CAS 760 Winter 2010

03 Techniques for Enhancing Traditional Logics

William M. Farmer

Department of Computing and Software
McMaster University

23 March 2010

Types

- A type system 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.
- 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 including dependent type constructors.
  - A universal type.
  - Possibly empty types.
  - Type variables.
  - Subtypes.
  - Dependent product and sum types.
- Implementation approaches:
  - The type system is built into the logic.
  - The type system is simulated using predicates.

Functions

- It is crucial that a general-purpose logic for mathematics and computing has strong support for reasoning with functions:
  - Function application.
  - Function abstraction.
  - Function types.
  - Partial functions.
  - Higher-order functions.
  - Quantification over functions.
  - Recursive definitions.
- Important issue: How should improper applications of partial functions be handled?

Basic Mathematical Objects

- Basic mathematical objects include truth values, numbers, and the following compound values:
  - Sets.
  - Functions.
  - Relations.
  - Tuples.
  - Sequences.
- A practical logic needs strong support for basic mathematical objects:
  - Appropriate types.
  - Constructors and selectors for compound values.
  - Quantification over compound values.

Partial Functions and Undefined Terms

- Partial functions and undefined terms are commonplace in mathematics and computing.
- Sources of undefinedness:
  1. Improper function applications.
  2. Improper definite descriptions.
  3. Improper indefinite descriptions.
- There are many approaches for handling partial functions and undefinedness.
- There is a traditional approach to undefinedness widely employed in mathematics but rarely used in computing.
- Most approaches fall into one of the following three categories:
  1. Approaches in a traditional logic.
  2. Approaches in a 2-valued logic with undefinedness.
  3. Approaches in a 3-valued logic with undefinedness.

Definite and Indefinite Description

- Benefits of definite and indefinite description:
  - Values can be described directly and naturally.
  - Definite description allows definitions to be explicit.
  - Indefinite description allows profiles to be explicit.
- Important issue: What value should an improper definite or indefinite description have?
- Definite and indefinite description works extremely well in a logic with undefinedness.
  - Improper definite and indefinite descriptions are undefined.
  - Partial functions can be defined using function abstraction and definite description.

Definition Principles

- Notational definitions
- Definitions and profiles of:
  - Constants.
  - Type constants and constructors.
  - Logical constants and other operators.
  - Variable binders.
- Recursive definitions.
  - Primitive recursion.
  - Monotone functionals.
- Algebraic data types.
  - Enumerated types.
  - Tuple and record types.
  - Disjoint union types.
  - Inductive data types.
- Important issue: How is conservativity ensured?

Two-Dimensional Notations

- Matrices.
  - Can concisely represent functions and relations.
  - Can be manipulated as algebraic objects.
- Graphs.
  - Can present complex relationships graphically.
  - Can finitely represent infinite structures.
- Tables.
  - Used to represent functions and relations.
  - Championed by David Parnas.

Reasoning about Syntax

- Reasoning about the semantics of expressions is often done via reasoning about the syntax of expressions.
- Examples:
  - Pattern matching.
  - Substitution.
  - Rules of inference.
  - Symbolic computation.
- Reasoning about syntax is usually performed in the metalogic.
- It is possible, but problematic, to reason about syntax directly in the logic.

Reasoning about Syntax in a Logic

- A system for reasoning about syntax in a logic needs:
  1. A representation of expressions as values in the logic's semantics.
  2. A quotation operation that associates an expression with its representation.
  3. An evaluation operation that associates a representation of an expression with the value of the expression.
- Representation approaches:
  - Abstract: An inductive data type.
  - Concrete: An encoding of expressions into numbers (Gödel numbers), trees, or sets.
- Important issues:
  - How is the liar paradox avoided?
  - How are expression fragments handled?