01—Introduction to CS3234; Propositional Calculus

CS 3234: Logic and Formal Systems

Martin Henz and Aquinas Hobor

August 12, 2010

1. Introduction to Logic and Formal Systems
2. Brief Introduction to CS3234
3. Administrative Matters

1. Introduction to Logic and Formal Systems
   - Origins of Mathematical Logic
   - Propositional Calculus
   - Predicate Calculus
   - Theorem Proving and Logic Programming
   - Systems of Logic
2. Brief Introduction to CS3234
3. Administrative Matters

What is logic?

1. the branch of philosophy dealing with forms and processes of thinking, especially those of inference and scientific method,
2. a particular system or theory of logic [according to 1].

(from “The World Book Dictionary”)

Origins of Mathematical Logic

Greek origins

The ancient Greek formulated rules of logic as *syllogisms*, which can be seen as precursors of formal logic frameworks.

Example of Syllogism

Premise

All men are mortal.

Premise

Socrates is a man.

Conclusion

Therefore, Socrates is mortal.

Historical Notes

Logic traditions in Ancient Greece

Stoic logic: Centers on propositional logic; can be traced back to Euclid of Megara (400 BCE)

Peripatetic logic: Precursor of predicate logic; founded by Artistotle (384–322 BCE), focus on syllogisms

Logic Throughout the World

Indian logic: Nyaya school of Hindu philosophy, culminating with Dharmakirti (7th century CE), and Gangea Updhyya of Mithila (13th century CE), formalized inference

Chinese logic: Gongsun Long (325–250 BCE) wrote on logical arguments and concepts; most famous is the "White Horse Dialogue"; logic typically rejected as trivial by later Chinese philosophers

Islamic logic: Further development of Aristotelian logic, culminating with Algazel (1058–1111 CE)

Medieval logic: Aristotelian; culminating with William of Ockham (1288–1348 CE)

Traditional logic: Port-Royal Logic, influential logic textbook first published in 1665

Remarks on Ockham

Ockham's razor (in his own words)

For nothing ought to be posited without a reason given, unless it is self-evident or known by experience or proved by the authority of Sacred Scripture.

Remarks on Ockham

Ockham's razor (in his own words)

For nothing ought to be posited without a reason given, unless it is self-evident or known by experience or proved by the authority of Sacred Scripture.

Ockham's razor (popular version, not found in his writings)

Entia non sunt multiplicanda sine necessitate.

Remarks on Ockham

Ockham's razor (in his own words)

For nothing ought to be posited without a reason given, unless it is self-evident or known by experience or proved by the authority of Sacred Scripture.

Ockham's razor (popular version, not found in his writings)

Entia non sunt multiplicanda sine necessitate.
English: Entities should not be multiplied without necessity.

Remarks on Ockham

Ockham's razor (in his own words)

For nothing ought to be posited without a reason given, unless it is self-evident or known by experience or proved by the authority of Sacred Scripture.

Ockham's razor (popular version, not found in his writings)

Entia non sunt multiplicanda sine necessitate.
English: Entities should not be multiplied without necessity.

Built-in Skepticism

As a result of this *ontological parsimony*, Ockham states that human reason cannot prove the immortality of the soul nor the existence, unity, and infinity of God.

Propositional Calculus

Study of atomic propositions

Propositions are built from sentences whose internal structure is not of concern.

Building propositions

Boolean operators are used to construct propositions out of simpler propositions.

Example for Propositional Calculus

Atomic proposition

One plus one equals two.

Atomic proposition

The earth revolves around the sun.

Combined proposition

One plus one equals two *and* the earth revolves around the sun.

Goals and Main Result

Meaning of formula

Associate meaning to a set of formulas by assigning a value *true* or *false* to every formula in the set.

Proofs

Symbol sequence that formally establishes whether a formula is always true.

Soundness and completeness

The set of provable formulas is the same as the set of formulas which are always true.

Uses of Propositional Calculus

Hardware design

The production of logic circuits uses propositional calculus at all phases; specification, design, testing.

Verification

Verification of hardware and software makes extensive use of propositional calculus.

Problem solving

Decision problems (scheduling, timetabling, etc) can be expressed as satisfiability problems in propositional calculus.

Predicate Calculus: Central ideas

Richer language

Instead of dealing with atomic propositions, predicate calculus provides the formulation of statements involving sets, functions and relations on these sets.

Quantifiers

Predicate calculus provides statements that all or some elements of a set have specified properties.

Compositionality

Similar to propositional calculus, formulas can be built from composites using logical connectives.

Progamming Language Semantics

The meaning of programs such as

```
if x >= 0 then y := sqrt(x) else y := abs(x)
```

can be captured with formulas of predicate calculus:

$$\forall x \forall y (x' = x \land (x \geq 0 \rightarrow y' = \sqrt{x}) \land (\neg(x \geq 0) \rightarrow y' = |x|))$$

Other Uses of Predicate Calculus

Specification: Formally specify the purpose of a program in order to serve as input for software design,

Verification: Prove the correctness of a program with respect to its specification.

Example for Specification

Let P be a program of the form

```
while a <> b do
   if a > b then a := a - b else a:= b - a;
```

The specification of the program is given by the formula

$$\{a \geq 0 \wedge b \geq 0\}\ P\ \{a = gcd(a, b)\}$$

Theorem Proving and Logic Programming

Theorem proving

Formal logic has been used to design programs that can automatically prove mathematical theorems.

Logic programming

Research in theorem proving has led to an efficient way of proving formulas in predicate calculus, called *resolution*, which forms the basis for *logic programming*.

Other Systems of Logic

Three-valued logic

A third truth value (denoting “don’t know” or “undetermined”) is often useful.

Intuitionistic logic

A mathematical object is accepted only if a finite construction can be given for it.

Temporal logic

Integrates time-dependent constructs such as (“always” and “eventually”) explicitly into a logic framework; useful for reasoning about real-time systems.

1. Introduction to Logic and Formal Systems
2. Brief Introduction to CS3234
   - Style: Broad, elementary, rigorous
   - Method: From Theory to Practice
   - Overview of Module Content
3. Administrative Matters

Style: Broad, elementary, rigorous

Broad: Cover a good number of logical frameworks

Elementary: Focus on a minimal subset of each framework

Rigorous: Cover topics formally, preparing students for advanced studies in logic in computer science

Method: From Theory to Practice

Cover theory and back it up with practical excercises that apply the theory and give new insights.

Overview of Module Content

1. Traditional logic (1 lectures, including today)
2. Propositional calculus (2 lectures)
3. Predicate calculus (3 lectures)
4. Program Verification (2 lectures)
5. Modal Logics (2 lectures)
6. Typing (2 lectures; to be confirmed)

Administrative Matters

- Use `www.comp.nus.edu.sg/~cs3234` and IVLE
- No textbook
- Assignments (one per week, starting next week; marked)
- Coq homework (every 2 weeks)
- Coq quiz (every 2 weeks)
- Discussion forums, announcements, webcast (IVLE)
- Labs (one per week); register!
- Tutorials (one per week); register!