics.
  - Version of NBG set theory with types and undefinedness.
  - Addresses issues 1–5.
- Chiron.
  - Version of NBG set theory with types, undefinedness, quotation, and evaluation.
  - Addresses issues 1–6.

Advantages Chiron has over NBG

- Types.
  - Has types as well as terms and formulas.
  - Types are assigned to terms.
  - Type system includes a universal type, type constructors, dependent function types, dependent pair types, possibly empty types, and subtypes.
- Operators.
  - Generalize function and predicate symbols.
  - Work with types, terms, formulas.
- Function application and abstraction.
- Undefinedness.
  - Follows the traditional approach to undefinedness.
  - Separate kinds of undefinedness for types, terms, formulas.
- Definite and indefinite description.
- Quotation and evaluation.